FLIGHT BOOKING SYSTEM

POC

Low Level Design (LLD)

Date: 15/08/2022

Current Document Version: [*1.0*]

DOCUMENT APPROVAL

Approvers of this document

Name	Department	Role	Signature	Date

DOCUMENT CHANGE HISTORY

Document Version #	Author	Date	Description
1.0	Gandavarapu Sai Shashank	24/07/2022	Flight Booking System LLD

Contents

1.0 Document Purpose

This document describes the solution architecture for Flight Booking System.

2.0 Intended Audience

This document is intended as a reference for the following roles and stakeholders who are interested in the Flight Booking System technical architecture.

Role	Nature of Engagement in WB Classics Portal Technical Architecture
Product Owners/SME	Key stakeholder to ensure that the architecture is aligned with business goals.
Business Analysts	Business analysts are one of the stakeholders who are informed with the key architectural decisions.
Enterprise Architects	Platform Architecture is aligned to business goals and architecture, architectural guidelines.
Solution Architects	To ensure solution design and architecture is aligned to business requirements, architectural guidelines.
Developers	Use Technical Architecture Document as the guiding document for detail design and implantation approach to align with Flight Booking System microservice.
End User	An end user can search flights, check flight fares, book and cancel flights.

3.0 Project Background, Objective(s)

3.1 Project Background

Flight Booking System leads to perform flight bookings where one can register themselves and perform various operations.

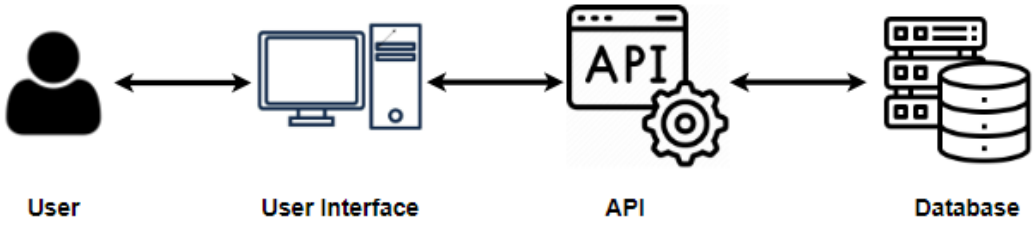
3.2 Project Objective

Flight Booking System will perform various operations like,
User Option - Home, Login, Search Flight, Book Flight, Cancel Booking, View Booking, Change Password, Logout.
Admin Option - Login, Check In, Add Flights data, Edit Flights data, delete Flights data, Add, Edit Customer data, Cancel Reservation.

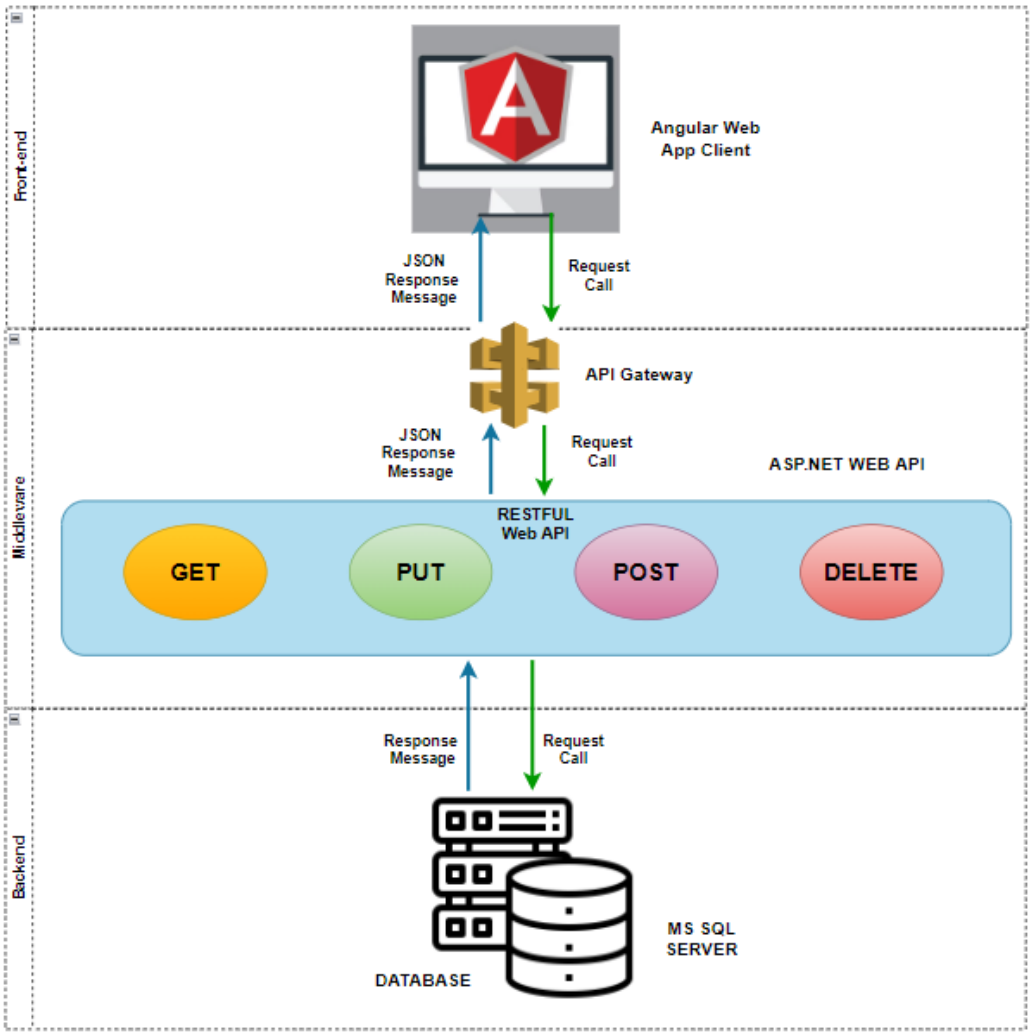
4.0 Design Pattern

#	Name	Description
1	ANGULAR	Angular is used for the frontend and designing of the application.
2	API	API is used as connection between frontend and database.
3	SQL SERVER	SQL server is used to store data to the database

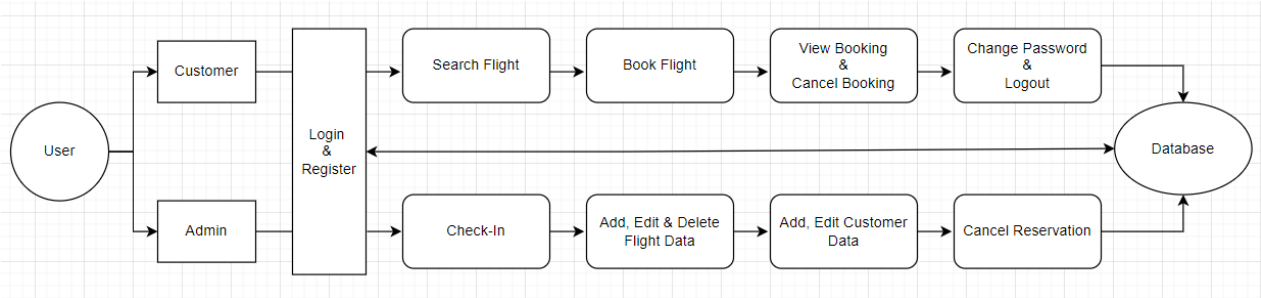
5.0 Solution Diagram



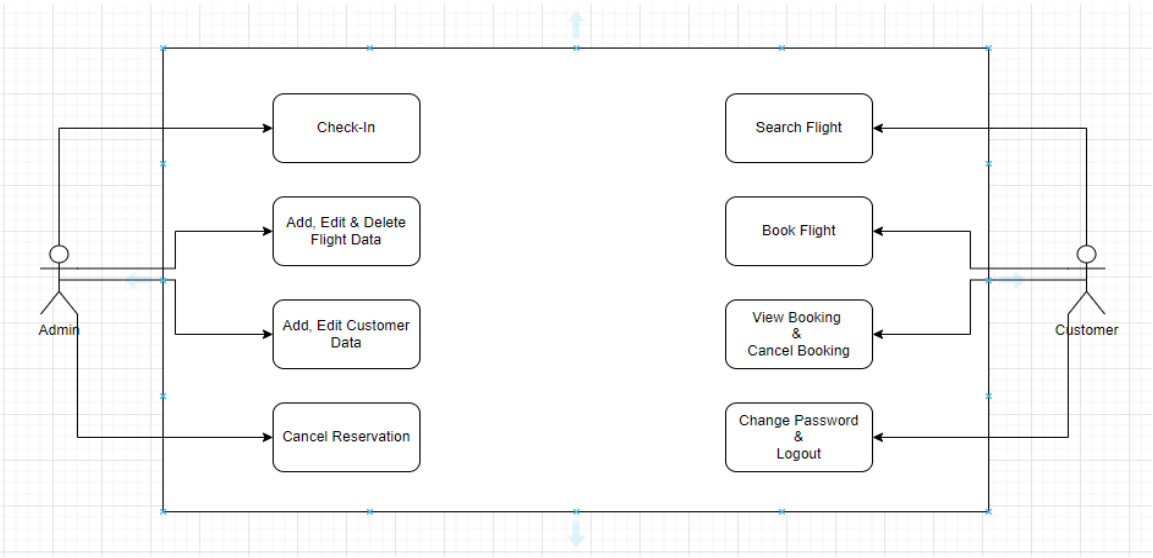
6.0 Architecture Diagram



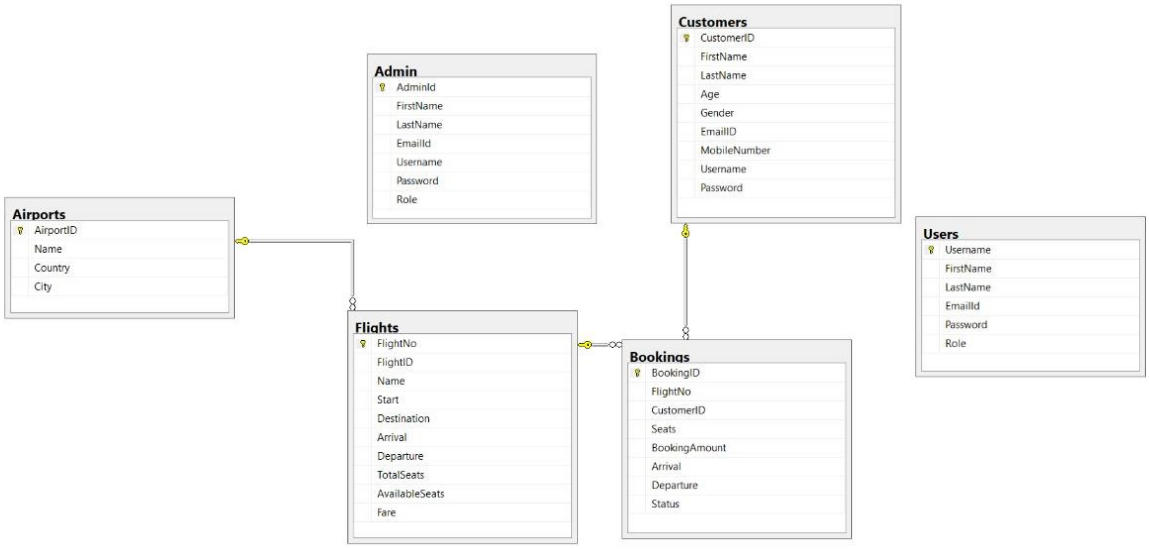
7.0 Flow Diagram



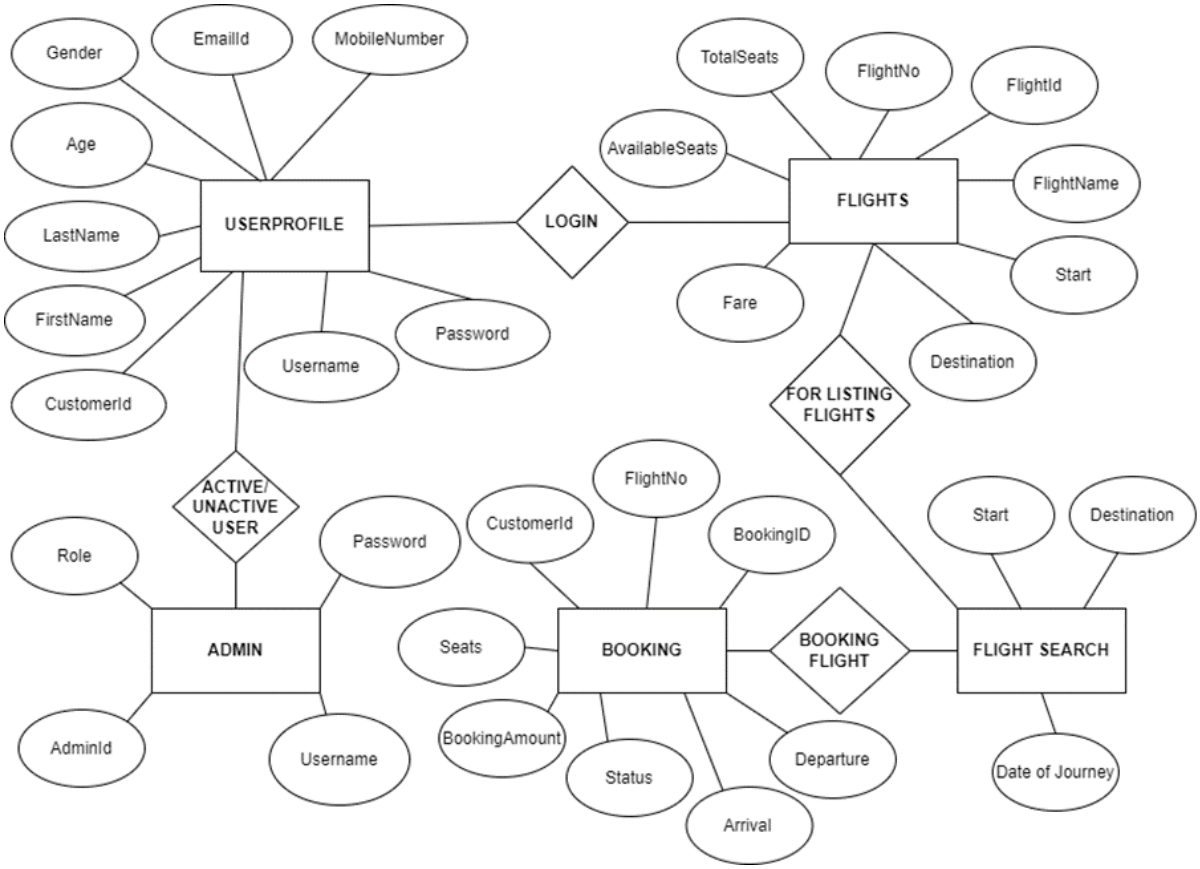
8.0 Use case Diagram



9.0 Class Diagram



10.0 E-R Diagram



11.0 User Requirements

11.1 Hardware

- Processor: Minimum 1.8 GHz. Recommended 2GHz or more
- Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)
- Hard Drive: Minimum 100 GB; Recommended 500GB or more
- Memory (RAM): Minimum 4 GB; Recommended 8 GB or above
- OS: Windows

11.2 Software

- Any Latest Browsers

12.0 Developer Requirements

12.1 Hardware

- Processor: Minimum 1.8 GHz. Recommended 2GHz or more
- Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)

- Hard Drive: Minimum 100 GB; Recommended 500GB or more
- Memory (RAM): Minimum 4 GB; Recommended 8 GB or above
- OS: Windows

12.2 Software

- Visual Studio 2022
- SQL Server Management Studio (SSMS)
- Node, Angular

12.3 Technology

- ASP.NET CORE 6.0 for backend
- SQL for the database operations
- Angular for frontend

13.0 Solution Steps

A. Customer Registration:

1. The user will select their role as a customer or Admin.
2. Customer will enter the required details such as Firstname, Lastname, Email, Phone Number, Gender, Age, Username, Password and click submit button to register.
3. The API call reaches respective controllers (Customer registration controller)
4. The input validation will have the Customer as argument to perform the validation,
   a) If validation fails, then it will return the error code and error description with status code
   b) If validation is successful, then the Customer details are stored in the database and success code is sent.
5. Success JSON response and HTTP status code 200 with corresponding success message.

B. Customer and Admin Login:

1. If customer or admin wants to login into their account then they have to enter Username and Password into the fields and click ‘Login’.
2. Admin can able to make customers account active/inactive and they can able to Check In, Add Flights data, Edit Flights data, delete Flights data, Add, Edit Customer data, Cancel Reservation and change password.
3. Customers can do the following options like Home, Login, Search Flight, Book Flight, Cancel Booking, View Booking, Change Password, Logout
4. The API call reaches the respective methods and controllers to make the changes or to fetch the data.
5. Then it will get updated in the database.
6. Success response code is returned.
7. Success JSON response and HTTP status code 200 with corresponding success message.

C. Search a Flight:

1. If Customer wants to search a flight, they have to enter FROM location, TO location, Date of Travel and click ‘Search’ button.
2. Now the API call reaches the controller and retrieves the data from database.
3. The list with all the flights are returned to the controller.
4. The controller returns the JSON response to API.
5. The available flights get displayed to the customers.
6. Success JSON response and HTTP status code 200 with corresponding success message.

D. Flight Booking:

1. Customer then chooses the required flight he/she wants and click the book option.
2. After, the Customer can enter his/her details, and create a booking by clicking on the CONFIRM button.
3. If booking is successful, the next confirmation screen is displayed with a booking reference number or booking id.

E. Cancel a booking:

1. If a booking to be cancelled, customer or admin must enter their Booking ID as input and click the ‘Cancel’ button.
2. If the Booking is present, it will delete from the database, it sends a response body with HTTP Success response code 200 else sends the not found response code 404.
3. The success JSON response and HTTP status code 200 with corresponding success message to the API gateway which will send to web app. Web app will displays Success message on the view.

F. View a booking:

1. If a booking to be viewed, user or admin must enter their Booking ID as input and click the ‘View’ button.
2. If the data is successfully fetched from the database, it sends the response body with the JSON data and HTTP Success response code 200 else sends the response code 404 not found
3. The success JSON response and HTTP status code 200 with corresponding success message to the API gateway. The web app will display the list of all the flights as searched with a ticket booking button ‘Book’.

G. Change the password:

1. If user or admin want to change the password, user or admin must enter the Username, old password and new password as input and click the ‘Submit’ button.
2. If the Username and the password are matched with the one on the database then it will change the password and sends the success code 200 else sends error code 400.

3. The success JSON response and HTTP status code 200 with corresponding success message to the API gateway which will send to web app. Web app will displays Success message on the view.

H. Updating Check-In status:

1. If the user wants to check in into a flight, user or admin must enter their Booking ID as input and click the 'Check In' button.
2. User inputs are validated by the angular web application:
   a) If validation fails, then it will return the error code and error description with status code 400
   b) If validation is successful, Call reaches the Web API gateway and it calls the API/controller 'Booking' and also the method 'CheckInAFlight()'.
3. If check-in is successful, it displays the confirmation message along with the booked seat number.

14.0 Classes/function

#	Class	Description
1	Customers.cs	Model holds the customers schema details
2	Flights.cs	Model holds the Flights schema details
3	Bookings.cs	Model holds the bookings schema details
4	Admin.cs	Model holds the admin schema details
5	Airports.cs	Model holds the airports schema details
6	Login.cs	Model holds the Login details schema details
7	CustomersController.cs	Controller Model holds the customers schema details
8	FlightsController.cs	Controller model holds the Flights schema details
9	BookingsController.cs	Controller model holds the bookings schema details
10	AdminController.cs	Controller model holds the admin schema details
11	AirportsController.cs	Controller model holds the airports schema details
12	LoginController.cs	Controller model holds the Login details schema details
13	DBContext.cs	Model holds the Db context schema details
14	Repository.cs	Repository holds the repository schema details of all the fields
15	Interface.cs	Interface holds the interface schema details of all the fields
16	Services.cs	Service holds the service schema details of all the fields

15.0 Data Model

Constraints	Columns	Data Types
Login details		
PK	Login ID	Varchar(15)
	Password	Varchar(16)
	Last Password Change	DateTime
	Logged In	Bit
FK	Role	Varchar(5)
Admin		
PK	Admin ID	Varchar(10)
	Name	Varchar(100)
	Role	Varchar(5)
Flights		
PK	Flight No	int
	Flight ID	Varchar(10)
	Name	Varchar(10)
FK	Start	Varchar(10)
FK	Destination	Varchar(10)
	Arrival	DateTime
	Departure	DateTime
	Seats	int
	Base price	Decimal
Customers		
PK	Customer ID	int
	Customer Name	Varchar(100)
	Email ID	Varchar(50)
	Mobile Number	Varchar(10)
	Gender	Varchar(5)
	Login ID	Varchar(15)
	Password	Varchar(16)
Bookings		
PK	Booking ID	Varchar(10)
FK	Customer ID	int
FK	Flight No	int
	Booking Amount	Decimal
	Arrival date	DateTime
	Checked In	Varchar(10)
	Seats	int

16.0 API Canvas

S.NO	Micro Service	Path	Verb	API Description	Role	Auth

1	Flight Booking System	/auth	GET	To login the user or admin account	No	True
2	Flight Booking System	/Customers	GET	To get customer details	No	True
3	Flight Booking System	/Customers	POST	To register a new customer	No	True
4	Flight Booking System	/Customers/id	GET	To get customers details by customer id	No	True
5	Flight Booking System	/Customers/id	PUT	To record edited customer details by id	No	True
6	Flight Booking System	/Customers/id	DELETE	To delete customers details by customer id	No	True
7	Flight Booking System	/Flights/start/dest/doj	GET	To search the flights on start place, destination and date of journey	No	True
8	Flight Booking System	/Flights/id	GET	To search the flights on a date and list all the flights	No	True
9	Flight Booking System	/Flights/Flightno	GET	To get the list of flights by flight number	Admin	True
10	Flight Booking System	/Flights/Flightno	PUT	To record edited flight data by flight number	Admin	True
11	Flight Booking System	/Flights/Flightno	DELETE	To delete flight data by flight number	Admin	True
12	Flight Booking System	/Booking/id	POST	To book a flight on a particular date	No	True
13	Flight Booking System	/Booking/id	GET	To all the view the bookings	Admin	True
14	Flight Booking System	/Booking/id	DELETE	To cancel a booking by its booking ID	No	True
15	Flight Booking System	/Booking/id	PUT	To record the checking Ins by the admin	Admin	True
16	Flight Booking	/Airports	GET	To get the available airports	Admin	True

	System					
17	Flight Booking System	/Airports	POST	To add airport data	Admin	True
18	Flight Booking System	/Airports/id	GET	To get the available airports by id	Admin	True
19	Flight Booking System	/Airport/id	PUT	To modify the airport data by id	Admin	True

17.0 HTTP Status Code

200 - Request succeeded
404 – Data not found
501 – Not Implemented

