**Problem Statement** - We are having real time streams data as flight and bus bookings, and I had to design a mini application that can store the Flight/Bus/Airport/BusStands data and all the flight bus route “connections” data in such a manner that it is scalable and fast for the queries like get the shortest route of Flight/Bus based on the choice for any particular data for pair of source and destination.

**Technologies used-**

Java, Spring Boot, Lombok etc and have kept data in Memory for this mini project but it is designed in manner that it is storable and easily extensible and scalable, and can get clipped compact graphs out of it easily as per the demand or business logic.

**Assumptions -**

1. In my solution I have not worked with TimeZones and I am assuming that the whole world used one TimeZone only.
2. If you search for a particular Day for Flights/Bus, I consider Flight/Bus for only the next two days (consider the cases where we have to take multiple indirect fights to reach there), and show my suggestion of least time taken based on those flights/bus.
3. Current solution is to have one airport and BusStand in each city and my storage actually is supportable and easily extensible for having other cases like multiple airports and buses.
4. I could have gone and would have added multiple APIs based on each criteria, but I guess that's not the intention of this assignment, and neither the time permitted with so many things going on in personal life and office working days, so just to show how we can use our graphs and get results better and quicker I have implemented few APIs which are kind of generic and work for both Flight and Bus, and handled the which one I should choose based on travelMode preference of the user.

**Design -**

1. I have tried to visualize everything in graphs and it is like a forest of graphs which are connected with nodes inside a graph and we have SuperNodes which connect these different components of graph, for e.g. IGraph of India and Graph of USA is joined with superNodes as International Airports and flights and we have several normal nodes Domestic Airports and connections, so yeah each connection has this connectionType property.
2. If you talk about a particular graph I have kept City as my first place to go when searching for a path, from there I will have airports and BusStands inside that city (This i call as TravelMode in my code),
   1. Let's say the TravelMode was only requested for flights, then from here I already have reached my source airport in this way, which has all the scheduled flights as its connections.
   2. The connection has all the data required which we will be using as filters for building a compact graph for pulling it out from Database, as the graph on which we will be applying any algorithm should be of size so that it should be supporting a machine’s memory size, and for quicker computation.
   3. After applying filters like travel date, connection type (international/domestic), available seats( which we can fetch from Flights data this connection references), etc and we can fetch and return.
3. From here we have our compact graph which we have fetched now, we can apply our algorithms based on the use cases.
   1. For now I have implemented the Dijkstra algorithm to find the shortest path and have implemented that as well to return a single shortest path from a source and a destination.
   2. The code is written in such a manner that we can easily switch to a better algorithm or optimise our Dijkstra only to return K shortest Paths as well.

**Implemented APIs**

1. **I have written 3 APIs that use Excel for quick loading our data and for creating our NETWORK. (please see all sample excel files in code inside directory “filesToUpload”) . First all these have to run in this sequence only to create sample data and network. (For Bus these columns are same just replace Airport with BusStand and Flight with Bus)**
   1. **POST - /uploadAirportsOrBusStands -** This is used to upload Airport data, which has the following columns in excel, it creates the base structure of Graph nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Airport Name | Airport Address | Airport Code | City | Airport Type (I/D) | Country |

* 1. **POST - /uploadFlightsOrBus -** This is used to upload data, which has the following columns in excel.

|  |  |  |
| --- | --- | --- |
| Flight Num | No of Business Class Seats | No of Economy Class Seats |

* 1. **POST- /uploadFlightOrBusSchedules-** This is used to upload the connection of a Flight or a Bus from a source to a destination and have all necessary information with it.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FLIGHT/BUS | Flight No | Src Airport | Dest Airport | Dep Time | Arr Time | Flight STATUS | Business Price | Economy Price | Currency |

1. I could have gone and would have added multiple APIs based on each criteria, but I guess that's not the intention of this assignment, so just to show how we can use our graphs and get results better and quicker, I have created an API, which gives takes source Airport/BusStand and destination Airport/BusStand
   1. Currently in API you should choose either both Airport or both Bus Stand, bus as we have our City as the main key inside a minimal graph, we can easily go to both Airports and BusStands from there, as we have our graph stored in that manner.
   2. Hence taking above consideration the implemented API also takes the TravelMode (FLIGHT/BUS) along with the src and dest.

**API- GET - /recommendFlightsOrBus/{src}/{dest}/{travelMode}/{date}**

* Sample - **localhost:8083/recommendFlight/KOLK/BHOP/FLIGHT/2020-11-11T00:01:01**

Where

* + src - source Airport/Bus code
  + Dest - destination Airport/BusStand code
  + travelMode - FLIGHT/BUS
  + Date in (yyyy-mm-ddThh:mm:ss) format.
* This returns a shortest path with all info about airports coming in between and the flightNumbers should be taken or Buses based on what travelMode chosen was.
* Rest as I said I could have gone and would have added many multiple API having different features, because my design supports them easily, but as I said, I guess that’s not the goal of the assignment, so I have implemented above features only and have designed the system easily extensible and scalable.

**I have added my postman collection also, with the code inside “fileToUpload” directory, in case anyone wants to go for quick testing.**