Airlines Reservation System

Submitted By:Areeba Amjad(2021-SE-17)
Submitted to:Miss Farwah Shah

**Department of Computer Science
University of Engineering and technology
Lahore(New Campus).**

Content

This page is intentionally left blank

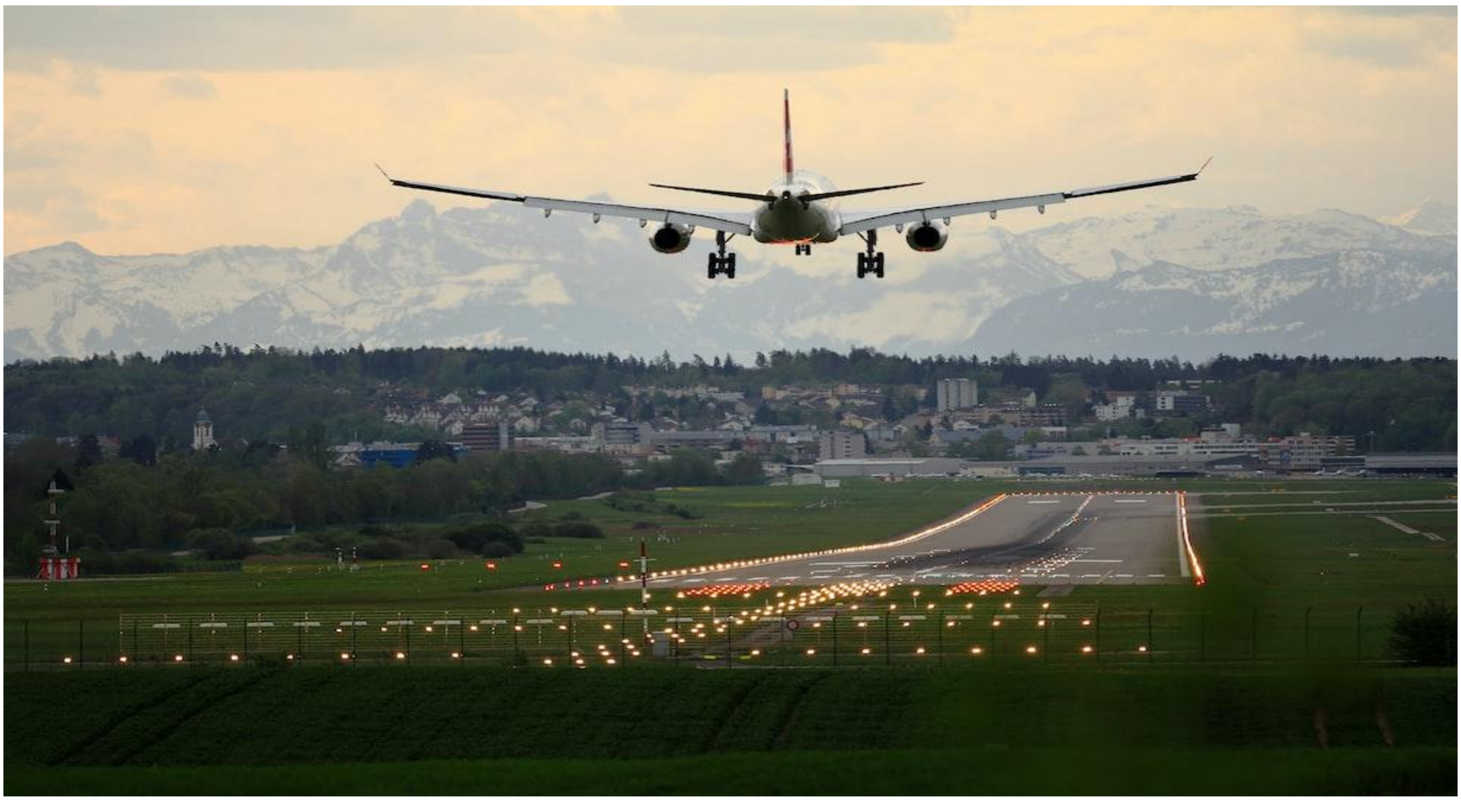
Introduction to Airlines Reservation System

What is ARS:

Airline reservation systems (ARS) are systems that allow an airline to sell their inventory (seats).
It contains information on schedules and fares and contains a database of reservations (or passenger name records) and of tickets issued (if applicable).
ARSs are part of passenger service systems (PSS), which are applications supporting the direct contact with the passenger.

History:

Founded in 1972, the ARS originally manufactured computer terminals for many uses throughout the aviation and travel sectors including airline reservation centers, airport operations and travel agency systems. Over 450,000 computer terminals were manufactured between 1972 and 2002 and although refurbishments are still supported today with many units still in use globally at airlines and airports, the company diversified into airline software development in 1987.



Market Analysis

Introduction:

The Airlines Reservation System is a crucial software tool for airlines that allows them to manage their flight schedules, ticket booking, and other operations. There are many similar products in the market that offer various functionalities for airlines. In this report, we will analyze at least 5 such products/projects in the market and compare their features, technologies, prices, and websites.

1-Amadeus:

Marketplace: Software house
Functions/Modules Implemented: Flight booking, itinerary management, fare pricing, ancillary services, departure control, inventory management, and business intelligence.
Technology used: Java, .NET, PHP, HTML, CSS, JavaScript, and Apache
Price: Customized pricing based on the client's requirements
Website: https://amadeus.com/
Home page screen: https://amadeus.com/en

2-Sabre:

Marketplace: Software house
Functions/Modules Implemented: Flight booking, travel itinerary management, and airline operations management.
Technology used: Java, .NET, PHP, HTML, CSS, JavaScript, and Apache
Price: Customized pricing based on the client's requirements
Website: https://www.sabre.com/
Home page screen: https://www.sabre.com/

3-Travelport:

Marketplace: Software house
Functions/Modules Implemented: Flight booking, itinerary management, fare pricing, ancillary services, departure control, inventory management, and business intelligence.
Technology used: Java, .NET, PHP, HTML, CSS, JavaScript, and Apache
Price: Customized pricing based on the client's requirements
Website: https://www.travelport.com/
Home page screen: https://www.travelport.com/

4-Expedia:

Marketplace: Online travel agency
Functions/Modules Implemented: Flight booking, hotel booking, car rental, and vacation packages.
Technology used: Java, .NET, PHP, HTML, CSS, JavaScript, and Apache
Price: Customized pricing based on the client's requirements
Website: https://www.expedia.com/
Home page screen: https://www.expedia.com/

5-Kayak:

Marketplace: Online travel agency
Functions/Modules Implemented: Flight booking, hotel booking, car rental, and vacation packages.
Technology used: Java, .NET, PHP, HTML, CSS, JavaScript, and Apache
Price: Customized pricing based on the client's requirements
Website: https://www.kayak.com/
Home page screen: https://www.kayak.com/

Conclusion:

All the above mentioned products have similar features and functionalities in terms of flight booking, itinerary management, fare pricing, and ancillary services.

They also use similar technologies for their development. The pricing is customized based on the client's requirements, and there is no fixed price. However, the marketplaces differ with **Amadeus**, **Sabre**, and **Travelport** being software houses.

While **Expedia** and **Kayak** are online travel agencies. The home page screens of all these products are user-friendly, and they provide easy navigation to their customers. Based on the analysis, it can be concluded that the **Airlines Reservation System** is a highly competitive market with several players offering similar functionalities to the airlines.

Project Introduction

Project Description

We will be designing a Airlines Reservations System in which both Admin and User can login to their accounts.Admin after his authentication can perform different functionalities such as add flight,delete flight,view scheduled flights.
While on the other hands User can book his flight,cancel his flight,view all schedules related to the flight etc.

Scope

The scope of an Airlines Reservation System includes the following functionalities:

1-Flight scheduling and inventory management: The system should allow airlines to manage their flight schedules and inventory effectively.

2-Ticket booking and reservations: The system should allow customers to book and reserve tickets for flights easily and efficiently.

3-User side management: The system should enable customers to manage their itineraries, including flight bookings, hotel reservations, and car rentals.

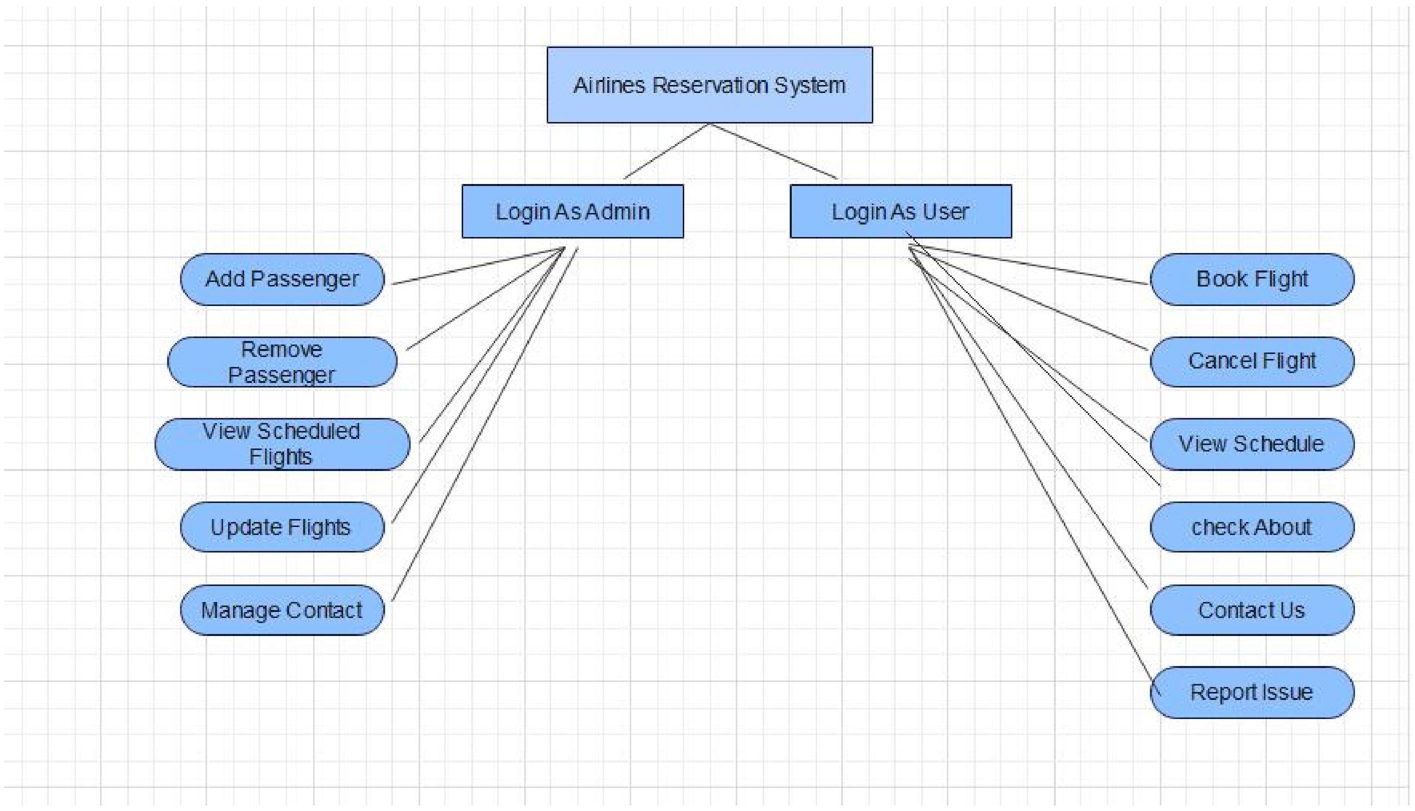
4-Fare pricing and ancillary services: The system should enable airlines to price their fares competitively and offer ancillary services such as seat upgrades, in-flight meals, and extra baggage.

6-**Departure control and boarding management:** The system should enable airlines to manage their departure and boarding processes efficiently, including checking in passengers, assigning seats, and boarding passengers.

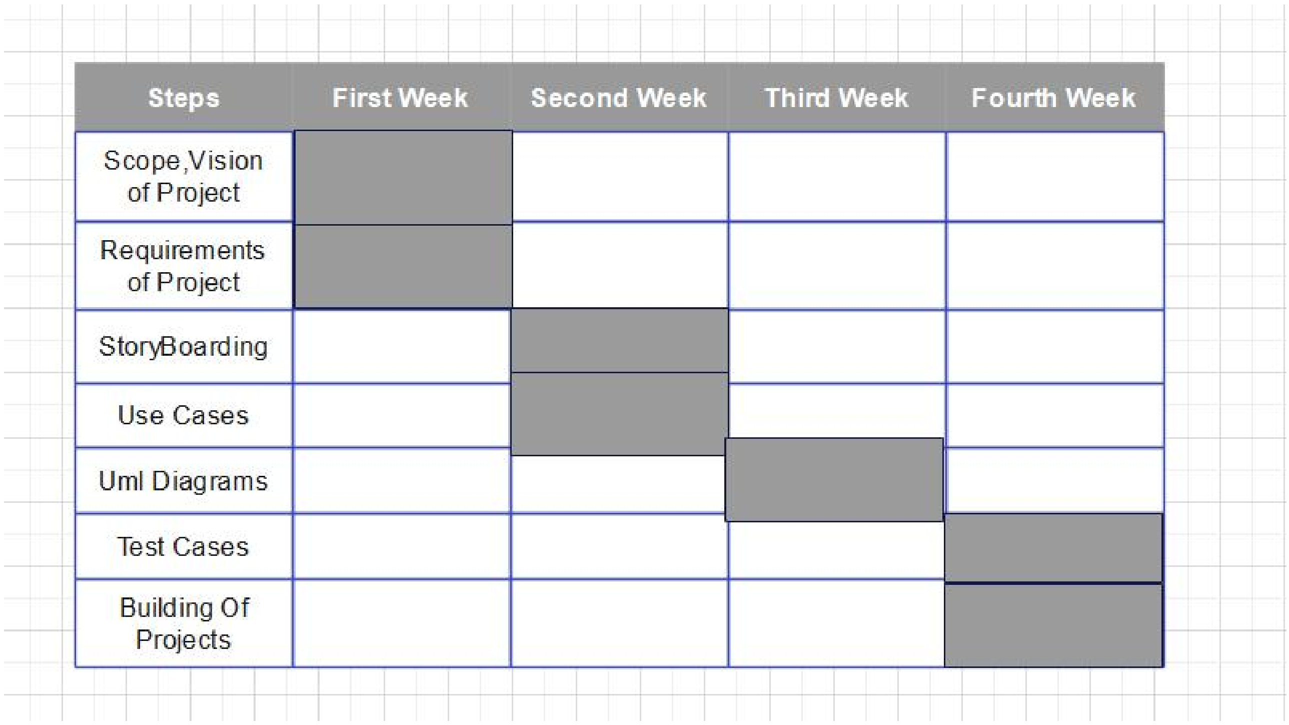
Vision

The vision of an Airlines Reservation System is to provide a seamless and efficient booking experience for customers while improving operations for airlines. The system should enable airlines to offer a wide range of services and options to customers, including online check-in, seat selection, and baggage tracking. The system should also provide real-time updates and notifications to customers, leading to increased customer satisfaction and loyalty.
The vision of an Airlines Reservation System is to provide a comprehensive and user-friendly booking platform for customers while enabling airlines to improve their operations, increase revenue, and offer a competitive advantage in the airline industry.

Work Break Down Structure



Gantt Chart



Steps	First Week	Second Week	Third Week	Fourth Week
Scope,Vision of Project	▨			
Requirements of Project	▨			
StoryBoarding		▨		
Use Cases		▨		
Uml Diagrams			▨	
Test Cases				▨
Building Of Projects				▨

Requirement Analysis

Functional Requirements

1-Flight Search:

The system should allow customers to search for flights based on their preferred dates, destinations, and departure times.

2-Flight Booking:

The system should allow customers to book flights by selecting a seat, meal options, and baggage allowance.

3-Flight Confirmation:

The system should provide confirmation of the booking to the customer and generate a ticket.

4-Booking Modification and Cancellation:

The system should allow customers to cancel or modify their bookings.

5-Online Check-in:

The system should allow customers to check-in online and print boarding passes.

6-Flight and Passenger Management:

The system should allow airline staff to manage flights, schedules, and availability.The system should allow staff to manage passenger information, including their personal details, bookings, and preferences.

7-Login:

The system should provide a login button for both Admin and the User.After clicking on that login button both admin and user will enter to their corresponding modes.

Non Functional Requirements

1-Availability and Scalability:

The system should be highly available, reliable, and scalable to handle a large number of users.

2-Security:

The system should be secure and protect user data from unauthorized access or theft.

3-Usability:

The system should be easy to use and have a user-friendly interface.

4-Compatibility:

The system should be compatible with multiple devices, including desktops, laptops, tablets, and smartphones.

5-Performance:

The system should have fast response times and be able to handle multiple concurrent transactions.

6-Reporting:

The system should be able to generate reports on flight schedules, bookings, and revenue.

7-Maintainability:

The system should be easily maintainable and upgradable.

Business Requirements

1-Revenue Generation:

The system should increase revenue for the airline by enabling more bookings and reducing overhead costs.

2-Competitive Advantage:

The system should provide a competitive advantage to the airline by offering a seamless and efficient booking experience to customers.

3-Customer Satisfaction:

The system should improve customer satisfaction by providing convenient booking and check-in options.

4-Operational Efficiency:

The system should increase operational efficiency by automating manual processes and reducing errors.

5-Marketing Insights:

The system should provide insights into customer preferences and booking patterns to enable targeted marketing campaigns.

6-Regulatory Compliance:

The system should comply with industry regulations and standards, such as those related to data privacy and security.

Business Rules

1-Availability of seats:

The reservation system must ensure that seats are available for booking based on the airline's schedule, aircraft capacity, and any other constraints.

2-Pricing:

The reservation system must calculate the fare for each ticket based on a variety of factors such as the route, time of booking, cabin class, and any promotions or discounts.

3-Payment:

The reservation system must provide a secure and reliable payment process for customers to purchase tickets. This may include support for various payment methods and fraud prevention measures.

4-Ticket issuance and delivery:

The reservation system must generate electronic tickets and provide them to customers via email or other means.

5-Customer data management:

The reservation system must securely store customer data such as personal information, payment details, and travel history. It must comply with relevant data protection regulations and ensure customer privacy.

6-Integration with other systems:

The reservation system must integrate with other airline systems such as inventory management, flight operations,

User Requirements

1-Refunds and Cancellations:

The system should provide clear information on the refund and cancellation policies, and allow users to request refunds or cancellations easily if necessary.

2-Special Assistance:

The system should allow users to request special assistance, such as wheelchair assistance or assistance for passengers with disabilities.

3-Customer Support:

The system should offer customer support options, including live chat, email, or phone support, to assist users with any issues they may encounter.

4-Accessibility and User-Friendliness:

The system should be accessible and user-friendly, with easy navigation, clear instructions, and support for multiple languages.

5-Booking Flights:

The system should allow users to search for available flights based on their preferred dates, destinations, and other relevant criteria. Users should be able to book a flight and choose their seat preference.

Physical Product Requirements

1. Backup and storage devices
2. Mobile devices
3. Computer hardware
4. Networking equipment
5. Peripheral devices

External Interface

1. **User interfaces:**
   Graphical user interfaces (GUIs), command-line interfaces (CLIs), or voice recognition systems.
2. **Application programming interfaces (APIs):**
   Interfaces through which the software system interacts with other software applications or services, such as web services, databases, or third-party software components
3. **Network interfaces:**
   These are the interfaces through which the software system communicates over a network, such as through protocols like TCP/IP or HTTP.
4. **Device interfaces:**
   These are the interfaces through which the software system interacts with external hardware devices, such as sensors, actuators, or data acquisition systems.
5. **File interfaces:**
   These are the interfaces through which the software system reads and writes files, such as text files, spreadsheets, or databases.

Development Constraints

1-Group Bookings:

This feature allows users to book flights for a group of people at once. It can also offer group discounts, assign group seats, and manage group bookings.

2-Seat Selection:

The ability to choose preferred seats, such as window or aisle seats, can be a significant feature for travelers. Users can select their seats during the booking process, or after booking their flight.

3-Flight Tracker:

A flight tracker can be used to provide real-time information on flight status, delays, and gate changes. It can also help users plan their travel time to the airport.

4-In-flight Entertainment:

Providing in-flight entertainment such as movies, TV shows, and music can enhance the travel experience for passengers.

5-Mobile App:

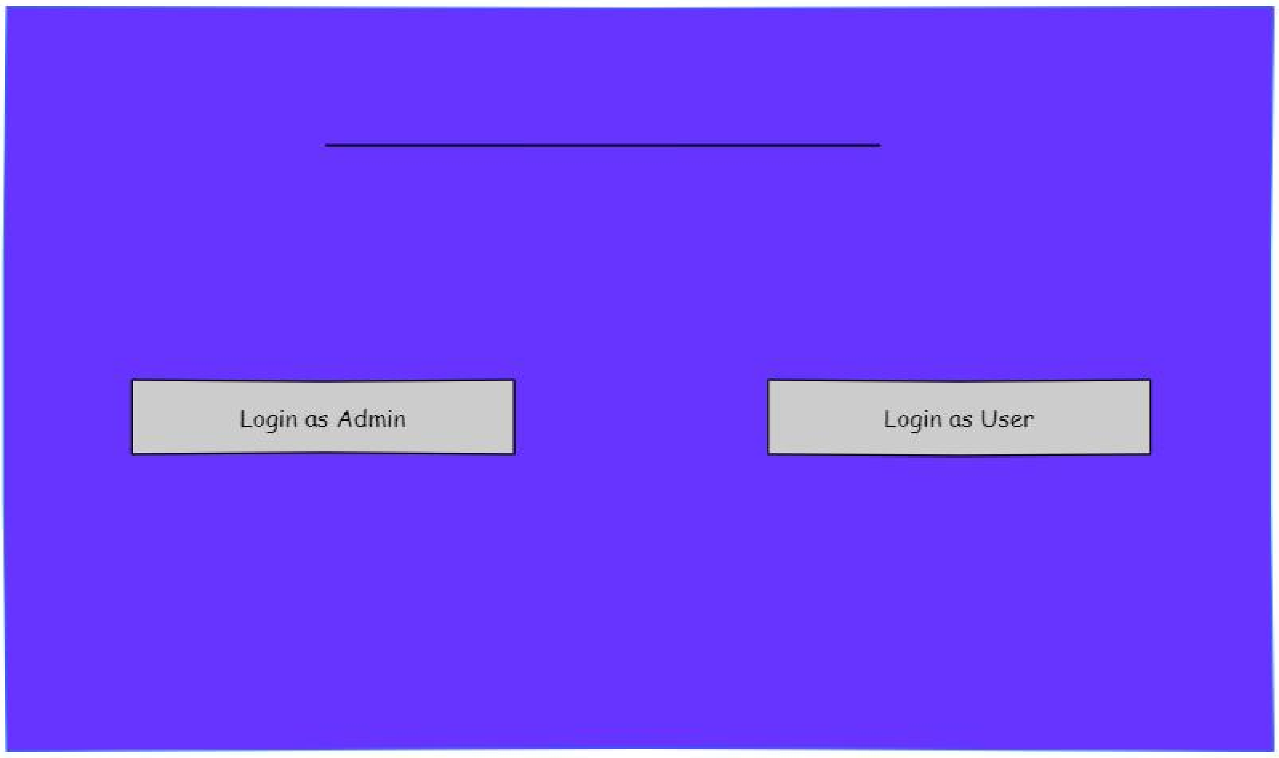
Developing a mobile app for your Airlines Reservation System can allow users to book their flights, view their itineraries, and check-in for their flights from their smartphones.

Wireframes and StoryBoarding

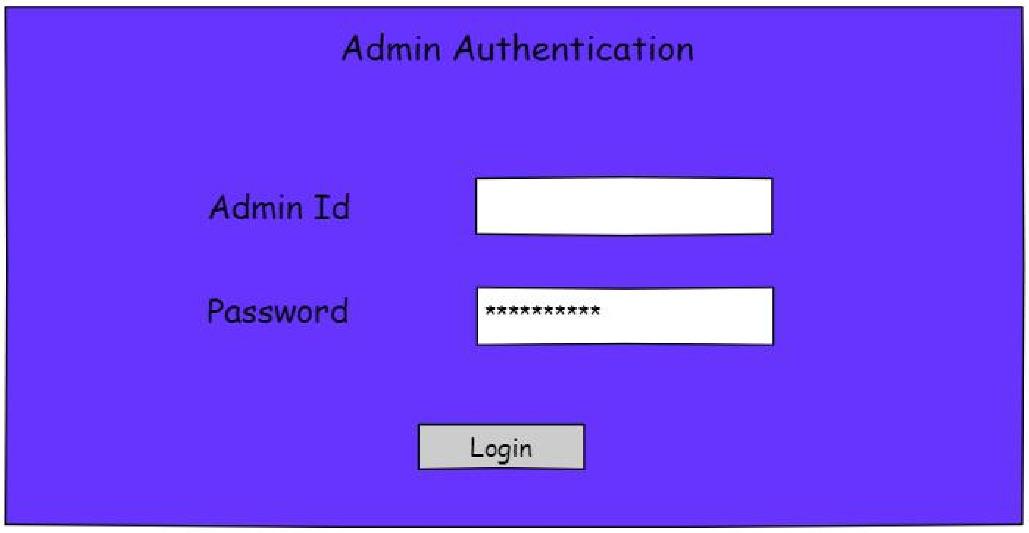
Wireframes

Admin End:

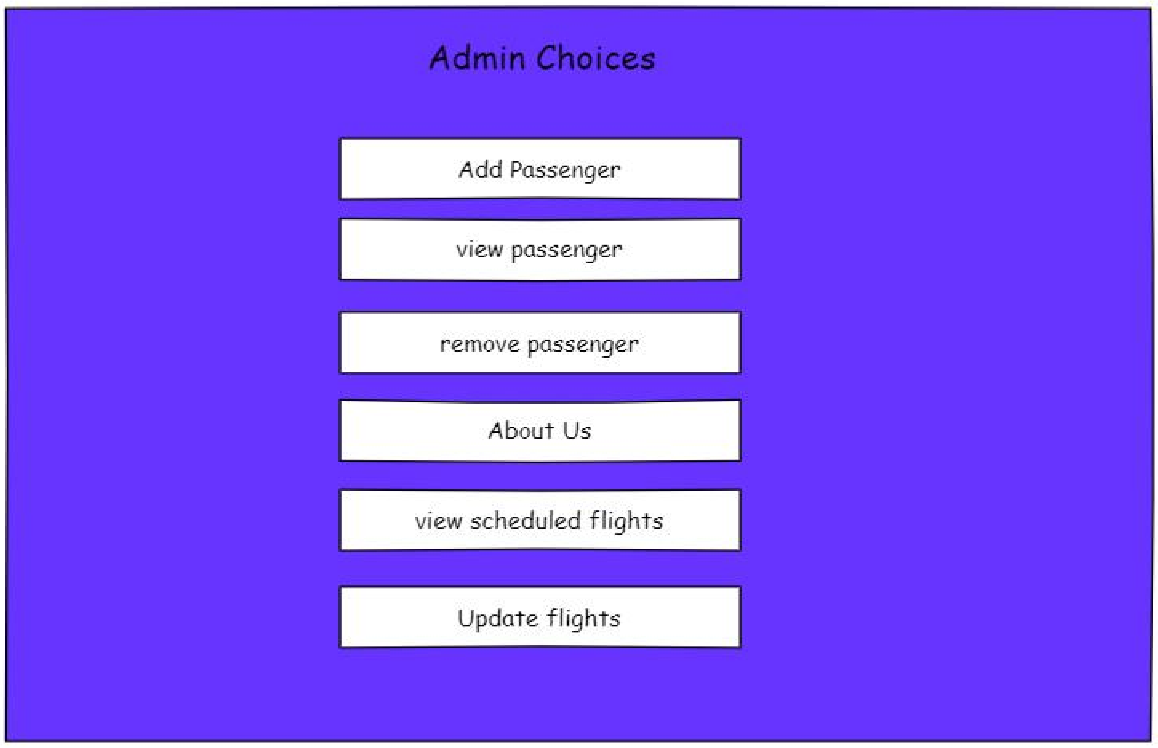
1-System will firstly ask that how(**“Login As”**) you want to enter into the system:



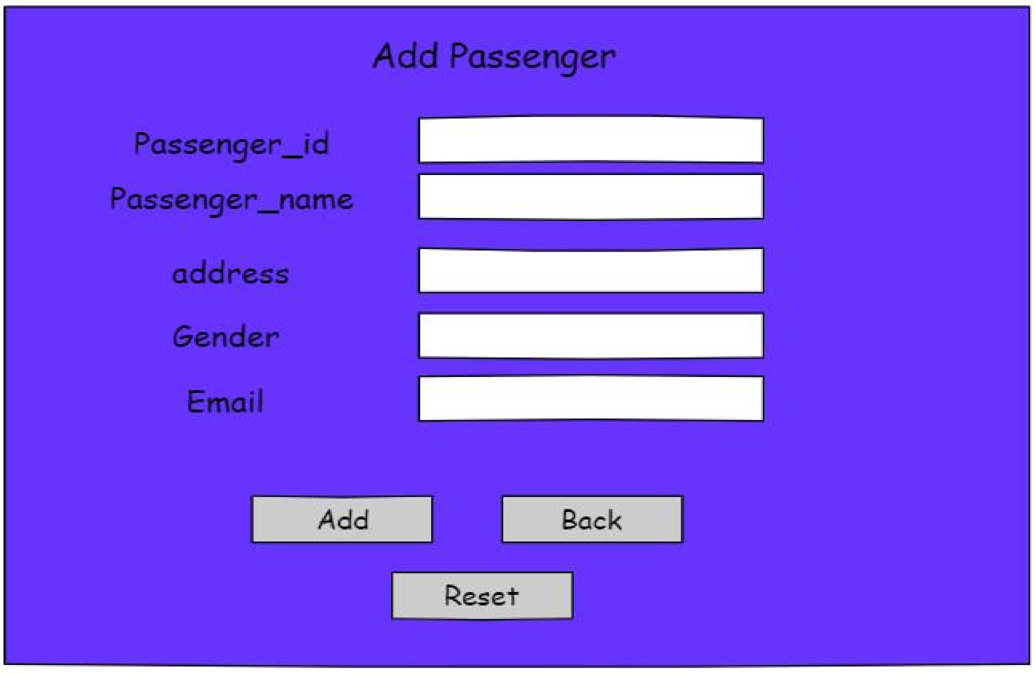
2-If the **“Login as admin ”** is clicked than system will authenticate the admin:



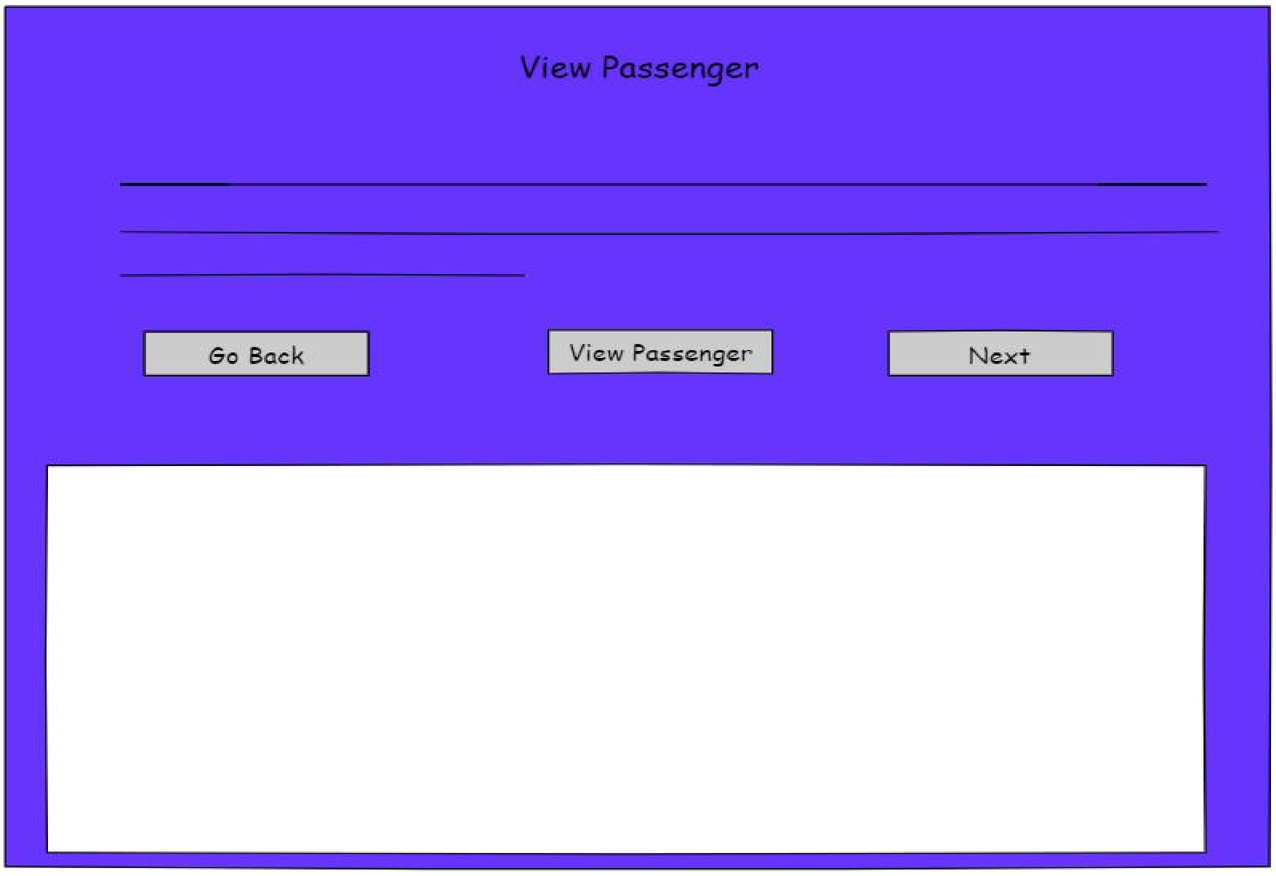
3-After authentication, **“admin choices”** will be appear on the screen:



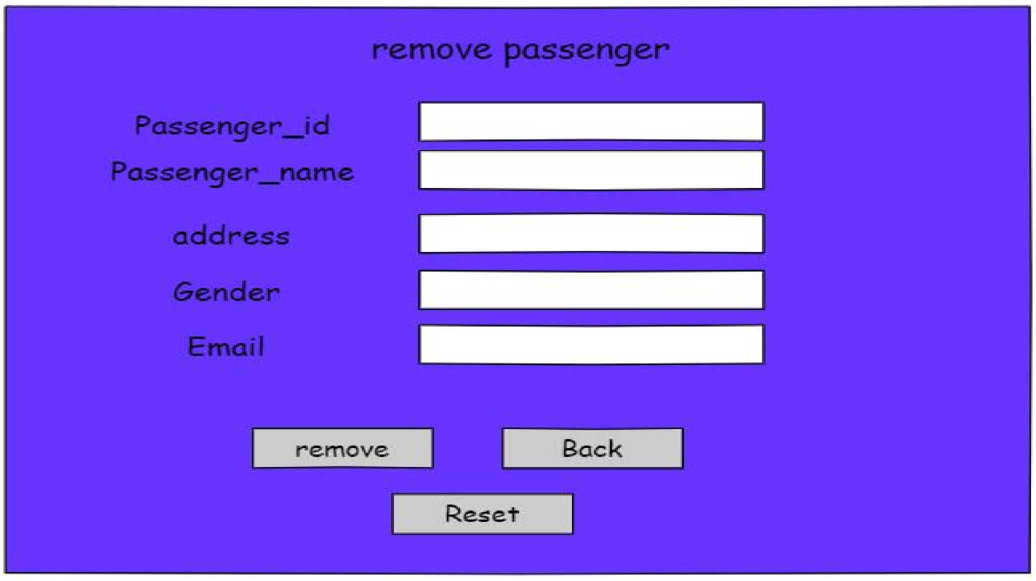
4-Admin will perform different activities according to his choice.Firstly he/she can **“Add Passenger”**:



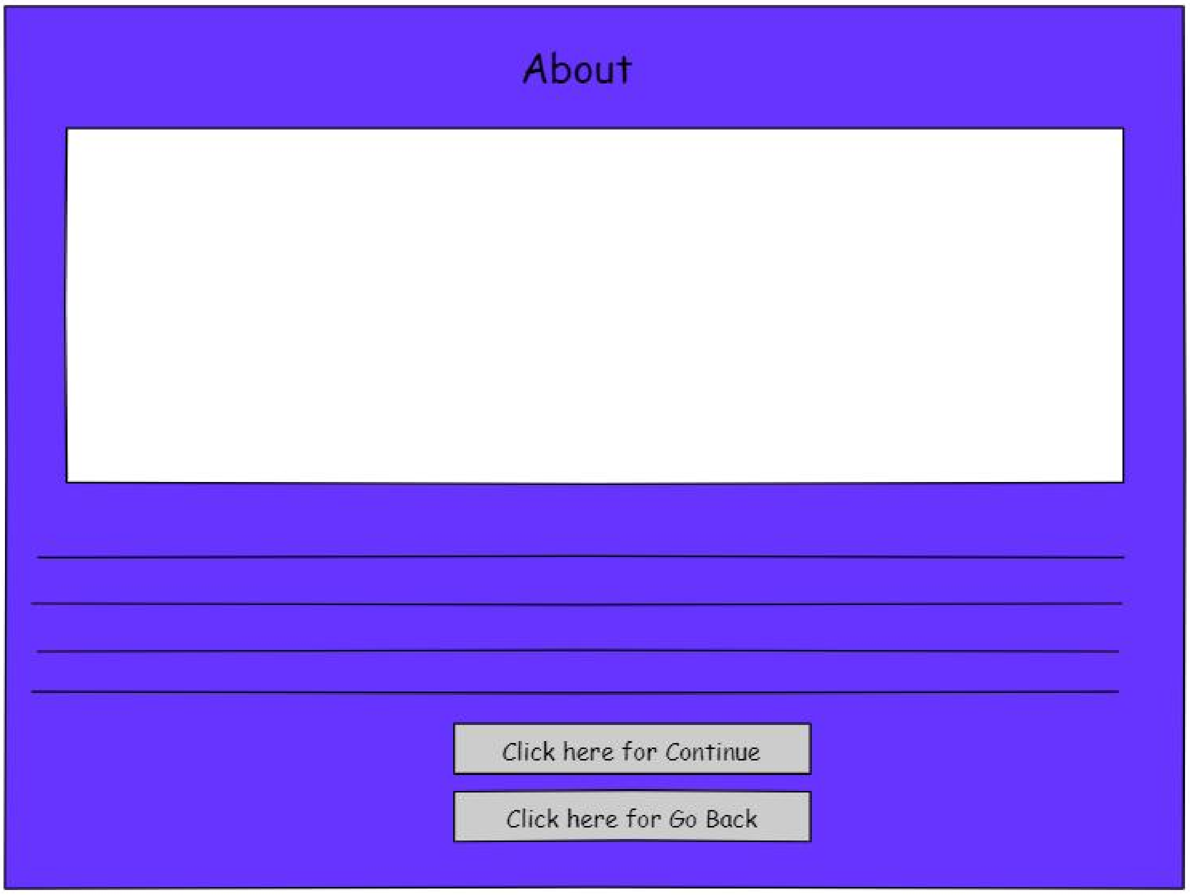
5-After adding a passenger into the system he/she can" **view the passenger"** from the system:



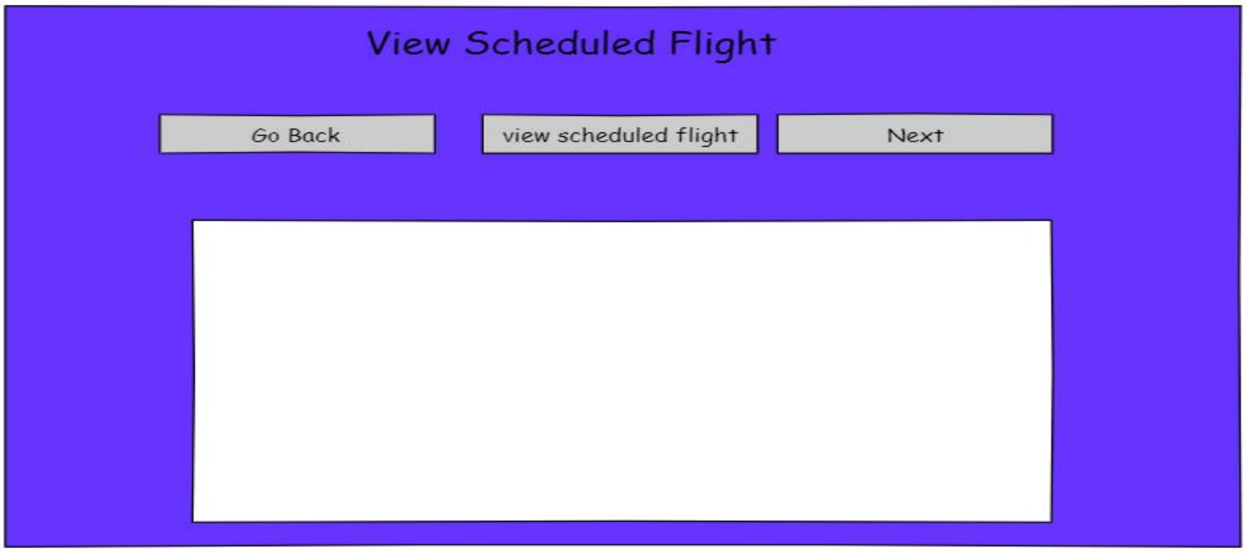
6-After adding a passenger into the system he/she can **"remove the passenger"** from the system:



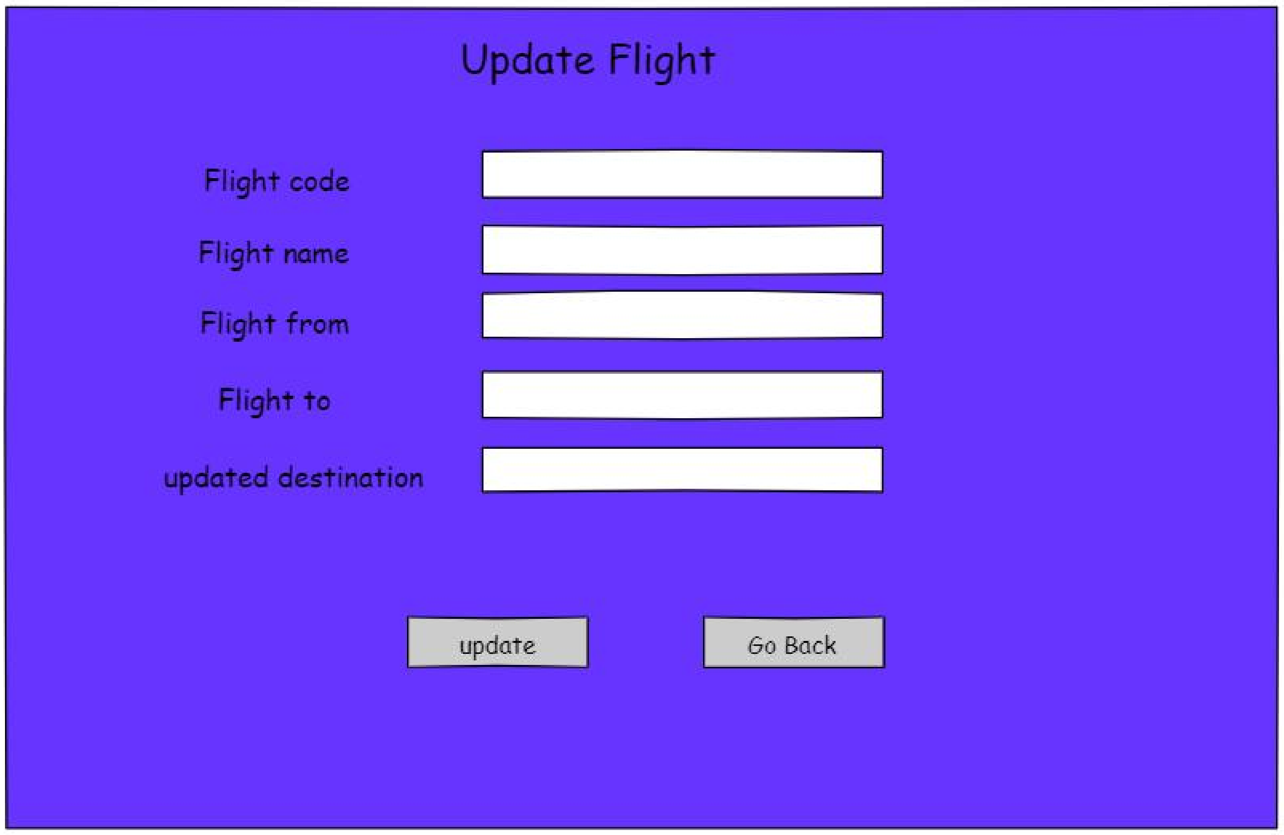
7-Admin can also view the **“About”** of the system:



8-Admin can also click on **“View Scheduled Flights”** to see the flights that have been scheduled:



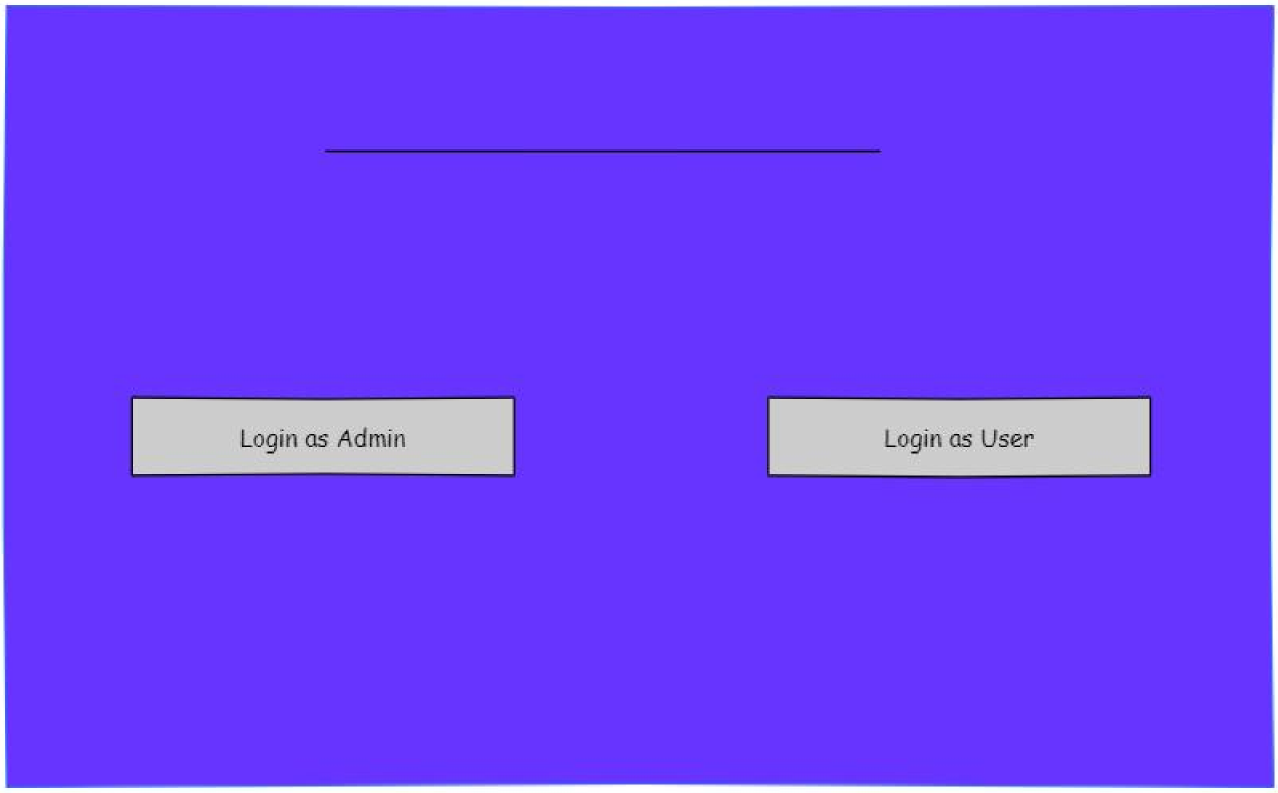
9-Admin can also **“Update Flight”** means he can make any change in the flights.



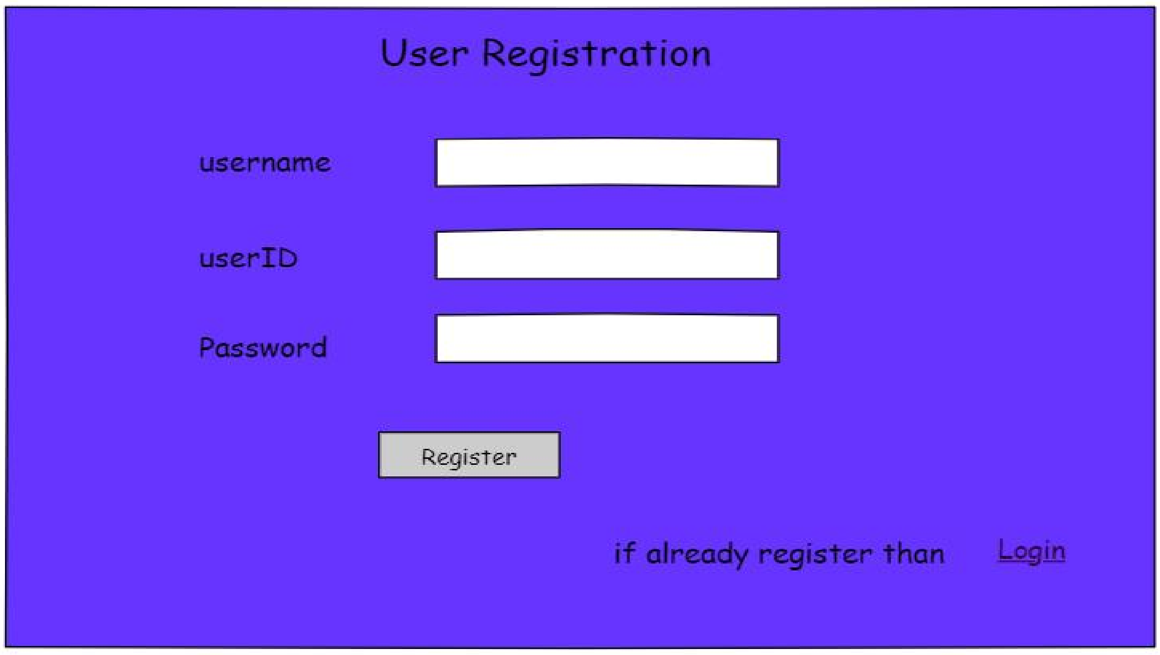
These are all the functions that an admin can perform on the **Airlines Reservation System .**
These can slightly vary from the original system.

User End:

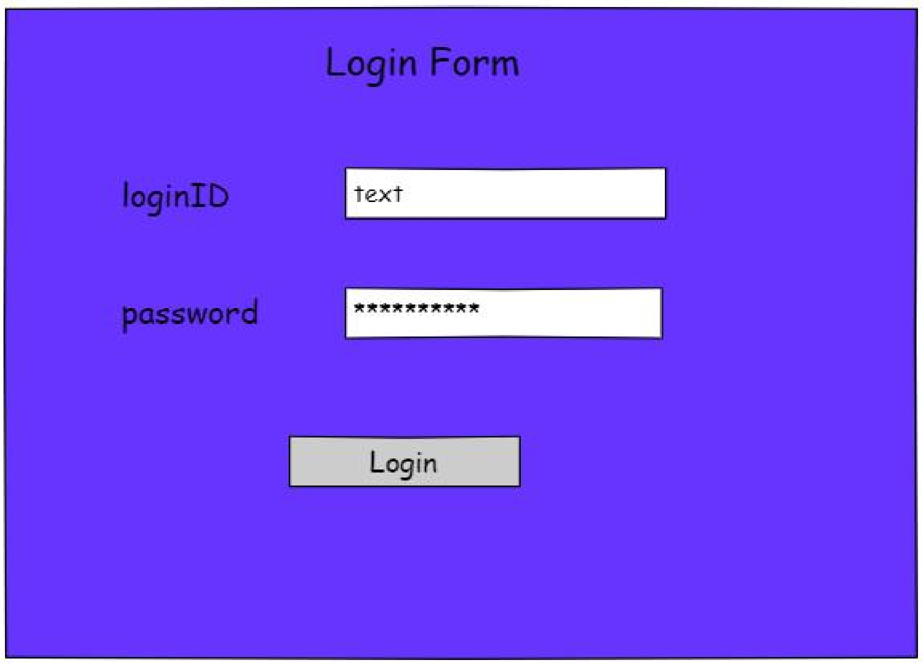
1-System will ask that how("Login AS") you want to enter into the system:



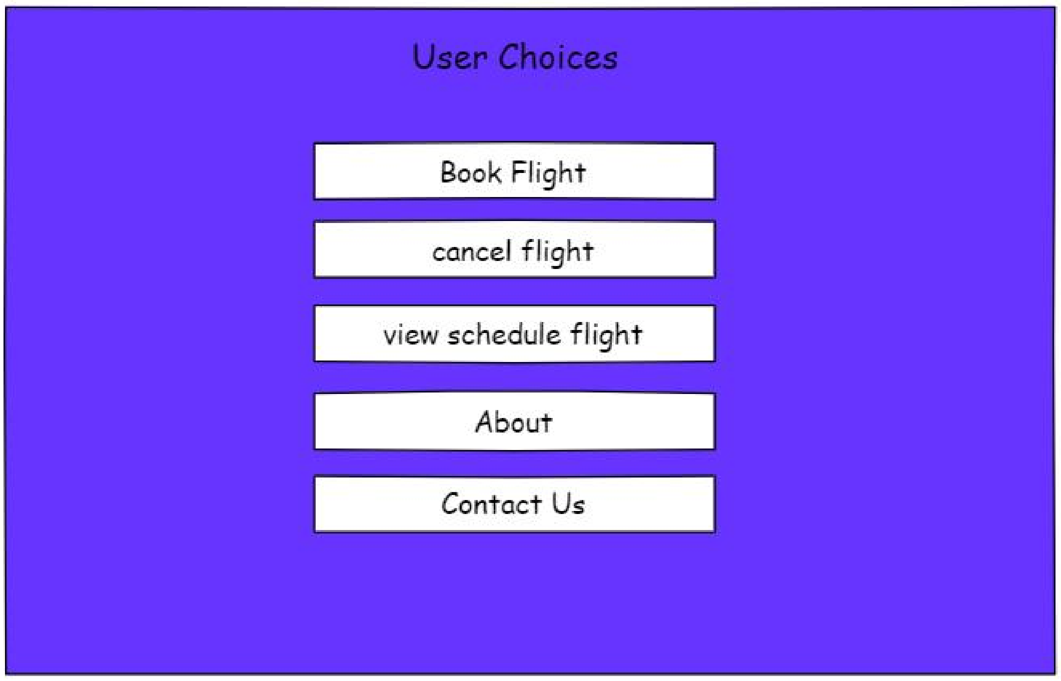
2-After clicking on **"Login as User"** the system will **"Register the student"**:



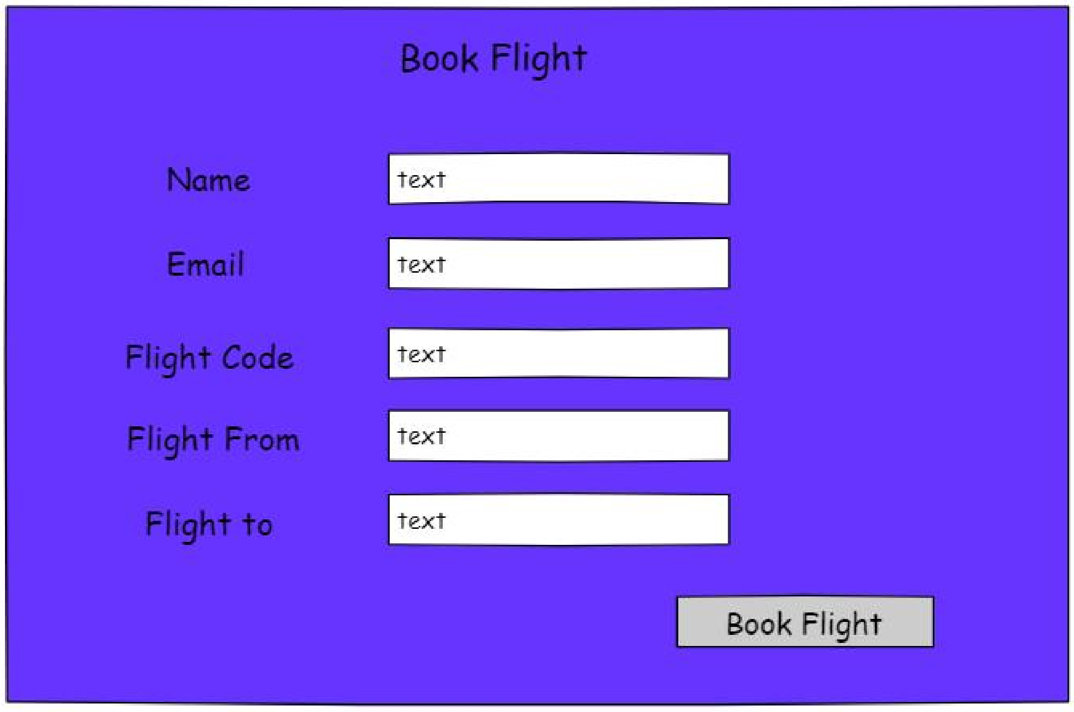
3-If a user is already registered he/she can click on **“Login”**:



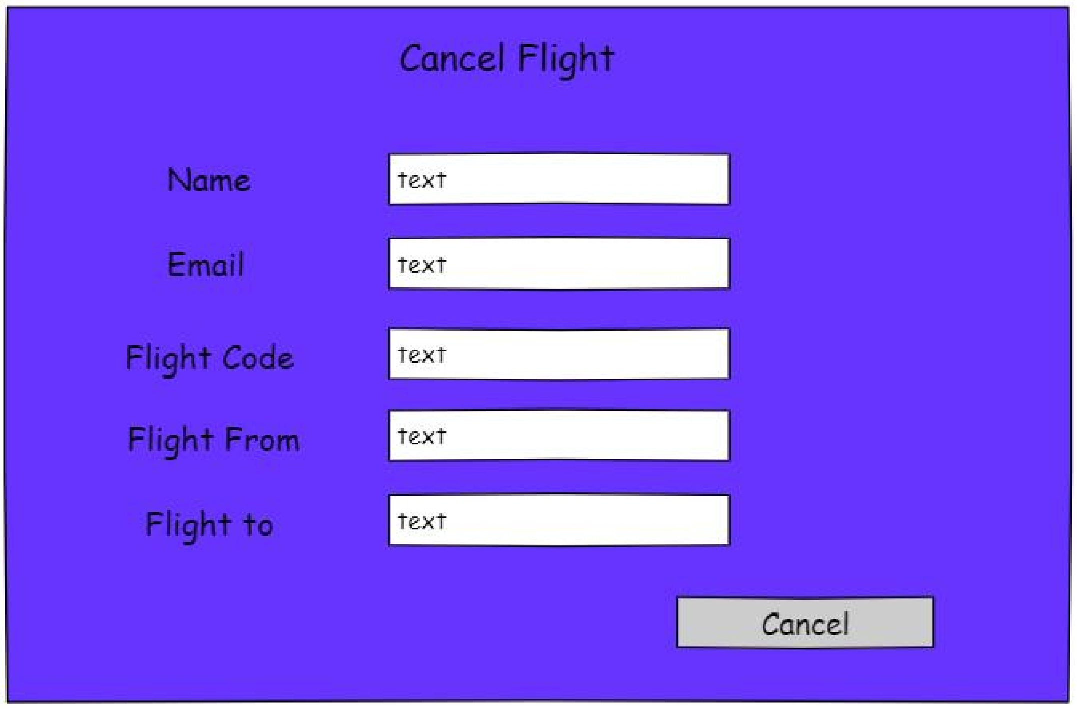
4-When a user is successfully login,then different **“choices”** will be appeared in front of him:



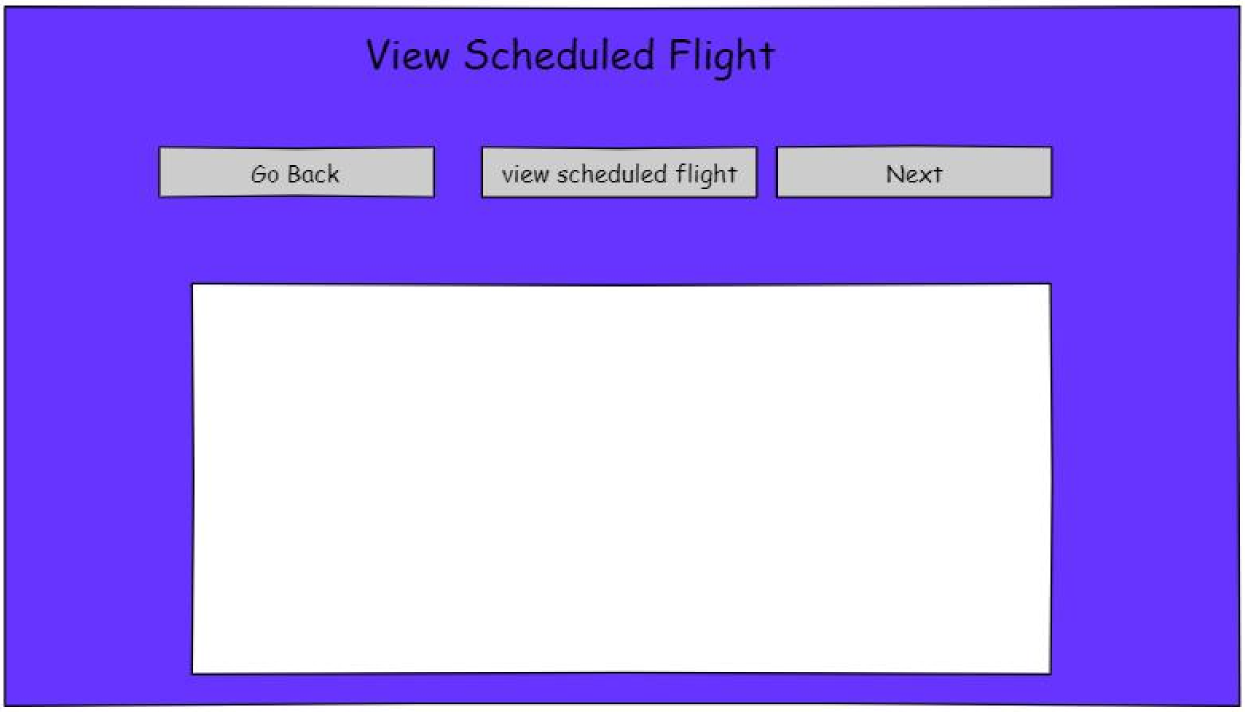
4-After that,User can **"Book Flight"** in order to book his flight:



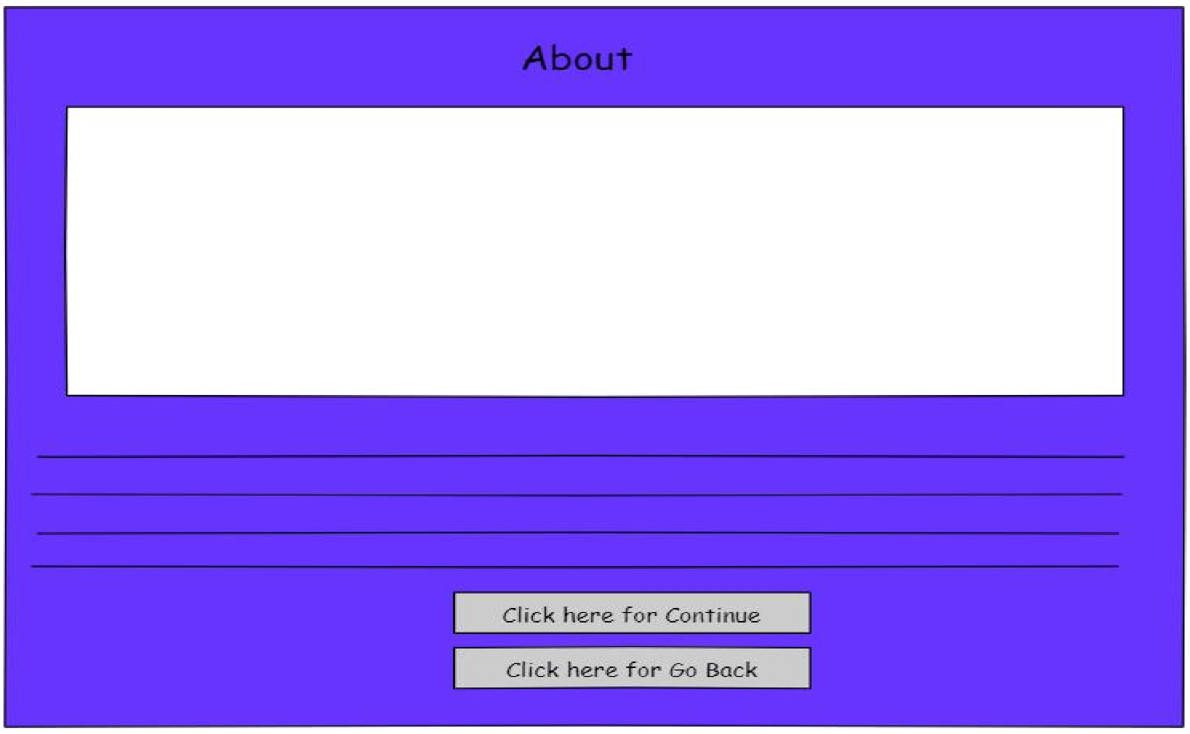
5-User can also **"Cancel Flight"** in order to cancel his flight:



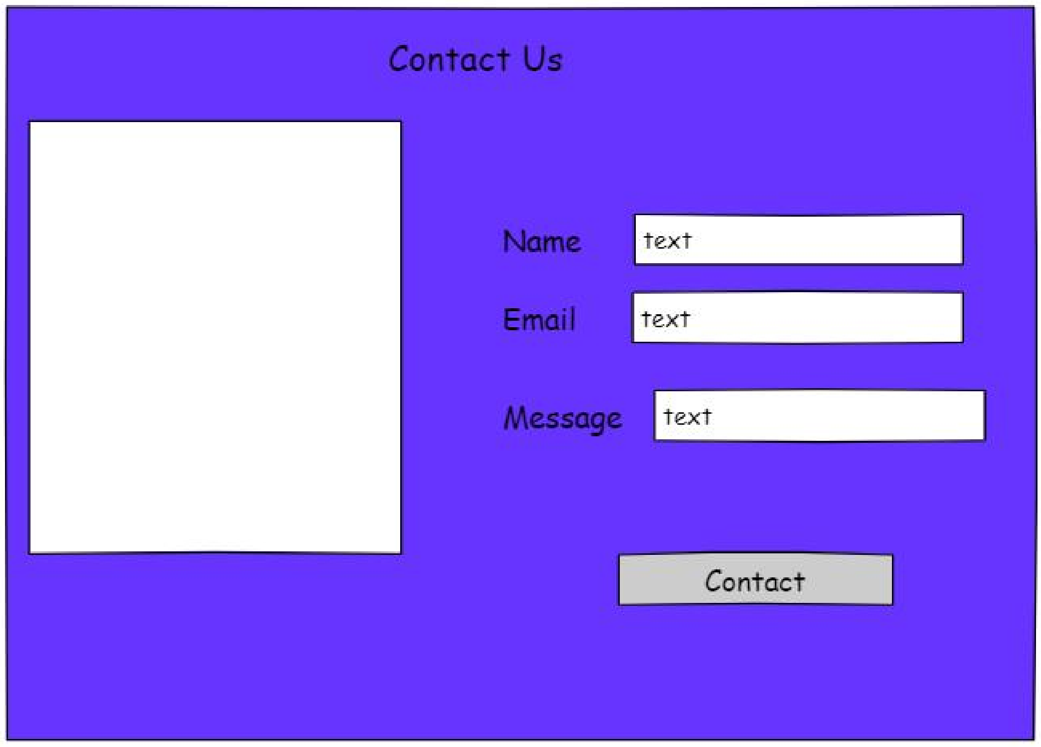
6-User can also see the **“View Scheduled Flights”** to see the flights that have been scheduled:



7-User can click **“About”** to see the about of the system:

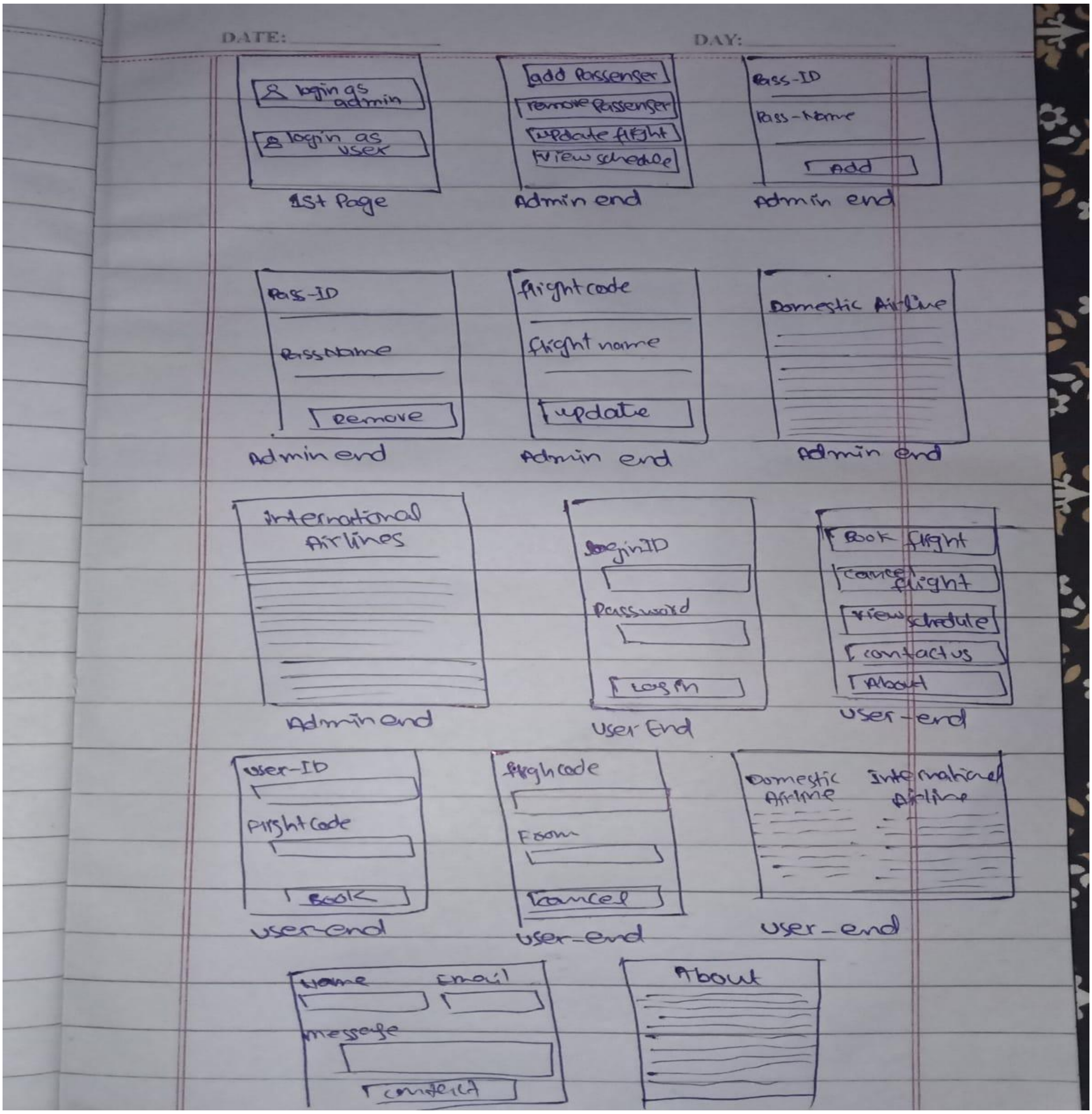


8-User can Click on **“Contact Us”** to contact with the system or in order to report any kind of issue:



When the user will click on Contact Button than his issue will be reported .

StoryBoarding



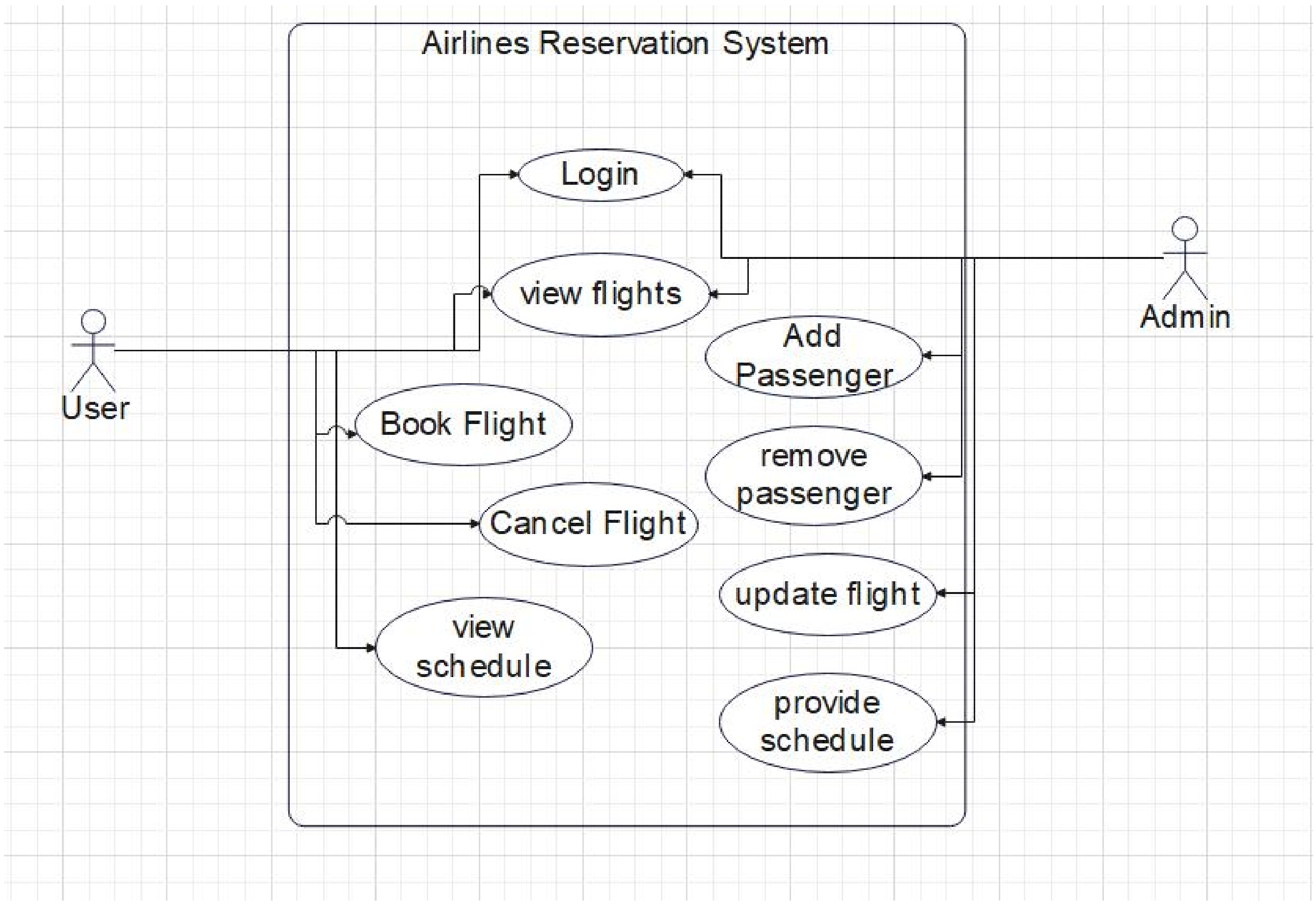
UML Modelling

Use Case Diagram

Description of use case:

A Use Case is a type of UML (Unified Modeling Language) diagram that is used to describe the functionality and behavior of a system from the user's perspective. The purpose of a Use Case is to provide a clear and detailed description of how a user or actor interacts with a system to achieve a specific goal or complete a particular task. In other words, a Use Case is a scenario or sequence of steps that describes how a user interacts with a system to accomplish a particular goal.

Use case of ARS:

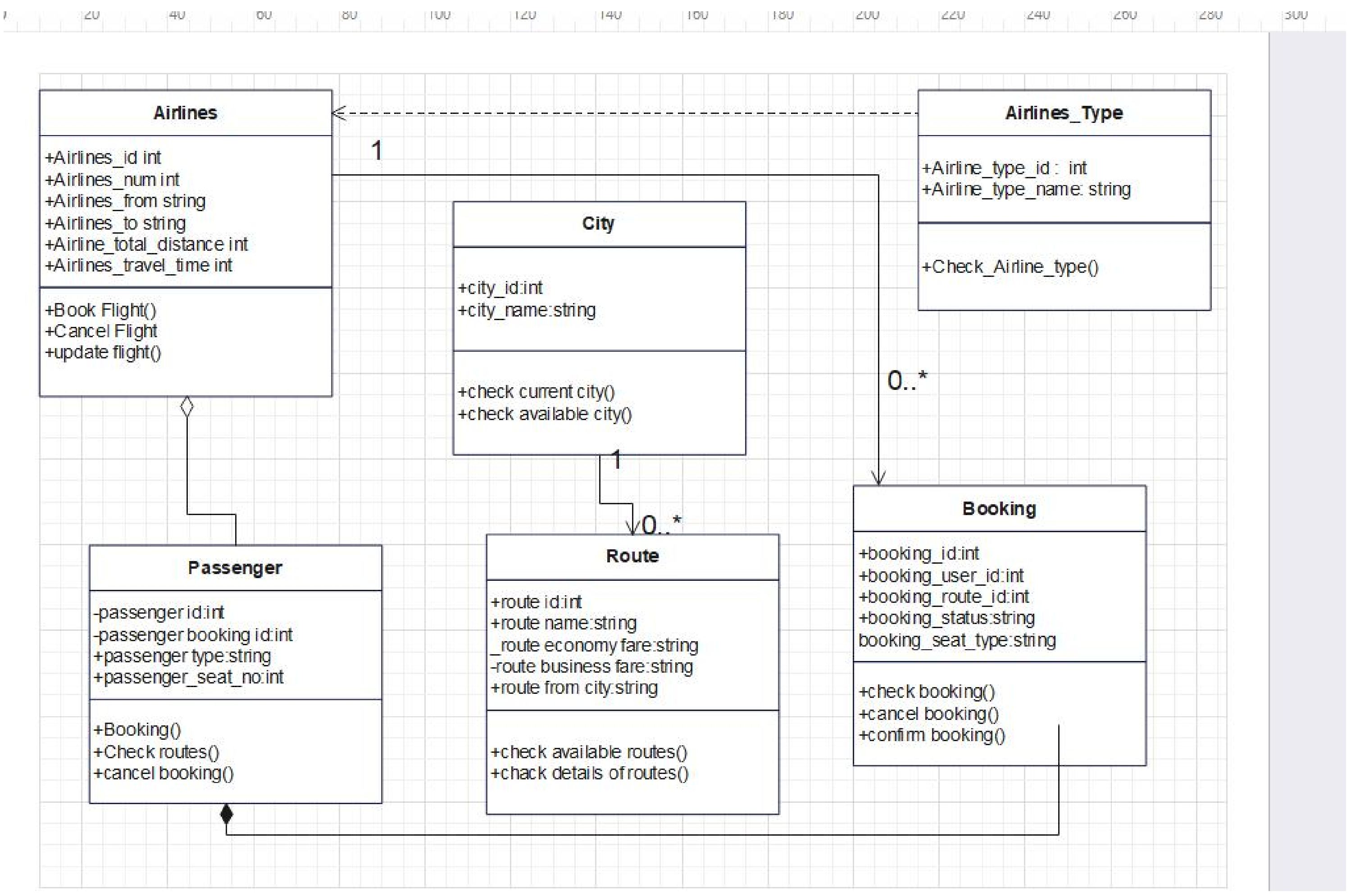


Class Diagram

Description of Class Diagram:

A Class diagram is a visual representation of the classes and objects in a system and their relationships, helping to provide a clear and concise overview of the system's structure and functionality. The purpose of a Class diagram is to provide a high-level view of the system's structure and the objects that make up the system.

Class Diagram of ARS:



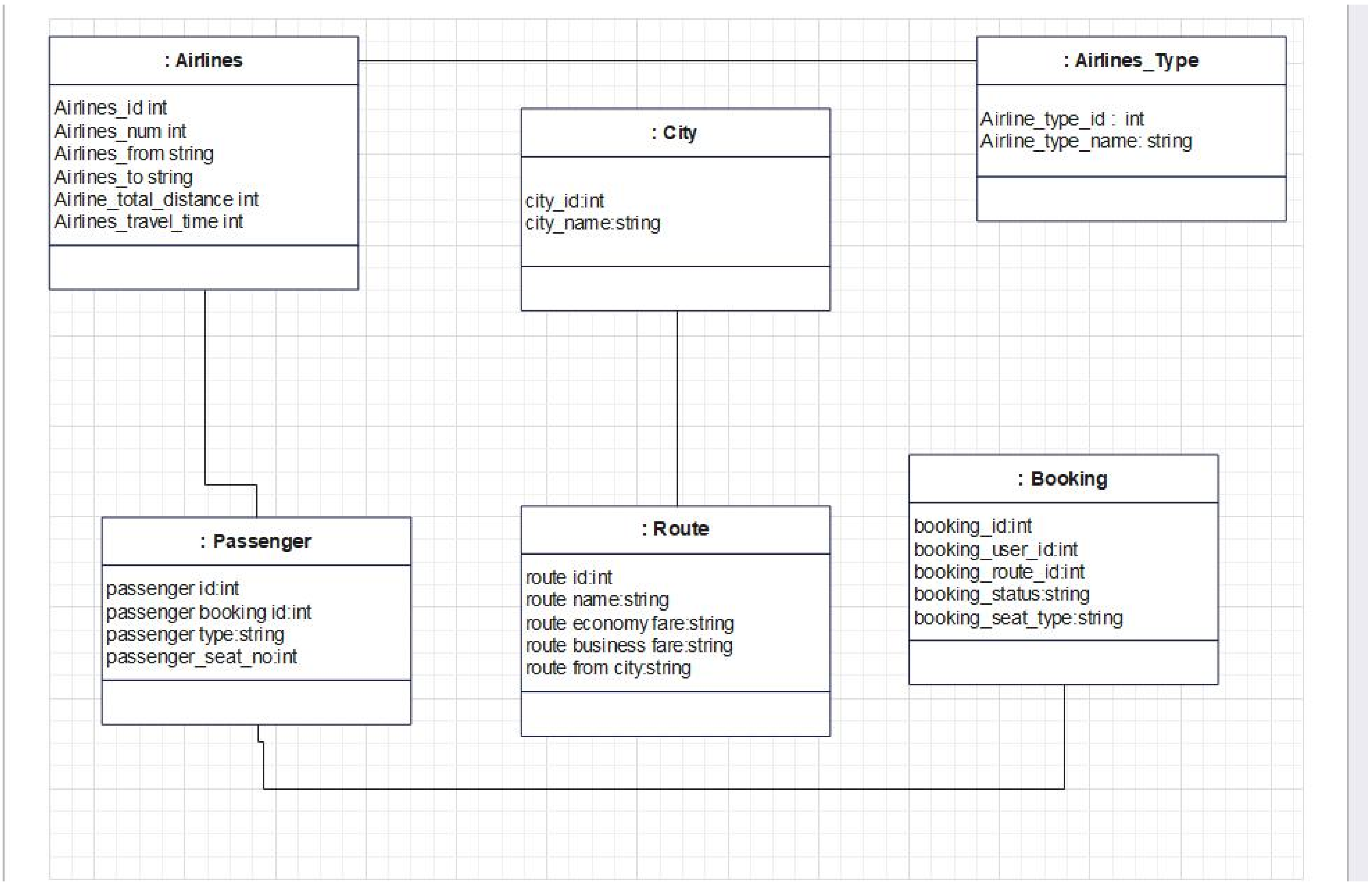
Object Diagram

Description of Object diagram:

An Object diagram is a type of UML (Unified Modeling Language) diagram that represents a specific instance of a class diagram at a particular moment in time. The purpose of an Object diagram is to provide a snapshot of the objects and their relationships in a system, including their attributes and the messages exchanged between them.

In other words, an Object diagram is a visual representation of the objects and their state at a specific point in time. It is useful for understanding the structure and behavior of a system, as well as for testing and debugging.

Object Diagram of ARS:



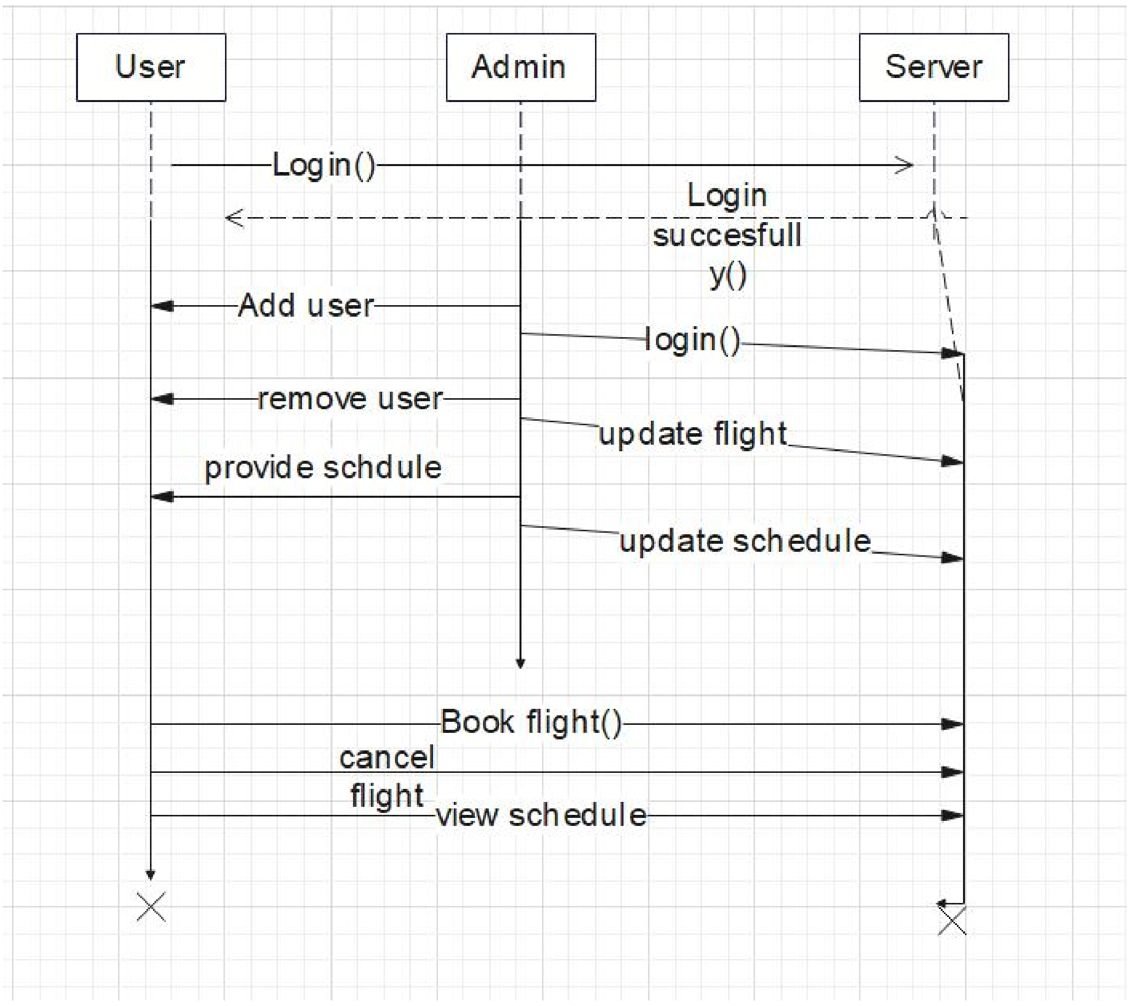
Sequence Diagram

Description of Sequence diagram:

A Sequence diagram is a type of UML (Unified Modeling Language) diagram that is commonly used to model and visualize the interactions between objects or components in a system over time. The purpose of a Sequence diagram is to show the flow of messages between the different objects or components involved in a particular scenario or use case.

In other words, a Sequence diagram helps to illustrate how the various parts of a system communicate and collaborate with each other to achieve a specific goal or accomplish a particular task.

Sequence Diagram of ARS:



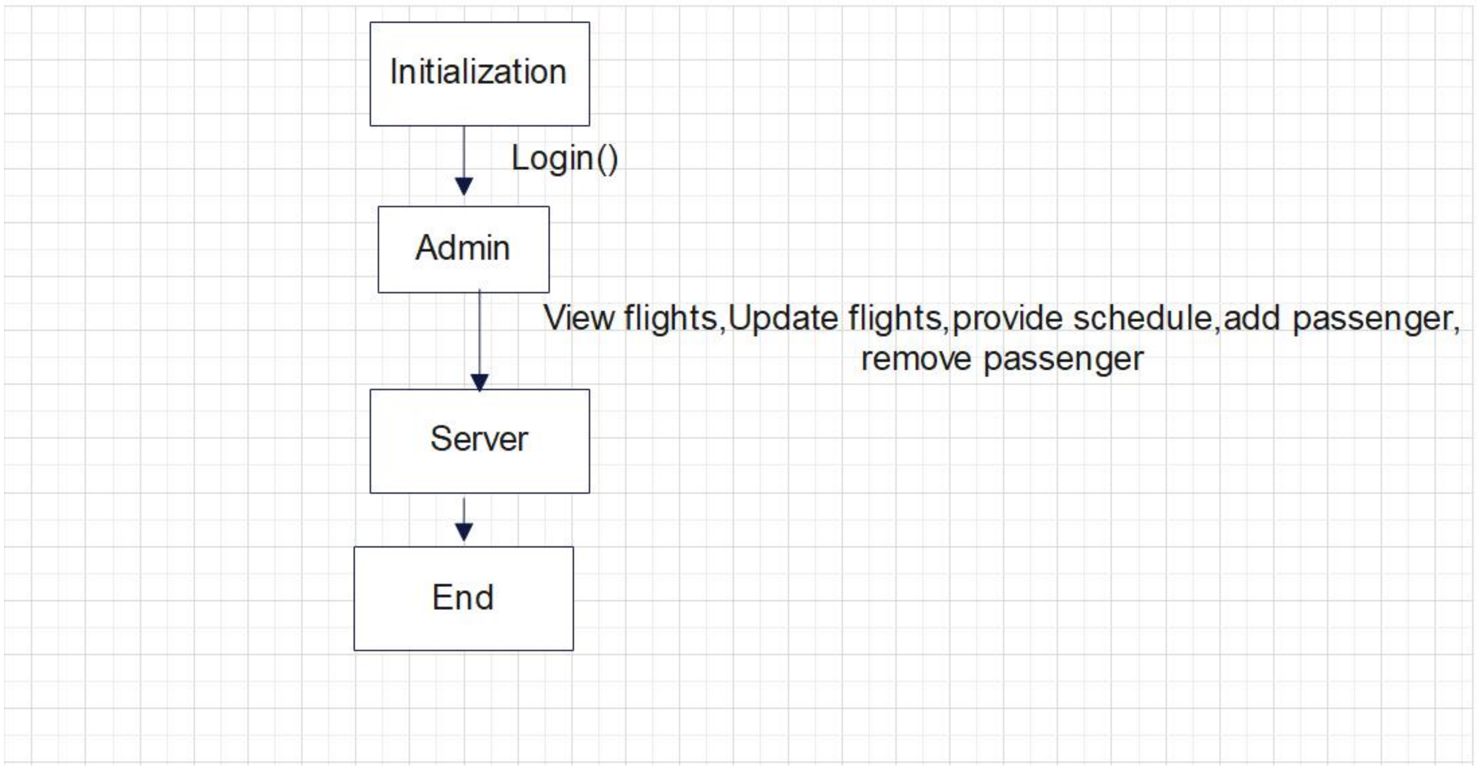
Collaboration Diagram

Description of Collaboration Diagram:

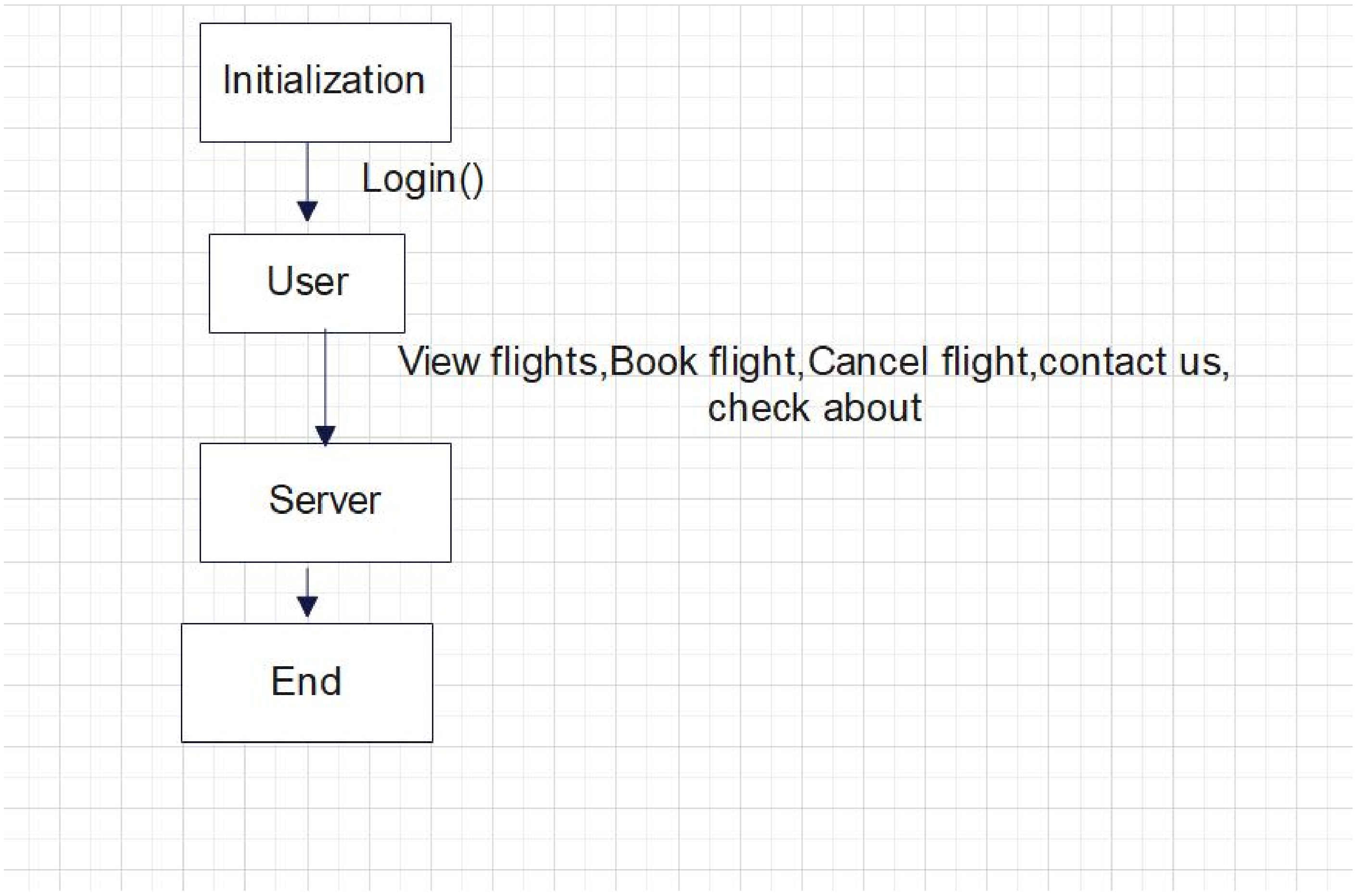
A Collaboration diagram, also known as a Communication diagram, is a type of UML (Unified Modeling Language) diagram that shows the interactions and relationships between objects or components in a system in terms of messages exchanged between them. The purpose of a Collaboration diagram is to provide a visual representation of how objects or components collaborate to accomplish a particular task or use case.

Collaboration Diagram of ARS:

1-Admin End


Initialization
Login()
Admin
View flights,Update flights,provide schedule,add passenger, remove passenger
Server
End

2-User Admin


Initialization
Login()
User
View flights,Book flight,Cancel flight,contact us, check about
Server
End

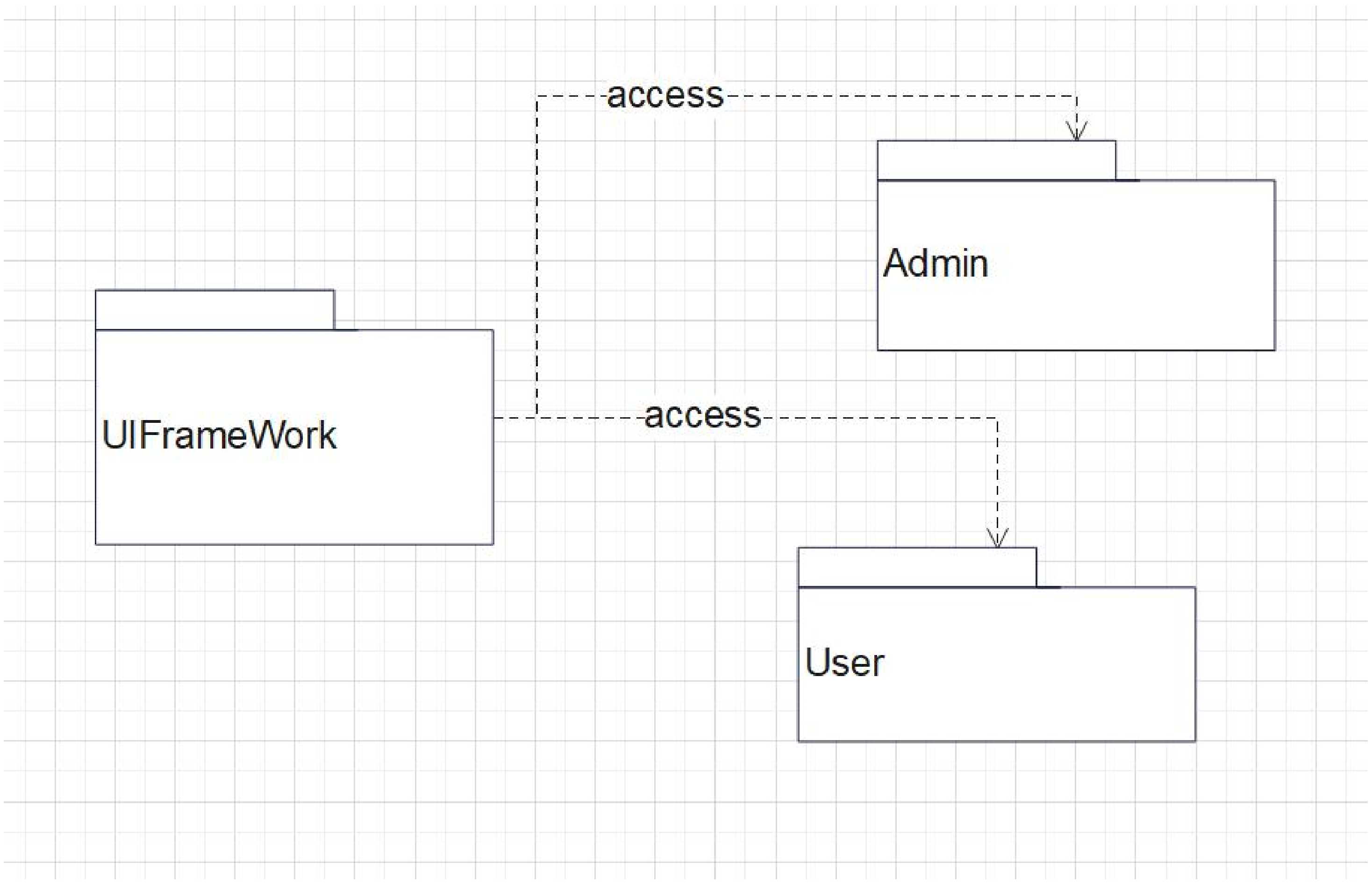
Since it is a Dynamic diagram so it has separate representation at both user end and admin end.

Package Diagram

Description of Package diagram:

Package diagrams are useful for visualizing and managing the complexity of large systems by breaking them down into smaller, more manageable units. The purpose of a package diagram in the Unified Modeling Language (UML) is to show the organization and dependencies of a system's components, which are grouped into packages.

Package Diagram of ARS:

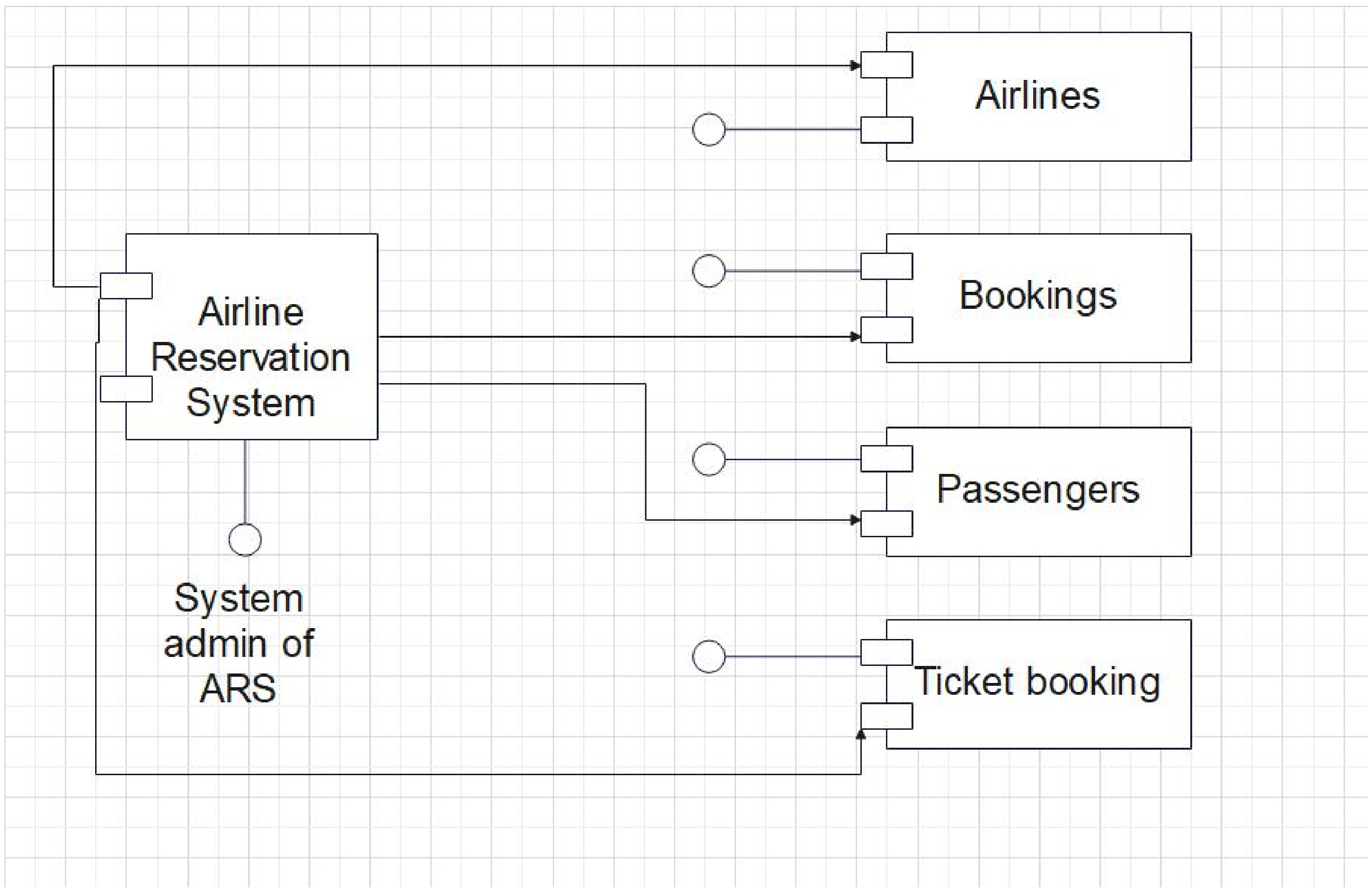


Component Diagram

Description of Component Diagram:

A Component diagram is a type of UML (Unified Modeling Language) diagram that shows the structural relationship between the components of a system and how they interact with each other. The purpose of a Component diagram is to provide a visual representation of the components that make up a system and how they are organized, helping to understand the structure and behavior of the system.

Component Diagram of ARS:

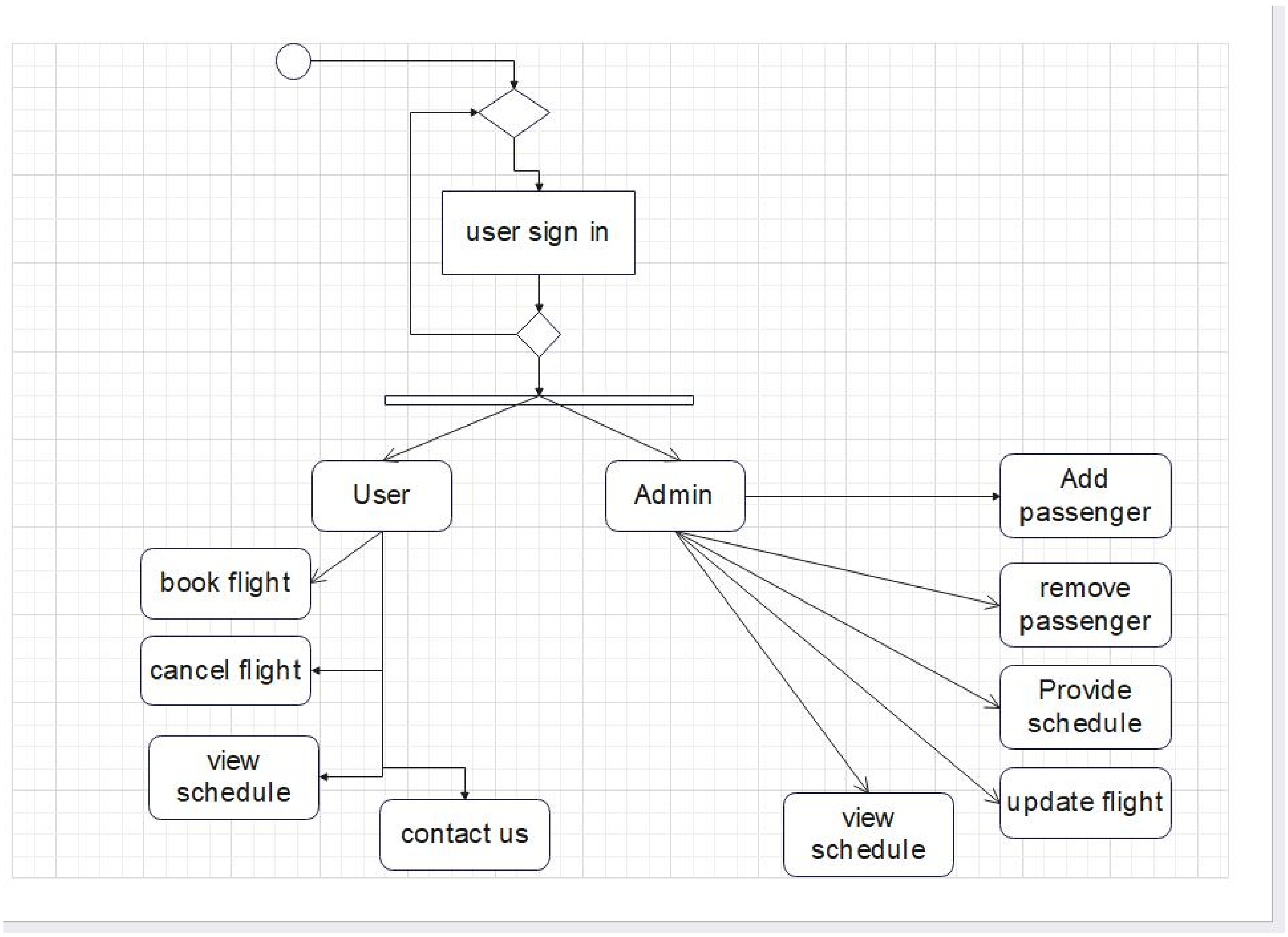


Activity Diagram

Description of Activity diagram:

An Activity diagram is a type of UML (Unified Modeling Language) diagram that represents the flow of activities or actions that occur in a system or process. The purpose of an Activity diagram is to model the behavior of a system or process by showing the flow of activities from start to finish.

Activity Diagram of ARS:

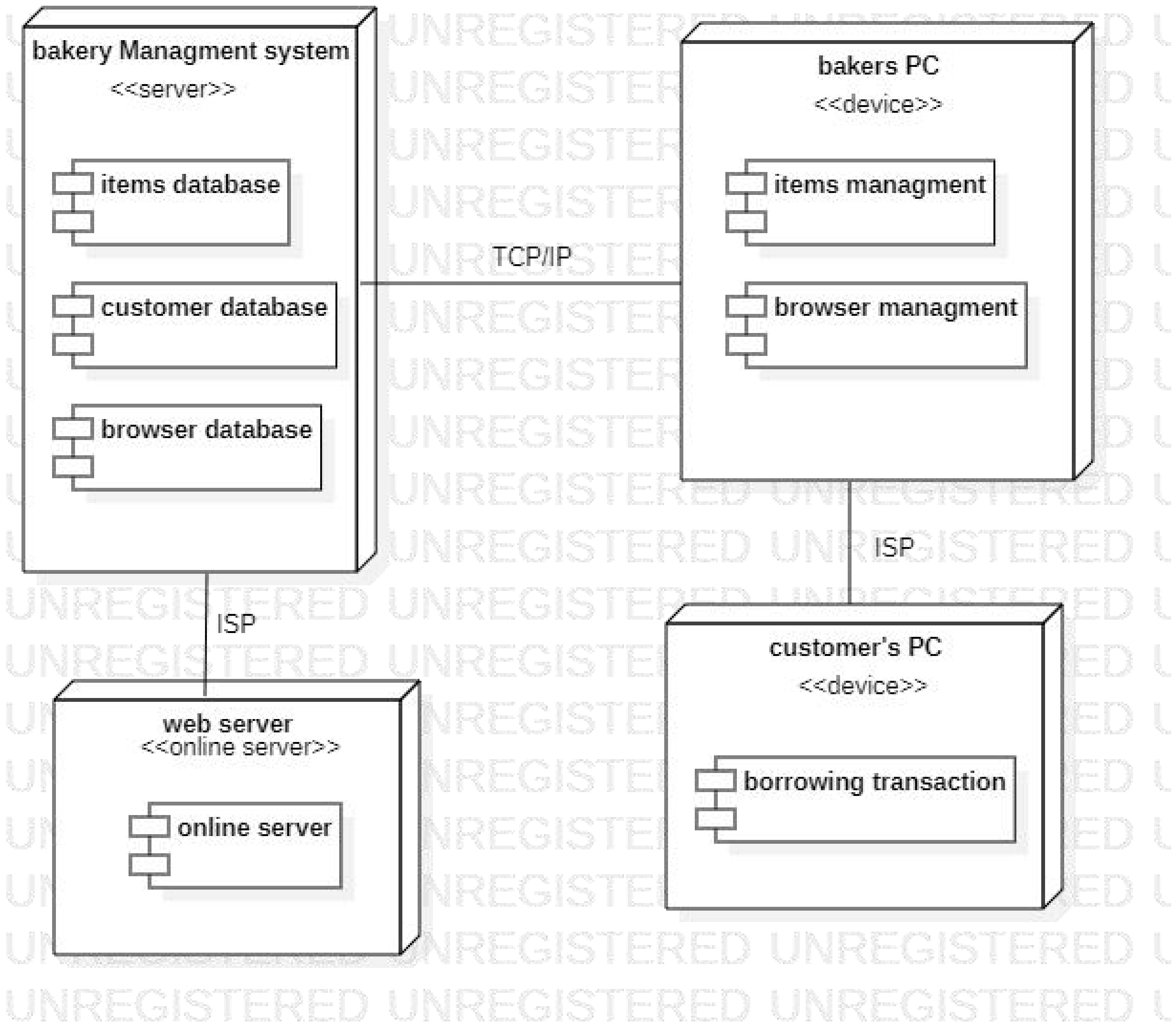


Deployment Diagram

Description of Deployment diagram:

The purpose of a Deployment diagram is to provide a visual representation of the hardware and software components that make up a system and how they are deployed in a network or infrastructure. Deployment diagram is a type of UML (Unified Modeling Language) diagram that shows the physical deployment of software and hardware components in a system or application.

Deployment Diagram of ARS:



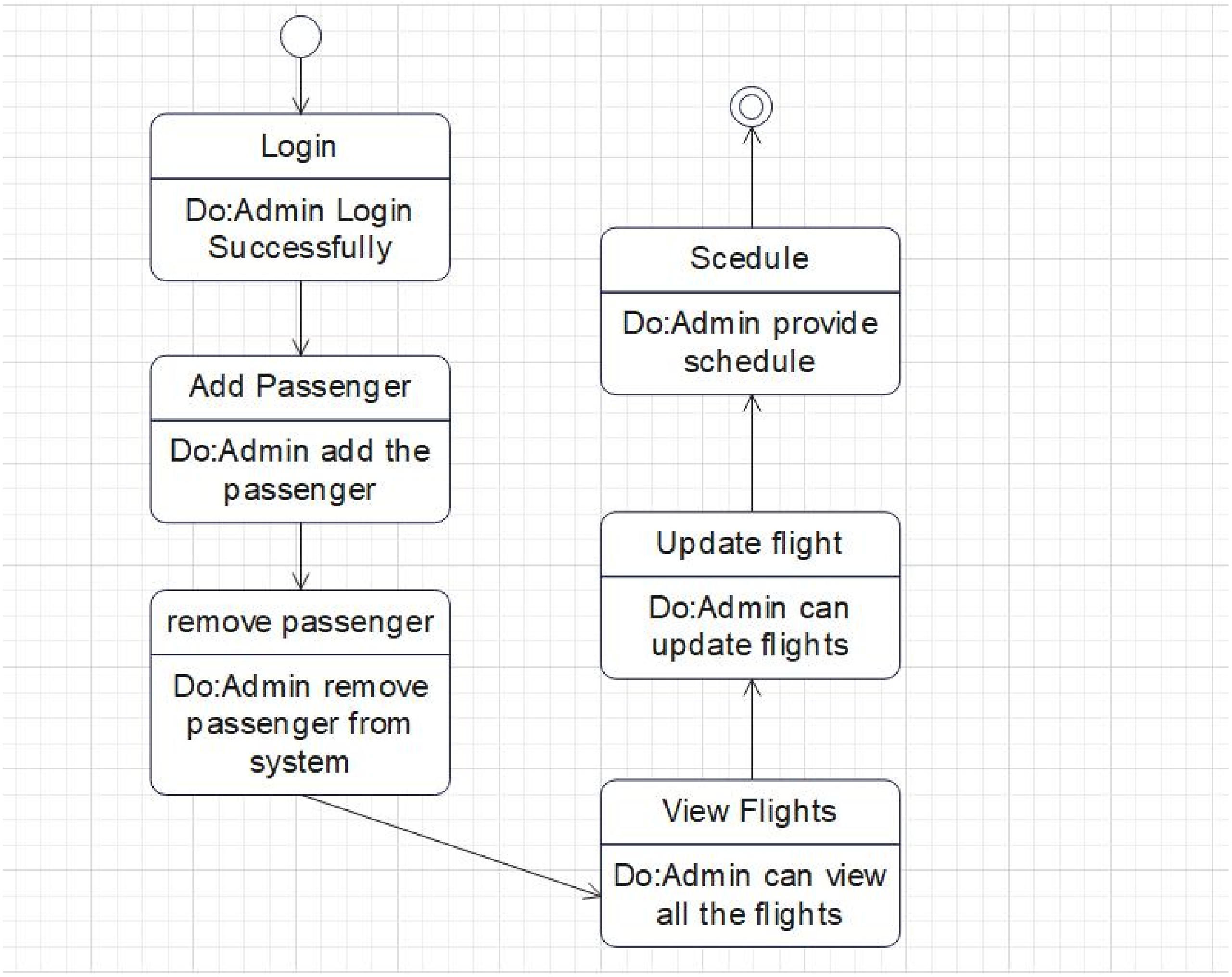
State Machine Diagram

Description of State machine diagram:

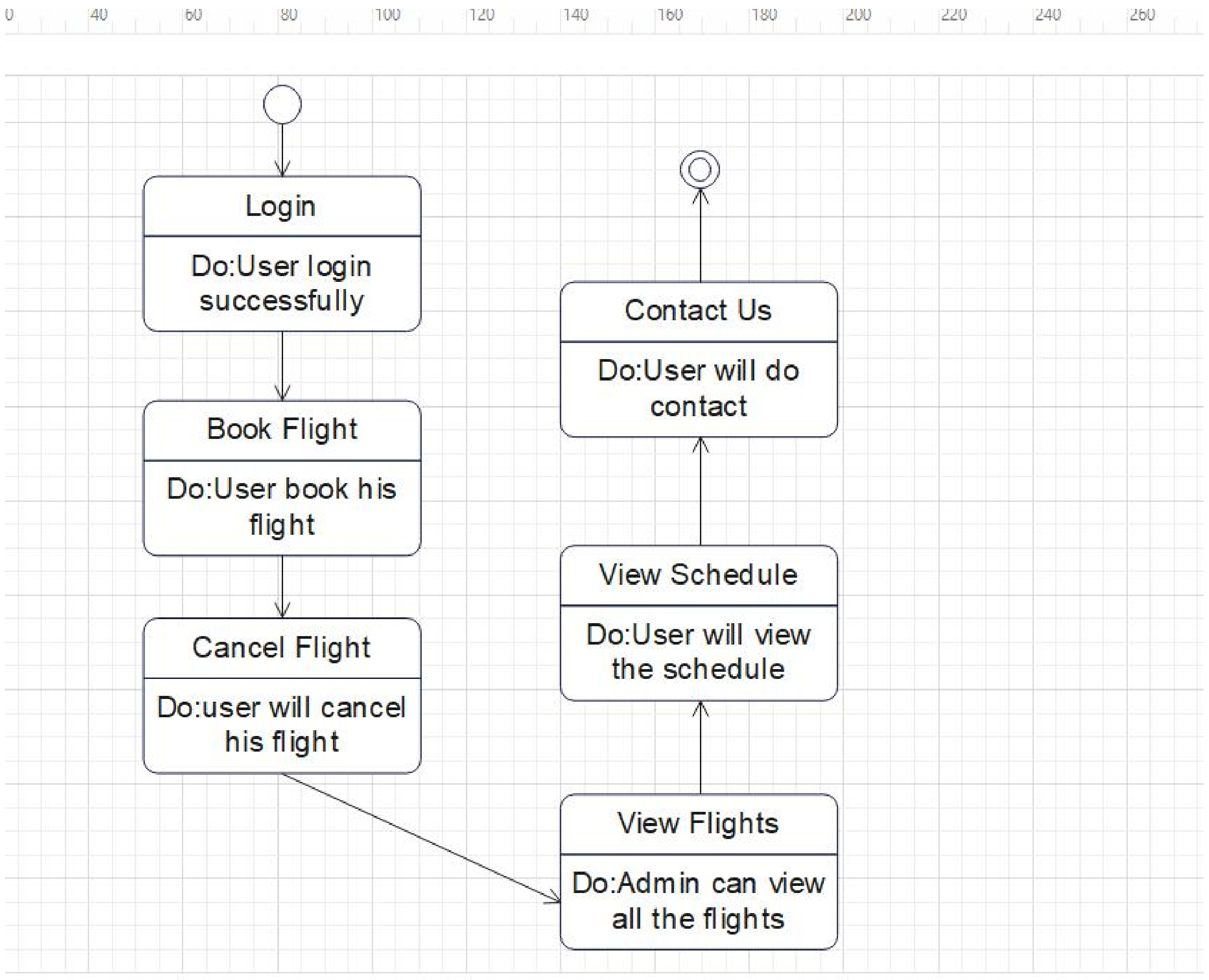
A State Machine diagram is a type of UML (Unified Modeling Language) diagram that models the behavior of a system or object over time, showing the different states that the system or object can be in and the transitions between those states. The purpose of a State Machine diagram is to provide a visual representation of how the system or object behaves in response to internal and external events.

State Machine Diagram of ARS:

1-Admin End:


Login
Do:Admin Login Successfully
Add Passenger
Do:Admin add the passenger
remove passenger
Do:Admin remove passenger from system
View Flights
Do:Admin can view all the flights
Update flight
Do:Admin can update flights
Scedule
Do:Admin provide schedule

2-User End:



Login
Do:User login successfully
Book Flight
Do:User book his flight
Cancel Flight
Do:user will cancel his flight
View Flights
Do:Admin can view all the flights
View Schedule
Do:User will view the schedule
Contact Us
Do:User will do contact

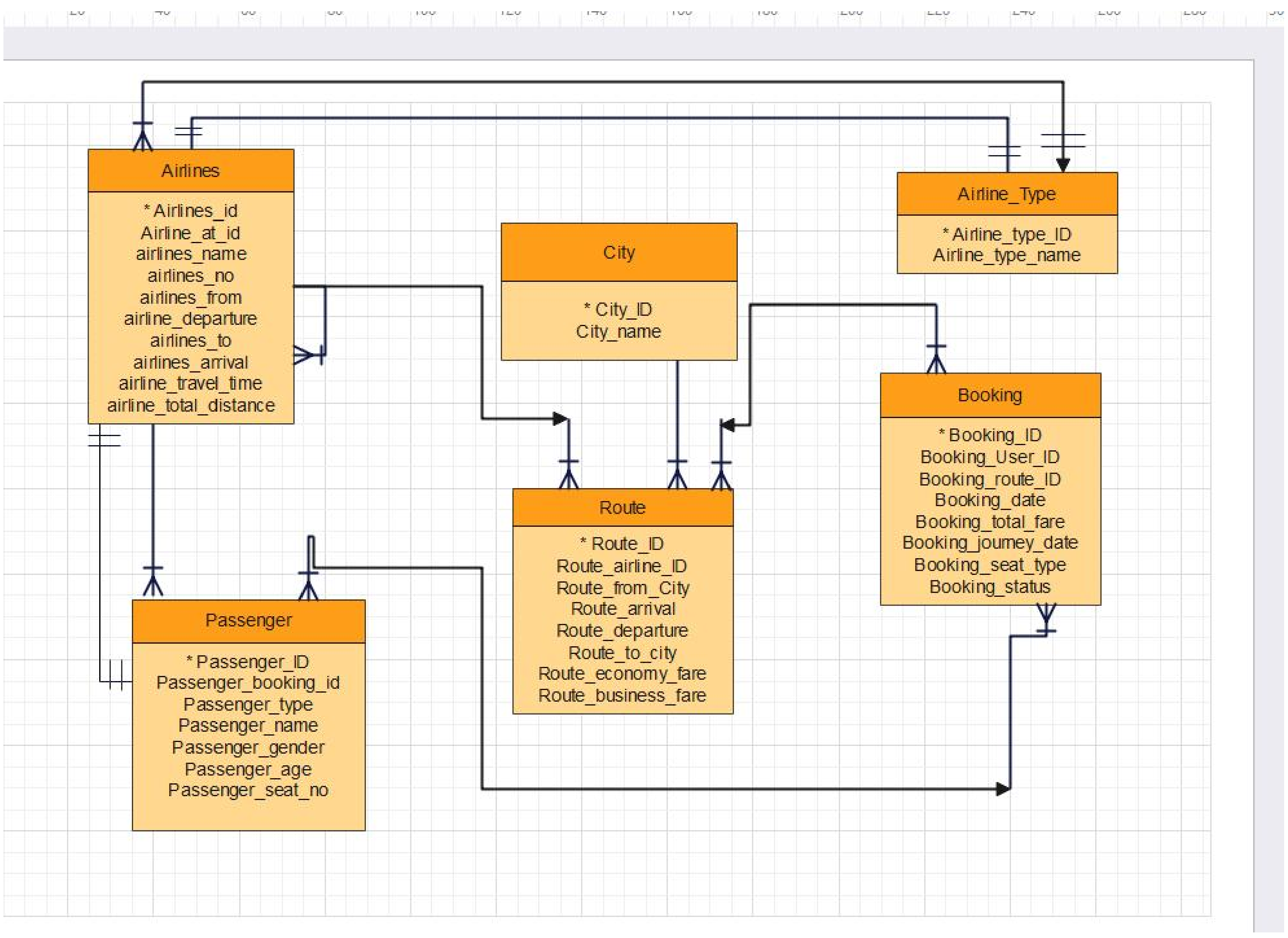
Since it is also a dynamic diagram,so it has different representations of both admin end and user end.

Entity Relation Diagram

1-Crows Foot Notation

Entity Relation Diagram:

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology (IT) system**.** Entity relation diagram of **airline reservations system** is as follows:



Explanation of Relations:

1-One Airline has only one type(**One-to-One**).

2-One Airline Type has many Airlines.For example PIA is airline type and many airlines can be of same type.(**One-to-Many**).

3-One Airline has many passengers(**One-to-Many**).

4-One passenger can be reserved at a time only in one airline(**One-to-One**).

5-One city have multiple routes(**One-to-Many**).

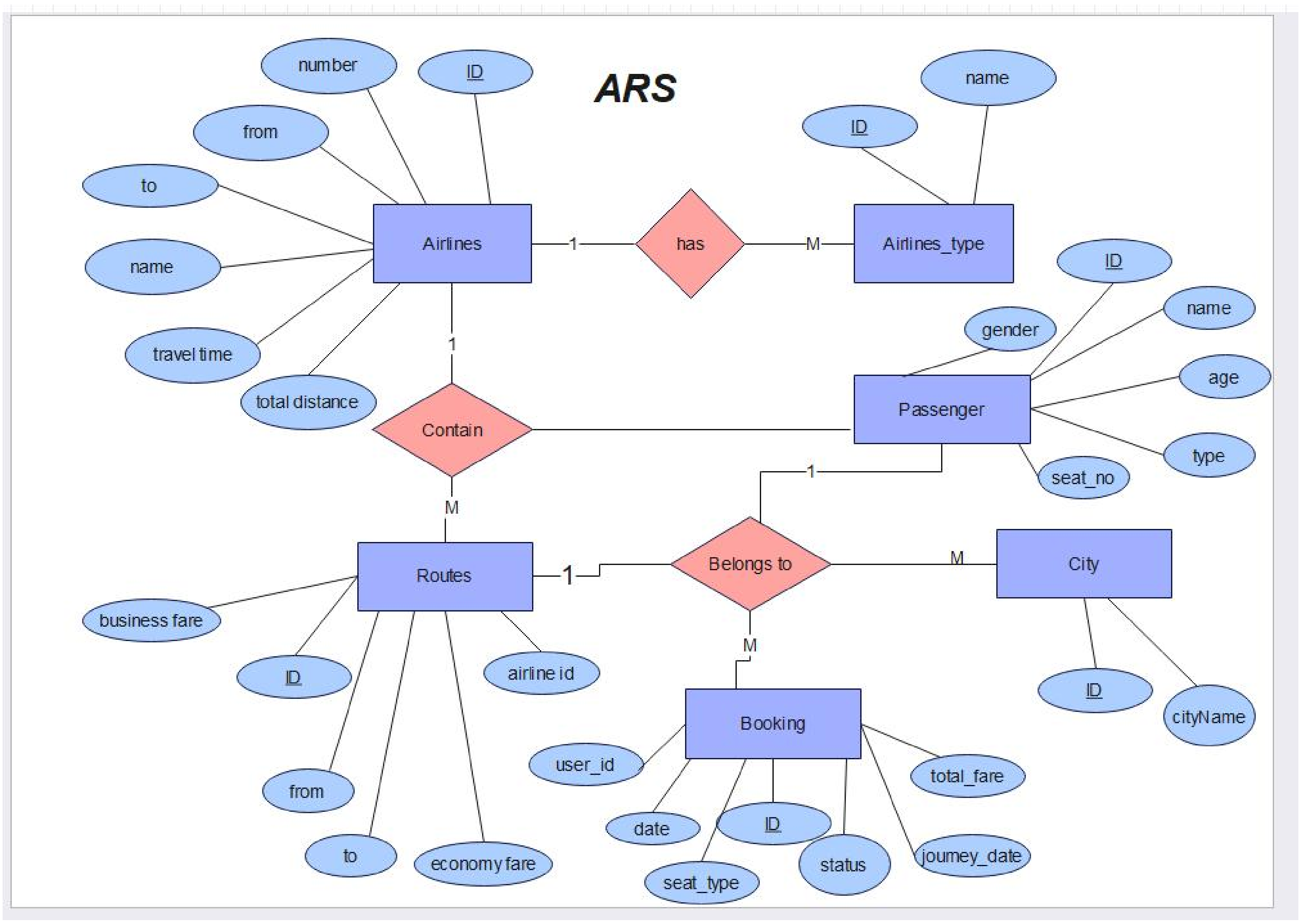
6-One airline have multiple routes and one route can have multiple airlines(**Many-to-Many**).

7-Many routes have many booking(**Many-to-Many**).

8-Many booking have many Passengers(**Many-to- Many**).

2-Chen Notation

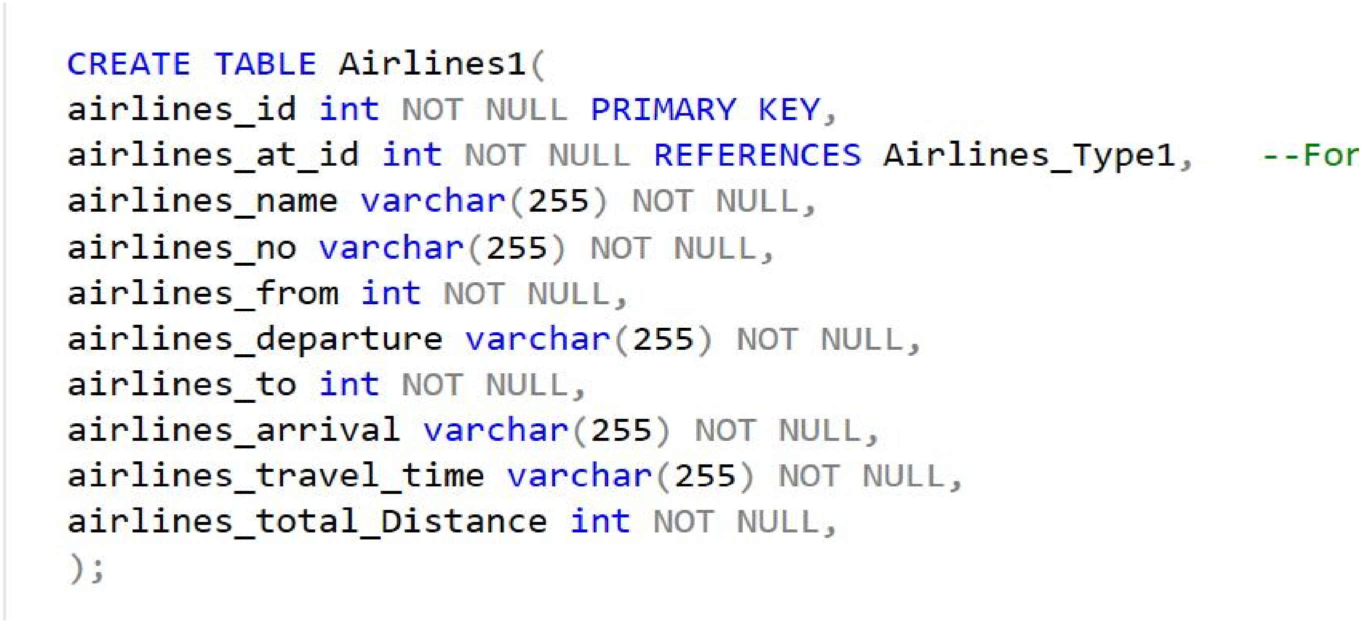
The Chen ERD notation is used and is considered to present a more detailed way of representing entities and relationships.
The Chen ERD notation for **airlines reservation system** is given below:



Normalization

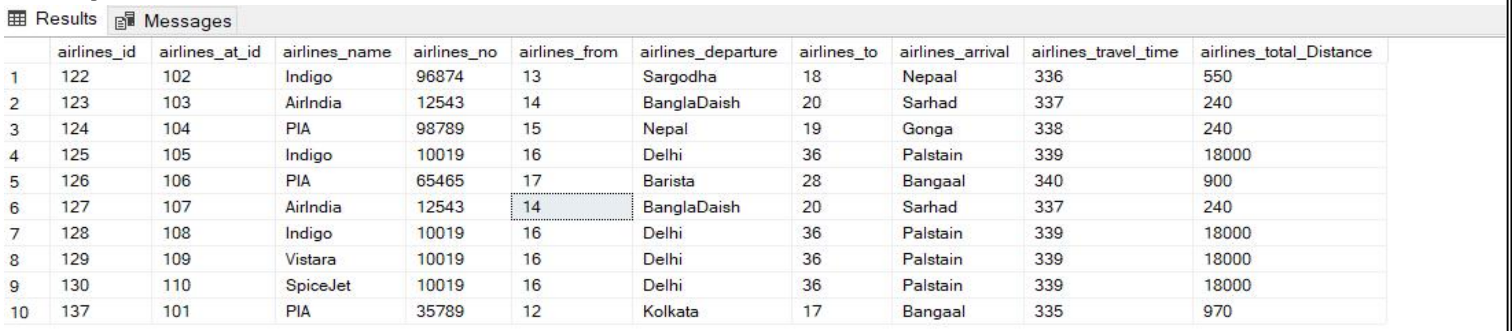
SQL Tables with their outputs

Table 01:



```sql
CREATE TABLE Airlines1(
airlines_id int NOT NULL PRIMARY KEY,
airlines_at_id int NOT NULL REFERENCES Airlines_Type1,   --For
airlines_name varchar(255) NOT NULL,
airlines_no varchar(255) NOT NULL,
airlines_from int NOT NULL,
airlines_departure varchar(255) NOT NULL,
airlines_to int NOT NULL,
airlines_arrival varchar(255) NOT NULL,
airlines_travel_time varchar(255) NOT NULL,
airlines_total_Distance int NOT NULL,
);
```

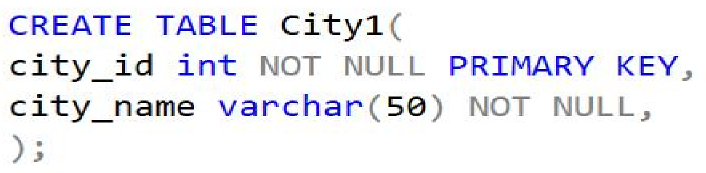
Output:



Results | Messages

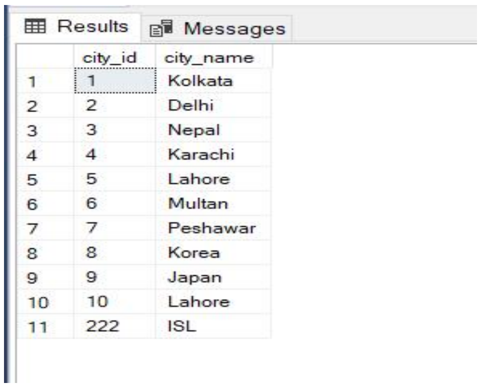
	airlines_id	airlines_at_id	airlines_name	airlines_no	airlines_from	airlines_departure	airlines_to	airlines_arrival	airlines_travel_time	airlines_total_Distance
1	122	102	Indigo	96874	13	Sargodha	18	Nepaal	336	550
2	123	103	AirIndia	12543	14	BanglaDaish	20	Sarhad	337	240
3	124	104	PIA	98789	15	Nepal	19	Gonga	338	240
4	125	105	Indigo	10019	16	Delhi	36	Palstain	339	18000
5	126	106	PIA	65465	17	Barista	28	Bangaal	340	900
6	127	107	AirIndia	12543	14	BanglaDaish	20	Sarhad	337	240
7	128	108	Indigo	10019	16	Delhi	36	Palstain	339	18000
8	129	109	Vistara	10019	16	Delhi	36	Palstain	339	18000
9	130	110	SpiceJet	10019	16	Delhi	36	Palstain	339	18000
10	137	101	PIA	35789	12	Kolkata	17	Bangaal	335	970

Table 02:



```sql
CREATE TABLE City1(
city_id int NOT NULL PRIMARY KEY,
city_name varchar(50) NOT NULL,
);
```

Output:



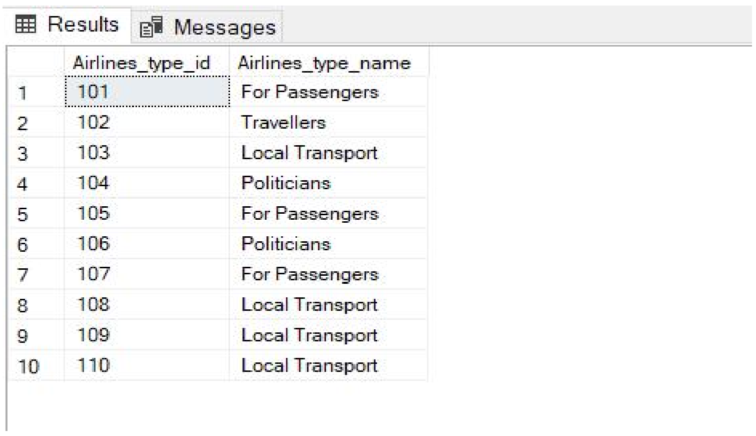
Results | Messages

	city_id	city_name
1	1	Kolkata
2	2	Delhi
3	3	Nepal
4	4	Karachi
5	5	Lahore
6	6	Multan
7	7	Peshawar
8	8	Korea
9	9	Japan
10	10	Lahore
11	222	ISL

Table 03:

```sql
CREATE TABLE Airlines_Type1(
Airlines_type_id int NOT NULL PRIMARY KEY,
Airlines_type_name varchar(255) NOT NULL,
);
```

Ouput:



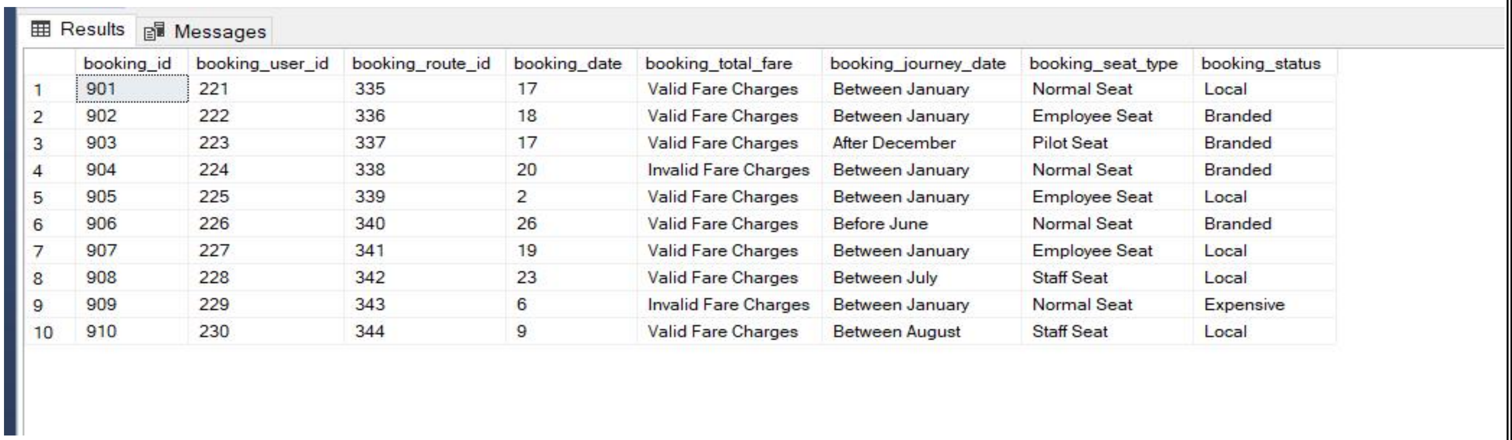
Results | Messages

	Airlines_type_id	Airlines_type_name
1	101	For Passengers
2	102	Travellers
3	103	Local Transport
4	104	Politicians
5	105	For Passengers
6	106	Politicians
7	107	For Passengers
8	108	Local Transport
9	109	Local Transport
10	110	Local Transport

Table 04:

```sql
CREATE TABLE Booking1(
booking_id int NOT NULL PRIMARY KEY,
booking_user_id int NOT NULL REFERENCES Passenger1,
booking_route_id int NOT NULL REFERENCES Route1,
booking_date varchar(255) NOT NULL,
booking_total_fare varchar(255) NOT NULL,
booking_journey_date varchar(255) NOT NULL,
booking_seat_type varchar(255) NOT NULL,
booking_status varchar(255) NOT NULL,
);
```

Output:



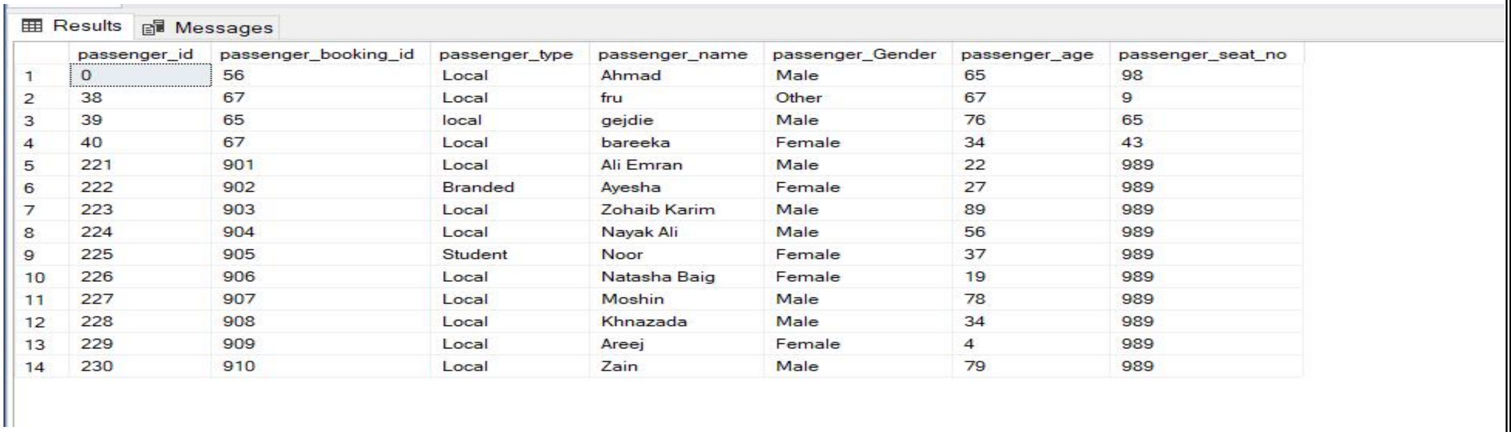
Results | Messages

	booking_id	booking_user_id	booking_route_id	booking_date	booking_total_fare	booking_journey_date	booking_seat_type	booking_status
1	901	221	335	17	Valid Fare Charges	Between January	Normal Seat	Local
2	902	222	336	18	Valid Fare Charges	Between January	Employee Seat	Branded
3	903	223	337	17	Valid Fare Charges	After December	Pilot Seat	Branded
4	904	224	338	20	Invalid Fare Charges	Between January	Normal Seat	Branded
5	905	225	339	2	Valid Fare Charges	Between January	Employee Seat	Local
6	906	226	340	26	Valid Fare Charges	Before June	Normal Seat	Branded
7	907	227	341	19	Valid Fare Charges	Between January	Employee Seat	Local
8	908	228	342	23	Valid Fare Charges	Between July	Staff Seat	Local
9	909	229	343	6	Invalid Fare Charges	Between January	Normal Seat	Expensive
10	910	230	344	9	Valid Fare Charges	Between August	Staff Seat	Local

Table 05:

```
CREATE TABLE Passenger1(
passenger_id int NOT NULL PRIMARY KEY,
passenger_booking_id int NOT NULL,
passenger_type varchar(255) NOT NULL,
passenger_name varchar(255) NOT NULL,
passenger_Gender varchar(255) NOT NULL,
passenger_age varchar(255) NOT NULL,
passenger_seat_no int NOT NULL
);
```

Output:

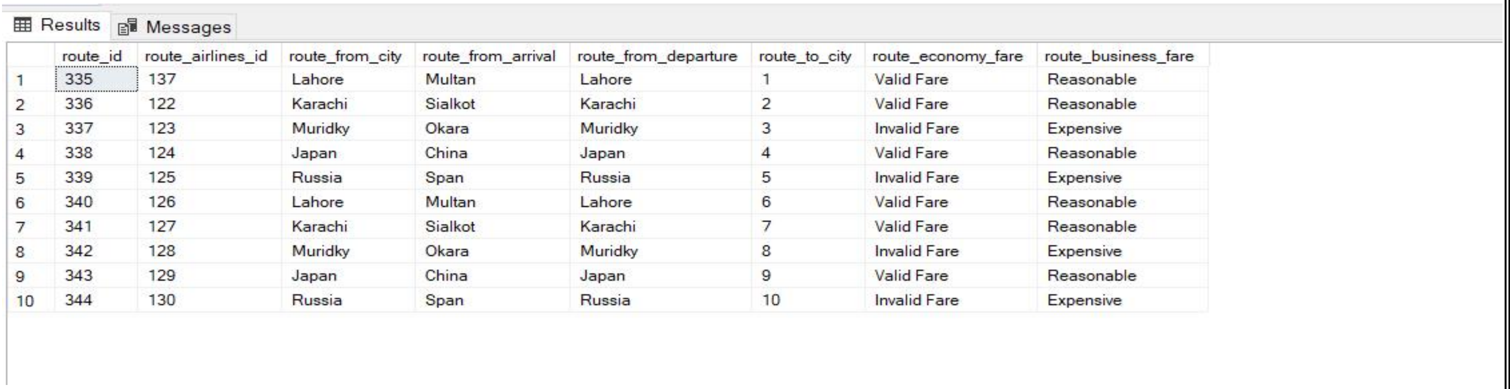


	passenger_id	passenger_booking_id	passenger_type	passenger_name	passenger_Gender	passenger_age	passenger_seat_no
1	0	56	Local	Ahmad	Male	65	98
2	38	67	Local	fru	Other	67	9
3	39	65	local	gejdie	Male	76	65
4	40	67	Local	bareeka	Female	34	43
5	221	901	Local	Ali Emran	Male	22	989
6	222	902	Branded	Ayesha	Female	27	989
7	223	903	Local	Zohaib Karim	Male	89	989
8	224	904	Local	Nayak Ali	Male	56	989
9	225	905	Student	Noor	Female	37	989
10	226	906	Local	Natasha Baig	Female	19	989
11	227	907	Local	Moshin	Male	78	989
12	228	908	Local	Khnazada	Male	34	989
13	229	909	Local	Areej	Female	4	989
14	230	910	Local	Zain	Male	79	989

Table 06:

```
CREATE TABLE Route1(
route_id int NOT NULL PRIMARY KEY,
route_airlines_id int NOT NULL REFERENCES Airlines1,
route_from_city varchar(255) NOT NULL,
route_from_arrival varchar(255) NOT NULL,
route_from_departure varchar(255) NOT NULL,
route_to_city int NOT NULL REFERENCES City1,
route_economy_fare varchar(255) NOT NULL,
route_business_fare varchar(255) NOT NULL,
);
```

Output:

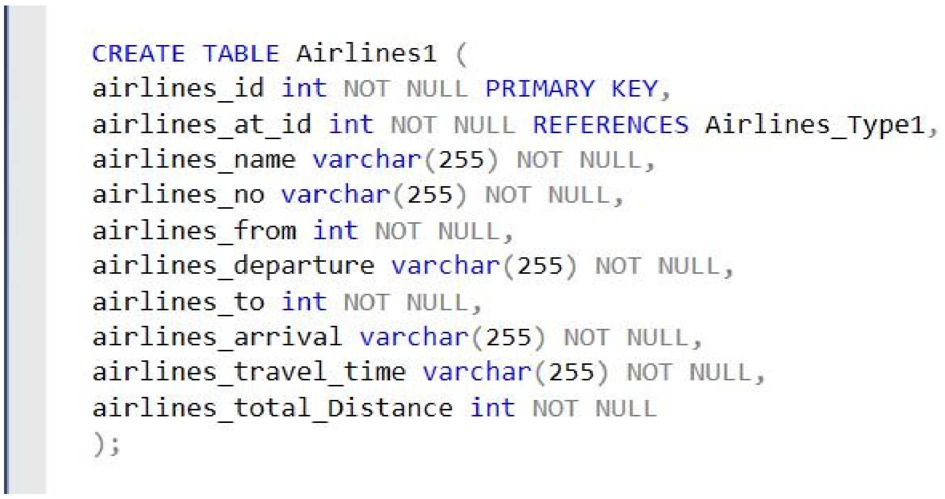


	route_id	route_airlines_id	route_from_city	route_from_arrival	route_from_departure	route_to_city	route_economy_fare	route_business_fare
1	335	137	Lahore	Multan	Lahore	1	Valid Fare	Reasonable
2	336	122	Karachi	Sialkot	Karachi	2	Valid Fare	Reasonable
3	337	123	Muridky	Okara	Muridky	3	Invalid Fare	Expensive
4	338	124	Japan	China	Japan	4	Valid Fare	Reasonable
5	339	125	Russia	Span	Russia	5	Invalid Fare	Expensive
6	340	126	Lahore	Multan	Lahore	6	Valid Fare	Reasonable
7	341	127	Karachi	Sialkot	Karachi	7	Valid Fare	Reasonable
8	342	128	Muridky	Okara	Muridky	8	Invalid Fare	Expensive
9	343	129	Japan	China	Japan	9	Valid Fare	Reasonable
10	344	130	Russia	Span	Russia	10	Invalid Fare	Expensive

First Normalization Form(1NF)

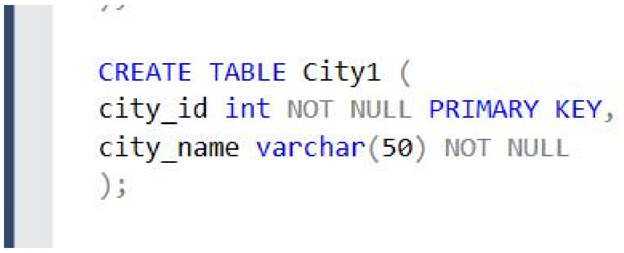
To convert to 1NF, we need to ensure that each column in the table contains only atomic values, and there are no repeating groups or arrays.

Table 01:



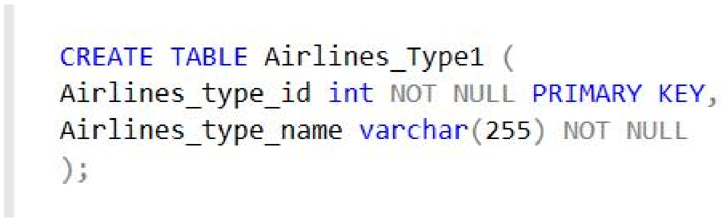
```sql
CREATE TABLE Airlines1 (
airlines_id int NOT NULL PRIMARY KEY,
airlines_at_id int NOT NULL REFERENCES Airlines_Type1,
airlines_name varchar(255) NOT NULL,
airlines_no varchar(255) NOT NULL,
airlines_from int NOT NULL,
airlines_departure varchar(255) NOT NULL,
airlines_to int NOT NULL,
airlines_arrival varchar(255) NOT NULL,
airlines_travel_time varchar(255) NOT NULL,
airlines_total_Distance int NOT NULL
);
```

Table 02:



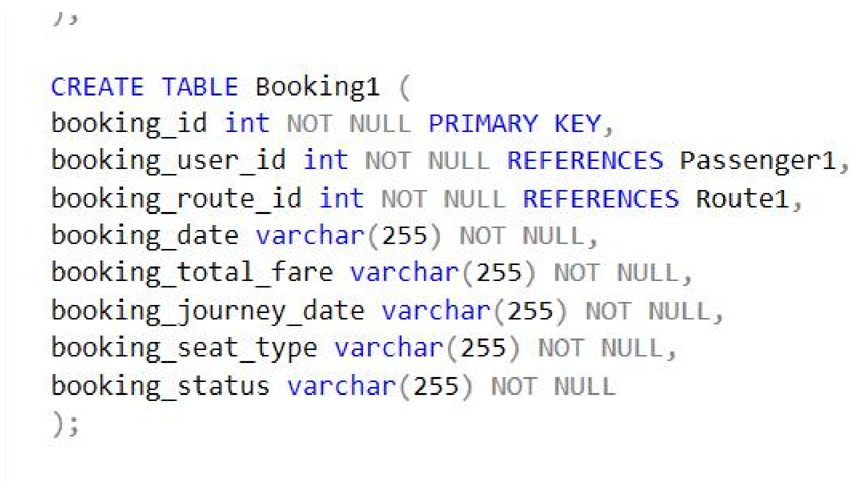
```sql
CREATE TABLE City1 (
city_id int NOT NULL PRIMARY KEY,
city_name varchar(50) NOT NULL
);
```

Table 03:



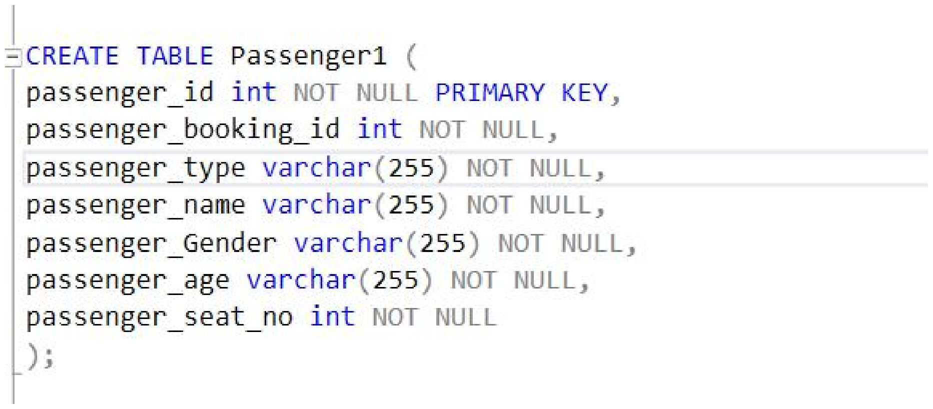
```sql
CREATE TABLE Airlines_Type1 (
Airlines_type_id int NOT NULL PRIMARY KEY,
Airlines_type_name varchar(255) NOT NULL
);
```

Table 04:



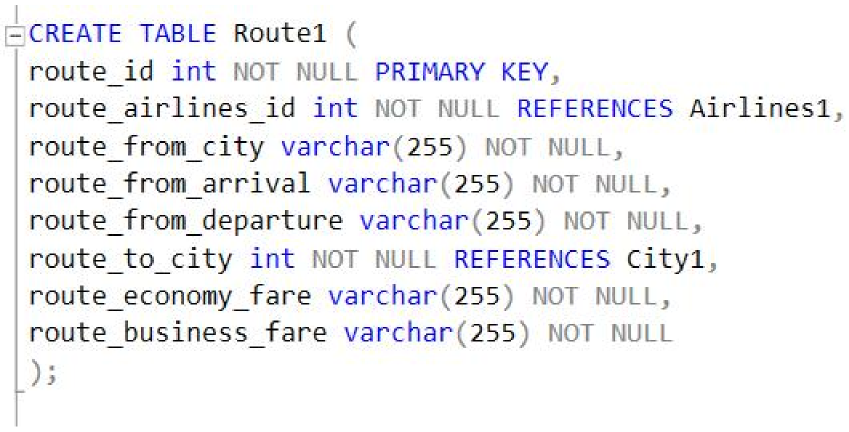
```sql
CREATE TABLE Booking1 (
booking_id int NOT NULL PRIMARY KEY,
booking_user_id int NOT NULL REFERENCES Passenger1,
booking_route_id int NOT NULL REFERENCES Route1,
booking_date varchar(255) NOT NULL,
booking_total_fare varchar(255) NOT NULL,
booking_journey_date varchar(255) NOT NULL,
booking_seat_type varchar(255) NOT NULL,
booking_status varchar(255) NOT NULL
);
```

Table 05:



```sql
CREATE TABLE Passenger1 (
passenger_id int NOT NULL PRIMARY KEY,
passenger_booking_id int NOT NULL,
passenger_type varchar(255) NOT NULL,
passenger_name varchar(255) NOT NULL,
passenger_Gender varchar(255) NOT NULL,
passenger_age varchar(255) NOT NULL,
passenger_seat_no int NOT NULL
);
```

Table 06:

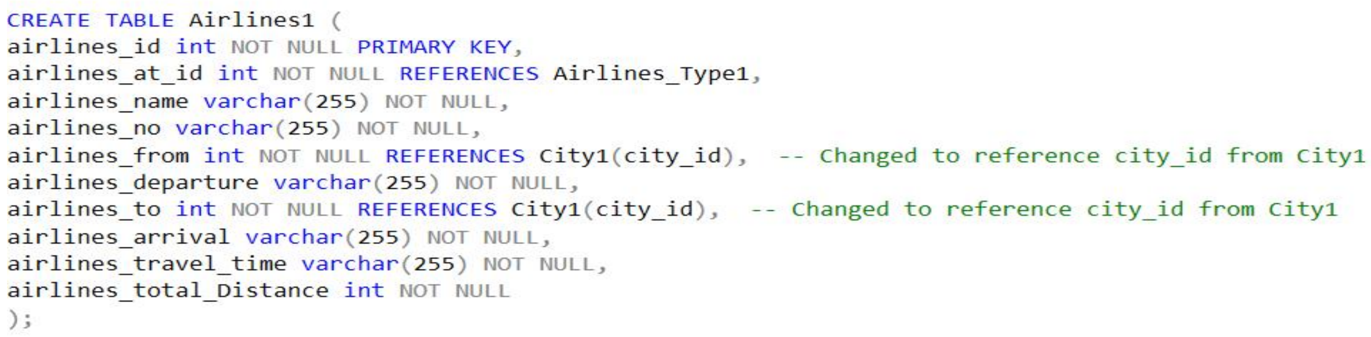


```sql
CREATE TABLE Route1 (
route_id int NOT NULL PRIMARY KEY,
route_airlines_id int NOT NULL REFERENCES Airlines1,
route_from_city varchar(255) NOT NULL,
route_from_arrival varchar(255) NOT NULL,
route_from_departure varchar(255) NOT NULL,
route_to_city int NOT NULL REFERENCES City1,
route_economy_fare varchar(255) NOT NULL,
route_business_fare varchar(255) NOT NULL
);
```

All the above tables are in First Normal Form because each column has atomic values.
There are no multi valued attributes.

Second Normalization Form(2NF)

Table 01:



```
CREATE TABLE Airlines1 (
airlines_id int NOT NULL PRIMARY KEY,
airlines_at_id int NOT NULL REFERENCES Airlines_Type1,
airlines_name varchar(255) NOT NULL,
airlines_no varchar(255) NOT NULL,
airlines_from int NOT NULL REFERENCES City1(city_id),  -- Changed to reference city_id from City1
airlines_departure varchar(255) NOT NULL,
airlines_to int NOT NULL REFERENCES City1(city_id),  -- Changed to reference city_id from City1
airlines_arrival varchar(255) NOT NULL,
airlines_travel_time varchar(255) NOT NULL,
airlines_total_Distance int NOT NULL
);
```

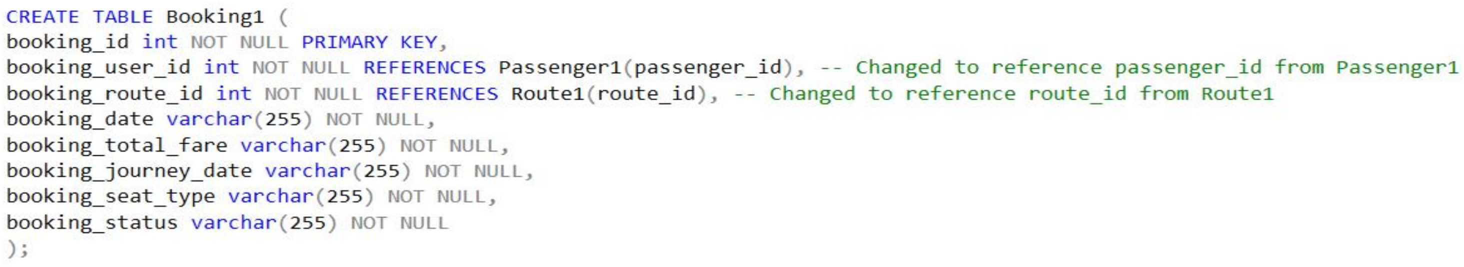
Table 02:

It is already in 2NF.

Table 03:

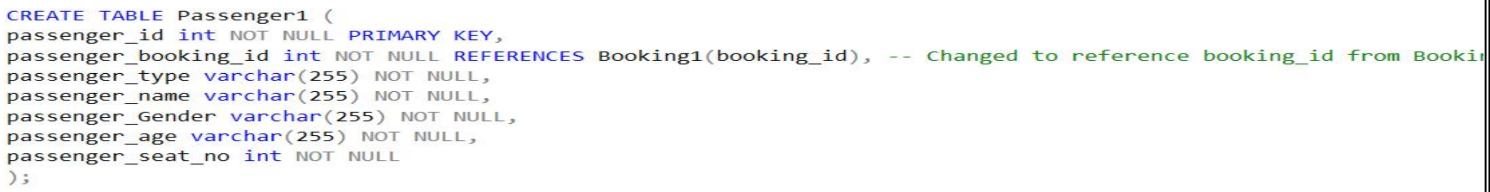
It is already in 2NF.

Table 04:



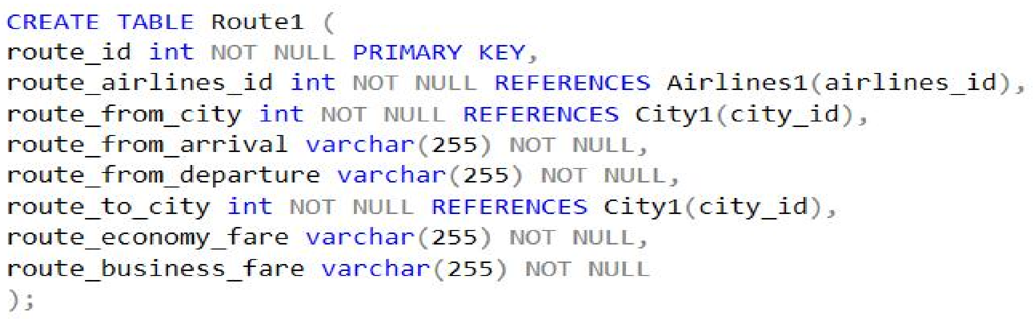
```
CREATE TABLE Booking1 (
booking_id int NOT NULL PRIMARY KEY,
booking_user_id int NOT NULL REFERENCES Passenger1(passenger_id), -- Changed to reference passenger_id from Passenger1
booking_route_id int NOT NULL REFERENCES Route1(route_id), -- Changed to reference route_id from Route1
booking_date varchar(255) NOT NULL,
booking_total_fare varchar(255) NOT NULL,
booking_journey_date varchar(255) NOT NULL,
booking_seat_type varchar(255) NOT NULL,
booking_status varchar(255) NOT NULL
);
```

Table 05:



```
CREATE TABLE Passenger1 (
passenger_id int NOT NULL PRIMARY KEY,
passenger_booking_id int NOT NULL REFERENCES Booking1(booking_id), -- Changed to reference booking_id from Booki
passenger_type varchar(255) NOT NULL,
passenger_name varchar(255) NOT NULL,
passenger_Gender varchar(255) NOT NULL,
passenger_age varchar(255) NOT NULL,
passenger_seat_no int NOT NULL
);
```

Table 06:



```
CREATE TABLE Route1 (
route_id int NOT NULL PRIMARY KEY,
route_airlines_id int NOT NULL REFERENCES Airlines1(airlines_id),
route_from_city int NOT NULL REFERENCES City1(city_id),
route_from_arrival varchar(255) NOT NULL,
route_from_departure varchar(255) NOT NULL,
route_to_city int NOT NULL REFERENCES City1(city_id),
route_economy_fare varchar(255) NOT NULL,
route_business_fare varchar(255) NOT NULL
);
```

Third Normalization Form(3NF)

Table 01:

```
CREATE TABLE Airlines1 (
airlines_id int NOT NULL PRIMARY KEY,
airlines_at_id int NOT NULL REFERENCES Airlines_Type1(Airlines_type_id),
airlines_name varchar(255) NOT NULL,
airlines_no varchar(255) NOT NULL,
airlines_departure varchar(255) NOT NULL,
airlines_arrival varchar(255) NOT NULL,
airlines_travel_time varchar(255) NOT NULL,
airlines_total_Distance int NOT NULL
);
```

Table 02:

```
CREATE TABLE City1 (
city_id int NOT NULL PRIMARY KEY,
city_name varchar(50) NOT NULL
);
```

Table 03:

```
CREATE TABLE Airlines_Type1 (
Airlines_type_id int NOT NULL PRIMARY KEY,
Airlines_type_name varchar(255) NOT NULL
);
```

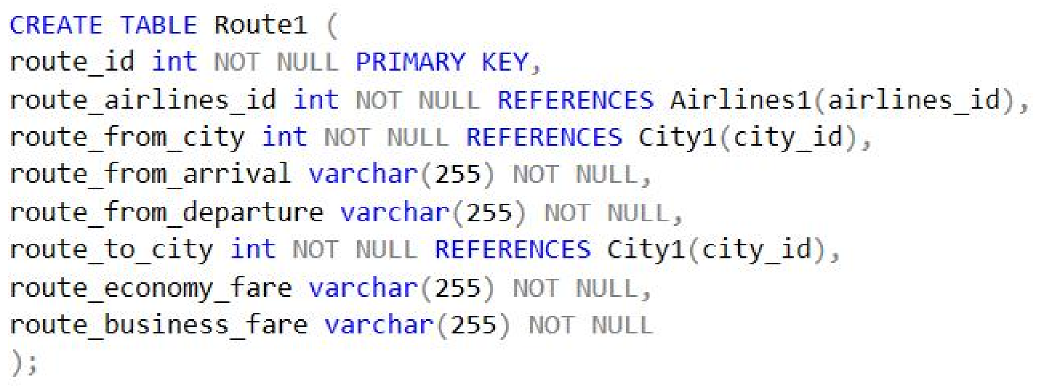
Table 04:

```
CREATE TABLE Booking1 (
booking_id int NOT NULL PRIMARY KEY,
booking_user_id int NOT NULL REFERENCES Passenger1(passenger_id),
booking_route_id int NOT NULL REFERENCES Route1(route_id),
booking_date varchar(255) NOT NULL,
booking_total_fare varchar(255) NOT NULL,
booking_journey_date varchar(255) NOT NULL,
booking_seat_type varchar(255) NOT NULL,
booking_status varchar(255) NOT NULL
);
```

Table 05:

```
CREATE TABLE Passenger1 (
passenger_id int NOT NULL PRIMARY KEY,
passenger_booking_id int NOT NULL REFERENCES Booking1(booking_id),
passenger_type varchar(255) NOT NULL,
passenger_name varchar(255) NOT NULL,
passenger_Gender varchar(255) NOT NULL,
passenger_age varchar(255) NOT NULL,
passenger_seat_no int NOT NULL
);
```

Table 06:

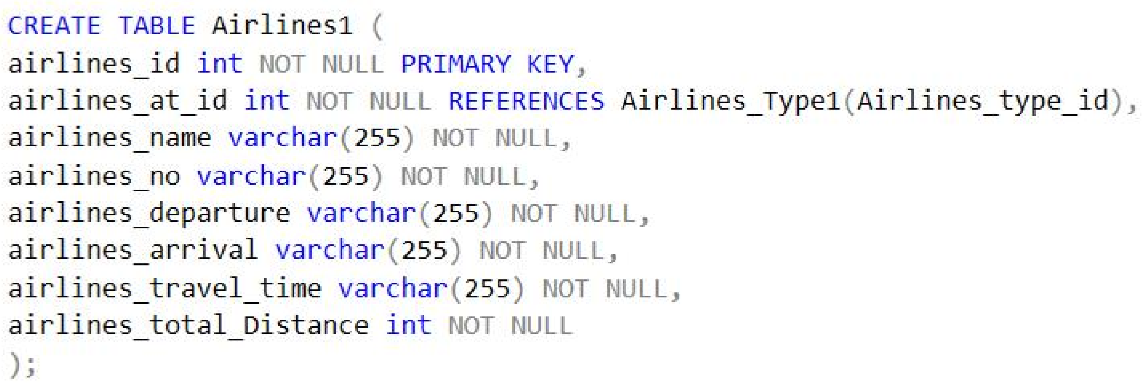


```
CREATE TABLE Route1 (
route_id int NOT NULL PRIMARY KEY,
route_airlines_id int NOT NULL REFERENCES Airlines1(airlines_id),
route_from_city int NOT NULL REFERENCES City1(city_id),
route_from_arrival varchar(255) NOT NULL,
route_from_departure varchar(255) NOT NULL,
route_to_city int NOT NULL REFERENCES City1(city_id),
route_economy_fare varchar(255) NOT NULL,
route_business_fare varchar(255) NOT NULL
);
```

In the above tables, there are no transitive dependencies, and each table has a single primary key that uniquely identifies each row, conforming to 3NF.

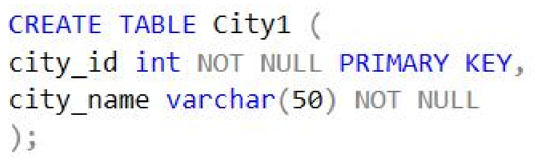
Boyce Code Normalization Form(BCNF)

Table 01:



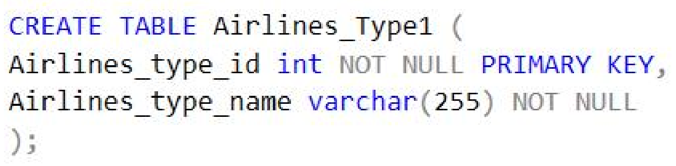
```
CREATE TABLE Airlines1 (
airlines_id int NOT NULL PRIMARY KEY,
airlines_at_id int NOT NULL REFERENCES Airlines_Type1(Airlines_type_id),
airlines_name varchar(255) NOT NULL,
airlines_no varchar(255) NOT NULL,
airlines_departure varchar(255) NOT NULL,
airlines_arrival varchar(255) NOT NULL,
airlines_travel_time varchar(255) NOT NULL,
airlines_total_Distance int NOT NULL
);
```

Table 02:



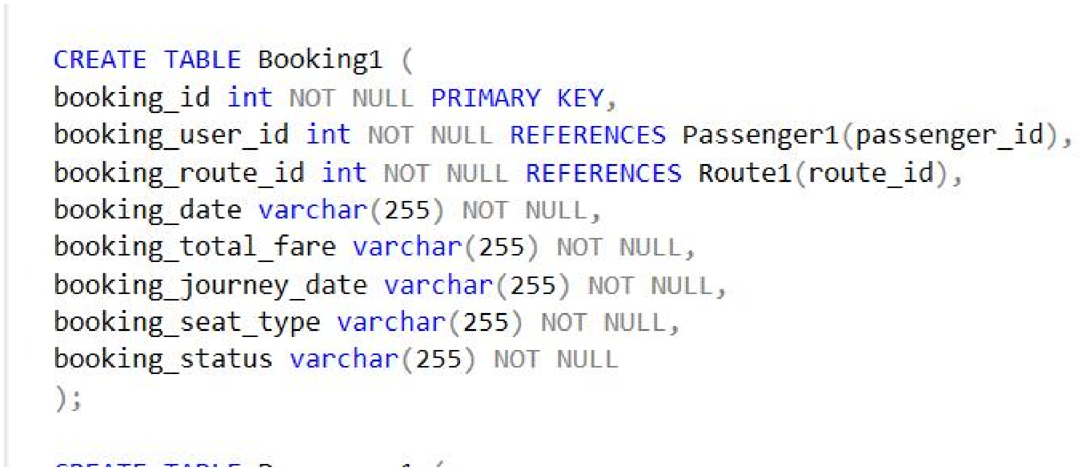
```
CREATE TABLE City1 (
city_id int NOT NULL PRIMARY KEY,
city_name varchar(50) NOT NULL
);
```

Table 03:



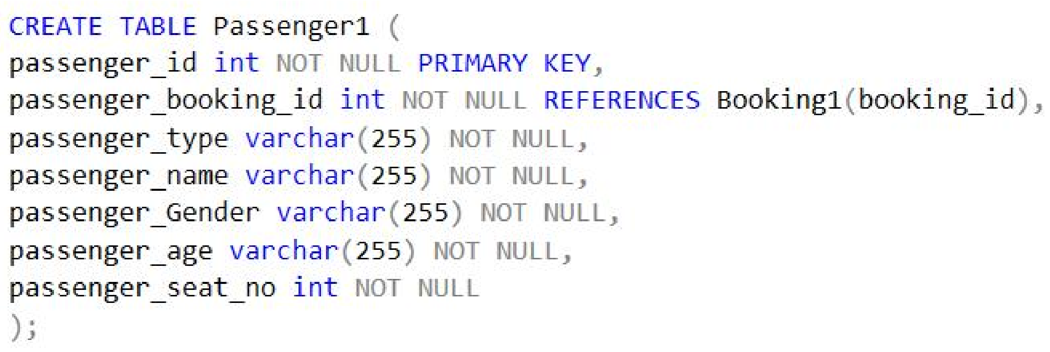
```
CREATE TABLE Airlines_Type1 (
Airlines_type_id int NOT NULL PRIMARY KEY,
Airlines_type_name varchar(255) NOT NULL
);
```

Table 04:



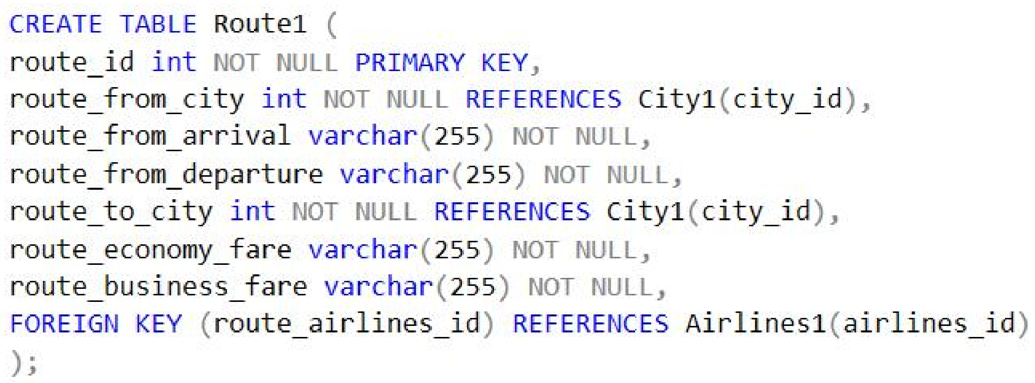
```
CREATE TABLE Booking1 (
booking_id int NOT NULL PRIMARY KEY,
booking_user_id int NOT NULL REFERENCES Passenger1(passenger_id),
booking_route_id int NOT NULL REFERENCES Route1(route_id),
booking_date varchar(255) NOT NULL,
booking_total_fare varchar(255) NOT NULL,
booking_journey_date varchar(255) NOT NULL,
booking_seat_type varchar(255) NOT NULL,
booking_status varchar(255) NOT NULL
);
```

Table 05:



```
CREATE TABLE Passenger1 (
passenger_id int NOT NULL PRIMARY KEY,
passenger_booking_id int NOT NULL REFERENCES Booking1(booking_id),
passenger_type varchar(255) NOT NULL,
passenger_name varchar(255) NOT NULL,
passenger_Gender varchar(255) NOT NULL,
passenger_age varchar(255) NOT NULL,
passenger_seat_no int NOT NULL
);
```

Table 06:



```
CREATE TABLE Route1 (
route_id int NOT NULL PRIMARY KEY,
route_from_city int NOT NULL REFERENCES City1(city_id),
route_from_arrival varchar(255) NOT NULL,
route_from_departure varchar(255) NOT NULL,
route_to_city int NOT NULL REFERENCES City1(city_id),
route_economy_fare varchar(255) NOT NULL,
route_business_fare varchar(255) NOT NULL,
FOREIGN KEY (route_airlines_id) REFERENCES Airlines1(airlines_id)
);
```

In the above tables, all the functional dependencies are satisfied, and each table has a single primary key that uniquely identifies each row, conforming to BCNF.

ThankYou !