CLDV6212 FINAL POE

A. List of Azure Components Utilized:

Component	Technology Choice	Hosting Model
Azure Function	Compute	Serverless
Azure Storage Table	Data Storage	PaaS

B. Motivation for Needed Change:

Considering the primary requirements presented by Aweh Productions, the motivation for needed change could be related to optimizing the retrieval of vaccination information, handling different data formats, and minimizing operational costs.

- **Retrieval Speed:** Azure Storage Tables are suitable for certain scenarios, but if the requirement is to retrieve vaccination information within seconds, you might need to consider a more performant data storage solution. Azure SQL Database, being a relational database, might provide faster querying capabilities compared to Azure Storage Tables.
- **Handling Different Data Formats:** If there are variations in data formats from different manufacturers, you may need a storage solution that is more flexible in handling diverse data structures. Azure Blob Storage, which allows storing unstructured data like documents, images, and other file types, could be considered. This provides flexibility in storing different data formats without the constraints of a fixed schema.
- **Operational Costs:** Azure Storage Tables are known for their cost-effectiveness, but depending on the scale and specific requirements, it might be beneficial to evaluate the overall operational costs. Azure Blob Storage and Azure SQL Database have different pricing models, and the cost implications should be considered based on factors like storage size, transaction volume, and retrieval patterns.

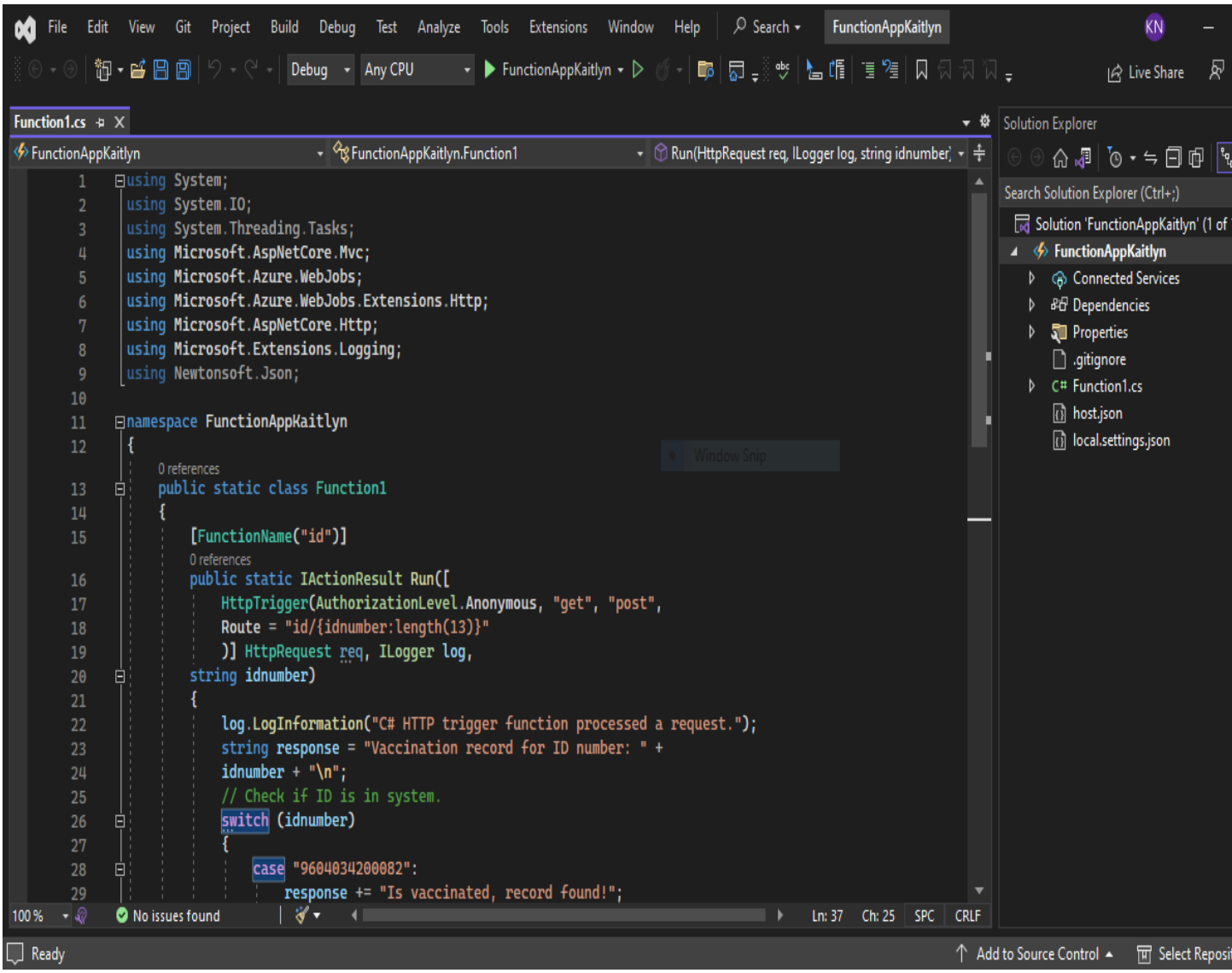
In summary, considering the need for faster retrieval, flexibility in handling diverse data formats, and cost optimization, a potential change could involve exploring Azure Blob Storage for handling varied data formats and evaluating the performance and cost implications of using Azure SQL Database for faster retrieval of vaccination information.

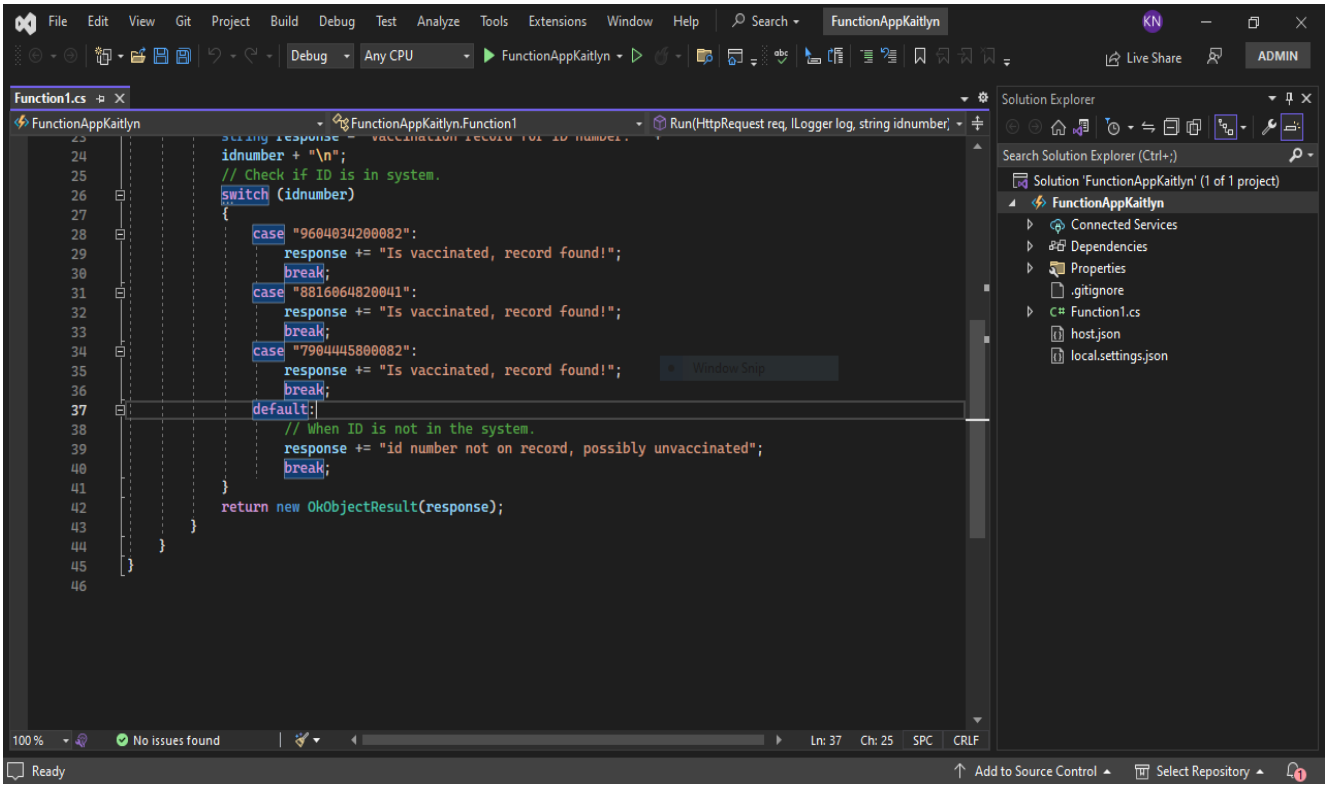
```
using Microsoft.WindowsAzure.Storage.Auth;
using Microsoft.WindowsAzure.Storage.Table;
using Microsoft.WindowsAzure.Storage;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace ConsoleApp3
{
  internal class Program
  {
    static void Main(string[] args)
    {
      try
      {
        // Get Storage Information
        var accountName = "azjhbrsgvckndnst1008bf3";
        var creds =
"w5e1CCry9j6++YHK5C4gj9fBqtlBxy9+6Din3gh8/6IOOxsGgQLeJeuUsbKQNJhItIwFIzvxQpX
A+AStfIN5pw==";


        // Set Auth
        var cred = new StorageCredentials(accountName, creds);
        var account = new CloudStorageAccount(cred, useHttps: true);
```

```
// Connect to Storage
var client = account.CreateCloudTableClient();
var table = client.GetTableReference("table1");


//User Input
Console.WriteLine("Vaccination Queue");
Console.WriteLine("Enter your ID number");
string id = Console.ReadLine();


Console.WriteLine("Enter the vaccination centre that you were administered at");
string vc = Console.ReadLine();


Console.WriteLine("Enter the vaccination date (DD/MM/YYYY)");
string vd = Console.ReadLine();


Console.WriteLine("Enter the vaccination serial number");
string vsn = Console.ReadLine();


Console.WriteLine("Enter the vaccination barcode");
string vb = Console.ReadLine();


//Connects to the Entity
```

```
            var obj = new Entity()
            {
                PartitionKey = id, // Must be unique
                RowKey = Guid.NewGuid().ToString(), // Must be unique
                Date = vd,
                Centre = vc,
                SerialNumber = vsn,
                Barcode = vb
            };
            var insertOperation = TableOperation.Insert(obj);
            table.Execute(insertOperation);
        }
        //If an error is to occur, the application will continue to run and print the error message
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            throw;
        }
    }


    public class Entity : TableEntity
    {
        public Entity(string id, string row)
        {
            this.PartitionKey = id; this.RowKey = row;
        }
```

```
        public Entity() { }
        public string Date { get; set; }
        public string Centre { get; set; }
        public string SerialNumber { get; set; }
        public string Barcode { get; set; }
    }
  }
}
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
FunctionAppKaitlyn
KN
Debug
Any CPU
FunctionAppKaitlyn
Live Share
Function1.cs
FunctionAppKaitlyn
FunctionAppKaitlyn.Function1
Run(HttpRequest req, ILogger log, string idnumber)
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;
namespace FunctionAppKaitlyn
{
0 references
public static class Function1
{
[FunctionName("id")]
0 references
public static IActionResult Run([
HttpTrigger(AuthorizationLevel.Anonymous, "get", "post",
Route = "id/{idnumber:length(13)}"
)] HttpRequest req, ILogger log,
string idnumber)
{
log.LogInformation("C# HTTP trigger function processed a request.");
string response = "Vaccination record for ID number: " +
idnumber + "\n";
// Check if ID is in system.
switch (idnumber)
{
case "9604034200082":
response += "Is vaccinated, record found!";
Window Snip
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'FunctionAppKaitlyn' (1 of
FunctionAppKaitlyn
Connected Services
Dependencies
Properties
.gitignore
Function1.cs
host.json
local.settings.json
100 %
No issues found
Ln: 37
Ch: 25
SPC
CRLF
Ready
Add to Source Control
Select Reposi


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
FunctionAppKaitlyn
Debug
Any CPU
FunctionAppKaitlyn
Live Share
ADMIN
Function1.cs
FunctionAppKaitlyn
FunctionAppKaitlyn.Function1
Run(HttpRequest req, ILogger log, string idnumber)
idnumber + "\n";
// Check if ID is in system.
switch (idnumber)
{
case "9604034200082":
response += "Is vaccinated, record found!";
break;
case "8816064820041":
response += "Is vaccinated, record found!";
break;
case "7904445800082":
response += "Is vaccinated, record found!";
break;
default:
// When ID is not in the system.
response += "id number not on record, possibly unvaccinated";
break;
}
return new OkObjectResult(response);
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'FunctionAppKaitlyn' (1 of 1 project)
FunctionAppKaitlyn
Connected Services
Dependencies
Properties
.gitignore
Function1.cs
host.json
local.settings.json
100 %
No issues found
Ln: 37
Ch: 25
SPC
CRLF
Ready
Add to Source Control
Select Repository