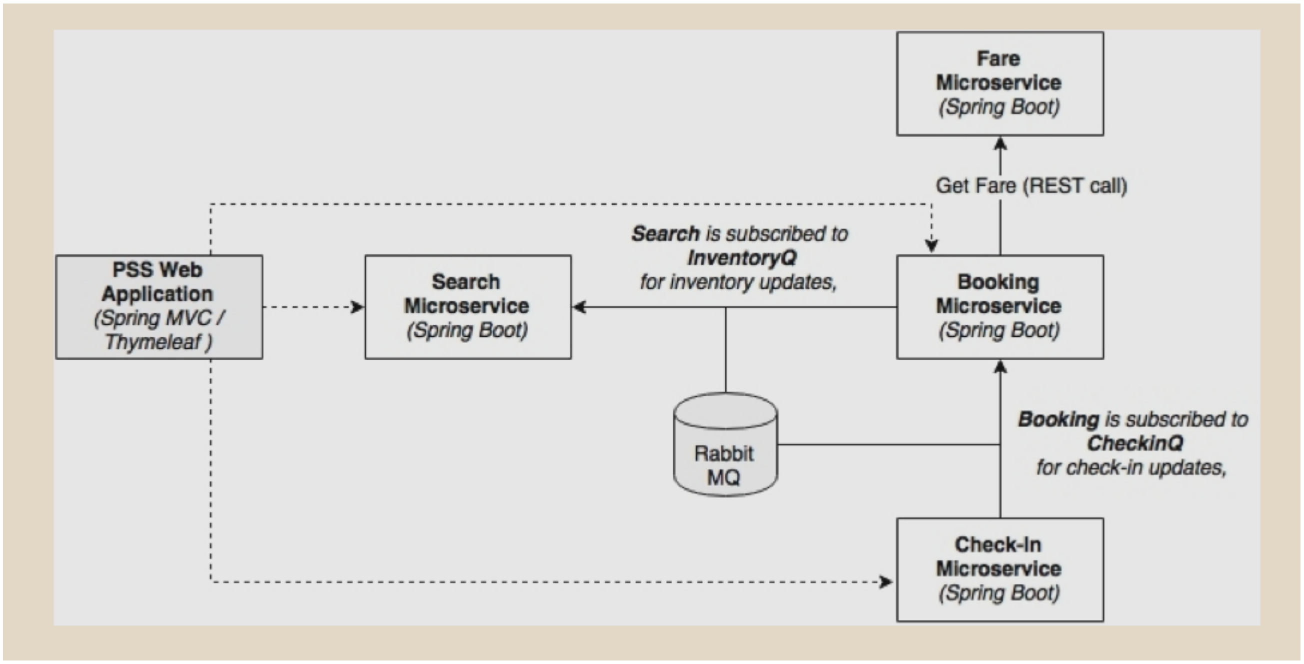
Flight Booking System

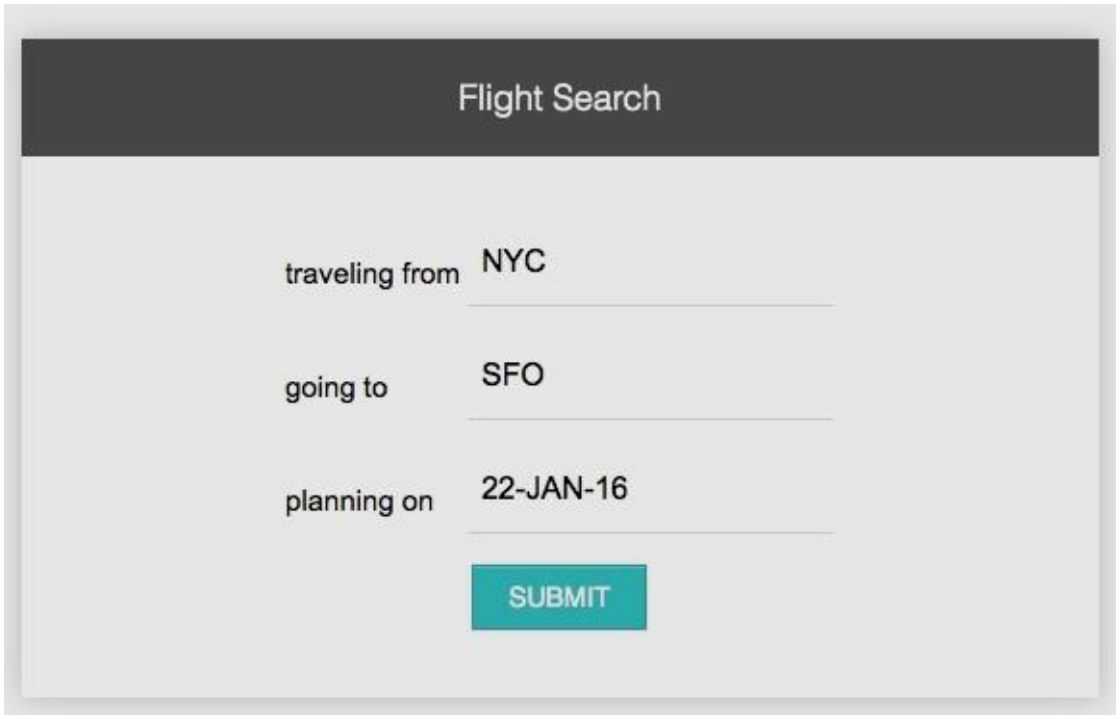
We will have to accomplish the following items in our microservices implementation:

1. Each microservice exposes a set of REST/ JSON endpoints for accessing business capabilities
2. Each microservice implements certain business functions using the Spring(Boot) framework.
3. Each microservice stores its own persistent data using H2 in-memory database
4. Microservices are built with Spring Boot, which has an embedded Tomcat server as the HTTP listener
5. RabbitMQ is used as an external messaging service. Search, Booking, and Check-in interact with each other through asynchronous messaging.



Fare Microservice (Spring Boot)
Get Fare (REST call)
PSS Web Application (Spring MVC / Thymeleaf)
Search Microservice (Spring Boot)
Search is subscribed to InventoryQ for inventory updates,
Booking Microservice (Spring Boot)
Rabbit MQ
Booking is subscribed to CheckinQ for check-in updates,
Check-In Microservice (Spring Boot)

As shown in the preceding diagram, we are implementing four microservices as an example: Search, Fare, Booking, and Check-in. In order to test the application, there is a website application developed using Spring MVC. The asynchronous messaging is implemented with the help of RabbitMQ. In this sample implementation, the default H2 database is used as the in-memory store for demonstration purposes
The browser asks for basic security credentials. Use guest or guest123 as the credentials. This example only shows the website security with a basic authentication mechanism. service-level security can be achieved using OAuth2.
Entering the correct security credentials displays the following screen. This is the home screen of our Flight Booking application:


Flight Search

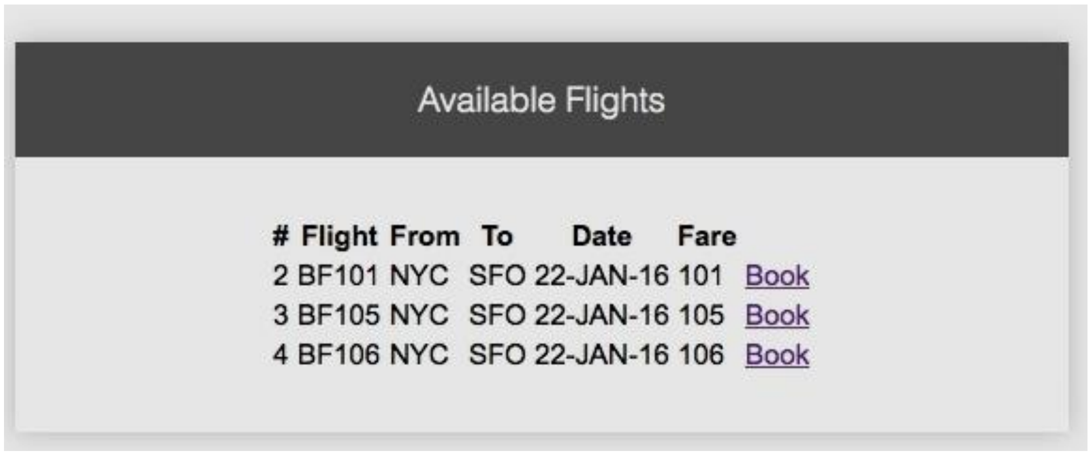
traveling from NYC

going to SFO

planning on 22-JAN-16

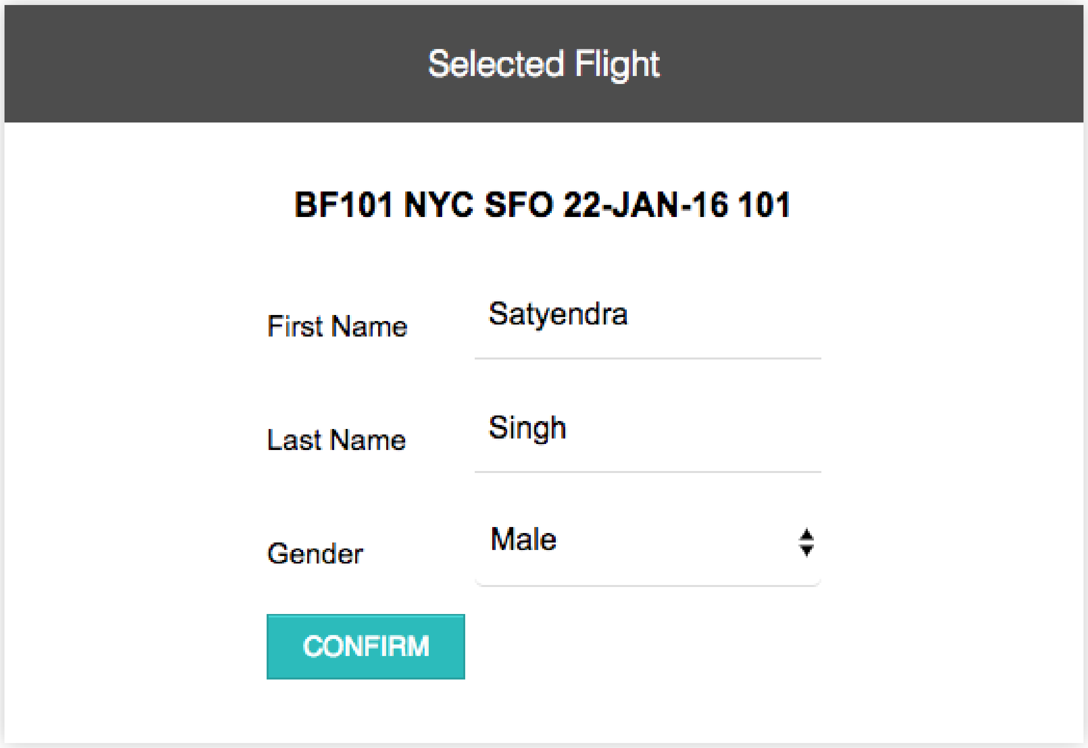
The SUBMIT button invokes the Search microservice to fetch the available flights that meet the conditions mentioned on the screen. A few flights are pre-populated at the startup of the Search microservice. Edit the Search microservice code to feed in additional flights, if required.

The output screen with a list of flights is shown in the next screenshot. The Book link will take us to the booking screen for the selected flight:



Available Flights

#	Flight	From	To	Date	Fare	
2	BF101	NYC	SFO	22-JAN-16	101	Book
3	BF105	NYC	SFO	22-JAN-16	105	Book
4	BF106	NYC	SFO	22-JAN-16	106	Book



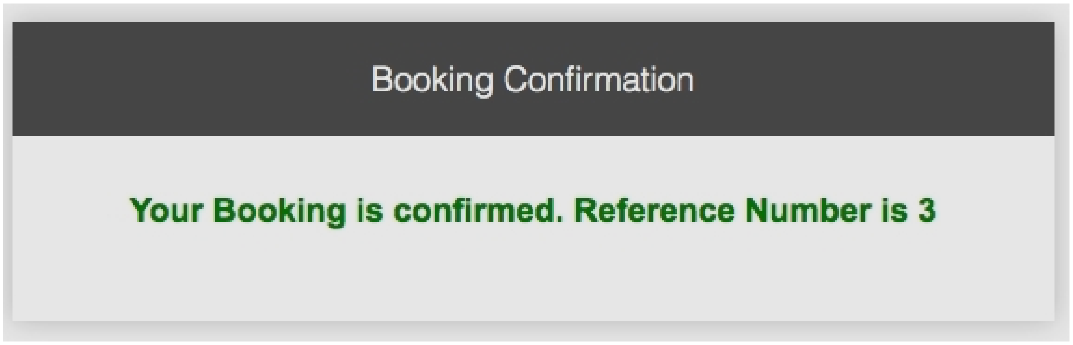
Selected Flight

BF101 NYC SFO 22-JAN-16 101

First Name	Satyendra
Last Name	Singh
Gender	Male

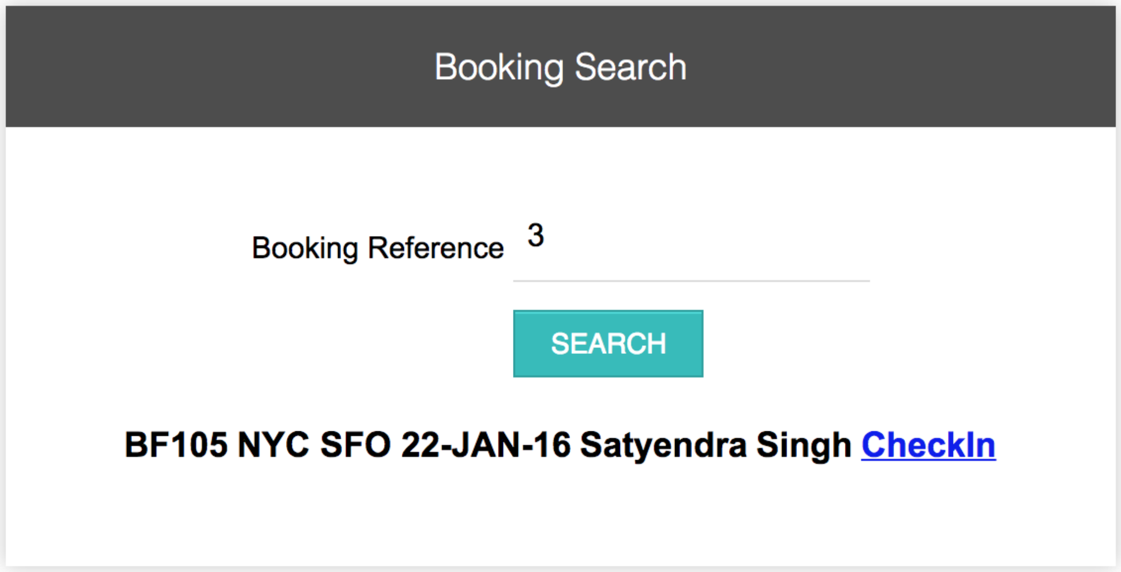
The following screenshot shows the booking screen. The user can enter the passenger details, and create a booking by clicking on the CONFIRM button. This invokes the Booking microservice, and internally, the Fare service as well. It also sends a message back to the Search microservice:

If booking is successful, the next confirmation screen is displayed with a booking reference number:



Your Booking is confirmed. Reference Number is 3

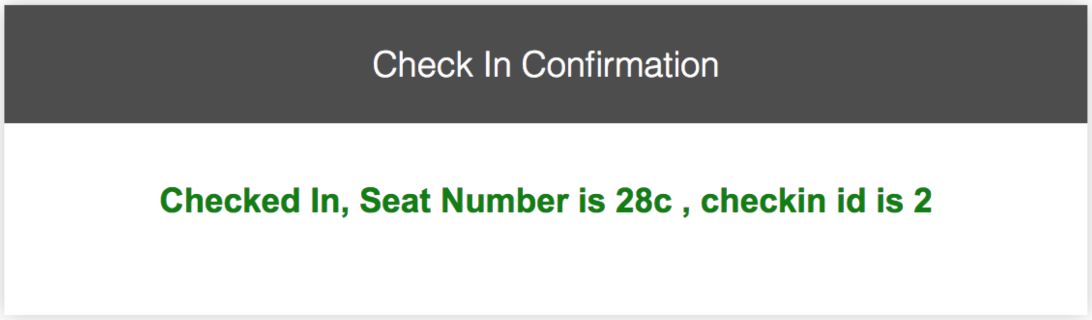
Clicking on the SEARCH button in the previous screen invokes the Booking microservice, and retrieves the booking information. Click on the CheckIn link to perform the check-in. This invokes the Check-in microservice:



Booking Reference 3

BF105 NYC SFO 22-JAN-16 Satyendra Singh CheckIn

If check-in is successful, it displays the confirmation message, as shown in the next screenshot, with a confirmation number. This is done by calling the Check-in service internally. The Check-in service sends a message to Booking to update the check-in status:



Checked In, Seat Number is 28c , checkin id is 2