IST 615

Final Project Report

Immanuel Odarteifio

Nagul Pandian

Project description: Syracuse University is an undergraduate and graduate educational institution that has about 22,000 students, 75 academic departments and more than 300 student organizations. In institutions as large as this it can sometimes be hard for information to be relayed to students regarding events that happen campus wide and for students of different disciplines to explore other options. In this project we are looking to create a cloud-based service that can help departments publicize their events and for students to be able to easily find functions and events they may be interested in around the campus.

The main functionality of the service is for event organizers to be able to post their event name, description, location, extra services provided at the event. The users, students, and other faculty can then access the service and are presented with all events posted on the service.

Cloud services to be used:

**Azure App Services:** We would like to host a html landing page where users can view all the events that have been posted so far.

**Azure SQL Database:** We will use a database I this project to store at least 5 tables for user information, organizer details, venue details, events and services provided. The tables will also serve as staging areas and final storage for any new information entered on the landing page.

**Intelligent Recommendations (Machine Learning):** We were hoping to have a search section where users could input certain buzz words and using intelligent recommendations, the events that closely match those words will be the first to be displayed.

I will need to establish a connection between the database and the landing page so I can show the available events, and for users to reserve to go for events which will update a table in the database to reflect the user selections. A connection with intelligent recommendations and the landing page will help bring up relevant events for the user.

**Cloud architecture:**

Azure SQL server

SQL Database

Azure App Services

The SQL Server hosts the SQL database to enable remote connections to the database from different devices with pre-approved ip addresses. The SQL database is where we stored our tables and data. App services is how we host our webpage for it to be accessed anytime. The app service is also connected to the database to facilitate the sharing of the database data through the flask app.

Azure SQL Server: We created a standard SQL server with the default configurations.

A screenshot of a computer

Description automatically generated

SQL Database: The database was created in the cloud on Microsoft azure. It was a general purpose, serverless database with 30gb of storage. After the database was created, we opened it in azure data studio, established the connection there with the username and password. We had an up/down script from a previous project that created about 14 tables and populated it with synthetic data. We added a few more rows of data to the events table, which was the table to be heavily used in this project. The events table had a few foreign key columns, so we had to create a join and then a view showing the desired columns. This view was called v\_events.

A screenshot of a computer

Description automatically generated

Python Code: To pull the data from our database to the webpage, we need to create a python app that connects to the database. The app is responsible for the connection and the local host for the webpage.

A screenshot of a computer program

Description automatically generated

Executing flask run creates the local host where we can view the webpage.

Below is the python connection to the SQL server and the database with the login credentials.

A screen shot of a computer

Description automatically generated

After connecting to the database using pyodbc we use cursor.execute() and a select all sql code to pull in the events view we created and assigned that to the variable “data”

A screen shot of a computer program

Description automatically generated

HTML Code: We created an html script to be published through the web application. We designed the page to have SU colors, attractive images showing aspects of the SU campus, an embedded video, and hyper links to other SU webpages. That was the reasonably easy part compared to what we had to do next. Our html script alone had no connection to the database to be able to display any data. To do that we had to establish a python connection but before that we needed to create a place to hold the data from the database. Using a for loop we go through the variable “data” and assign each column of the row to an h tag to be displayed. We displayed the available events as a list.

A screenshot of a computer program

Description automatically generated

App Services: On the deployment process using Visual Studio Code (VS Code) for our web application, we experienced a streamlined and efficient workflow. The process began with setting up our development environment in VS Code, which is well-equipped for handling a variety of programming languages and frameworks. After writing our Python code, which integrated the `pyodbc` and `Flask` libraries, we moved towards configuring our Azure environment.  
Using the Azure extension in VS Code, we were able to easily manage our cloud resources. This integration allowed for a seamless transition from coding to deployment. We set up an Azure WebApp service, a platform that enables the hosting and management of web applications without the complexity of infrastructure management. The WebApp service was chosen for its scalability, reliability, and integration capabilities with Azure's database services.  
Next, we focused on the database aspect. The Azure-hosted database was prepared to store and manage the data our application would use. We utilized an ODBC connection string from the Azure database, which was crucial for establishing a secure and reliable connection between our application and the database.  
Once the application was ready and the database was set up, we deployed our code directly from VS Code to the Azure WebApp service using Azure's continuous deployment options. This feature automatically deploys our code whenever we commit changes, ensuring that our application is always up-to-date.  
Finally, we tested the deployed application to ensure it was running smoothly on the Azure platform. The Flask framework facilitated the running of our HTML code, and the application successfully interacted with the Azure database. Data retrieval and display on the web application were functioning as expected, demonstrating the successful integration of the various technologies and services we utilized.  
In summary, the deployment process using Visual Studio Code provided a cohesive and efficient pathway from development to deployment, fully leveraging Azure’s cloud capabilities and services. The result was a robust, scalable web application with a reliable database connection, hosted on a platform that ensures high availability and performance.

Issues encountered:

* The first issue we had was the database connection. We created one database and server so in order for us to work on it together we had to share the connection, which was difficult because of the firewall exception and our ip addresses constantly changing. We decided to create a second database and server to solve connectivity issues.
* The second struggle we had was the code intensive approach we took. At the end of the day it seemed like a cool and fun way to leverage the capabilities of cloud tools but after a while it felt like a chore especially having to extensively research into coding languages like python and html to learn hoe to implement the functionality that we required. This also took up a large chunk of our time.
* In creating the azure app service we mistakenly created it in a separate resource group from the one which had the database and sql server, so when it came to deploying we had a few errors because the app service couldn’t find the database we asked it to connect to. We fixed this issue by using a tcp in the python code that had the server connection.

A screen shot of a computer

Description automatically generated

In doing that it told the app to go to a different resource group for the sql server.

* We also wanted to incorporate a search engine that would allow users to type in a keyword and then bring up events that match that key word using Azure intelligent recommendations. Not only would this have been a difficult python and html code to write but too tedious to incorporate into our architecture because intelligent recommendations had its own prerequisites to run that would have clashed with our infrastructure.