

Architecture Diagram and Labs User Stories

Globally Distributed App

Click2Cloud Inc.

Email: contact@click2cloud.net

Document Version – V1.1(Draft)

Contents

[Overview 3](#_Toc492333340)

[Proposed Architecture 4](#_Toc492333341)

[Labs Outline 4](#_Toc492333342)

[LAB 1: Globally-distributed apps using Azure Cosmos DB Service 5](#_Toc492333343)

[Problem Statement 5](#_Toc492333344)

[Solution 5](#_Toc492333345)

[Scenario 6](#_Toc492333346)

[Technology Used 8](#_Toc492333347)

[LAB 2: Using Table API of Azure Cosmos DB Service 8](#_Toc492333348)

[Problem Statement 8](#_Toc492333349)

[Solution 8](#_Toc492333350)

[Scenario 8](#_Toc492333351)

[Technology Used 9](#_Toc492333352)

[LAB 3: Using Graph (Gremlin) API of Azure Cosmos DB Service 9](#_Toc492333353)

[Problem Statement 9](#_Toc492333354)

[Solution 9](#_Toc492333355)

[Scenario 10](#_Toc492333356)

[Technology Used 10](#_Toc492333357)

[LAB 4: Connect Chat Bot with Cosmos DB 10](#_Toc492333358)

[Problem Statement 11](#_Toc492333359)

[Solution 11](#_Toc492333360)

[Scenario 11](#_Toc492333361)

[Technology Used 12](#_Toc492333362)

[LAB 5: 12](#_Toc492333363)

[Problem Statement 12](#_Toc492333364)

[Scenario 12](#_Toc492333365)

[Technology Used 13](#_Toc492333366)

[LAB 6: 13](#_Toc492333367)

[Problem Statement 13](#_Toc492333368)

[Scenario 13](#_Toc492333369)

[Technology Used 14](#_Toc492333370)

# Overview

This document contents the proposed Labs and architecture for Globally distributed application on Immersion Labs.

Proposed lab would be called as; Globally Distributed Apps, one of the innovative strategy of Microsoft designed to deliver apps for FY18. Once the App would get ready, it would be used across the globe to validate Microsoft’s vision and strategy.

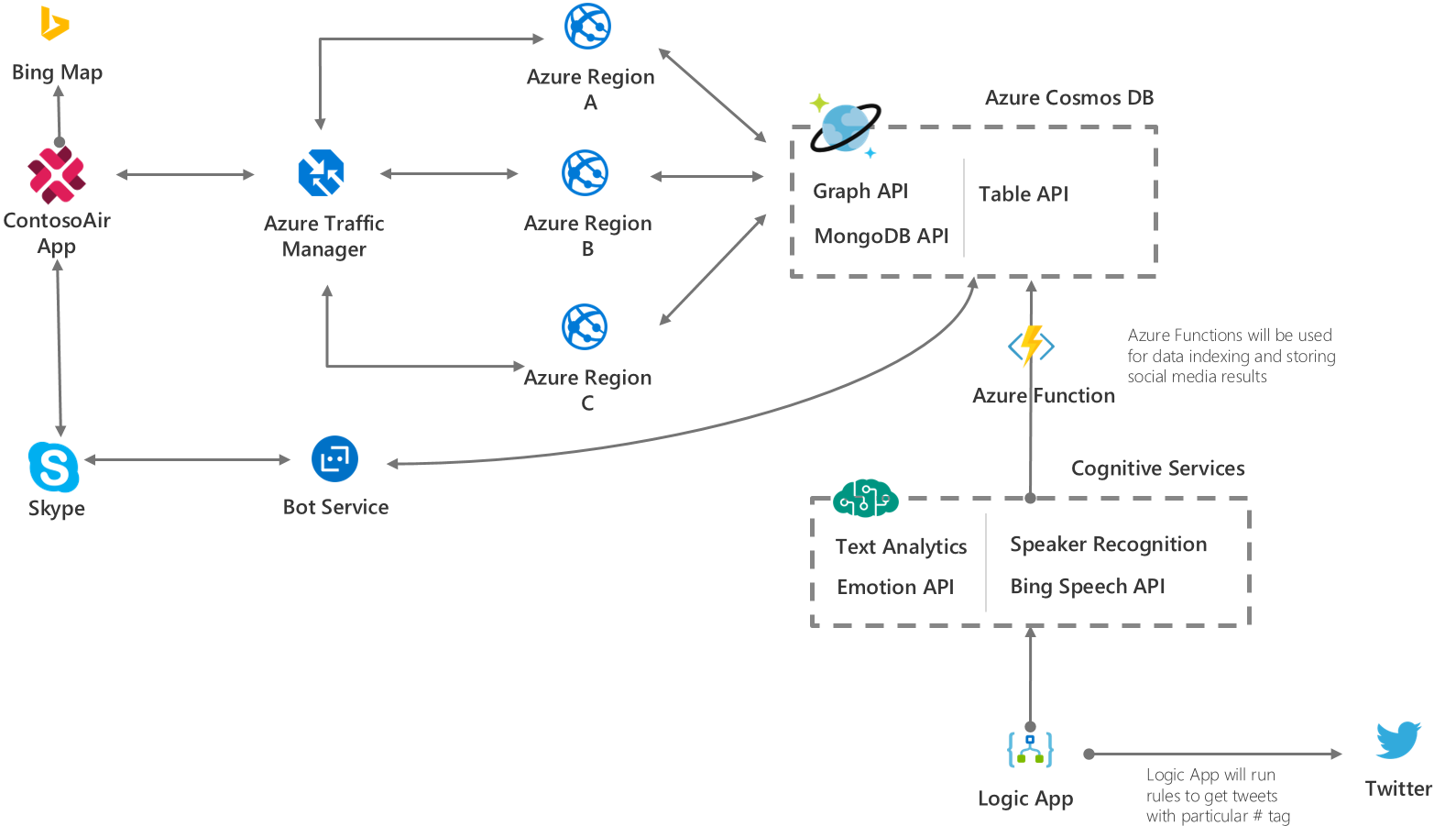
The globally distributed app would be an extension of the “ContosoAir.Com” demo, which is a customer facing app. The front end of the app code is available (can be used after some ticks) and entire backend would be developed from scratch. The app would be using majorly the Cosmos DB. Each lab should present different aspects of global distributed applications.

There would be total 6 labs around following technologies. Each can lab use a single or the combination of below mentioned technologies.

* Cosmos DB
* Azure Functions
* Cognitive Services
* BOT
* Logic App
* Bing Map
* Machine Learning
* Azure Search

Each lab to have its own user stories and should run in isolation with reference to another labs.

# Proposed Architecture



# Labs Outline

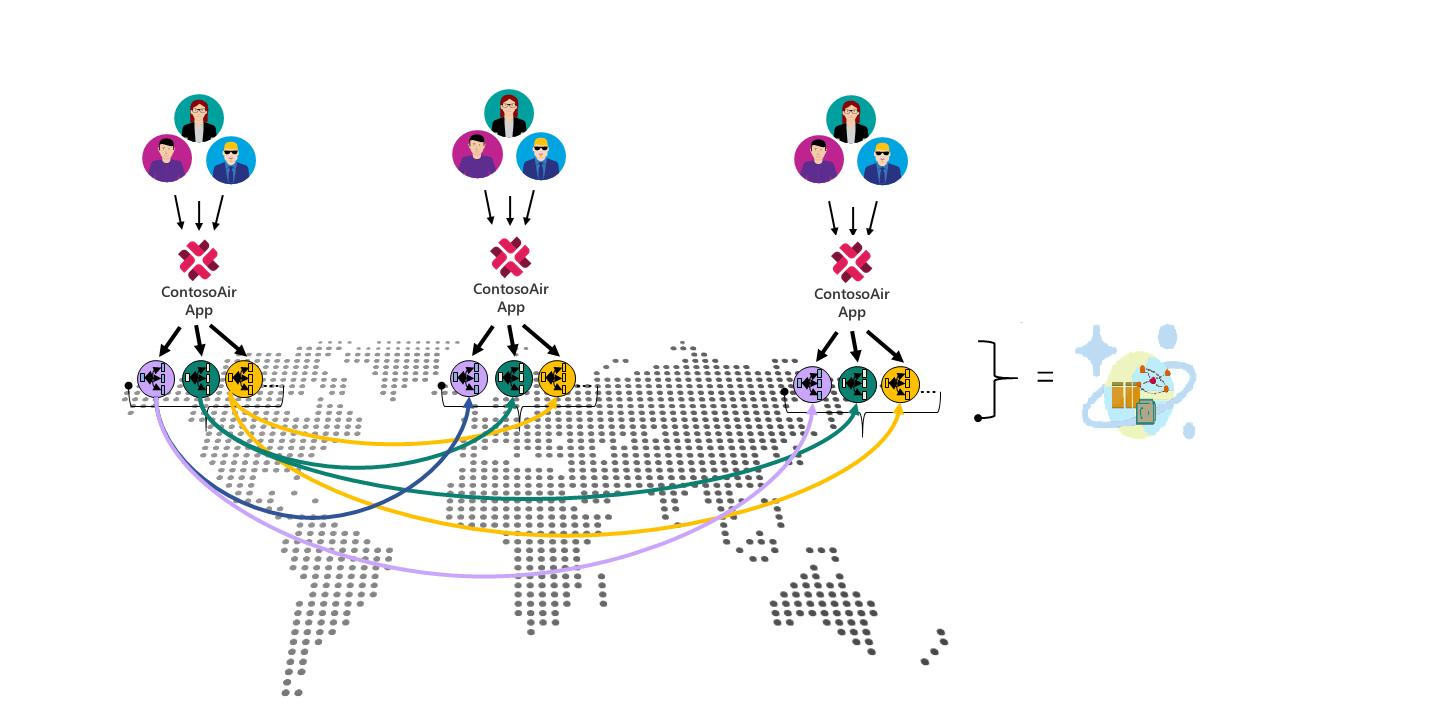
|  |  |  |  |
| --- | --- | --- | --- |
| # | Outline | Scenarios | Technologies |
| 1 | Globally-distributed apps using Azure Cosmos DB Service | 1. Setup lab environment by using DevOps tools 2. A single Azure Cosmos DB collection partitioned and distributed across multiple Azure regions. 3. Tunable data consistency levels in Azure Cosmos DB | App Service, Cosmos DB, ARM Templates, Ansible Playbooks |
| 2 | Using Table API of Azure Cosmos DB Service | 1. Build application using Table API of Cosmos DB 2. Retrieve flights details from Cosmos DB and plot details using Bing Map | App Service, Cosmos DB, Bing Map |
| 3 | Using Graph (Gremlin) API of Azure Cosmos DB Service | 1. Build application using Gremlin API of Cosmos DB 2. Retrieve details from Cosmos DB and plot on graph | App Service, Cosmos DB, Graph API |
| 4 | Connect Chat Bot with Cosmos DB | 1. Bot framework and connect Bot service with Cosmos DB 2. Read data from Cosmos DB and share result as a response on Bot connector | Cosmos DB, Bot Services, Bing Map |
| 5 | DevOps Tools (<https://azure.microsoft.com/en-in/try/devops/>) |  |  |
| 6 | Cosmos DB, Logic App, Cognitive Services and Azure Function |  |  |
| 7 | Machine Learning |  |  |

# LAB 1: Globally-distributed apps using Azure Cosmos DB Service

## Problem Statement

Traditionally, it takes a long time and a lot of tedious effort to build a globally distributed database and there will be some issue like latency and consistency if the data stored in centralized location. So, to solve this we will use Azure Cosmos Database.

## Solution



## Scenario

|  |  |
| --- | --- |
| 1 | Scenario walkthrough – Setup lab environment using DevOps Tools  The lab environment setup will include, deploying the front-end application, service API and Cosmos DB.  Prerequisites:   1. Lab will be setup to run the Ansible Playbooks 2. ARM Templates to setup application resources 3. Access to VSTS   Instructions:   1. To setup the lab environment, the user will launch the Ubuntu bash shell and update the required information such as Subscription details, Resource group etc in Ansible playbook. 2. Now run the Ansible playbook to setup the environment 3. Now, login to Azure Portal and verify the application resource in provided Resource group |
| 2 | Scenario walkthrough – A single Azure Cosmos DB collection partitioned and distributed across multiple Azure regions  Latency is the delay between a client request, probably a request made by you at your computer, and a response to that request.   1. The Developer runs an application of ContosoAir. 2. The application which he deployed, is available in only one region, for eg. East US. 3. The developer than try to access the application and do all the operation like searching and booking a ticket and so on. 4. As in our example the Cosmos DB is initially available in only Singapore region, so user will notice late response as both Cosmos DB and Web Service, which request for data, is hosted on different regions. 5. After that, the developer again change the region, in ARM template, and extend the support of Cosmos DB in East US region. 6. The developer again repeat the same process of searching and booking a ticket and this time he/she will get a low latency. |
| 3 | Scenario walkthrough – Tunable data consistency levels in Azure Cosmos DB  Different regions have their own copy of database, and they may not always be consistent if read / write operations happen frequently.   1. There is a possibility where another user book the same seat from another region which is already booked recently. 2. So, the user from this region should be aware about this changes. 3. So, to test this scenario developer can deploy this application in two different regions and browse the application and will try to book an air ticket for a seat. 4. After that, the developer will open another application deployed in another region and will try to book an air ticket for a same seat. 5. Developer can set the different consistency level of the cosmos database to see the impact of the how frequently the data updates. |
| 4 | Scenario walkthrough – Multi-Model API Support   1. From the Contosoair application, user will search the flights records which is retrieve using Table API. 2. The user can see the flight feedback posted by different user which will be stored using the Gremlin API and show in graphical format. |
| 5 | Scenario walkthrough – The planet scale  Prerequisite:  Visual Studio Ultimate edition   1. User will launch the Visual Studio Ultimate edition. 2. Using Visual Studio, user will make 1000 concurrent request to the ContosoAir application for single region. 3. Using the Microsoft Azure web portal, observer the Cosmos DB is scaled automatic. |

## Technology Used

App Service: For hosting front end and service layer

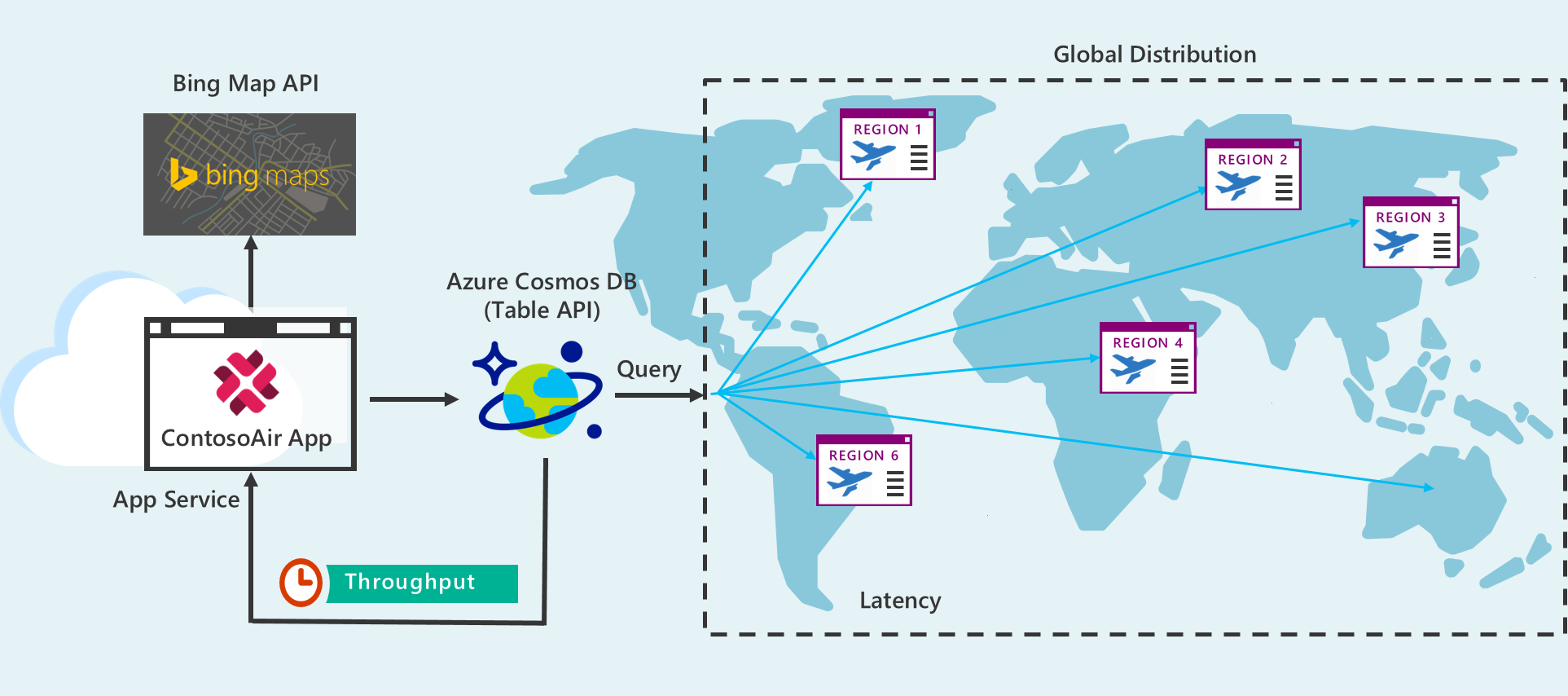
ComosDB with MogoDB API: It is used to store and retrieve data of countries for plotting it into the Bing Map

# LAB 2: Using Table API of Azure Cosmos DB Service

## Problem Statement

A ContosoAir customer is planning for vacation overseas and wanted to search for available location which doesn’t require VISA for travelling.

## Solution



## Scenario

|  |  |
| --- | --- |
| 1 | Scenario walkthrough – Cosmos DB (Table API)   1. Customer visit on contosair.com website for Air Ticket Booking. 2. Fill all the details required for Ticket Booking. 3. Enter Place name where he/she wants to visit. 4. It will display Place details whether it requires VISA or not and other details. 5. By clicking Find Other Placesbutton contosoair.com will display all places that doesn’t required VISA for selected citizen. |
| 2 | Scenario walkthrough – Bing Map API   1. After clicking on Find other places button contosoair.com will fetch all countries from CosmosDB Database. 2. And plot all fetched countries on Map using Bing Map API by Azure 3. Then user can also select other available places which does not required VISA 4. And Complete Ticket Booking Process on contosoair.com |

## Technology Used

Cosmos DB: It is used to store and retrieve data of countries for plotting it into the Bing Map

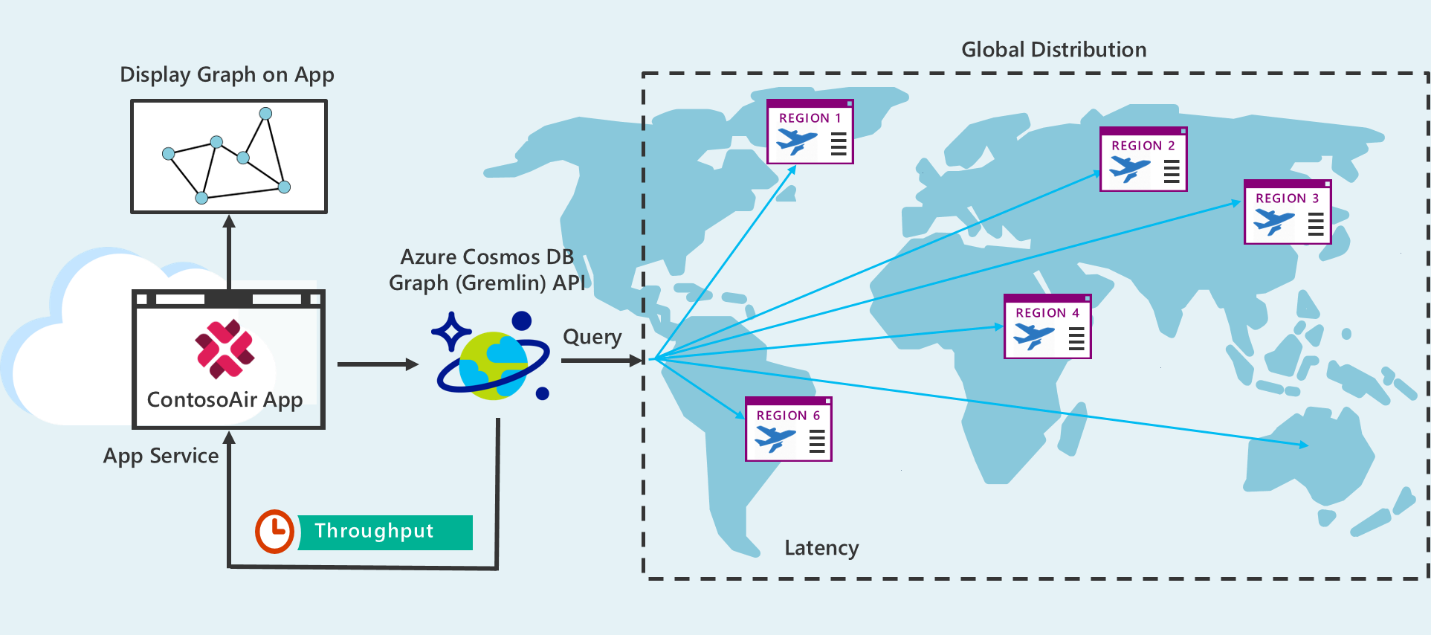
Bing Map: Bing Map service is used to Plot all the countries who doesn’t need VISA on Map

# LAB 3: Using Graph (Gremlin) API of Azure Cosmos DB Service

## Problem Statement

When customers want to travel by air, they book for specific airline. But, in a situation, they may experience that they are not going to travel with same airline, which they have booked for, but with a different airline. This happens because of codeshare, which is a business arrangement where two or more airlines share the same flight. To avoid customer inconvenience, they can check codeshare for booked airline by clicking Check Codeshare button and this will show all codeshare airline details of booked flight.

## Solution



## Scenario

|  |  |
| --- | --- |
| 1 | Scenario walkthrough – Cosmos DB   1. Customer visit on contosair.com website for Air Ticket Booking. 2. Fill all the details required for Ticket Booking. Like Trip Type, From Date, To Date, Depart Date, Return Date & Passenger 3. It will list all available flights with their Airline Provider Name and fare. 4. Select flight appropriate flight as per need 5. By clicking Check Codeshare button user can see all available Codeshare for selected Airline provider. |
| 2 | Scenario walkthrough – Cosmos DB Graph (Gremlin) API   1. After clicking on Check Codeshare button, it will display codeshare. 2. Codeshare will display in graphical format by using Cosmos DB Graph (Gremlin) API. 3. And click on Book Ticket Button to complete Ticket booking process. |

## Technology Used

Cosmos DB: Cosmos DB is used to store and retrieve the Airline Codeshare.

Cosmos DB Graph API: Cosmos DB Graph API is used for display the codeshare airlines in graphical format.

# LAB 4: Connect Chat Bot with Cosmos DB

## Problem Statement

The customer is driving towards the airport and gets stuck in traffic, now there are chances that he may miss the important flight. So, to get the timely information he communicates with Azure bot service to know the status of his flight, as well as boarding gate number.

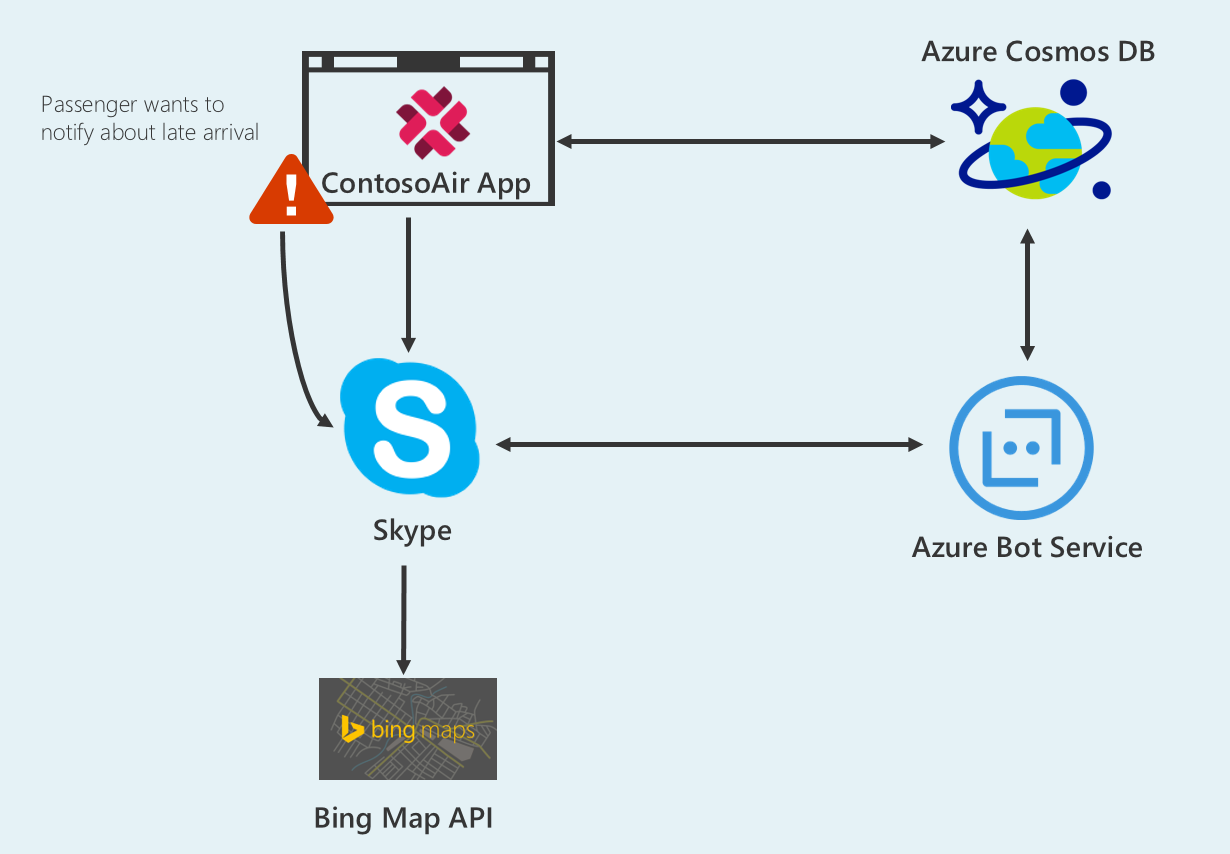
For that, bot will ask some question to the user like his flight number and will check with Cosmos DB to send appropriate information about the flight and boarding gate number which help him to reach as early as possible at the airport, the bot also send Bing map link of airport to find the shortest path or alternate to reach the airport. Once he reaches the airport the bot also send the terminal internal map to the user.

OR

For a passenger travelling through connecting flights and the first flight gets delayed and second flight is on the correct time, because of the delay caused in the first flight, there is very less time to board the next flight and so causes inconvenience for the traveler.

For that, bot will ask some question to the user like his flight number or travel details and will check with Cosmos DB to send appropriate information about the flight and alternate flight details. Based on the result, passenger can decide flight cancellation or reschedule flight.

## Solution



## Scenario

|  |  |
| --- | --- |
| 1 | Scenario walkthrough – Chat Bot   1. User booked the flight ticket from contosoair.com portal. 2. User booked cab to reach the airport. 3. While traveling toward the airport, he may get late to reach the airport because of traffic or some other problem. 4. The bot will interact will user and collect the information about the flight. 5. The bot will communicate with Cosmos DB using collected information from user. 6. The bot will send the flight status and boarding gate number to the user which will help the user to reach the airport as early as possible. |
| 2 | Scenario walkthrough – Bing map / Internal map view   1. The bot will send the airport map link. 2. Once user click on that link, Bing map will open with default airport location. 3. User will locate the route in the map and find the shortest path to reach the airport. |
| 3 | Scenario walkthrough – Send terminal overview image   1. Once user reach the airport, the bot service will send the terminal internal map. 2. User will locate the gate number to reach the terminal as early as possible. |

## Technology Used

Cosmos DB: Cosmos DB is used to store and retrieve the Airline data.

Bot Services: For knowledge based answering to the user regarding travel details.

Bing Map: Bing Map service is used to share possible shortest location to airport

# LAB 5:

## Problem Statement

A customer wants to travel to a tourist destination for an adventure. He/ She doesn’t aware more about the destination, to explore more about the destination he can take help of ContosoAir application and get an idea about the events and concerts happening during his/ her travel period, crime rates, climate situation etc.

## Scenario

|  |  |
| --- | --- |
| 1 | Scenario walkthrough – Cosmos DB, Logic App, Cognitive Services and Azure Function   1. The passenger logged-in into the Contosoair.com website. 2. The passenger will search for the flight where he/she wants to go. The cotosoair.com helps passenger to find the flights in the best rate with all the comfort. 3. The passenger will book the flight from location ABC to location XYZ. 4. After choosing the destination, the overall rating about that location is displayed on the website. 5. The Logic App will be configured to run in background which will continuously populate the tweets regarding the places. The tweets will be processed using Text Analytics of cognitive services for the sentimental analysis. It will keep the result in the Cosmos DB with the help of Azure functions. 6. The report will be shown to end customer while booking the tickets. So, that he/ she can judge the overall situation and plan the trip. 7. After booking the flight from location ABC to XYZ, the daily tweets about the location will be mailed to the user so that customer will be aware of the current situation. |

## Technology Used

Cosmos DB, Logic app, Cognitive Services (Text Analytics), and Azure function

# LAB 6:

## Problem Statement

Flight delay is one of the most remembered performance indicator of any transportation system. When a customer wants to travel from Seattle to Barcelona, if he book the ticket with one the of Airline companies who is having the bad record for the flight delays to Barcelona. It may put a person in a trouble. This can be resolved through Spark Connector and Machine Learning service of Azure.

## Scenario

|  |  |
| --- | --- |
| 1 | Scenario walkthrough – ContosoAir Website   1. The user goes to ContosoAir website. 2. User selects the Airline with which he/she wants to travel, Enter the details of source and destination under ticket booking section and clicks on the Delay prediction button. |
| 2 | Scenario walkthrough – Spark Connector   1. After clicking on the Delay prediction button, it will fetch the data from Spark Connector which will be processed through machine learning service of Azure. |
| 3 | Scenario walkthrough – Machine Learning   1. Fetched data from Spark Connector will processed by rules defined in the machine learning service of the Azure for predicting the flights status; whether it will be delayed or not. 2. According to the predictive result, user will decide to book the ticket with that Airline or not. |

## Technology Used

Spark Connector, Machine Learning