databricks

Delivering Personalized & Realtime Context for LLM

Databricks Feature & Function Serving

Aakrati Talati
TL Feature Serving - Databricks

Mingyang Ge
ML/AI Engineer - Databricks

FEATURE STORE SUMMIT 2023

Organized by HOPSWORKS

Outline

Delivering personalized context using Feature Serving

- Traditional hotel booking website
- Bring LLM to hotel booking
  - Context
  - Enterprise governance
- Demo
- Conclusion & Questions

Traditional hotel booking website

FEATURE STORE
SUMMIT
2023

Any …

Traditional hotel booking website

- Website is built on
  - Search
  - Ranking
  - And Personalization
- Personal data in the lakehouse

Bringing LLM to hotel booking website

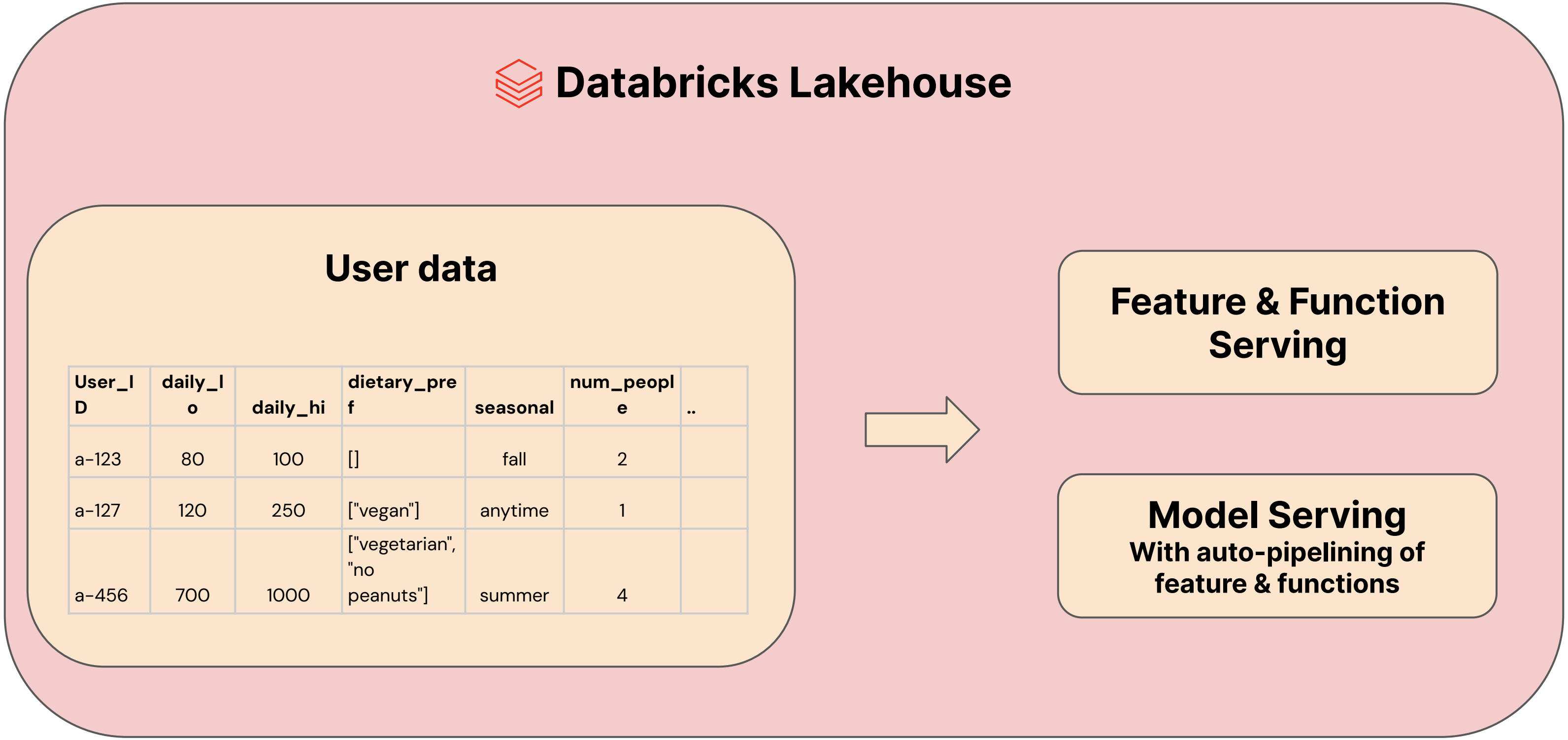
Generic chatbot

Out-of-the-box LLM without personalized context

'input': 'Plan a 7-day vacation to the Amalfi Coast around September for user id "a-456".',

'output': "Sure! I can help you plan a 7-day vacation to the Amalfi Coast in September. However, I am an AI language model and I don't have access to your personal information. I suggest you contact a travel agency or use a travel booking website to plan your trip. They can help you with flights, accommodations, and activities. Have a great trip!"

User data in the Lakehouse



Databricks Lakehouse

User data

User_ID	daily_lo	daily_hi	dietary_pref	seasonal	num_people	..
a-123	80	100	[]	fall	2	
a-127	120	250	["vegan"]	anytime	1	
a-456	700	1000	["vegetarian", "no peanuts"]	summer	4	

Feature & Function Serving

Model Serving
With auto-pipelining of feature & functions

Feature Engineering in Unity Catalog

Your Lakehouse is your feature store

- Any table is a feature table - A primary key is all you need
- One copy of your data, accessible from all workspaces
- No more training/serving skew!
- Lineage and governance aware

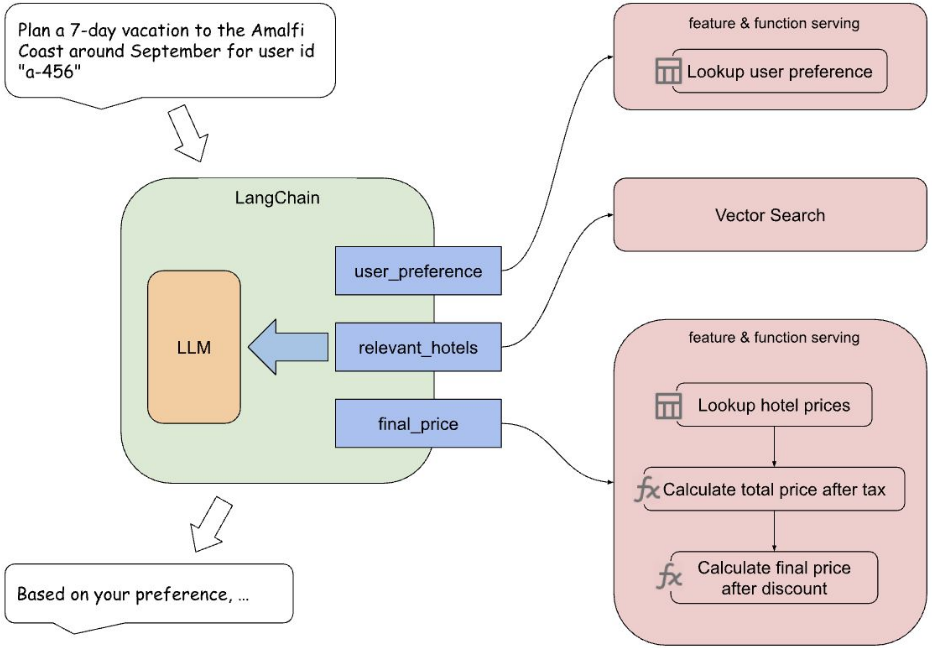
Real-time Feature + Function Serving

Your Lakehouse supports real-time AI at scale

- For those features that require real-time calculation
- Lookups and calculations can both be run in realtime
- Can support arbitrary python functions
- Can support arbitrary function chaining
- Increased contextual understanding leads to better chatbots

Personalized context from lakehouse

Bringing it all together!



Plan a 7-day vacation to the Amalfi Coast around September for user id "a-456"
LangChain
LLM
user_preference
relevant_hotels
final_price
feature & function serving
Lookup user preference
Vector Search
feature & function serving
Lookup hotel prices
Calculate total price after tax
Calculate final price after discount
Based on your preference, ...

Workflow

What do I know about this user?

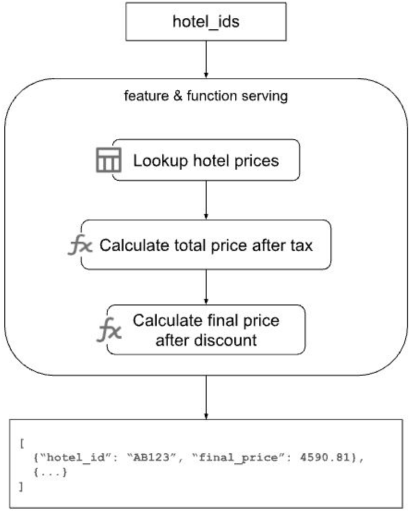
> **Feature & Function Serving:** Get the user’s budget and preferences

What are the best hotels based on the user's preferences?

> **Vector search**: Get top-K hotels from vector search based on user's preferences

What is the total amount your trip will cost ?

> **Feature & Function Serving:** Get real time availability and prices, and calculate total prices using function chaining



hotel_ids

feature & function serving

Lookup hotel prices

Calculate total price after tax

Calculate final price after discount

```
[
  {"hotel_id": "AB123", "final_price": 4590.81},
  {...}
]
```

AI driven travel chatbot:

Demo with personal context from Feature & Function Serving

FEATURE STORE
SUMMIT
2023

Key Takeaways

Feature & Function Serving - brings personalized context at your fingertips!

- Feature Engineering in Unity Catalog
  - Your Lakehouse is your feature store
  - Lineage and governance aware

- Real-time Feature + Function Serving
  - For those features that require real-time calculation
  - Arbitrary python functions + Function chaining

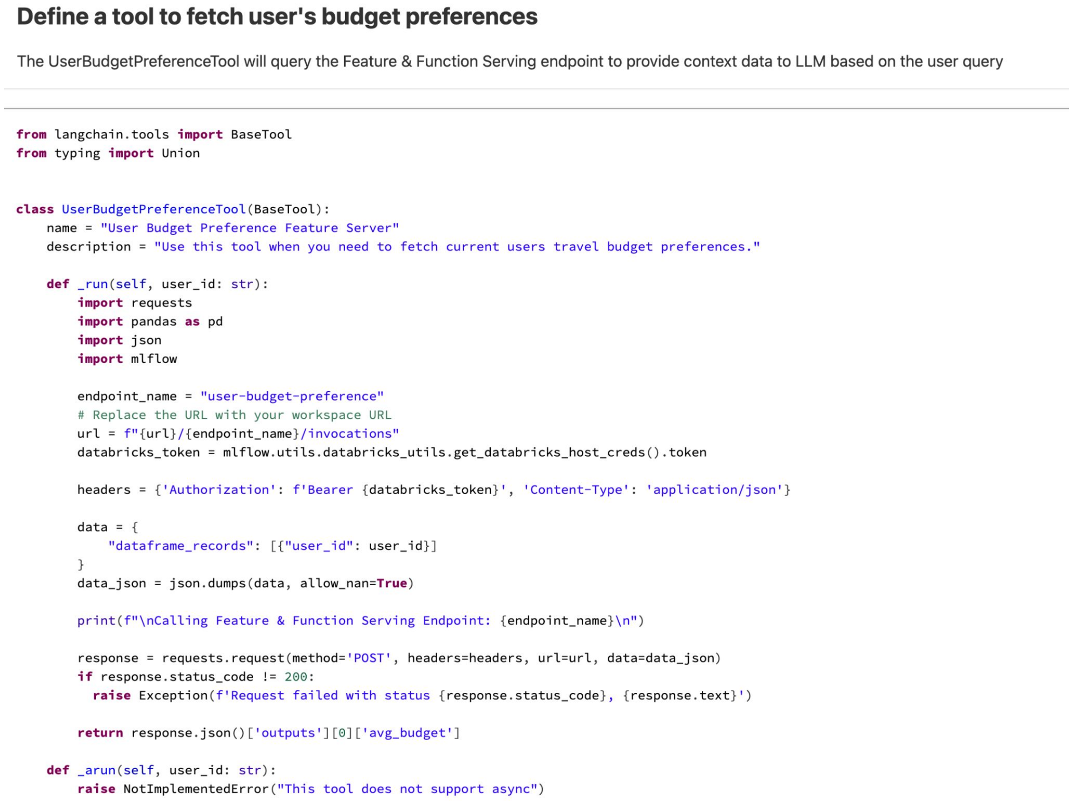
Questions?

Email: feature-serving@databricks.com

Code Snippets - Create Feature Serving Endpoint

```
hotel_endpoint_name = "hotel-final-price"
# Create a lookup to fetch features by key.
features=[
  FeatureLookup(
    table_name="feature_serving.travel_recommendations.hotel_prices",
    lookup_key="hotel_id",
  ),
  FeatureFunction(
    udf_name="feature_serving.travel_recommendations.hotel_total_price",
    input_bindings={
      "price": "price",
      "tax_rate": "tax_rate",
      "num_days": "num_days"
    },
    output_name="total_price",
    inclu
  ),
  FeatureFunction(
    udf_name="feature_serving.travel_recommendations.discount_price",
    input_bindings={
      "price": "total_price",
    },
    output_name="final_price"
  )
]

# Create endpoint
status = fc.create_feature_serving_endpoint(name=hotel_endpoint_name, config = EndpointCoreConfig(servables=Servable(features=features)))
print(status)
```

Code Snippets - LangChain Tools

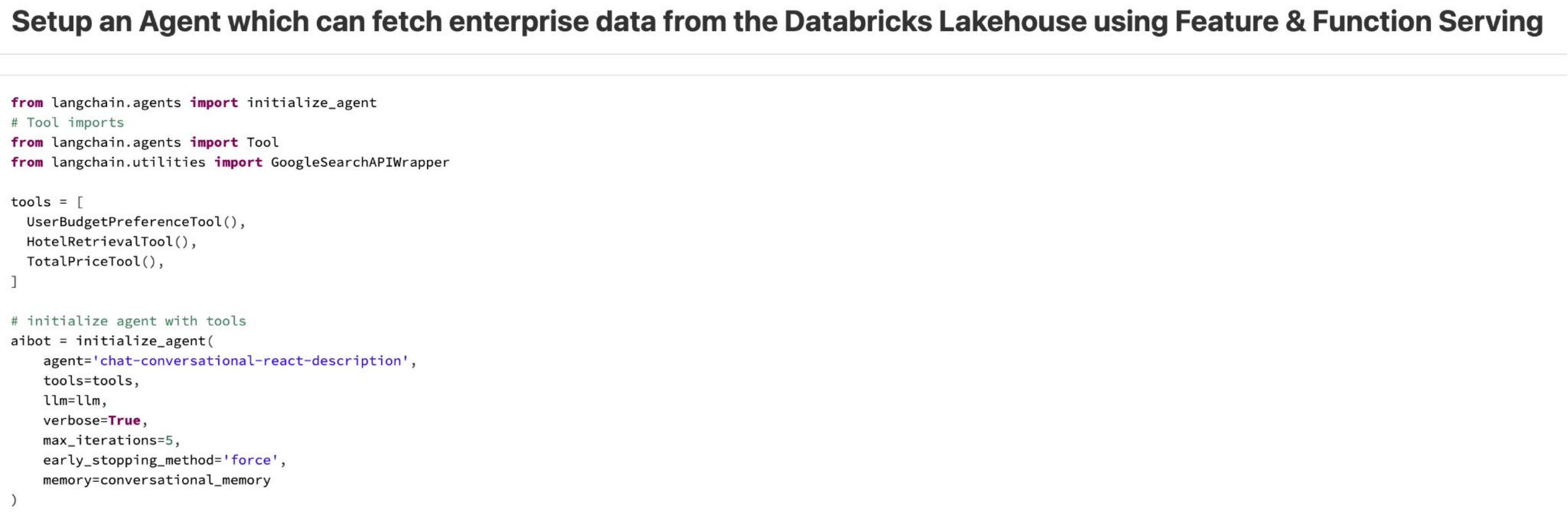


Define a tool to fetch user's budget preferences

The UserBudgetPreferenceTool will query the Feature & Function Serving endpoint to provide context data to LLM based on the user query

```python
from langchain.tools import BaseTool
from typing import Union


class UserBudgetPreferenceTool(BaseTool):
    name = "User Budget Preference Feature Server"
    description = "Use this tool when you need to fetch current users travel budget preferences."

    def _run(self, user_id: str):
        import requests
        import pandas as pd
        import json
        import mlflow

        endpoint_name = "user-budget-preference"
        # Replace the URL with your workspace URL
        url = f"{url}/{endpoint_name}/invocations"
        databricks_token = mlflow.utils.databricks_utils.get_databricks_host_creds().token

        headers = {'Authorization': f'Bearer {databricks_token}', 'Content-Type': 'application/json'}

        data = {
            "dataframe_records": [{"user_id": user_id}]
        }
        data_json = json.dumps(data, allow_nan=True)

        print(f"\nCalling Feature & Function Serving Endpoint: {endpoint_name}\n")

        response = requests.request(method='POST', headers=headers, url=url, data=data_json)
        if response.status_code != 200:
          raise Exception(f'Request failed with status {response.status_code}, {response.text}')

        return response.json()['outputs'][0]['avg_budget']

    def _arun(self, user_id: str):
        raise NotImplementedError("This tool does not support async")
```

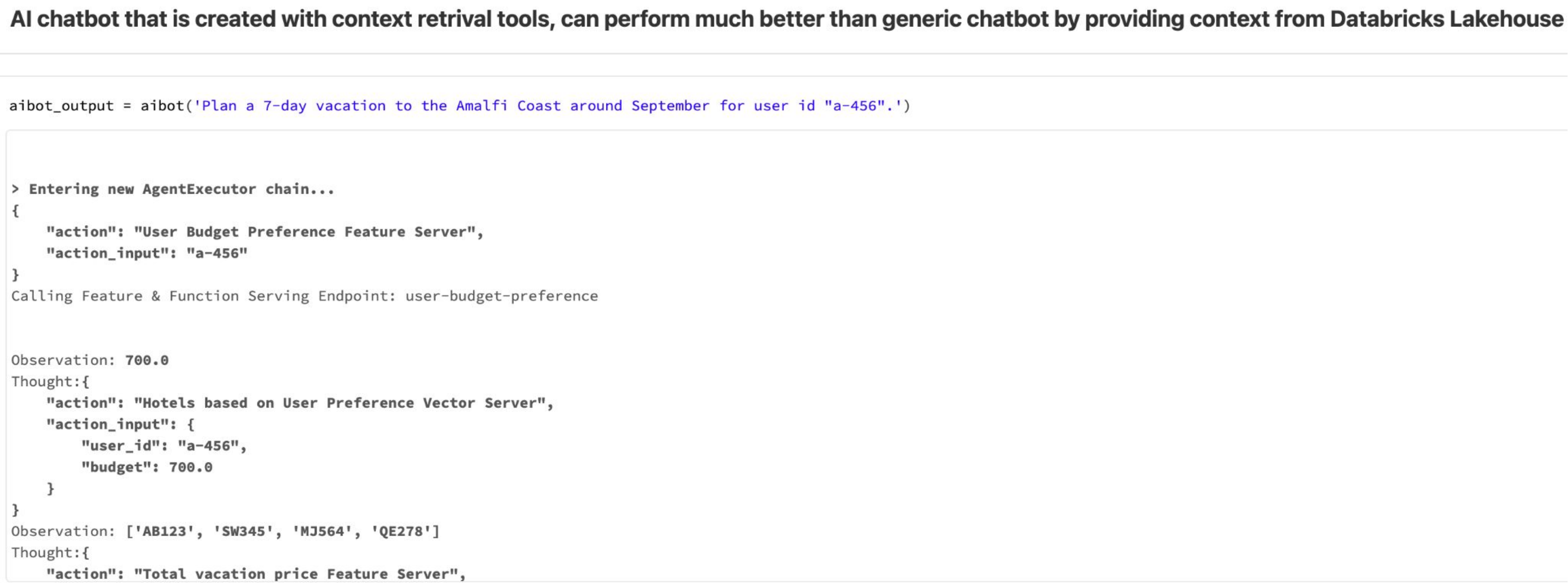
Code Snippets - AI bot with Feature & Function Serving



Setup an Agent which can fetch enterprise data from the Databricks Lakehouse using Feature & Function Serving

```
from langchain.agents import initialize_agent
# Tool imports
from langchain.agents import Tool
from langchain.utilities import GoogleSearchAPIWrapper

tools = [
  UserBudgetPreferenceTool(),
  HotelRetrievalTool(),
  TotalPriceTool(),
]

# initialize agent with tools
aibot = initialize_agent(
    agent='chat-conversational-react-description',
    tools=tools,
    llm=llm,
    verbose=True,
    max_iterations=5,
    early_stopping_method='force',
    memory=conversational_memory
)
```

Code Snippets - AI bot with Feature & Function Serving

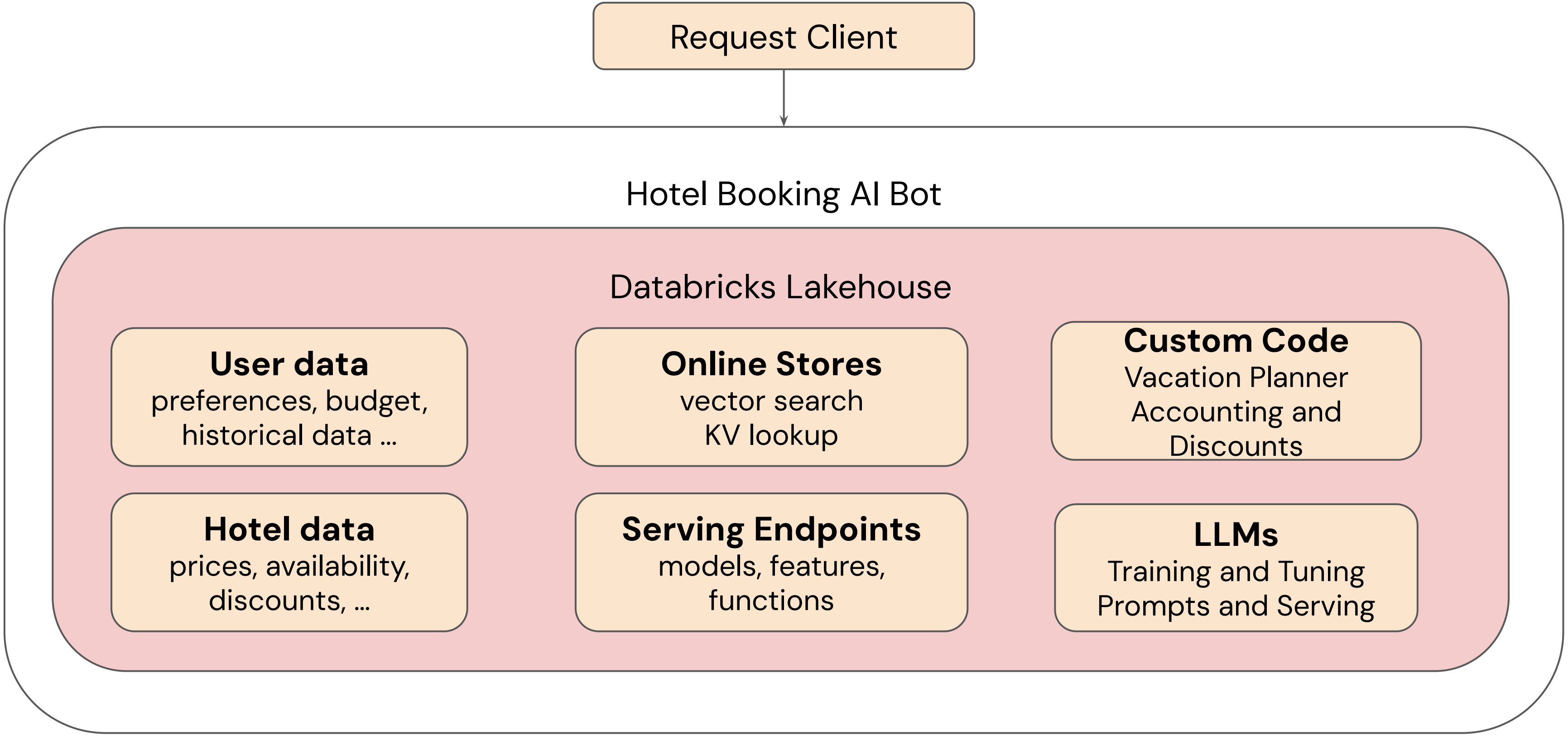


AI chatbot that is created with context retrival tools, can perform much better than generic chatbot by providing context from Databricks Lakehouse

```
aibot_output = aibot('Plan a 7-day vacation to the Amalfi Coast around September for user id "a-456".')
```

```
> Entering new AgentExecutor chain...
{
    "action": "User Budget Preference Feature Server",
    "action_input": "a-456"
}
Calling Feature & Function Serving Endpoint: user-budget-preference


Observation: 700.0
Thought:{
    "action": "Hotels based on User Preference Vector Server",
    "action_input": {
        "user_id": "a-456",
        "budget": 700.0
    }
}
Observation: ['AB123', 'SW345', 'MJ564', 'QE278']
Thought:{
    "action": "Total vacation price Feature Server",
```

Power of LLMs + Personalized context



Request Client

Hotel Booking AI Bot

Databricks Lakehouse

User data
preferences, budget, historical data …

Online Stores
vector search
KV lookup

Custom Code
Vacation Planner
Accounting and Discounts

Hotel data
prices, availability, discounts, …

Serving Endpoints
models, features, functions

LLMs
Training and Tuning
Prompts and Serving