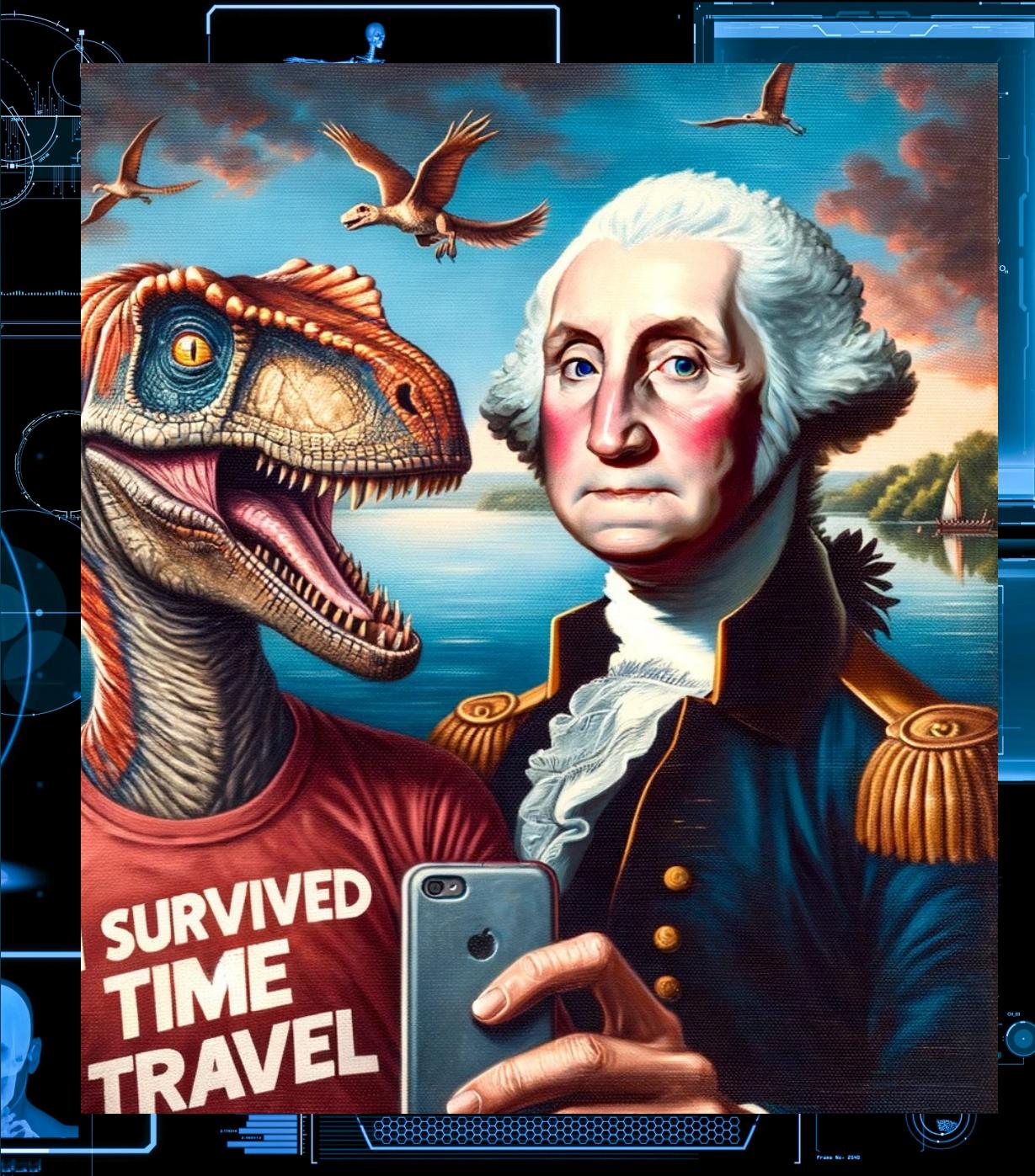


Team 3

# DALL-E 3 Prompt Generator

Jalesa Jones  
Adrian Torres  
Shintaro Osuga  
Samantha Zygmont



# Overview

---

- What is DALL-E
- Statement of Purpose
- Stakeholder Benefits & Applications
- Key Features of the Database
- Product Work-Flow
- Resources & Tools Involved
- Database Architecture
- SQL Code & Explanation
- Python Code Explanation
- Conclusion & Reflection





# What is DALL-E?

---

- DALL-E is an AI art generation model which uses user given prompts to create unique art
- The name DALL-E stems from the combination of famous Spanish artist Salvador Dali and Pixar character WALL-E

# Statement of Purpose

---





# Problem Statement

---

- Currently, when generating an image on the DALL·E 3 system, there is a lack of predictability and consistency between images causing prompt engineers to struggle to recreate images in the same style as before. The solution proposed is to create a database application capable of saving prompts and displaying visual cues per keyword or completed prompt. In the application the keywords stay constant while everything around those keywords can be varied to create unique images while maintaining art style consistency.

# Scope & Boundary of Product

---

## Within Boundary

- Providing Prompts for Engineer

## Outside Boundary

- Providing AI for Generation



# Stakeholder Benefits & Applications

---



# Stakeholders & Benefits

---

## **Stakeholders**

- DBMS/Script Developers & Maintainers
- Businesses
- Individuals
- R&D teams

## **Benefits**

- Organized image-prompt database
- Ability to create and re-create images with higher confidence
- Ability to share and gain input and ideas
- Curate and hone prompt engineering

# Applications

---

## Commercial Apps

- Social Media Like
- Many Users, Less Quality

## Professional Apps

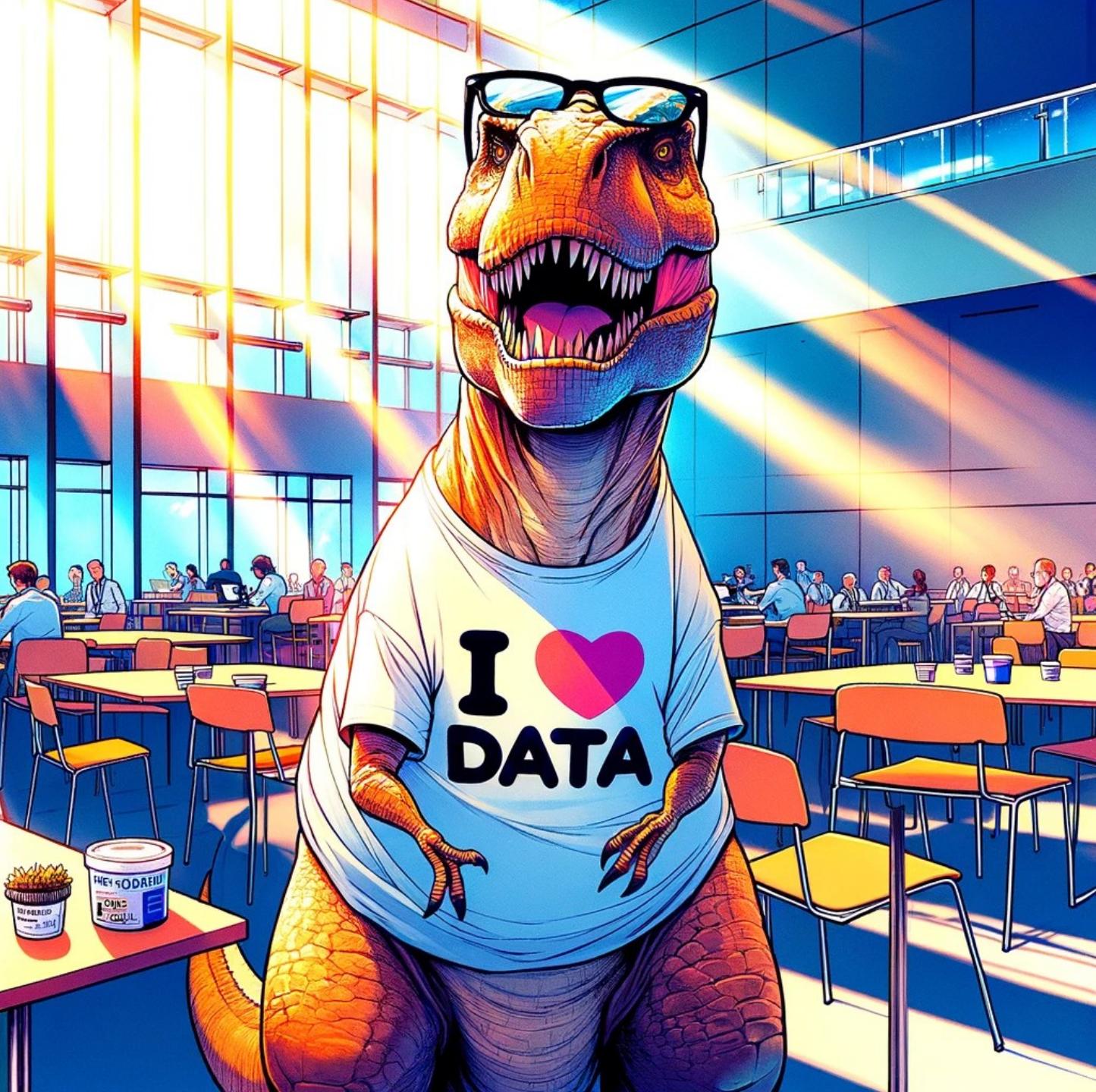
- Focused on Prompt Engineering
- Less Users, More Quality

## Research

- Focused on Prompt Engineering, Reverse Engineering
- Even Less Users, Even More Quality

# Key Features

---



# Key Features

---

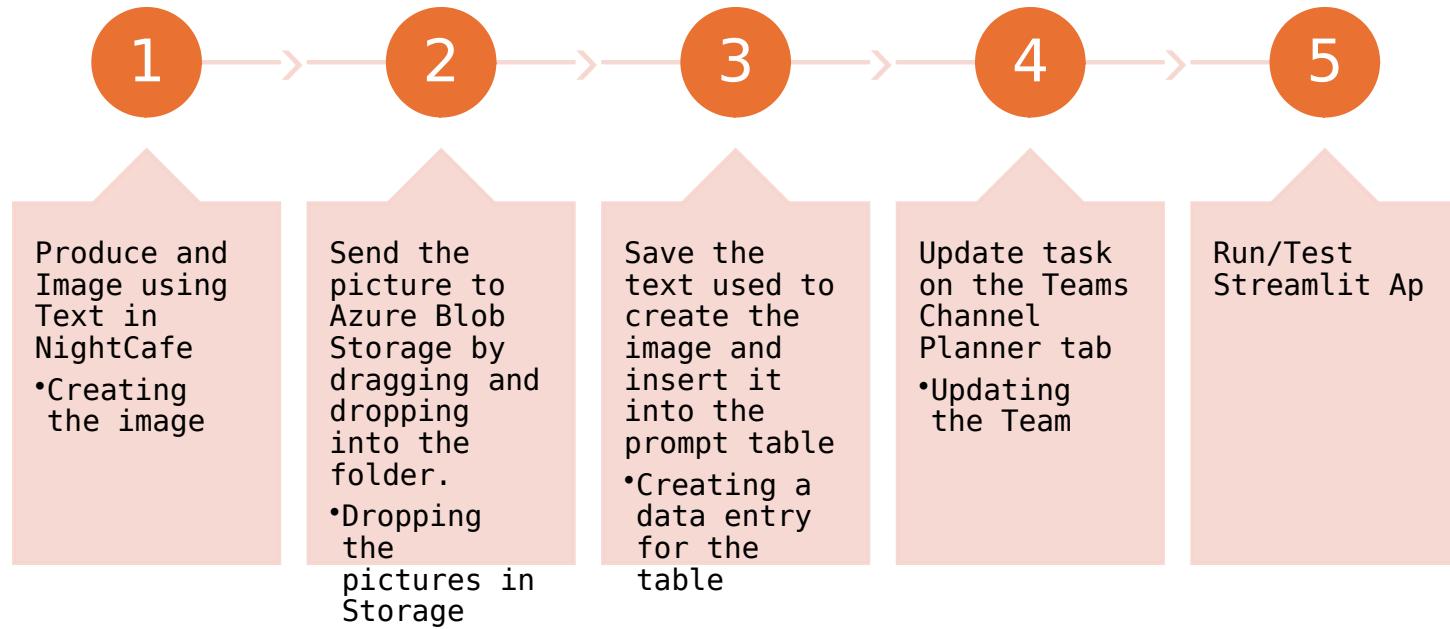
## Business Features (Requirements)

- A Database which stores image to prompt information which can be retrieved and references easily.
- A user-friendly platform for users to upload, look up, and recreate images from prompts.

## Functional Features (Requirements)

- Image to prompt search
- Prompt to image search
- Categorical search
- Rating and commenting on image-prompts

# Project Workflow



# Azure SQL Server and Database

The screenshot shows the Azure portal interface for managing an Azure SQL Server. The main title bar indicates the server name is '24'. The left sidebar contains a navigation menu with sections such as Overview, Activity log, Access control (IAM), Tags, Quick start, Diagnose and solve problems, Settings (including Microsoft Entra ID, SQL databases, SQL elastic pools, DTU quota, Properties, Locks), Data management (Backups, Deleted databases, Failover groups, Import/Export history), Security (Networking, Microsoft Defender for Cloud, Transparent data encryption, Identity, Auditing), Intelligent Performance (Automatic tuning, Recommendations), and Monitoring.

**Essentials**

- Resource group ([move](#)) : DefaultResourceGroup-EUS
- Status : Available
- Location : East US
- Subscription ([move](#)) : Azure for Students
- Subscription ID : 9e04def2-c7e5-4b28-9485-90daa0d0e1fe
- Tags ([edit](#)) : [Add tags](#)
- Notifications (0)
- Features (6) [All](#) [Security \(4\)](#) [Performance \(1\)](#) [Recovery \(1\)](#)

**Configured Features:**

- Microsoft Entra admin**: Allows you to centrally manage identity and access to your Azure SQL databases. Status: CONFIGURED
- Microsoft Defender for SQL**: Vulnerability Assessment and Advanced Threat Protection. Status: NOT CONFIGURED
- Automatic tuning**: Monitors and tunes your database automatically to optimize performance. Status: CONFIGURED
- Auditing**: Track database events and writes them to an audit log in Azure storage. Status: NOT CONFIGURED
- Transparent data encryption**: Encryption at rest for your databases, backups, and logs. Status: SERVICE-MANAGED KEY
- Failover groups**: Automatically manages replication, connectivity and failover for a set of databases. Status: NOT CONFIGURED

**Available resources**

Name	Type	Status	Pricing tier
First_DB	SQL database	Online	General Purpose - Serverless: Standard-series (Gen5), 2 vCores

# Azure Blob Storage

Home > dalle3storage | Containers >

2244 Container

Search  X <>

Upload Change access level Refresh Delete Change tier Acquire lease Break lease View snapshots Create snapshot Give feedback

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Shared access tokens

Access policy ★

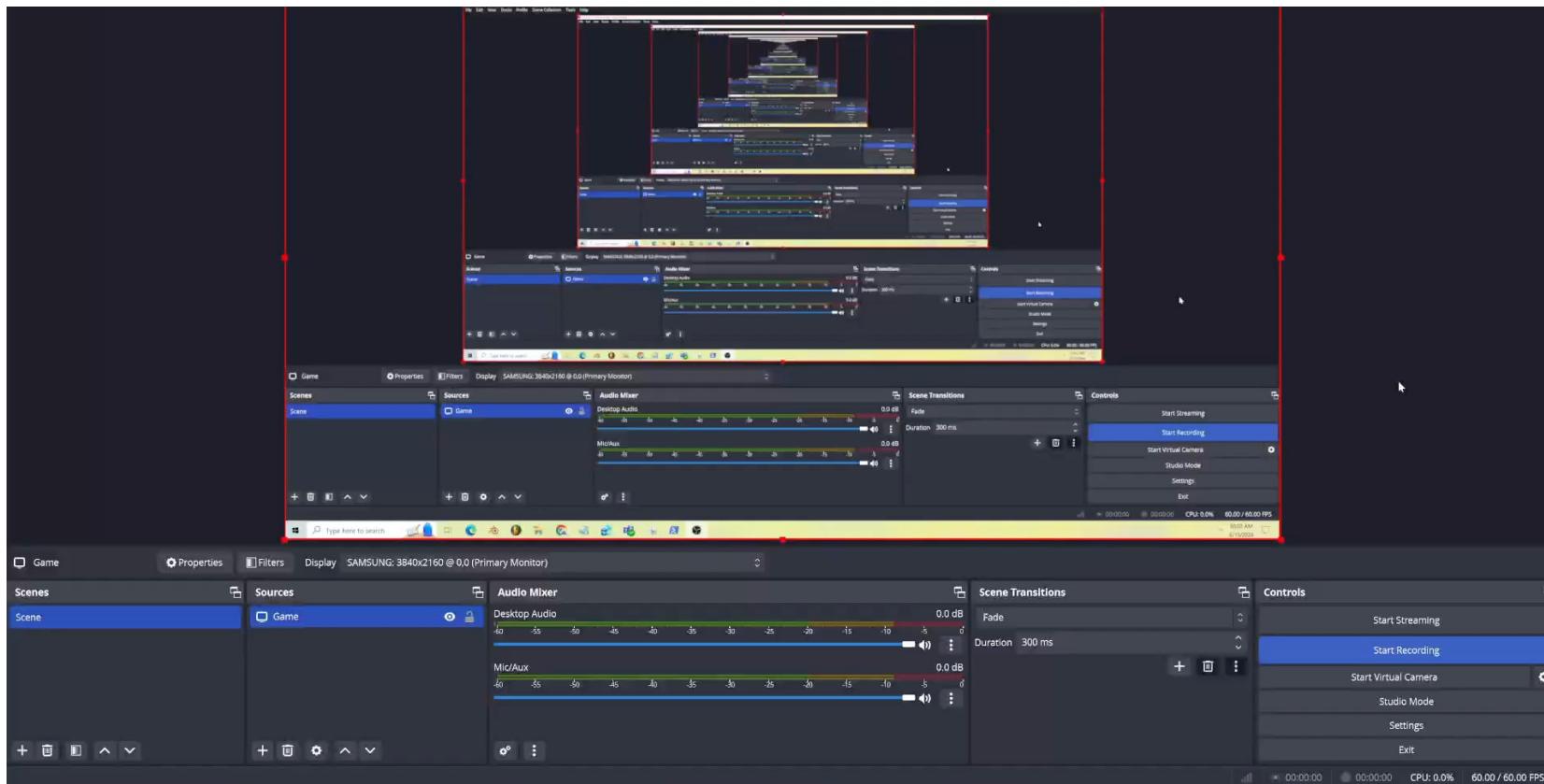
Properties

Metadata

File	Last Modified	Tier	Type	Size	Status	Actions
Prompt_ID_1.jpg	6/8/2024, 12:47:25 PM	Cool (Inferred)	Block blob	343.45 KiB	Available	...
Prompt_ID_10.jpg	6/8/2024, 12:48:01 PM	Cool (Inferred)	Block blob	225.71 KiB	Available	...
Prompt_ID_11.jpg	5/25/2024, 3:28:17 PM	Cool (Inferred)	Block blob	23.16 KiB	Available	...
Prompt_ID_12.jpg	6/4/2024, 12:58:26 PM	Cool (Inferred)	Block blob	147.08 KiB	Available	...
Prompt_ID_13.jpg	6/4/2024, 12:58:26 PM	Cool (Inferred)	Block blob	288.96 KiB	Available	...
Prompt_ID_14.jpg	6/4/2024, 12:58:26 PM	Cool (Inferred)	Block blob	178.52 KiB	Available	...
Prompt_ID_15.jpg	6/4/2024, 12:58:26 PM	Cool (Inferred)	Block blob	147.52 KiB	Available	...
Prompt_ID_16.jpg	6/4/2024, 12:58:26 PM	Cool (Inferred)	Block blob	206.93 KiB	Available	...
Prompt_ID_17.jpg	6/4/2024, 12:58:27 PM	Cool (Inferred)	Block blob	288.55 KiB	Available	...
Prompt_ID_18.jpg	6/4/2024, 12:58:27 PM	Cool (Inferred)	Block blob	273.19 KiB	Available	...
Prompt_ID_19.jpg	6/4/2024, 12:58:27 PM	Cool (Inferred)	Block blob	172.04 KiB	Available	...
Prompt_ID_2.jpg	6/4/2024, 1:02:18 PM	Cool (Inferred)	Block blob	473.64 KiB	Available	...
Prompt_ID_20.jpg	5/25/2024, 3:28:17 PM	Cool (Inferred)	Block blob	17.32 KiB	Available	...
Prompt_ID_21.png	6/4/2024, 1:41:35 PM	Cool (Inferred)	Block blob	659.65 KiB	Available	...
Prompt_ID_22.png	6/4/2024, 1:41:35 PM	Cool (Inferred)	Block blob	647.8 KiB	Available	...
Prompt_ID_23.png	6/4/2024, 1:41:35 PM	Cool (Inferred)	Block blob	1.02 MiB	Available	...
Prompt_ID_24.png	6/4/2024, 1:41:35 PM	Cool (Inferred)	Block blob	674.98 KiB	Available	...
Prompt_ID_25.png	6/4/2024, 1:41:35 PM	Cool (Inferred)	Block blob	507.92 KiB	Available	...
Prompt_ID_26.png	6/4/2024, 1:41:35 PM	Cool (Inferred)	Block blob	389.64 KiB	Available	...
Prompt_ID_27.png	6/4/2024, 1:41:35 PM	Cool (Inferred)	Block blob	767.92 KiB	Available	...
Prompt_ID_28.png	6/4/2024, 1:41:36 PM	Cool (Inferred)	Block blob	884.76 KiB	Available	...
Prompt_ID_29.png	6/4/2024, 1:41:35 PM	Cool (Inferred)	Block blob	596.87 KiB	Available	...
Prompt_ID_3.jpg	6/4/2024, 1:02:18 PM	Cool (Inferred)	Block blob	467.63 KiB	Available	...
Prompt_ID_30.png	6/4/2024, 1:41:35 PM	Cool (Inferred)	Block blob	538.11 KiB	Available	...
Prompt_ID_31.jpeg	6/4/2024, 1:18:52 PM	Cool (Inferred)	Block blob	385.16 KiB	Available	...
Prompt_ID_32.jpeg	6/4/2024, 1:18:52 PM	Cool (Inferred)	Block blob	284.91 KiB	Available	...
Prompt_ID_33.jpeg	6/4/2024, 1:18:52 PM	Cool (Inferred)	Block blob	313.39 KiB	Available	...

Load more

# App Demonstration



# Resources and Tools Involved

Azure Blob Storage

Picture Storage

Azure Data Studio

SQL Queries & Table Creation

Python

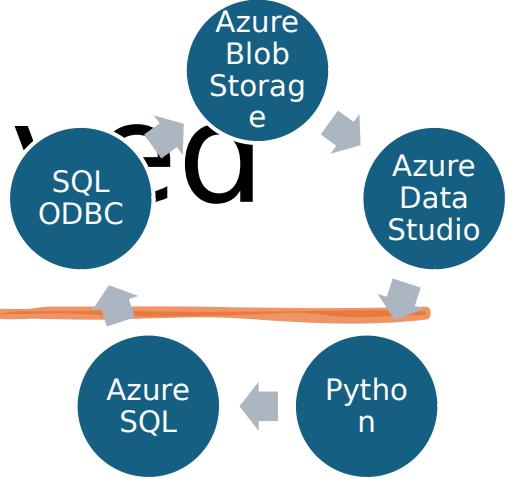
Streamlit App Creation

Azure SQL

Database Residence

SQL ODBC

The Bridge to our Database



# Database Architectu re

---



# Tables

---

## Prompts

Prompt_ID
Prompt_Category
Prompt_Tags
Prompt_Creation_Date
Prompt_Creator_User_ID

## Users

User_ID
User_Username
User_Firstname
User_Lastname
User_Email
User_Registration_Date

## Images

Image_ID
Image_Prompt_ID
Image_Path

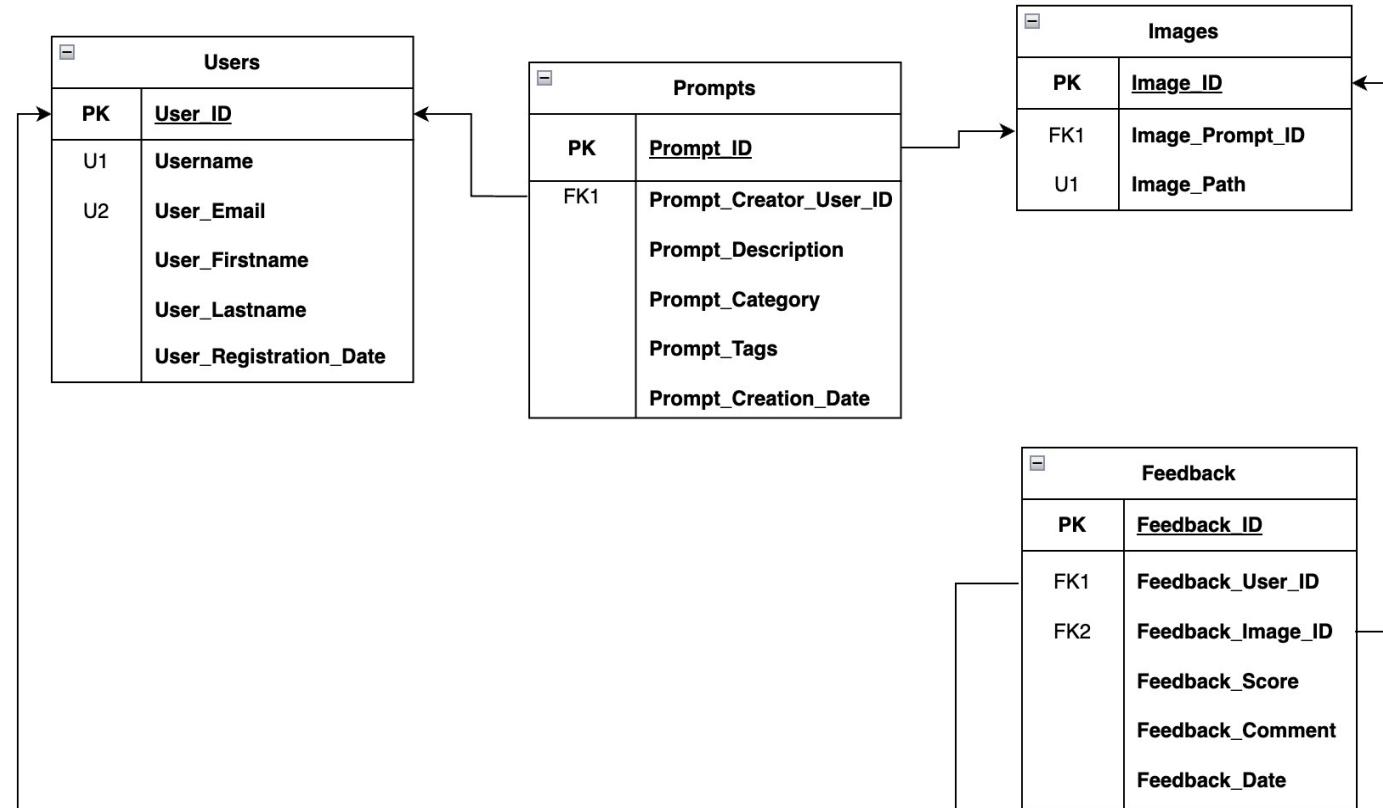
## User\_Feedback

Feedback_ID
Feedback_Image_ID
Feedback_User_ID
Feedback_Score
Feedback_Comment
Feedback_Date

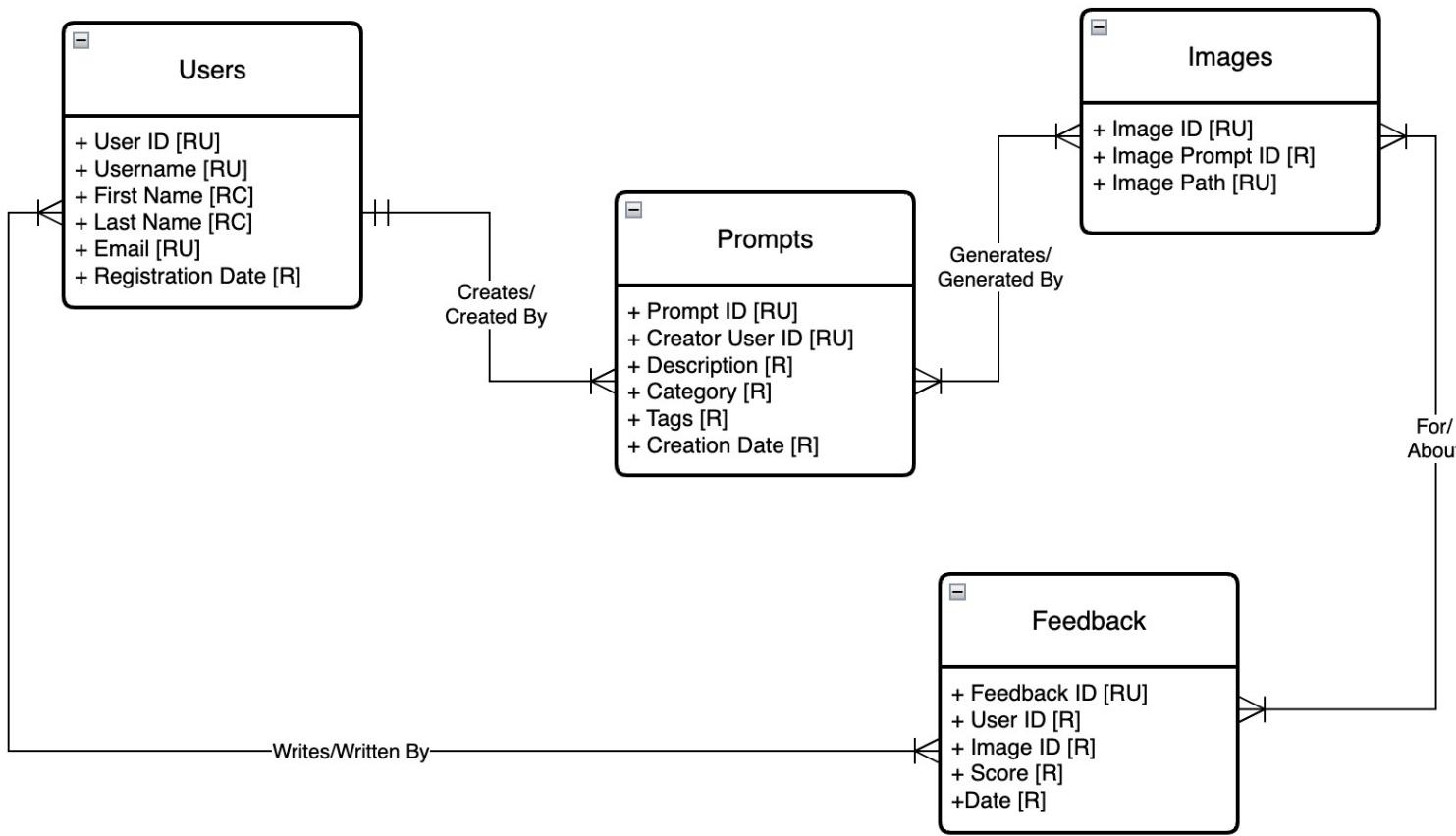
# Logical Model

The logical model to the side shows the relationships between the attributes and tables.

The tables are mostly connected via foreign keys which allow for quick referencing of related records in other tables.



# Conceptual Model



The conceptual model to the left shows the type of connection each table has with one another.

Most of the relationships between tables are Many-to-Many except for the Users to Prompts which is a One-to-Many relationship.

# SQL Code

---



# SQL Tables

```
create table Users (
    User_ID int identity not null,
    Username varchar(50) not null,
    User_Firstname varchar(50) not null,
    User_Lastname varchar(50) not null,
    User_Email varchar(50) not null,
    User_Registration_Date date not null,
    constraint pk_User_ID primary key (User_ID),
    constraint u_User_Email unique (User_Email),
)
GO
```

User_ID	Username	User_Firstname	User_Lastname	User_Email	User_Registration_Date
1	PennyWiseGuy	Penny	Wise	PennyWise@yahoo.com	2024-01-12
2	CadeCrusader	Barry	Cade	BarryCade@gmail.com	2024-02-23
3	AnaCondaRules	Anna	Conda	Annaconda@hotmail.com	2024-03-08
4	PollyMorphic	Polly	Mer	PollyMer@yahoo.com	2024-04-17
5	SandyShores	Sandy	Beach	SandyBeach@gmail.com	2024-05-01
6	PhilharmonicFunk	Phil	Harmonic	PhilHarmonic@hotmail.com	2024-01-19
7	WillPowered	Will	Power	WillPower@yahoo.com	2024-02-03
8	BarbWireQueen	Barb	Dwyer	BarbDwyer@gmail.com	2024-03-28
9	JustinTimeTraveler	Justin	Thyme	JustinThyme@hotmail.com	2024-04-11
10	AlDenteDish	Al	Dente	AlDente@yahoo.com	2024-05-15

# SQL Tables

```
create table Users (
    User_ID int identity not null,
    Username varchar(50) not null,
    User_Firstname varchar(50) not null,
    User_Lastname varchar(50) not null,
    User_Email varchar(50) not null,
    User_Registration_Date date not null,
    constraint pk_User_ID primary key (User_ID),
    constraint u_User_Email unique (User_Email),
)
GO
```

User_ID	Username	User_Firstname	User_Lastname	User_Email	User_Registration_Date
1	PennyWiseGuy	Penny	Wise	PennyWise@yahoo.com	2024-01-12
2	CadeCrusader	Barry	Cade	BarryCade@gmail.com	2024-02-23
3	AnaCondaRules	Anna	Conda	Annaconda@hotmail.com	2024-03-08
4	PollyMorphic	Polly	Mer	PollyMer@yahoo.com	2024-04-17
5	SandyShores	Sandy	Beach	SandyBeach@gmail.com	2024-05-01
6	PhilharmonicFunk	Phil	Harmonic	PhilHarmonic@hotmail.com	2024-01-19
7	WillPowered	Will	Power	WillPower@yahoo.com	2024-02-03
8	BarbWireQueen	Barb	Dwyer	BarbDwyer@gmail.com	2024-03-28
9	JustinTimeTraveler	Justin	Thyme	JustinThyme@hotmail.com	2024-04-11
10	AlDenteDish	Al	Dente	AlDente@yahoo.com	2024-05-15

# SQL Tables

---

## DALL·E 3 Prompt Saver



Image ID: 15 - "Serene, Majestic Mountain Peaks  
Glowing in Moonlight: Snow-capped, Pine Trees  
Alpine Wilderness Impressionist Cool Tones: Blues,  
Greens, Purples Oil Painting"

Average Score for Image 15: 4.80

Comments:

I could get lost in this scenery.

Generate Prompt and Image

```
score_query = f"SELECT AVG(Feedback_Score) AS AvgScore FROM User_Feedback WHERE Feedback_User_ID = {image_id}"
```

```
comment_query = f"SELECT Feedback_Comment FROM User_Feedback WHERE Feedback_Image_ID = {image_id}"
```

# Python Code

---



# Connecting to the DB

---

```
def connect_to_db():
    # Use the environment variable directly in the connection string with f-string formatting
    db_password = os.getenv('DB_PASSWORD')  # Correctly use the same key as defined in the .env file
    conn_str = f"""
        Driver={{ODBC Driver 18 for SQL Server}};
        Server=tcp:24.database.windows.net,1433;
        Database=First_DB;
        Uid=AT;
        Pwd={db_password};
        Encrypt=yes;
        TrustServerCertificate=no;
        ConnectionTimeout=30;
    """
    try:
        conn = pyodbc.connect(conn_str)
        logging.info("Database connection successful")
        return conn
    except Exception as e:
        logging.error("Error connecting to database", exc_info=True)
        st.error(f"Failed to establish database connection: {e}")
        return None
```

# App Creation

```
def load_images_and_generate():
    conn = connect_to_db()
    if conn is not None:
        try:
            query = """
SELECT i.Image_ID, i.Image_Path, p.Prompt_ID, p.Prompt_Description, p.Prompt_Tags
FROM Images i
JOIN Prompts p ON i.Image_Prompt_ID = p.Prompt_ID
"""
            images_df = pd.read_sql(query, conn)
            # Create a dictionary for the sidebar selectbox outside the loop
            image_options = {f"ID {row['Image_ID']} - {row['Prompt_Description']}": row for index, row in images_df.iterrows()}
            selected_option = st.sidebar.selectbox("Select an Image", list(image_options.keys()))
            selected_row = image_options[selected_option]

            # Display the selected image details
            st.image(selected_row["Image_Path"], width=300, caption=f"Image ID: {selected_row['Image_ID']} - {selected_row['Description']}")

            # Additional functionalities for selected image
            display_average_score(selected_row['Image_ID'])
            display_comments(selected_row['Image_ID'])

            # Generating new prompt and image
            tags = selected_row['Prompt_Tags']
            generate_button = st.button("Generate Prompt and Image", key=f"generate{selected_row['Image_ID']}")

            if generate_button:
                prompt = create_prompt_from_tags(tags)
                if prompt:
                    st.text(prompt) # Display the generated prompt
                    image_url = generate_image(prompt)
                    if image_url:
                        st.image(image_url, caption="Generated Image")

        finally:
            if conn:
                conn.close()
```

# App Creation

---

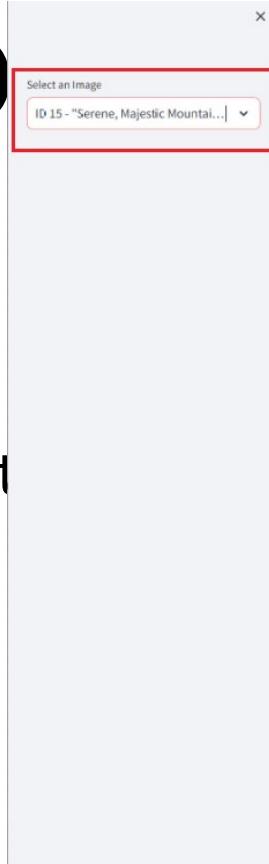
```
def display_average_score(image_id):
    conn = connect_to_db()
    if conn:
        try:
            score_query = f"SELECT AVG(Feedback_Score) AS AvgScore FROM User_Feedback WHERE Feedback_User_ID = {image_id}"
            score_result = pd.read_sql(score_query, conn)
            avg_score = score_result.iloc[0]['AvgScore']
            st.write(f"Average Score for Image {image_id}: {avg_score:.2f}")
        finally:
            conn.close()

def display_comments(image_id):
    conn = connect_to_db()
    if conn:
        try:
            comment_query = f"SELECT Feedback_Comment FROM User_Feedback WHERE Feedback_Image_ID = {image_id}"
            comments_result = pd.read_sql(comment_query, conn)
            st.write("Comments:")
            for index, row in comments_result.iterrows():
                st.text(row['Feedback_Comment'])
        finally:
            conn.close()
```

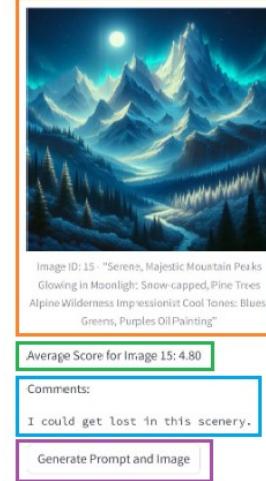
# Python App

---

- **Red** - Image Selection
- **Orange** - Image/ID & Prompt
- **Green** - Average Image Rating
- **Blue** - User Comments
- **Purple** - New Image Generator



DALL-E 3 Prompt Saver



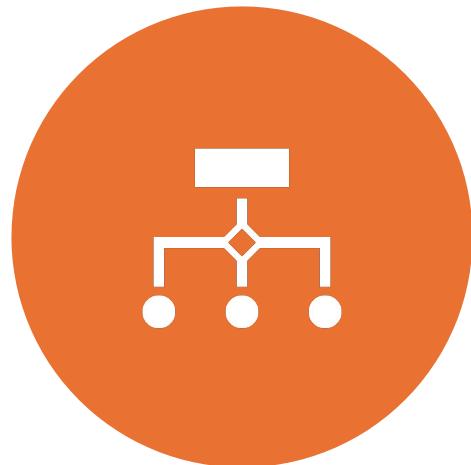
# Conclusion & Reflections

---



# Reflections & Future Improvements

---



IMPLEMENTATION OF A TRACKING SYSTEM FOR PROMPTS TO FOLLOW THE GENERATION OF PROMPTS WITH EACH GENERATION.



IMPLEMENTATION OF A COMPARISON SYSTEM FOR UNDERSTANDING THE SUBTLE CHANGES THERE ARE WHEN GENERATING IMAGES FROM ONE PROMPT

# Conclusion

---

- Even as it stands in its current position, the DALL-E Image-Prompt database will prove to be an invaluable piece in furthering the commercial use of AI image generation and integration of them in everyday life.

