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

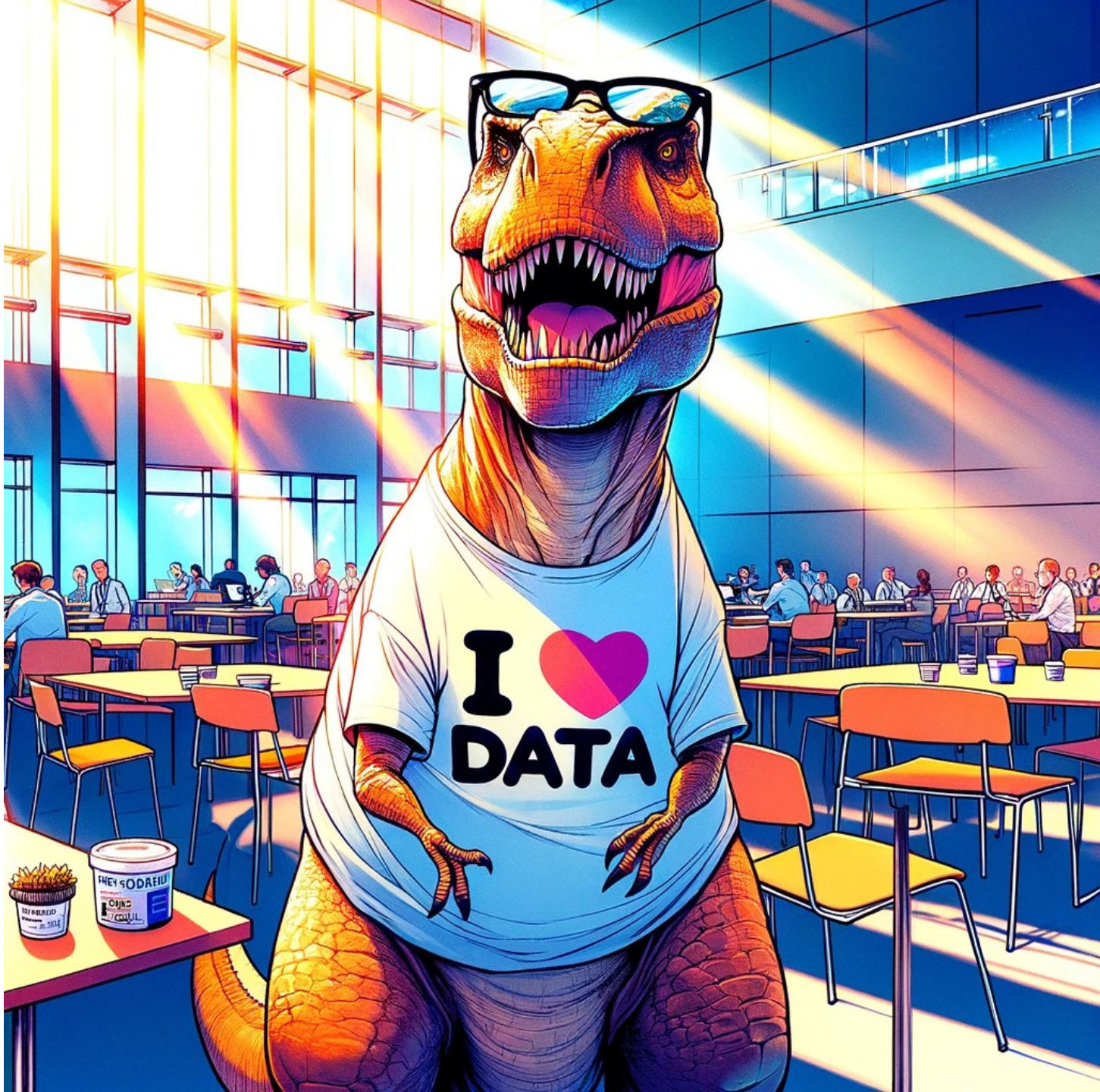
Professional Apps

- 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.

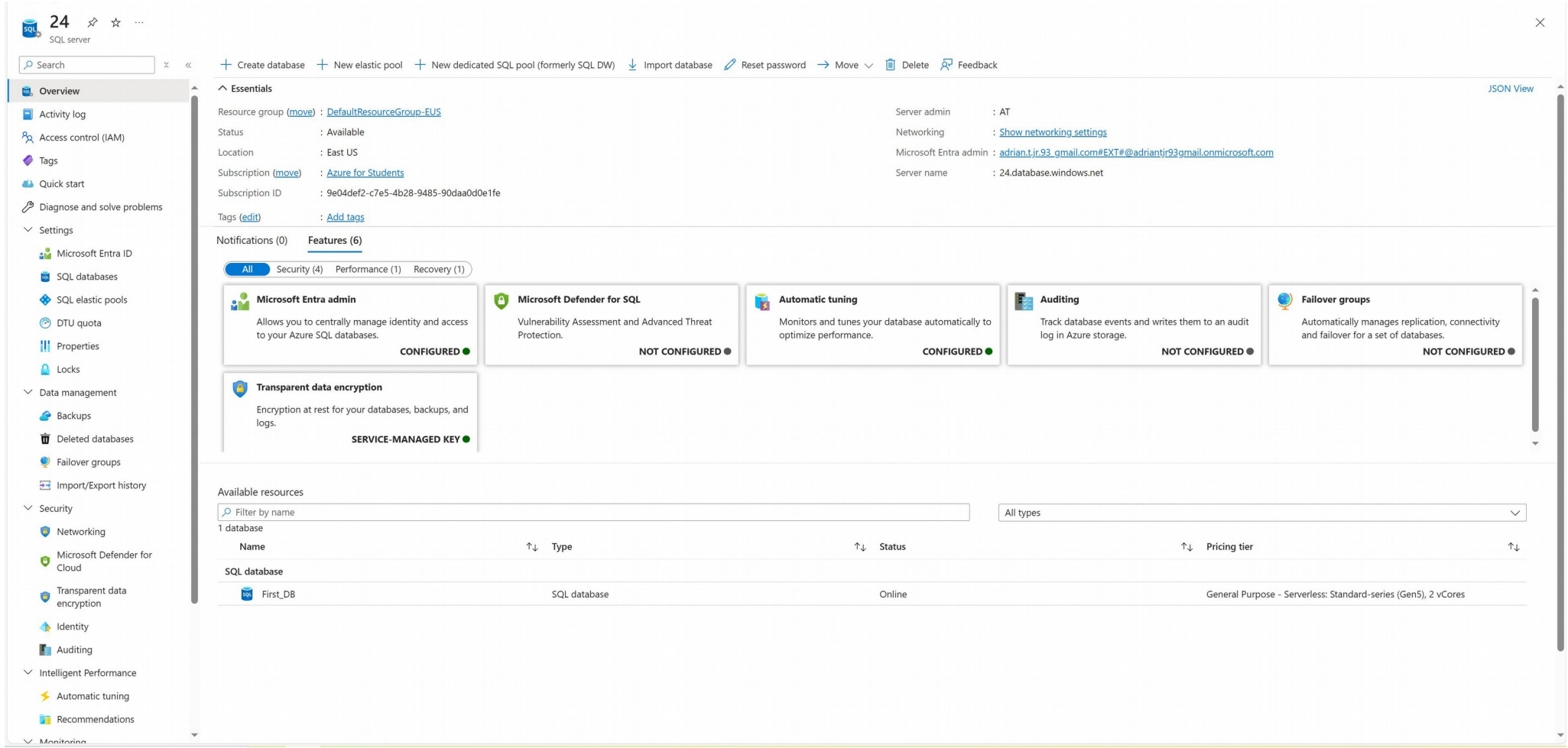
Functional Features (Requirements)

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

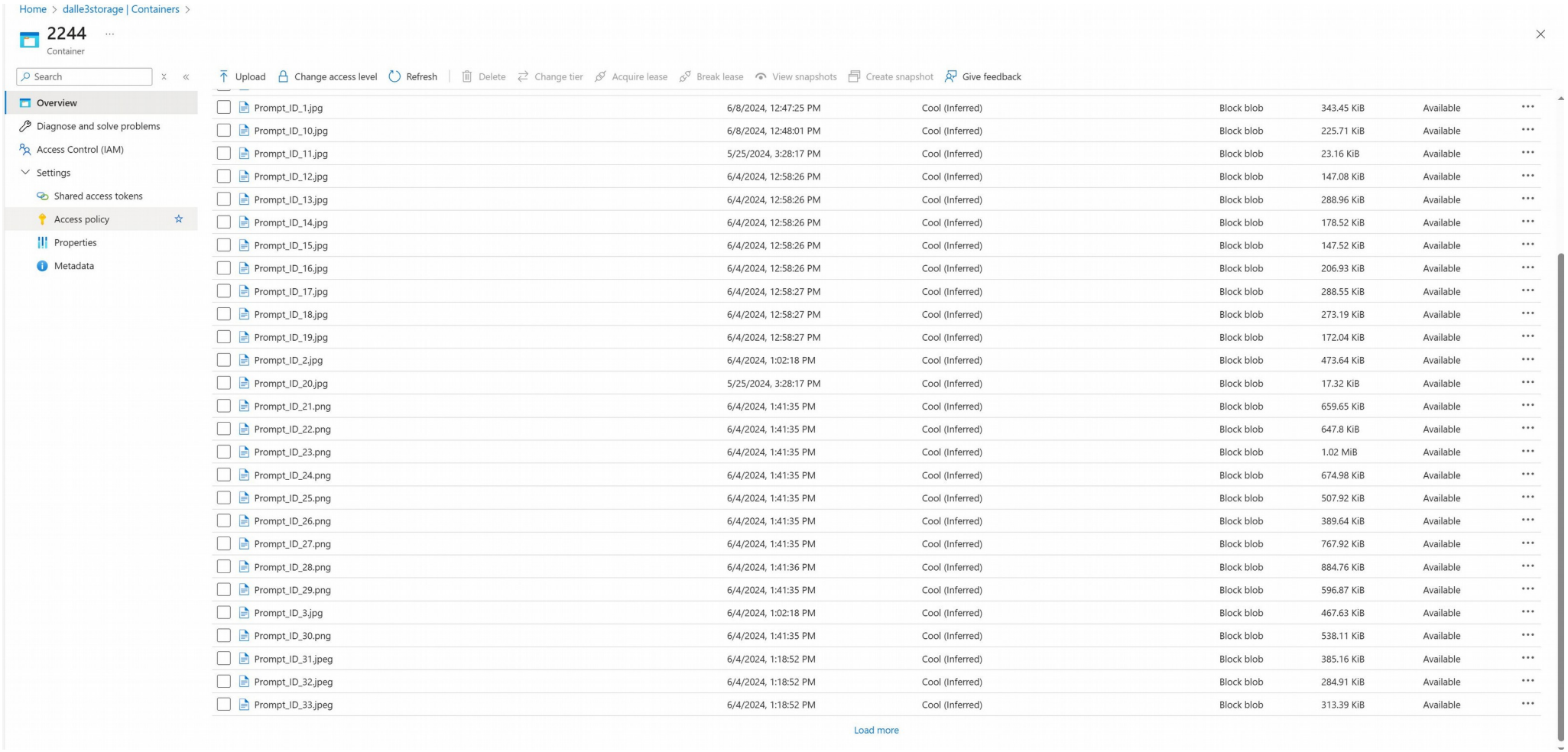
Project Workflow

1. Produce and Image using Text in NightCafe
   - Creating the image
2. Send the picture to Azure Blob Storage by dragging and dropping into the folder.
   - Dropping the pictures in Storage
3. Save the text used to create the image and insert it into the prompt table
   - Creating a data entry for the table
4. Update task on the Teams Channel Planner tab
   - Updating the Team
5. Run/Test Streamlit Ap

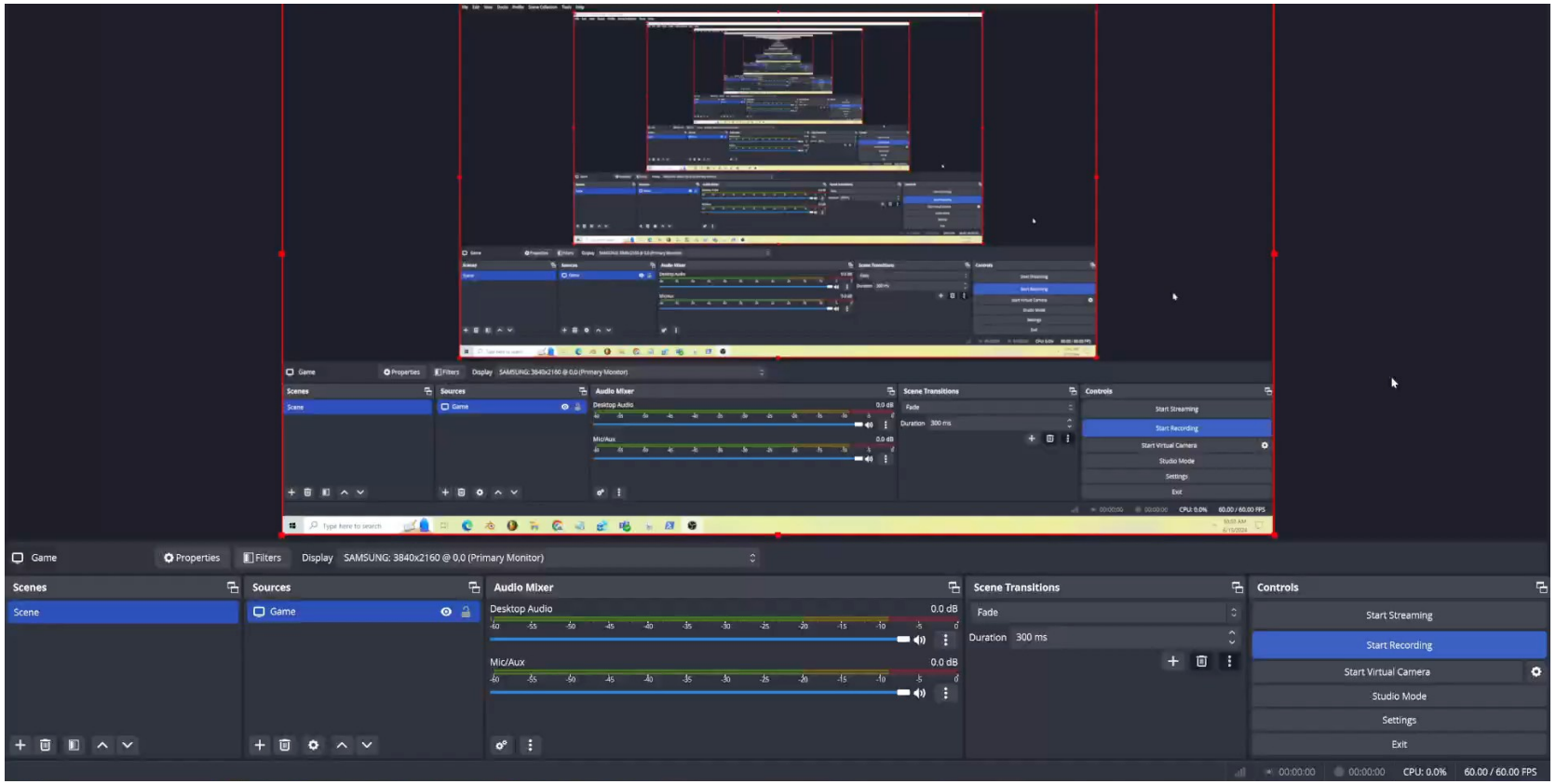
Azure SQL Server and Database



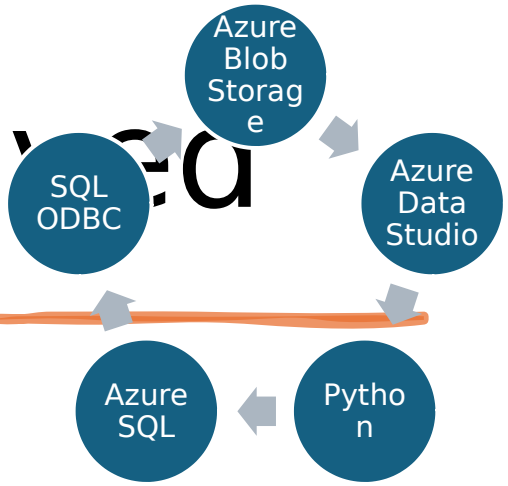
Azure Blob Storage



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 Architecture

Tables

Prompts

- Prompt_ID
- Prompt_Category
- Prompt_Tags
- Prompt_Creation_Date
- Prompt_Creator_User_ID

Users

- User_ID
- User_Username
- User_Firstname
- User_Lastnaeme
- User_Email
- User_Registration_Date

Images

- Image_ID
- Image_Prompt_ID
- Image_Path

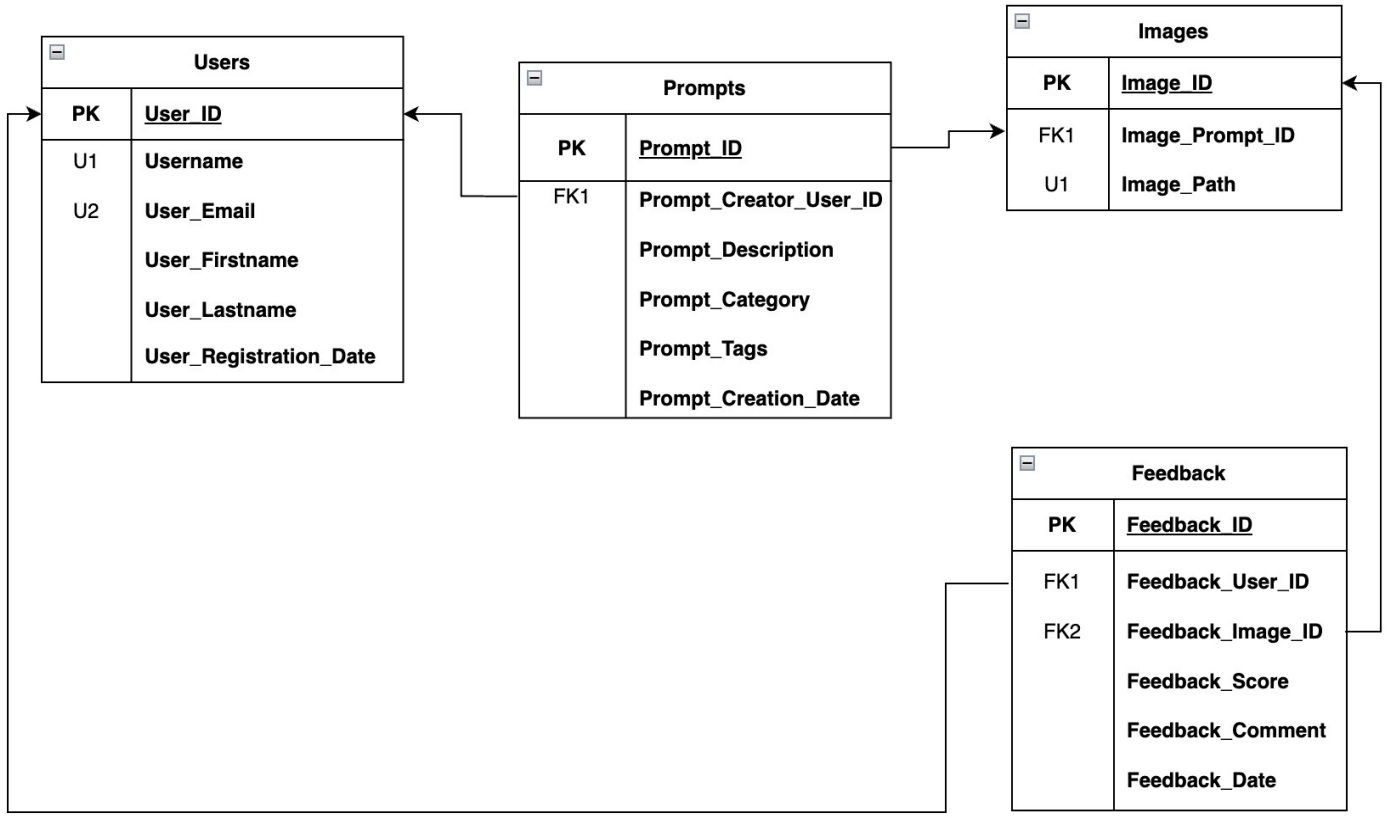
User_Feedback

- Feedback_ID
- Feedback_Image_ID
- Feedback_User_ID
- Feedback_Score
- Feedback_Comment
- Feedback_Date

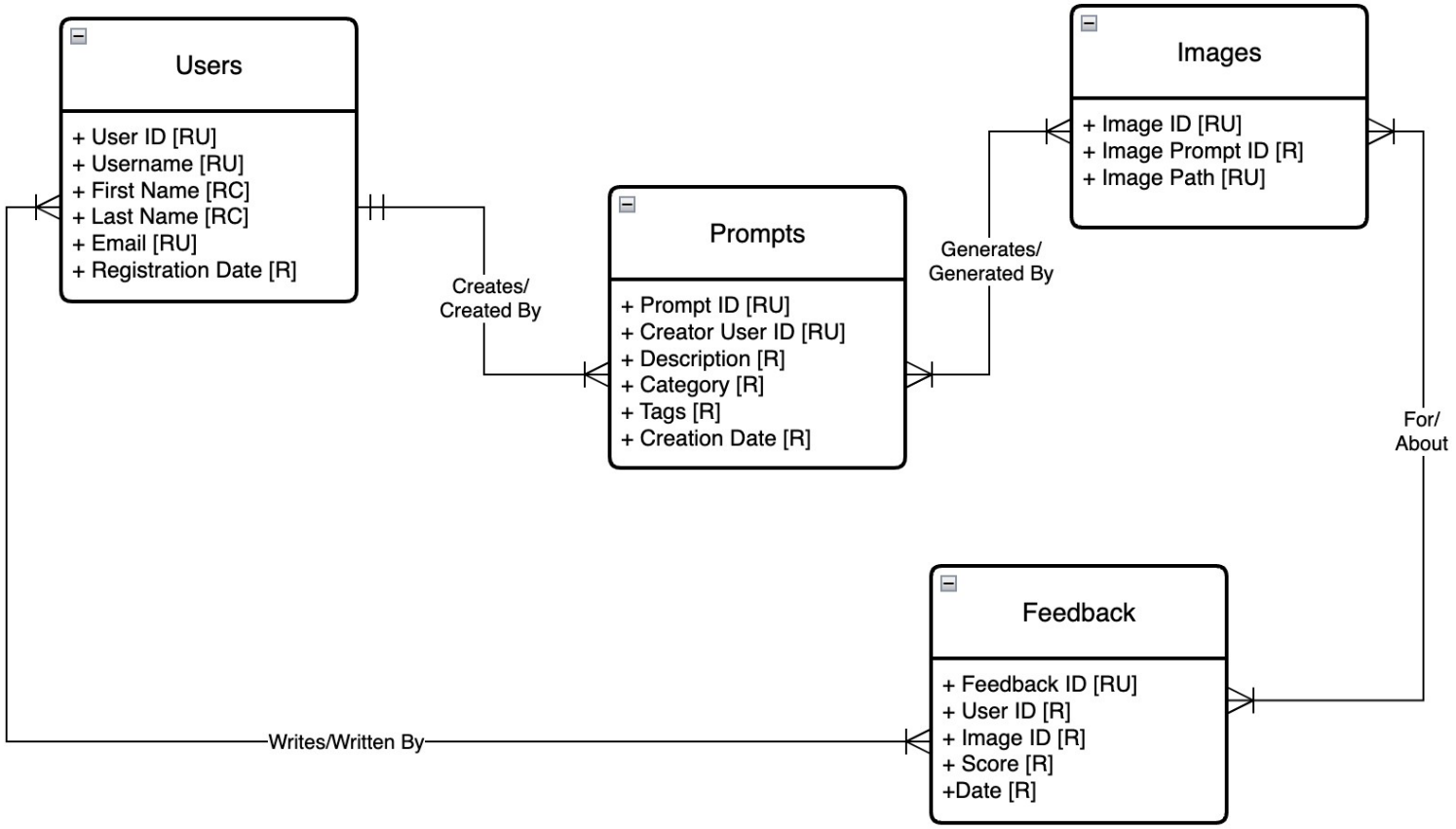
Logical Model

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


```
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

```python
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

```python
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()
```

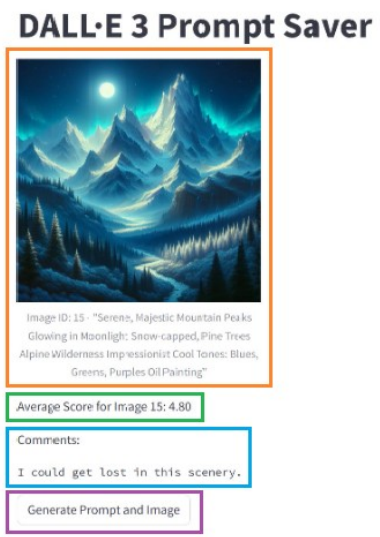
App Creation

```python
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



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.