# **Mastercard Challenge**

1. **Resource endpoint :**

<http://localhost:8080/connected?origin=city1&destination=city2>

**1.1 Input :**

* String city1 (Optional)
* String city2 (Optional)

**1.2 Output :**

* yes – if connected
* no – if not connected or error

1. **Running the application :**
   1. Copy the **city.txt** input file to the same location as the jar file.
   2. Execute the following command:

**java -jar challenge-1.0.jar**

1. **Implementation details :**
2. **ChallengeApplication :**

Springboot application class. Defines main method to start the application.

1. **AppInitializer :**

Read the city.txt file as a stream().

Build a graph with city as vertex and add an edge between the vertices if the cities are connected.

Create cityIndexMap which will have city name as key and index as value. We need this map as graph contains integer values as vertex instead of city name string.

e.g. :- Boston – 1

Philadelphia – 2

This **graph** acts as a database for this application.

1. **RouteController :**

Defines the handler method for rest endpoint “/connected”. This takes city1 and city2 as input.

This method invokes the **RouteService** to process the request.

1. **RouteService :**

Get the index for the city1 and city2 from the CityIndexMap.

If the index value is -1, then city is not defined in the input file **city.txt**. Return response as **“no”** as the input is invalid.

If validation is successful, then invoke **Graph.isconnected()** method to check the connectivity.

1. **Graph :**

This class defines methods for creating the graph.

This will create the graph as a n x n adjacency matrix.

addEdge() method adds the edge between the 2 vertices.

isConnected(int source, int destination) method checks if there is an edge between the source and destination.

1. **CityData :**

This class holds the cityIndexMap and graph as static members.

This acts as a database for the application.

1. **TestSuite :**
   1. **RouteControllerTest**

Defines tests for positive and negative cases using RestTemplate.

* 1. **RouteControllerMockTest**

Defines tests for positive and negative cases using MockMVC framework and mocking the service responses.

* 1. **ChallengeApplicationTests**

Very context loads.

* 1. **SmokeTest**

Verify controller objects are not null.