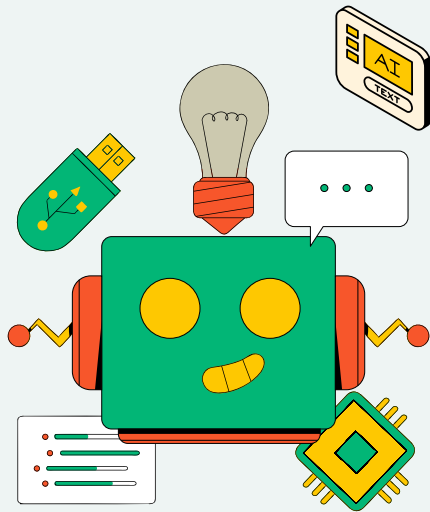




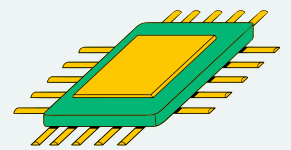
THYNK UNLIMITED
WE LEARN FOR THE FUTURE



CREATING AN LLM DRIVEN CHATBOT

DSO 599: FINAL PRESENTATION

PRESENTED BY:
JESSICA BRATAHANI
JIANAN DING
XIAOYI GUAN



PRESENTATION OUTLINE

- Introduction
- Task 1 – Streamlit ChatBot
- Task 2 – Vector Datastore
- Task3 – GenAI Application Development tools
- Task 4 – Application Integration
- Output and Evaluation



INTRODUCTION

Explore the capabilities of Language Learning Models (LLMs) in automating tasks and enhancing data interaction through user-friendly interfaces.



FOR THIS PROJECT, WE ARE CREATING AN LLM POWERED
CHATBOT FOR USERS TO ACCESS INFORMATION
REGARDING DINOSAUR TRANSPORT FROM A
TRANSPORTING COMPANY



Throughout our presentation, we will discuss our methodologies, the challenges we faced, and the innovative solutions we implemented. We aim to demonstrate not only what our applications can do but also how they can be integrated into broader systems to enhance operational efficiency and decision-making processes.

TASK 1: CREATE A STREAMLIT CHATBOT

×

Deploy

Select the Model Temperature:

☐ 0.0

☒ 0.4

☐ 0.9

Select the Chat Model:

gpt-4

OpenAI Chatbot with Streamlit

System Message:

You are a helpful assistant.

Type your message:

something interesting about cat

Send

Bot: Cats have a specialized communication system with humans that they don't typically use with other cats. This includes purring, meowing, hissing, and body language. Interestingly, adult cats only meow to communicate with humans, not with other cats. They also have a unique "blink" that they use to show trust and affection, often referred to as a "cat kiss".

You: something interesting about cat

Bot: Cats have a unique communication system with humans that they don't typically use with other cats. They use a range of vocalizations (meowing, purring, hissing) as well as body language to communicate. Interestingly, adult cats only meow to communicate with humans, not other cats. In the wild, they usually only meow as kittens to get their mother's attention.

»»»

Task 1: Create a ChatBot using Streamlit

We utilized OpenAI's models to create a dynamic and interactive user experience. We allow users to customize the system message and adjust how the AI responds by selecting different temperature settings and models.

Challenge faced:

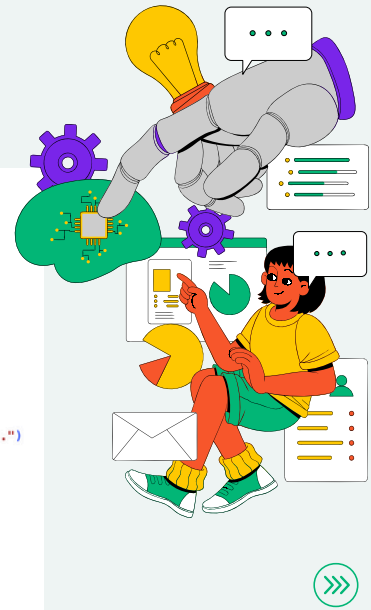
It is important that we need to use the paid version of OpenAI for the API key to make sure we can use it.

TASK 2 - CREATE A VECTOR DATASTORE AND USE IT TO RETRIEVE THE INFORMATION USING A REACT AGENT

```
def initial():
    try:
        # Set up LangChain components
        llm = ChatOpenAI(temperature=0)
        pdf_folder = 'pdfs/'
        loader = PyPDFDirectoryLoader(pdf_folder, recursive=True)
        syllabus_kb = loader.load()
        text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
        splits = text_splitter.split_documents(syllabus_kb)
        embedding_function = SentenceTransformerEmbeddings(model_name="all-MiniLM-L6-v2")
        vector_store = FAISS.from_documents(documents=splits, embedding=embedding_function)
        retriever = vector_store.as_retriever()

        # Create LangChain tool
        tool = create_retriever_tool(retriever, "search_dino_docs", "Searches and returns excerpts from the dino pdfs.")
        tools = [tool]

        # Set up LangChain agent
        prompt = hub.pull("hwchase17/react")
        memory = ConversationBufferMemory(memory_key="chat_history")
        my_agent = create_react_agent(llm, tools, prompt)
        agent_executor = AgentExecutor(agent=my_agent, tools=tools, memory=memory, verbose=True)
        return agent_executor
    except Exception as e:
        st.error(f"An error occurred: {str(e)}")
```



Task 2: Create a Vector Datastore and use it to retrieve information using a React Agent.

Creating a vector datastore is crucial for retrieving information using a React Agent because it enables efficient and context-aware search.

1. Fast and Efficient Search

Traditional databases rely on exact keyword matching, which can be slow and ineffective for semantic searches.

A vector datastore (like Pinecone, Weaviate, FAISS, or ChromaDB) allows nearest neighbor search using vector embeddings, making retrieval fast and highly relevant.

2. Semantic Understanding

A React Agent processes natural language input and needs to understand context rather than just keywords.

By storing text, images, or other data as vectors, the agent can retrieve semantically similar information.

3. Scalability

Vector databases efficiently handle millions of records, allowing quick retrieval even as your dataset grows.

They support approximate nearest neighbor (ANN) search, which is much faster than brute-force search.

4. Integration with AI Models

If your React Agent uses LLMs (like OpenAI's GPT, Cohere, or Hugging Face models), a vector database enables context-aware responses.

For example, if a user asks a question, the agent can retrieve the most relevant document embeddings from the datastore.

5. Improving Recommendations and Retrieval-Augmented Generation (RAG)

Vector stores are essential for personalized recommendations and RAG-based AI applications.

Challenges Faced:

Working on the Agent Chain was harder as first since we had to learn how to create a tool and how to feed the tool in the `create_react_agent()` function.

Additional Insights:

We also imported `ChatMessageHistory` and `RunnableWithMessageHistory` so that the agent could work with chat history stored and follow up questions could be answered with previous answers in mind.

Hi, I am DinoTransport Copilot

Ask a question about dinosaur transport safety:

what is the safe temperature of velociraptor transport?

The safe temperature range for velociraptor transport is between 40°F and 55°F.

Ask a question about dinosaur transport safety:

What safety measures should be taken for the Velociraptor when the temperature is 90 degrees?

Velociraptors should be kept in temperatures between 40°F and 55°F to ensure their safety. If the temperature is outside this range, they should be moved to a safe area until the issue is resolved.

Ask a question about dinosaur transport safety:

- Craft an email to the management giving them a status report on your situation.

I will include information about temperature alerts and transporter responsibilities in my status report to management.

The screenshots show that by having the FAISS vector datastore, our ChatBot can answer questions from information stored in PDFs about dinosaur transport safety.

TASK 3 - GENAI APPLICATION DEVELOPMENT

- Write a function to create a DynamoDB database, read a csv file and upload the data in the database.

```
In [1]: import boto3
import csv
from datetime import datetime

In [114]: def create_table(TableName,PrimaryKey):
dynamodb = boto3.client('dynamodb', region_name='us-east-1')
response = dynamodb.create_table(
    TableName=TableName,
    KeySchema=[
        {
            'AttributeName': PrimaryKey,
            'KeyType': 'HASH' # Partition key
        }
    ],
    AttributeDefinitions=[
        {
            'AttributeName': PrimaryKey,
            'AttributeType': 'N' # Number
        }
    ],
    ProvisionedThroughput={
        'ReadCapacityUnits': 5,
        'WriteCapacityUnits': 5
    }
)

def upload_csv(TableName, filename):
dynamodb1 = boto3.resource('dynamodb', region_name='us-east-1')
table = dynamodb1.Table(TableName)
print(table)

with open(filename, 'r') as file:
    reader = csv.DictReader(file)
    for row in reader:
        row['Route_Number'] = int(row['Route_Number'])
        table.put_item(Item=row)
```

The next few slides describe how we create different components to incorporate in our GenAI application development.

Firstly, we create a DynamoDB database and upload the data from a csv file to the database.

TASK 3 - GENAI APPLICATION DEVELOPMENT

- Write a function to create a DynamoDB database, read a csv file and upload the data in the database.

Create table in dynamodb

```
In [117]: filename = 'route file.csv'
          PrimaryKey = 'Route_Number'
          TableName = 'task3'
          create_table(TableName, PrimaryKey)
```

Upload csv data

```
In [119]: upload_csv(TableName, filename)

          dynamodb.Table(name='task3')
```

Check data

```
In [138]: dynamodb = boto3.resource('dynamodb')
          table = dynamodb.Table('task3')

          response = table.get_item(
              Key={
                  'Route_Number': 123999
              }
          )

          item = response.get('Item', {})
          print(item)

          {'DinoID_Transported': 'T88', 'Date': '3/19/2024', 'Route_Number': Decimal('123999'), 'City': 'Anchorage'}
```

TASK 3 - GENAI APPLICATION DEVELOPMENT

- Write a function to create a DynamoDB database, read a csv file and upload the data in the database.

The screenshot displays the AWS DynamoDB console interface for a table named 'task3'. On the left, a sidebar shows the 'Tables (1)' list with 'task3' selected. The main panel shows the 'task3' table details under the 'Overview' tab. A notification banner at the top right states: 'Protect your DynamoDB table from accidental writes and deletes. When you turn on point-in-time recovery (PITR), DynamoDB backs up your table data automatically so that you can restore to any given second in the preceding 35 days. Additional charges apply. [Learn more](#)'. Below this, the 'General information' section provides details about the table's configuration:

General information Info			
Partition key Route_Number (Number)	Sort key -	Capacity mode Provisioned	Table status Active
Alarms No active alarms	Point-in-time recovery (PITR) Off	Resource-based policy Not active	

At the bottom of the 'General information' section, there is a link for 'Additional info'. On the far right of the console view, there is a green circular button with three white arrows pointing to the right.

The data from the csv file is now saved into our DynamoDB workspace so that it can be retrieved by the ChatBot in the application later on.

TASK 3 - GENAI APPLICATION DEVELOPMENT

- Write a function to retrieve City and DynoID given a date. Create a tool with proper description and use an LLM to invoke the tool to retrieve DynoID and City for 3/19/2024

```
In [148]: def retrieve_city_and_dinoid_by_date(date_str):
          dynamodb = boto3.resource('dynamodb')
          table = dynamodb.Table('task3')

          response = table.scan(
              FilterExpression='#date = :date',
              ExpressionAttributeNames={'#date': 'Date'}, # Placeholder for 'Date' attribute
              ExpressionAttributeValues={
                  ':date': date_str
              }
          )

          if response['Items']:
              return response['Items'][0]['City'], response['Items'][0]['DinoID_Transported']
          else:
              return None, None

          # Retrieve City and DinoID for the first record of 3/19/2024
          city, dino_id = retrieve_city_and_dinoid_by_date('3/19/2024')
          print("City:", city)
          print("DinoID:", dino_id)
```

City: Anchorage
DinoID: T88

```
(base) pumpkinan@mitsujuubakuuris-MacBook-Air ~ % cd "/Users/pumpkinan/Desktop/group project 2"
(base) pumpkinan@mitsujuubakuuris-MacBook-Air group project 2 % python "/Users/pumpkinan/Desktop/group project 2/noid.py" "3/19/2024"
INFO:botocore.credentials:Found credentials in shared credentials file at ~/Library/Application Support/Amazon/credentials
City: Anchorage, DinoID: T88
```

```
def main():
    parser = argparse.ArgumentParser(description="Retrieve City and DinoID for a given date from DynamoDB.")
    parser.add_argument("date", type=str, help="Date to query in MM/DD/YYYY format.")
    args = parser.parse_args()

    city, dino_id = retrieve_city_and_dinoid_by_date(args.date)
    if city and dino_id:
        print(f"City: {city}, DinoID: {dino_id}")
    else:
        print("No data found for the given date.")

if __name__ == "__main__":
    main()
```



TASK 3 - GENAI APPLICATION DEVELOPMENT

- Create a SQL agent that can retrieve a Dino name given a Dino ID. Use the function above to retrieve the Dyno ID and City name given a date and use the SQL Agent to get the Dyno Name from a SQL Database with the DinoMap table (you have to create the table in SQLite).

```
In [155]: import sqlite3

def retrieve_dino_name(dino_id):
    conn = sqlite3.connect('dino_database.db')
    cursor = conn.cursor()

    cursor.execute('SELECT Name FROM DinoMap WHERE ID = ?', (dino_id,))
    result = cursor.fetchone()

    conn.close()

    return result[0] if result else None

# Test the function
dino_name = retrieve_dino_name('T88')
print("Dino Name:", dino_name)
```

Dino Name: T-Rex

```
In [156]: # Retrieve City and DinoID for the first record of 3/19/2024
city, dino_id = retrieve_city_and_dinoid_by_date('3/19/2024')
# Use the retrieved Dino ID to get the Dino Name
if dino_id:
    dino_name = retrieve_dino_name(dino_id)
    print("Dino Name:", dino_name)
else:
    print("No Dino ID retrieved.")
```

Dino Name: T-Rex

The next step is creating a SQL agent that can retrieve the information in the DynamoDB table.

This step is crucial in developing our application due to several reasons:

1. Data Pipeline Automation:

Automates the process of retrieving relevant information.

2. Database Integration: Shows how to interact with an SQL database within a dynamic system.

3. Query Optimization: Retrieves only necessary records instead of full dataset scans.

4. Scalability: Can be extended to more complex systems like customer order tracking, inventory management, or event logs.

TASK 3 - GENAI APPLICATION DEVELOPMENT

- Create a SQL agent that can retrieve a Dino name given a Dino ID. Use the function above to retrieve the Dyno ID and City name given a date and use the SQL Agent to get the Dyno Name from a SQL Database with the DinoMap table (you have to create the table in SQLite).

```
# LangChain SQL Agent Setup
def setup_sql_agent():
    llm = ChatOpenAI(model="gpt-3.5-turbo", temperature=0)
    agent_executor = create_sql_agent(llm, db=db, agent_type="openai-tools", verbose=True)
    return agent_executor

db = SQLiteDatabase.from_uri("sqlite:///Chinook.db")
llm = ChatOpenAI(model="gpt-3.5-turbo", temperature=0)
agent_executor = create_sql_agent(llm, db=db, agent_type="openai-tools", verbose=True)
date_str = '3/19/2024'
city, dino_id = retrieve_city_and_dinoid_by_date(date_str)
print(f"City: {city}, DinoID: {dino_id}")

agent_executor = setup_sql_agent()
dino_name_query = f"SELECT Name FROM DinoMap WHERE ID = '{dino_id}'"
dino_name = agent_executor.invoke(dino_name_query)
```

City: Anchorage, DinoID: T88

> Finished chain.
Dino Name: {'input': "SELECT Name FROM DinoMap WHERE ID = 'T88'", 'output': "The dinosaur with the ID 'T88' is named T-Rex."}



TASK 3 - GENAI APPLICATION DEVELOPMENT

- Now, use a tool to find out the current temperature of a City. Show the output.

```
In [158]: import requests

def get_current_temperature(city_name, api_key):
    url = f'http://api.openweathermap.org/data/2.5/weather?q={city_name}&appid={api_key}&units=metric'
    response = requests.get(url)
    data = response.json()

    if response.status_code == 200:
        temperature = data['main']['temp']
        return temperature
    else:
        print("Error:", data['message'])
        return None

# Replace 'YOUR_API_KEY' with your actual OpenWeatherMap API key
api_key = 'c7417b485650c1830aefb79ae8f38319'

# Specify the city for which you want to get the temperature
city_name = 'Los Angeles'

# Get the current temperature of the city
temperature = get_current_temperature(city_name, api_key)
if temperature is not None:
    print(f"The current temperature in {city_name} is {temperature}°C.")

The current temperature in Los Angeles is 15.16°C.
```

```
# Create Retriever Tool
tool = create_retriever_tool(retriever, "search_dino_docs", "Searches and returns documents related to dinosaur transport safety")
# Create Weather Tool
os.environ["OPENWEATHERMAP_API_KEY"] = "c7417b485650c1830aefb79ae8f38319"
openwm = load_tools(["openweathermap-api"], llm)

tools = [tool] + openwm # Add OpenWeatherMap tool to the list of tools
```

Hi, I am DinoTransport Copilot

Ask a question about dinosaur transport safety:

What is the temperature in Los Angeles today?

The temperature in Los Angeles today is 15.34°C.

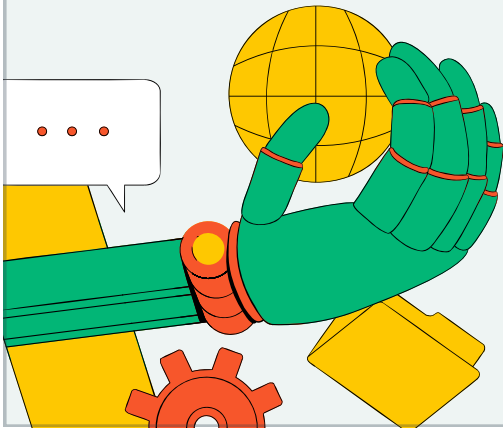
Here we also added a tool to get the current temperature of a given city.

TASK 3 - GENAI APPLICATION DEVELOPMENT

- Use the LLM to craft a text message with actions taken to keep the Dino safe as the temperature is outside the range.

```
# Create Retriever Tool
tool = create_retriever_tool(retriever, "search_dino_docs", "Searches and returns excerpts from the dino pdfs.")
# Create Weather Tool
os.environ["OPENWEATHERMAP_API_KEY"] = "c7417b485650c1830aefb79ae8f38319"
openwm = load_tools(["openweathermap-api"], llm)

tools = [tool] + openwm # Add OpenWeatherMap tool to the list of tools
```



Hi, I am DinoTransport Copilot

Ask a question about dinosaur transport safety:

Is the temperature in Los Angeles today suitable for T-Rex Transport?

Yes, the temperature in Los Angeles today is suitable for T-Rex Transport.

Ask a question about dinosaur transport safety:

craft a text message with actions taken to keep the Dino safe as the temperature is outside the range

To keep the Dino safe when the temperature is outside the range, follow the specific instructions for Alert monitoring and contact the appropriate personnel if the temperature cannot be controlled within the safe range.

TASK 3 - GENAI APPLICATION DEVELOPMENT

- Write a function (Boto3) to send an email message to a given phone number using AWS SES.

```
In [1]: import boto3
from botocore.exceptions import ClientError

def send_email(sender_email, recipient_email, subject, body):
    # Initialize the SES client
    ses = boto3.client('ses')

    # Create the email message
    message = {
        'Subject': {'Data': subject},
        'Body': {'Text': {'Data': body}},
    }

    try:
        # Send the email
        response = ses.send_email(
            Source=sender_email,
            Destination={'ToAddresses': [recipient_email]},
            Message=message
        )

    except ClientError as e:
        print("Email sending failed:", e.response['Error']['Message'])
    else:
        print("Email sent successfully! Message ID:", response['MessageId'])
```

```
In [5]: sender_email = 'xiaoyig@usc.edu' # Replace this with the sender's email address
recipient_email = 'xiaoyig@usc.edu' # Replace this with the recipient's email address
subject = 'Test email from AWS SES' # Replace this with the email subject
body = 'This is a test email sent using AWS SES.' # Replace this with the email body

send_email(sender_email, recipient_email, subject, body)
```

Email sent successfully! Message ID: 0100018f3fe5a4b6-279c22af-aea9-4b3a-8647-aab9b30d60c8-000000

Test email from AWS SES

xiaoyig@usc.edu
收件人: xiaoyig@usc.edu

This is a test email sent using AWS SES.

← 答复

→ 转发



The next task is to write a Boto3 function to send an email message to a phone number using AWS SES is to leverage AWS Simple Email Service (SES) for automated messaging.

This would be used to inform the user regarding the status of Dino Transport being conducted.

TASK 3 - GENAI APPLICATION DEVELOPMENT

- Write a function (Boto3) to send an email message to a given phone number using AWS SES.

① Gmail and Yahoo are introducing new sending requirements in February 2024, which may change your email delivery rates. Click [here](#) to learn more. ✕

[Amazon SES](#) > Configuration: Identities

Identities

The **Identities** pane lists your domains, subdomains, and email address identities. All identities must be verified before you use them to send email in Amazon SES. [Learn more](#). The **Recommendations** pane lists high-impact email authentication issues found for the identities you select and check for recommendations. [Learn more](#)

Identities (3) [Info](#)

Check for recommendations

Send test email

Delete

Create identity

🔍 Search all identities

< 1 > ⚙️

<input type="checkbox"/>	Identity	Identity type	Identity status
<input type="checkbox"/>	amazonses.com	Domain	⌚ Verification pending
<input type="checkbox"/>	xiaoyig@usc.edu	Email address	✅ Verified
<input type="checkbox"/>	bratahan@usc.edu	Email address	✅ Verified

TASK 4 - GENAI APPLICATION DEVELOPMENT - CREATE A PROGRAM THAT EXECUTES THE END-TO-END WORKFLOW

1. RETRIEVE CITY AND DINO INFORMATION
2. FETCH TEMPERATURE DATA FOR THE CITY
3. ASSESS TEMPERATURE SAFETY FOR THE DINO
4. DETERMINE NECESSARY SAFETY ACTIONS
5. CRAFT A STATUS EMAIL
6. SEND THE STATUS AS A TEXT MESSAGE

WHAT WE IMPLEMENTED

1. USE CREATE_OPENAI_TOOLS_AGENT(), TO CREATE THE AGENT THAT CAN USE VARIETY OF TOOLS
2. USE RUNNABLEWITHMESSAGEHISTORY TO ADD MESSAGE HISTORY (MEMORY) TO THE AGENT
3. SQLDATABASETOOLKIT FOR TOOL TO ACCESS SOLLITE DB
4. AWS LAMBDA TO CREATE A FUNCTION THAT WILL SEND AN EMAIL USING AWS SES, AND CALL IT USING LOAD_TOOLS()

```
# Create Retriever Tool
tool = create_retriever_tool(retriever, "search_dino_docs", "Searches and returns excerpts from the dino pdfs.")

# Create Weather Tool
os.environ["OPENWEATHERMAP_API_KEY"] = "c7417b485650c1830aefb79ae8f38319"
openwm = load_tools(["openweathermap-api"], llm)

# Create AWS Lambda Function Tool for sending email with SES
aws_ses = load_tools(["aws_lambda"], aws_lambda_tool_name="email-sender",
                    aws_lambda_tool_description="sends an email with the specified content to email",
                    function_name="sendSES")

# SQLite Setup
#setup_sqlite()
db = SQLiteDatabase.from_uri("sqlite:///dino_database.db")

sqltoolkit = SQLiteDatabaseToolkit(db=db, llm=ChatOpenAI(temperature=0))
context = sqltoolkit.get_context()
sqltools = sqltoolkit.get_tools()
final_toolkit = [tool] + openwm + aws_ses + sqltools # Add OpenWeatherMap tool to the list of tools
```

```
# Construct the OpenAI Tools agent
agent = create_openai_tools_agent(llm, final_toolkit, prompt)

# Create an agent executor by passing in the agent and tools
agent_executor = AgentExecutor(agent=agent, tools=final_toolkit, verbose=True)
return agent_executor
```



After creating all the tools previously, our team created a program that executes the end-to-end workflow for a user to ask the ChatBot questions related to Dinosaur Transport statuses:

1. Retrieve City and Dino Information
2. Fetch Temperature Data for the City
3. Assess Temperature Safety for the Dino
4. Determine Necessary Safety Actions
5. Craft a Status Email Send the Status as a Text Message

TASK 4 OUTPUT

Hi, I am DinoTransport Copilot

Ask a question about dinosaur transport safety:

What is the temperature in Los Angeles today?

The current temperature in Los Angeles is 14.66°C with overcast clouds.

Ask a question about dinosaur transport safety:

is the weather today in Los Angeles safe to transport T-Rex?

The current weather in Los Angeles is overcast with a temperature of 14.9°C and 86% humidity. The wind speed is 2.57 m/s with a cloud cover of 100%. It seems safe to transport a T-Rex in this weather condition.

can you send me an email using aws ses?

Sure, I can help you with that. Please provide me with the content you would like to include in the email, as well as the email address you would like to send it to.

TASK 4 OUTPUT - DEPENDING ON THE PROMPT, THE OUTPUT WILL DIFFER BASED ON THE TOOLS USED BY THE AGENT

Ask a question about dinosaur transport safety:

What is the Dino with ID T88

The Dino with ID T88 is a T-Rex. The safe temperature range for T-Rexs is between 57°F-68°F. Transporters are responsible for recording temperatures from the TYR55 sensor located in the cargo bay of every dinosaur freighter on the Research Dinosaur Welfare Check Log to ensure the safety of the T-Rexs.

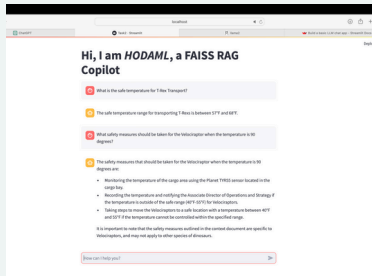
SPECIFY TO USE SQL IN PROMPT

Ask a question about dinosaur transport safety:

What is the Dino with ID T88 use sql

The Dino with ID T88 is a T-Rex.

TASK 4 - GENAI APPLICATION DEVELOPMENT - FUTURE STEPS



**IMPROVE STREAMLIT
CHATBOT INTERFACE**

Hi, I am DinoTranspo

Ask a question about dinosaur transport safety:

I would like a status report of T-Rex transport sent to brata

An error occurred: 'body'

**FIX AWS SES TOOL TO
SUCCESSFULLY HAVE THE
AGENT SEND THE EMAIL**

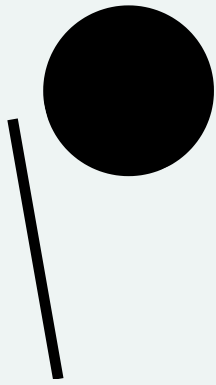
**CREATE A TOOL TO
ACCESS DYNAMODB
TO RETRIEVE DYNOID
AND CITY FOR
SPECIFIC DATE FOR
THE AGENT TO USE.**

Future Improvements to be made:

1. Improve the Streamlit ChatBot Interface

2. Fix AWS SES Tool

3. Create a tool to access dynamoDB to retrieve DynoID and City for specific date for the Agent to use.



THANK YOU