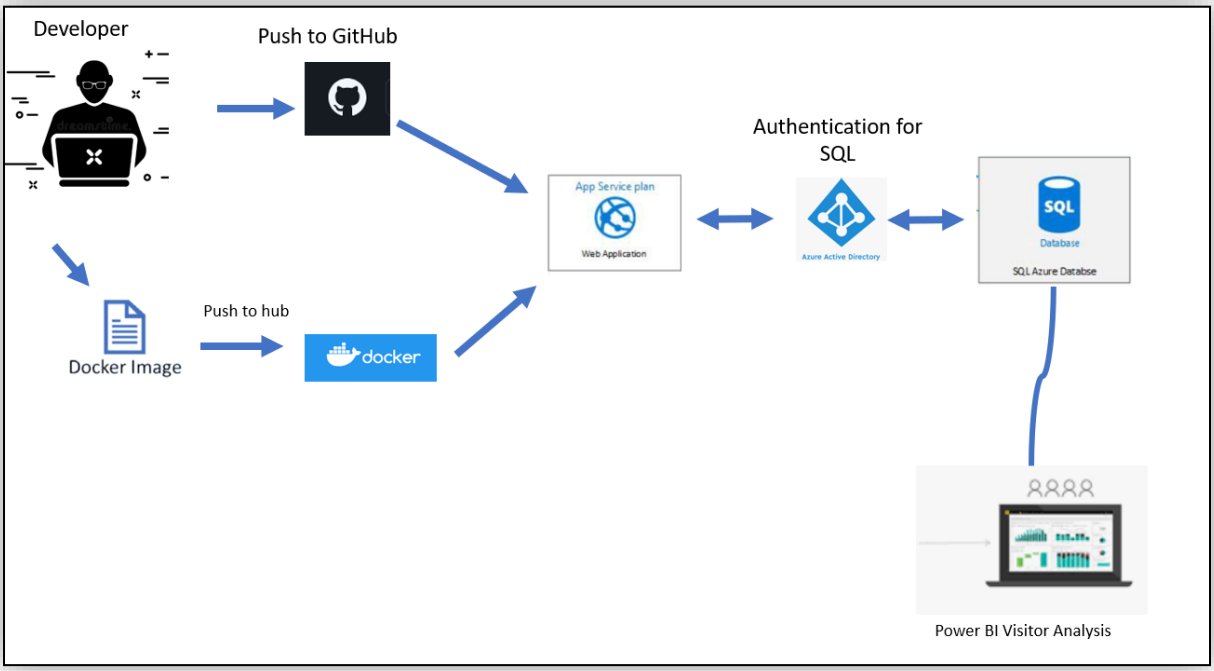
Developing Reception Voice Assistant and Visitors Analytics using (Azure SQL ,Power BI, Active Directory ,Azure Web App & Docker)

Problem statement –

Traditional reception systems lack personalized and engaging experiences for visitors entering establishments such as malls, hospitals, and restaurants. Additionally, there is a need for efficient tracking and analysis of visitor data to enhance customer service and make informed business decisions..

Proposed Solution: The proposed solution is a web application that utilizes user input to generate a personalized welcome message with the visitor's name. The application will capture visitor details, such as name, arrival time, and purpose of visit, and store them in a database for tracking and analysis purposes. It will also provide features to generate reports, visualize visitor data, and extract insights to improve business operations.

Solution Architecture Diagram :-



Let's look into technical details and the Implementation of the solution:

You need the following software/ Azure Account, please find Following are the Azure service used to create the solution:

- **Storage Account**
- **WebApp service**
- **Application insight**
- **Azure Active Directory (for security)**
- **SQL Database**
- **Power BI Desktop**
- **Docker**

Azure Account:- get an azure account by clicking on the following link
https://azure.microsoft.com/en-us/free/

There is free credit for students and 200 USD credit if you want to get started with Azure
https://azure.microsoft.com/en-us/free/students/

Visual Studio 2019 Community
https://visualstudio.microsoft.com/downloads/

The community edition is free for students and open-source contributors for non-commercial use.
Visual Studio Code (Optional if you have Visual Studio 2019 for Azure Development)
https://code.visualstudio.com/download

PROJECT DEPLOYED LINK:

https://receptionistvoice.azurewebsites.net/

Github Link:

https://github.com/mohdp6728/blogathonproject

Docker Link:

https://hub.docker.com/repository/docker/mohd6728/docker

Required Packages :

Azure identiy
Playsound and GTTs- for voice libraries

Technologies : Python FLASK

Driver code :

```
from azure.identity import DefaultAzureCredential
from flask import Flask, render_template, request
import pyodbc
from gtts import gTTS
import os
import subprocess
# from azure.identity import DefaultAzureCredential
# from azure.keyvault.secrets import SecretClient

app = Flask(__name__)

# Connect to Azure SQL DB
server = 'boomlet.database.windows.net'
database = 'boomlet'
username = 'maddy'
password = 'Viratkohli18'

# password = 'Paneer@1234'
driver = '{ODBC Driver 17 for SQL Server}'
# Use Azure AD authentication
credential = DefaultAzureCredential()
# cnxn =
pyodbc.connect(f'DRIVER={driver};SERVER={server};DATABASE={database};UID={user
name};PWD={password}')
```

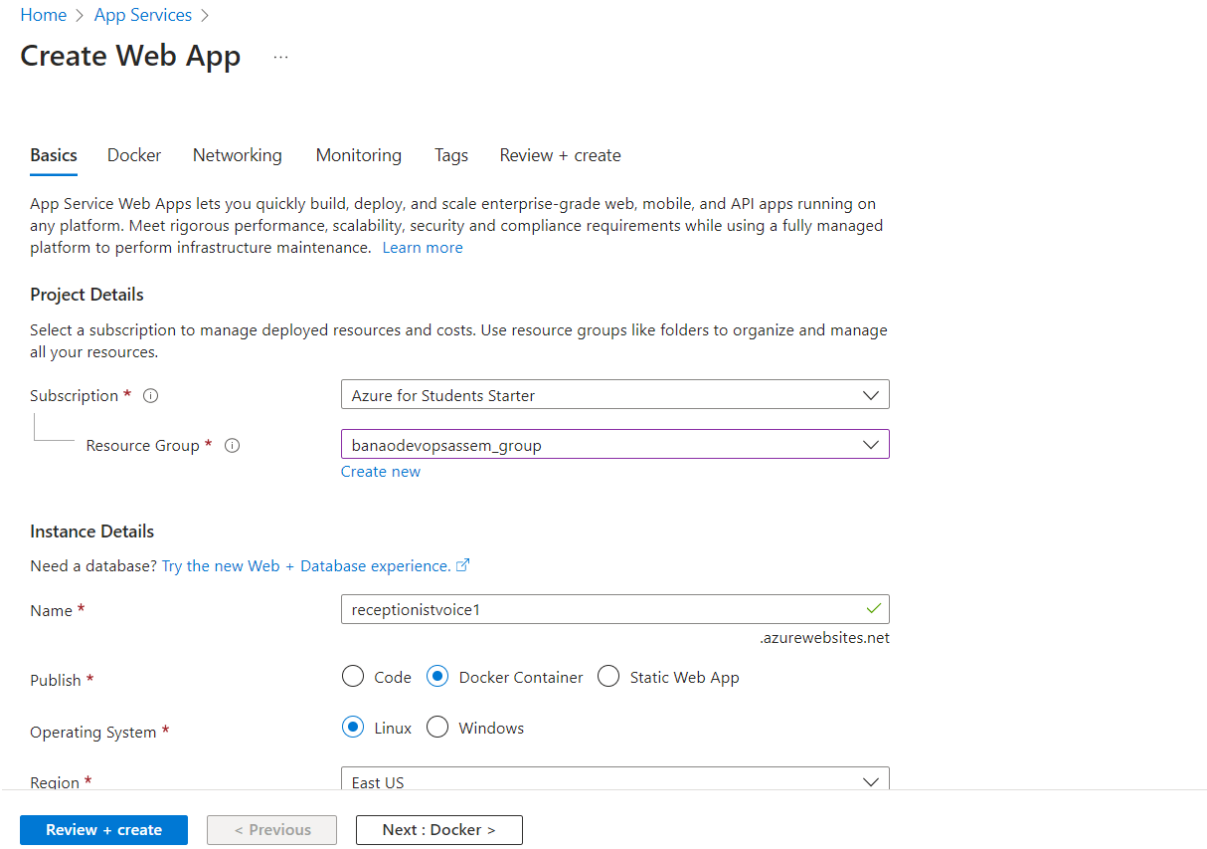
```
cnxn =
pyodbc.connect(f'DRIVER={driver};SERVER={server};DATABASE={database};Authentic
ation=ActiveDirectoryPassword;UID=maddy@trentboult446gmail.onmicrosoft.com;PWD
={password}')

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        return submit()
    return render_template('index.html')

@app.route('/submit', methods=['POST'])
def submit():
    full_name = request.form.get('full_name')
    purpose = request.form.get('purpose')
    number = request.form.get('number')
    welcome_message = f"Welcome {full_name} to BoomLet Media"

    # Insert user details into boomlet table
    cursor = cnxn.cursor()
    insert_query = f"INSERT INTO users (full_name, purpose, number) VALUES
('{full_name}', '{purpose}', '{number}')"
    cursor.execute(insert_query)
    cnxn.commit()

    # Generate audio file using gTTS
    tts = gTTS(welcome_message)
    tts.save(f"static/{full_name}.mp3")

    return render_template('success.html', full_name=full_name,
purpose=purpose, number=number)

@app.route('/database', methods=['GET'])
def database():
    # Select all records from boomlet table
    cursor = cnxn.cursor()
    select_query = "SELECT * FROM users"
    cursor.execute(select_query)
    rows = cursor.fetchall()

    return render_template('database.html', rows=rows)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=False)
```

Steps to Create and Connect Azure WebApp and Docker:

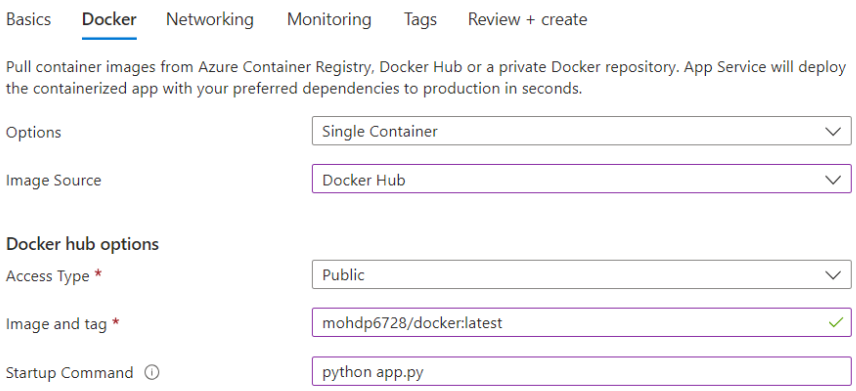
Step 1: Create an WebApp

Azure Web App is a fully managed platform for hosting and deploying web applications. It provides a scalable and secure environment to run your web applications without worrying about infrastructure management.

NOTE : We can either Deploy our app by using Docker or Github Actions here we choose Docker



Next we configure the docker Settings like startup command and source url



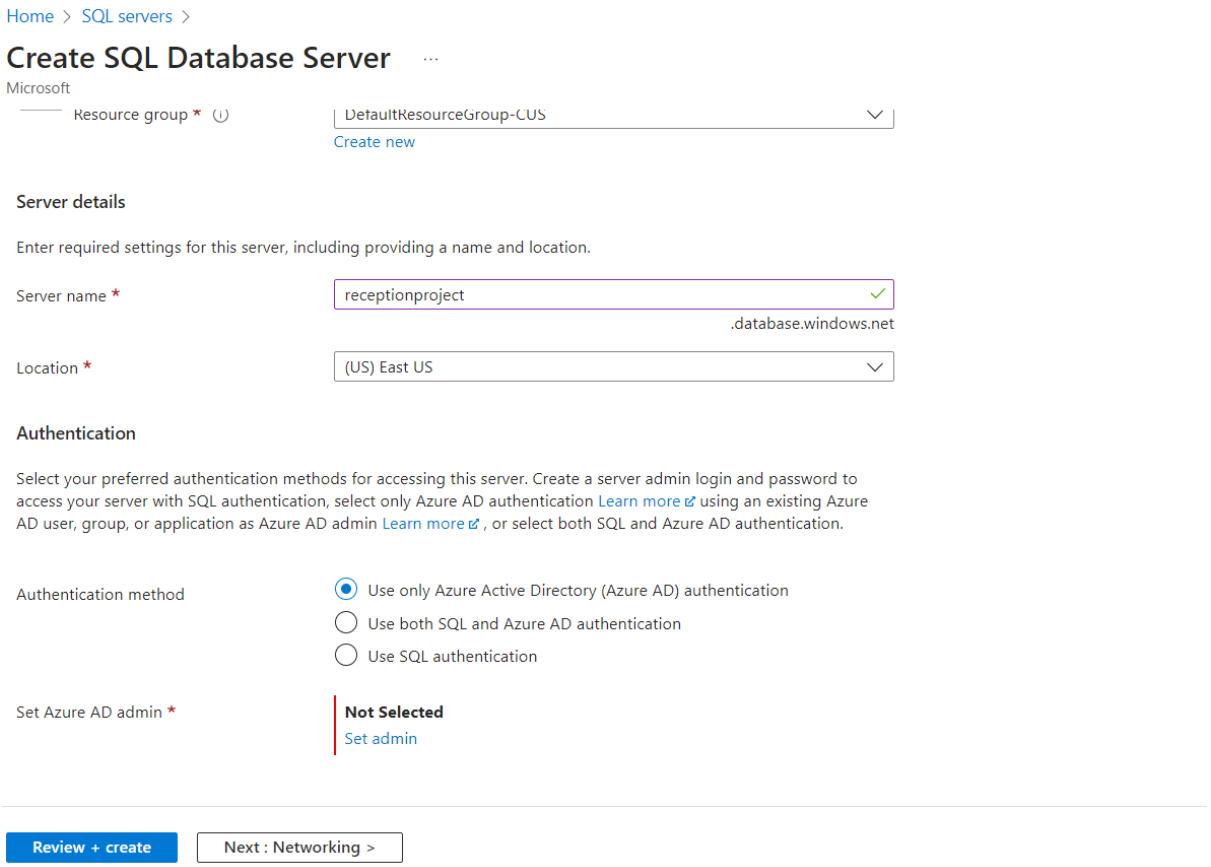
Step 2: Creating Azure SQL DB and Azure Active Directory

SQL DB with Azure Active Directory integration allows you to leverage Azure Active Directory for authentication and authorization in your SQL database. It provides enhanced security and centralized access control for your database, allowing you to manage user identities and permissions more effectively.

Query to schema :

```
CREATE TABLE [dbo].[users](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [full_name] [varchar](50) NULL,
    [purpose] [varchar](100) NULL,
    [number] [varchar](20) NULL,
    [gender] [varchar](10) NULL,
    [date] [date] NULL,
    [time] [time] NULL
)
```

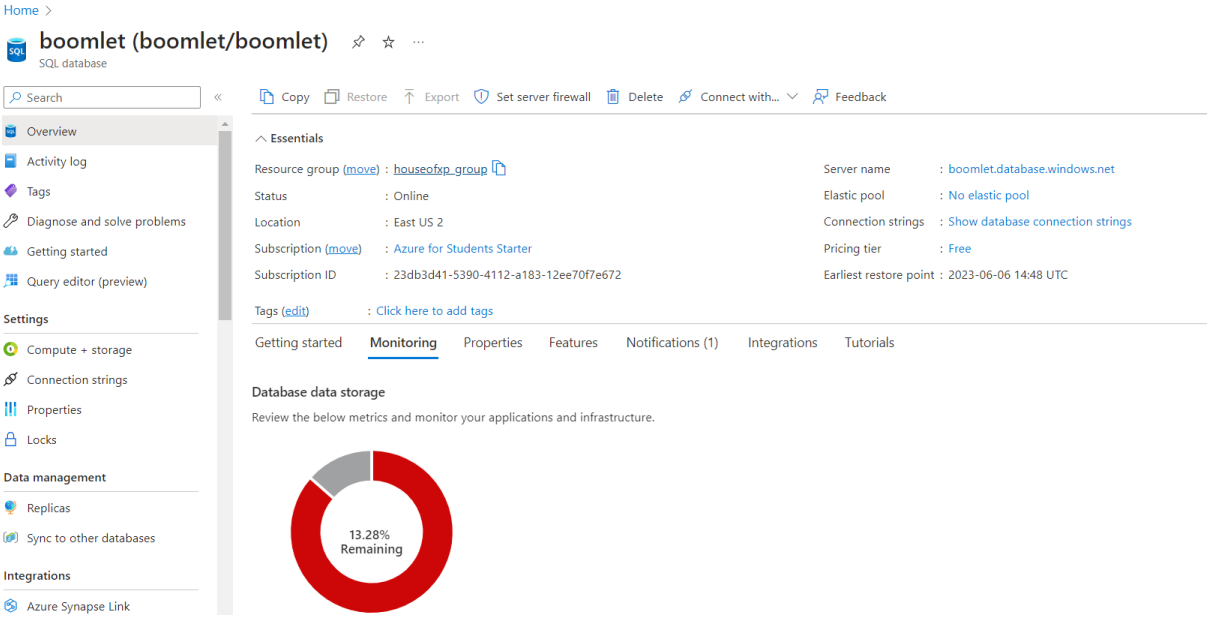
SQL SERVER:



Firewall rules

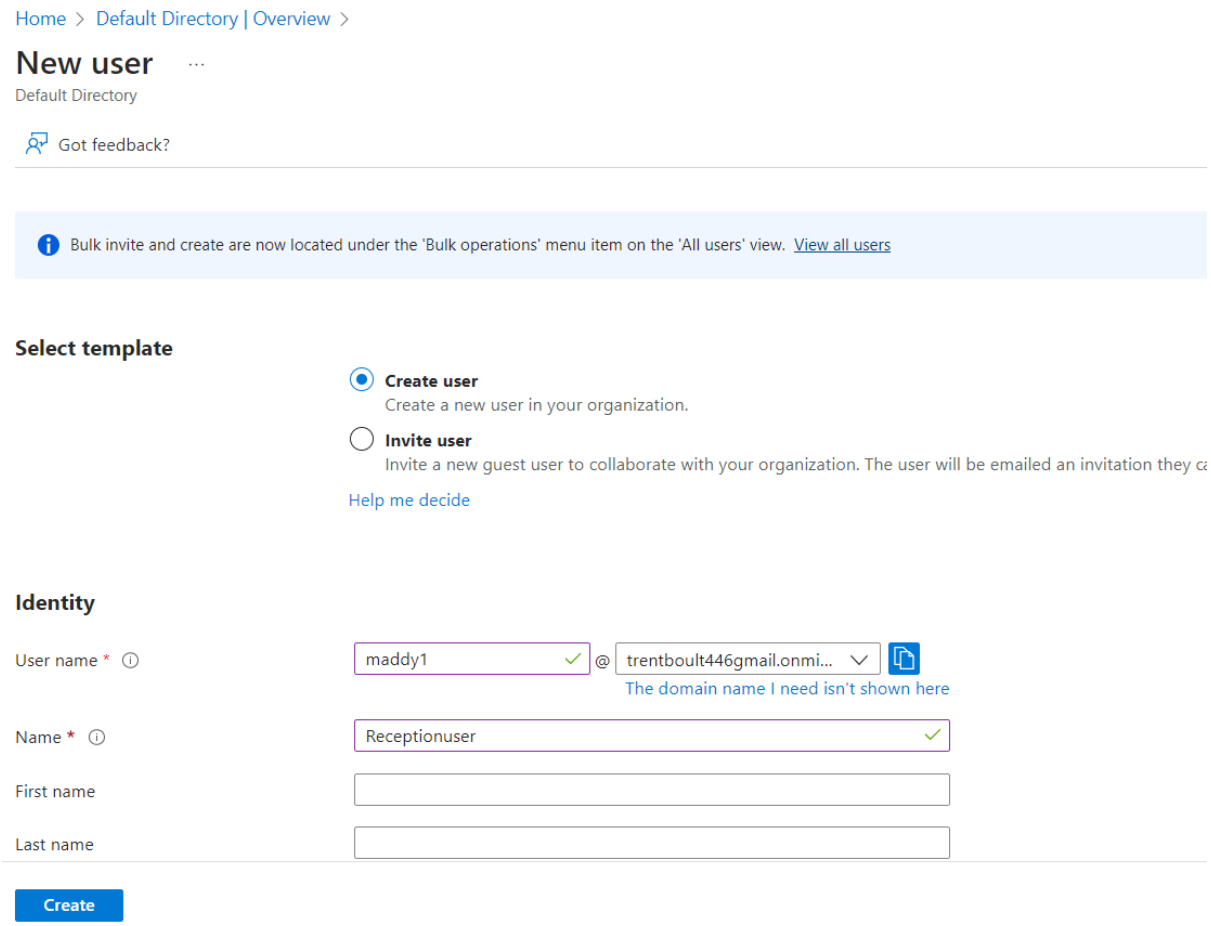
Allow Azure services and resources to access this server
Yes

SQL DB:

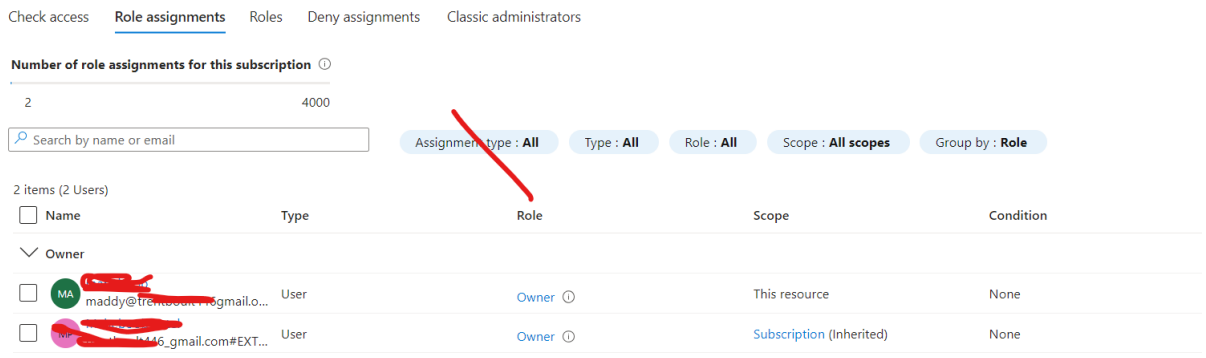


STEP 3 : Azure Active Directory Configuration:

In the Azure portal, go to "Azure Active Directory."
Click on "Users" and then "New user."
Fill in the required information to create a new user.
Make sure to assign a username and password for the user.
Note down the username (user principal name) and password for future reference.

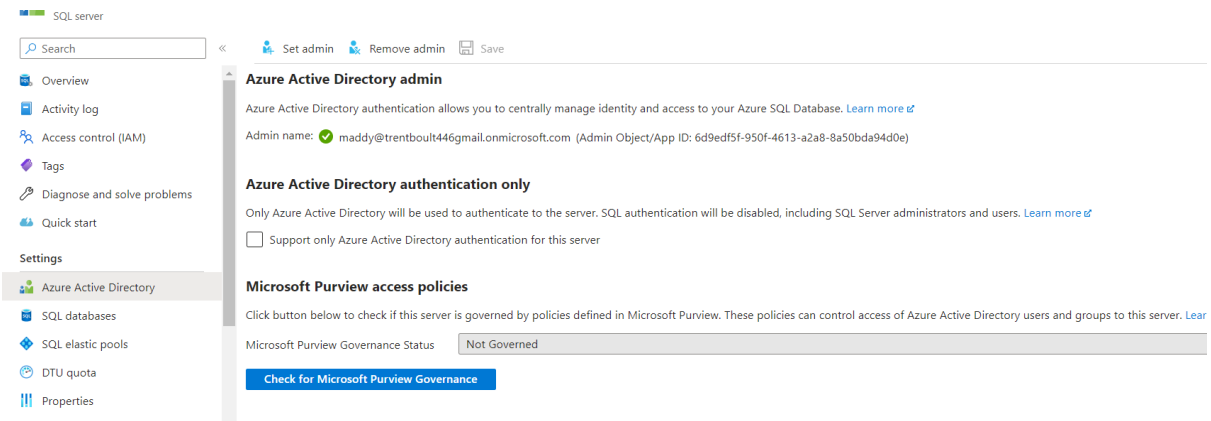


Assign the user appropriate permissions to the Azure SQL DB:



In the Azure portal, go to the Azure SQL DB resource.
Click on "Access control (IAM)" in the left-hand menu.
Click on "Add" and then "Add role assignment."
Select the appropriate role (e.g., "Contributor" or "SQL Server Contributor").
In the "Select" field, search and select the user you created in the previous step.
Click on "Save" to assign the role to the user.
Configure Azure SQL DB for Azure AD authentication:

In the Azure portal, go to the Azure SQL DB resource.
Click on "Firewalls and virtual networks" in the left-hand menu.
Ensure that "Allow Azure services and resources to access this server" is set to "Yes" to allow access from the App Service.
Click on "Active Directory admin" in the left-hand menu.
Click on "Set admin" and select the user you created in step 2 as the admin.
Click on "Save" to set the admin.



Modify the connection string in your Flask application:

Remove the existing username and password variables from your code.
Add a new variable for the Azure AD tenant ID at the beginning of your code:
python
Copy code
tenant_id = '<your-azure-ad-tenant-id>'
Modify the connection string in the cnxn initialization to include the Authentication=ActiveDirectoryPassword parameter:
python
Copy code
cnxn =
pyodbc.connect(f'DRIVER={driver};SERVER={server};DATABASE={database};UID={username}@{tenant_id};Authentication=ActiveDirectoryPassword')

STEP 4: DOCKER BUILD

Docker with Azure: Docker with Azure enables you to run containerized applications on the Azure platform. It provides a flexible and scalable environment for deploying and managing containers, allowing you to easily package your applications with their dependencies and run them consistently across different environments. Docker with Azure simplifies the deployment and management of containerized applications in the cloud

We can use docker build -t imagename

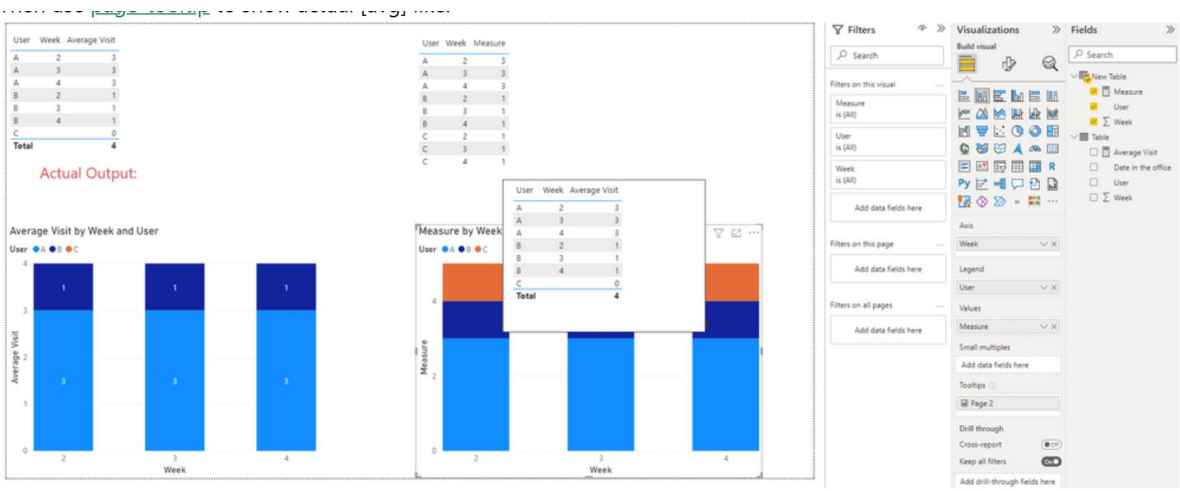
Here is the code for docker file

```
app.py M   success.html M   index.html M   Dockerfile ●   requirements.txt ●
Dockerfile > ...
# Use the official Python base image
FROM python:3.9-slim

# Set the working directory
WORKDIR /app
# Copy and install pip requirements
COPY requirements.txt .
RUN python -m pip install --no-cache-dir --upgrade pip
# Install system dependencies
RUN apt-get update && apt-get install -y --no-install-recommends \
    unixodbc-dev \
    gcc \
    g++ \
    curl \
    gnupg2 \
    && rm -rf /var/lib/apt/lists/*
# Install Microsoft ODBC Driver for SQL Server
RUN curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add -
RUN curl https://packages.microsoft.com/config/debian/10/prod.list > /etc/apt/sources.list.d/mssql-release.list
RUN apt-get update && ACCEPT_EULA=Y apt-get install -y msodbcsql17
RUN python -m pip install --no-cache-dir -r requirements.txt

# Copy the application files
COPY . /app
EXPOSE 5000

# Set the default command to run the application
CMD ["python", "app.py"]
```

And than after successful build :
We can tag and push the image to docker hub:

```
mohdp6728/docker            latest1   fb99a10451c1   3 weeks ago     481MB
PS C:\Users\trent\OneDrive\Documents\New folder\boomlet> docker tag newdocker mohdp6728/docker:latest
PS C:\Users\trent\OneDrive\Documents\New folder\boomlet> docker push  mohdp6728/docker:latest
The push refers to repository [docker.io/mohdp6728/docker]
bde87955fd9a: Preparing
8767aa780040: Preparing
```

STEP 5: further the collected data can be used for visualizing the data using Power BI :



Challenges Faced:

- **Azure Configuration: Setting up and configuring Azure services, understanding authentication methods and security considerations.**
- **Dependencies and Packages: Managing dependencies, resolving version conflicts, and ensuring compatibility between packages.**
- **Integration and Communication: Integrating components, establishing secure communication channels.**
- **Scalability and Performance: Handling concurrent visitors, optimizing queries, and managing resource allocation.**
- **Security and Privacy: Safeguarding visitor data, ensuring secure authentication, and complying with privacy regulations.**

Business Benefits:

- **Enhanced Customer Experience: Improved customer satisfaction by providing personalized welcome messages and tracking visitor details for better engagement and analysis.**
- **Efficient Visitor Management: Streamlined visitor registration process, eliminating manual paperwork, and enabling easy access to visitor information.**
- **Data Analysis and Insights: Leveraging visitor data for analyzing patterns, identifying trends, and making informed business decisions for optimizing operations.**
- **Versatile Application Usage: Applicable across various industries like malls, hospitals, and restaurants, catering to diverse reception needs and enhancing brand image.**

– By Sahil Shaikh
Senior Developer