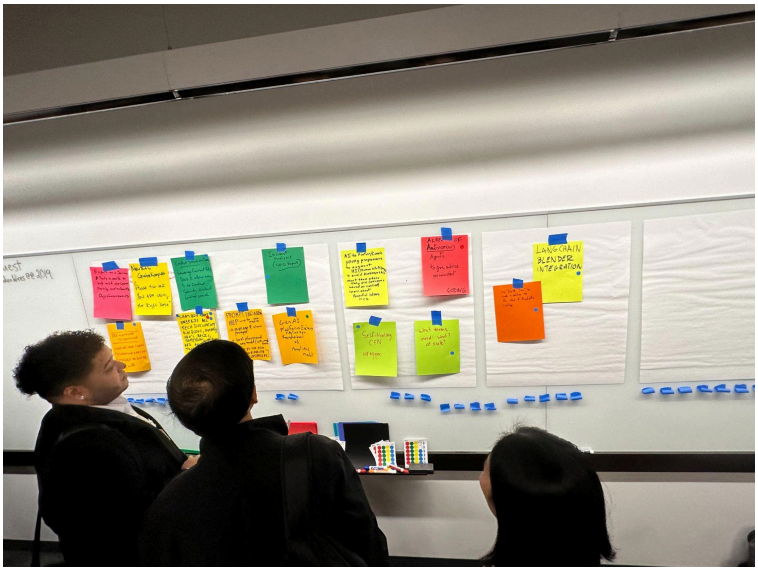
DevOps GAI Hackathon Projects and Readouts

There were around 60 attendees from various industries (large banks, healthcare, services, software vendors, and students). This workshop focused on solving IT infrastructure and operations problems related to DevOps and DevSecOps. For selection projects, we used an OpenSpaces format (i.e., sticky note/dot voting). Following some brief brainstorming, the teams self-organized and got to work. Amazing results were achieved.



The nine groups worked on DevOps and DevSecOps-related projects for the better part of the afternoon.

1) HuggingFace Project

John Willis gives an overview of what Joseph Enochs built on Hugginface. Joseph Enochs built an impressive tool hosted on HuggingFace that allows you to chunk and search text data easily. It shows the embeddings and similarity of text chunks to help you.

Video:

https://drive.google.com/file/d/1LVvF2XISCXcTBT3dkdyx9d4qmOJk4Pc6/view?usp=sharing
(Begins at 0:00 and ends at 5:18)

HuggingFace

https://huggingface.co/llmlocal

2) Void Project

John Willis loaded a database of public outage postmortems called "The Void" into MongoDB Atlas Vector Search. He was then able to ask questions about outages by specific companies in 2023. This demonstrates the power of ingesting PDF data into vector search.

Video:

https://drive.google.com/file/d/1LVvF2XISCXcTBT3dkdyx9d4qmOJk4Pc6/view?usp=sharing
(Begins at 5:18 and ends at 5:18)

Note Book Code

https://github.com/OperationalizingAI/Hackathon-1-24-24/blob/main/MDBLoad-SM-Void.ipynb
https://github.com/OperationalizingAI/Hackathon-1-24-24/blob/main/MDBRetrieve-SM-Void.ipynb

Void Database

https://www.thevoid.community/database

3) Identify issues, root causes, and hypotheses during an incident

The group presented a design for an AI agent to help quickly identify issues, root causes, and hypotheses during an incident at 2 am.

The agent has four main components:

- Memory - stores the state of the current situation

- Tools - API calls to get external data like change logs
- SME Experts (Mixture of Experts) - simulated experts to analyze things like stack traces
- Loop - identifies top issues and mitigation steps

There was a proof of concept to simulate an SME expert analyzing stack traces to identify a hypothesis of what went wrong. The presentation focused on tools and designs to leverage AI to quickly troubleshoot incidents and outages by ingesting relevant data sources.

1. The group experimented with ingesting stack traces from GitHub issues into a vector database to see if it could provide more helpful information than ChatGPT.
2. They imported 25 issues with stack traces but found that queries did not return helpful information from the rag (retrieved and generated knowledge base). Instead, ChatGPT's default answers were returned.
3. This is likely because LangChain is abstracting away the prompt sent to the LLM. The chain takes the ingested data and stuffs it into a prompt, but you lose control over exactly what data the LLM sees.
4. One option is not to use LangChain and carefully construct the prompt using only the relevant ingested data.
5. The group has more experiments planned around identifying configuration changes as a root cause of issues, using their AI agent's "change control expert" module.

In summary, the group learned that care must be taken in how ingested data is presented to the LLM via prompts to get useful results. But the hackathon has been a good experience for initial experiments and learning.

Videos:

https://drive.google.com/file/d/1LVvF2XISCXcTBT3dkdyx9d4qmOJk4Pc6/view?usp=sharing
(Begins at 8:25 and ends at 12:01)

https://drive.google.com/file/d/1LTRrdFxf-imZe_WqpCof0_pdxsDe0Qou/view?usp=sharing
(Begins at 0:00)

Note Book Code

(get from group)

4) Ingesting Markdown documentation into a vector database to power a "RAG" (retrieved and generated knowledge base) to make documentation and training more accessible.

Here is a summary of the key points from the presentation:

1. One group experimented with ingesting Markdown documentation into a vector database to power a "rag" (retrieved and generated knowledge base) to make documentation and training more accessible.
2. Key lessons:
    - Harvesting Markdown from websites into a rag is easy with an OpenAI API subscription
    - Important to identify questions people ask that don't get answered to improve documentation
    - Can use those unanswered questions to get a feedback loop to improve the documentation continuously
3. Another approach is using LLM models like LaMDA to check if an organization's practices adhere to industry best practices based on the documentation ingested.
    - The vector store should contain the "rules of engagement" reference data rather than shifting operational data.
4. Tools like LangeSmith provide observability into how LangChain processes ingested data and prompts behind the scenes.
    - Can also set up question/answer pairs to evaluate whether queries are getting better over time.

In summary, key lessons focused on documentation improvements, comparing operations to best practices, getting visibility into prompt processing, and setting up feedback loops for continuous improvements.

Videos:

https://drive.google.com/file/d/1lPxC-iFj79gtEO5ojYsSEEaGY1URE1Ti/view?usp=sharing

Note Book Code

NO CODE

5) LangChain to interact with Blender 3D design software via an OpenAPI (formerly Swagger) specification

Here is a summary of the key points from the presentation:

1. The group explored using LangChain to interact with Blender 3D design software via an OpenAPI specification.
2. Blender has a scene graph representing 3D objects, locations, rotations, etc. Operations like adding objects or transforming them modify the scene graph.
3. They created a FastAPI server with endpoints corresponding to Blender operations that return the updated scene graph JSON.
4. This allows execution of Blender operations by calling the REST API, which renders updated images after each call showing the scene changes.
5. Though they didn't have time to finish it, the goal is to use LangChain to handle natural language interactions to manipulate the Blender scene by calling the underlying API.
6. This could enable conversational creation and editing of 3D scenes and assets by end users without 3D expertise.
7. There are opportunities to expand this by generating assets with AI algorithms to import into scenes, applying textures and materials, animating objects, etc.

In summary, they demonstrated a proof of concept for using AI to power more intuitive 3D scene creation by linking natural language interactions to an API that operates on a 3D design application.

Videos:

https://drive.google.com/file/d/11d29FO2M1M3su_Y4pOOaMPiLeE256zbQ/view?usp=sharing
(Begins at 0:00)

Note Book Code

https://github.com/michaelgold/blendchain

6) Create an AI system that could recommend GitHub templates/starters for a user's desired project.

Here is a summary of the key points from the presentation:

1. The group set out to create an AI system that could recommend GitHub templates/starters for a user's desired project.

2. The proposed system would ingest data from GitHub repositories, including contents and READMEs, into a database.
3. Users could ask natural language questions that would be compared against the vector embeddings of the GitHub data to find relevant repositories.
4. Additional context from the selected repositories would be combined to provide an answer detailing the recommended template and reasoning why it fits the user's needs.
5. They demonstrated a proof of concept that took sample repository data with descriptions and then successfully recommended one that matched the requirements outlined in a complex sample question.
6. Next steps are to automate the continuous ingestion of GitHub data, fine-tune the model to improve context-based reasoning and recommendations from questions, and fully develop the application.

In summary, they designed an intelligent system for recommending GitHub templates based on repository analysis and presented early promising results from their prototype.

Videos:

https://drive.google.com/file/d/1tT3qUZq4e439DGYXcLqYuYKuxAMmwhRV/view?usp=sharing (Begins at 0:00)

Note Book Code

No Code

7) (Part 2) Create a system to ingest organizational documents like runbooks and manuals so users can ask natural language questions instead of spending hours searching documentation.

Here is a summary of the key points from the presentation:

1. The group set out to create a system to ingest organizational documents like runbooks and manuals so users could ask natural language questions instead of spending hours searching documentation.
2. They first tried ingesting Roland product manuals from a website into Amazon Q. But it took too long to sync and fine tune prompts.
3. Next, they uploaded 3 PDF manuals into S3 and connected them to Amazon Q, but could still not get good responses to questions.

4. Finally, they used Amazon Kendra to connect to the S3 bucket and automatically process/OCR the PDFs, allowing Amazon Q to index and vectorize the content for better responses.
5. They demonstrated Kendra + Q answering queries about making guitar sounds by pointing users to relevant sections of the equipment manuals.
6. They almost got a similar integration working with Superblocks, but the PDFs were too large.
7. Key lessons learned were ease of use with Kendra for ingestion/OCR compared to more customization possible with Superblocks and potential next steps for leveraging the indexed data with services like Bedrock.

In summary, they experimented with different techniques for ingesting and querying documentation, overcoming initial challenges to demonstrate a working solution with Kendra and Amazon Q.

Videos:

https://drive.google.com/file/d/1n2X2UISEtyPX0GkYQkV5X5xNm0vakFyu/view?usp=sharing (Begins at 0:00)

Note Book Code

Get code from the team

8) (Part 2) Amazon Q has a native retriever and integration with Kendra for ingesting and indexing data that can then be queried.

Here is a summary of the key points from the presentation:

1. Amazon Q has a native retriever and integration with Kendra for ingesting and indexing data that can be queried.
2. It also has adapters to import directly from sources like Confluence, which could be useful for documentation and runbooks.
3. While the native Q retrieval may be more limited, it provides a quick serverless option. For more customization, Kendra and Bedrock could be used together.
4. They set up a prototype that imported product manuals from S3 into Amazon Q and demonstrated it, answering some sample questions.
5. Some challenges included figuring out S3 access permissions and finding public URL access within Q. But overall, it was a valuable hands-on experience in a short time.
6. The key next steps would be integrating the Q chatbot into tools like Slack for easy conversational access and building custom prompts/rules with Bedrock.

In summary, they successfully built an initial working prototype to import documents and answer questions while identifying future areas for customization and integration to create more useful applications.

Videos:

https://drive.google.com/file/d/1C2wegRFf0QRA2LBZdkq01d8-SSOlyxsh/view?usp=sharing
(Begins at 0:00)

Note Book Code

Get code from the team

9) Design an AI tutor/mentor to assist people in learning to code.

Here is a summary of the key points from the presentation:

1. The group set out to design an AI tutor/mentor to assist people in learning to code.
2. Key components included:
    - Capturing context like coding language, skill level, problem being solved to personalize assistance
    - Summarizing the conversation to confirm understanding
    - Allowing users to select terms to learn more about
    - Providing a reflection window to give feedback on what went well/poorly
3. They implemented capturing user context and linking it to sample coding projects of appropriate complexity.
4. Additional areas designed but not implemented due to time included: conversation summarization, selecting terms to learn, and reflection.
5. The group presented screenshots of a prospective front-end to capture user input and back-end logic to provide personalized learning recommendations based on context.

In summary, they designed a framework for an AI coding tutor that adapts guidance based on user context and tools to improve understanding between the user and AI. Though only partially implemented, they successfully demonstrated initial components to personalize learning.

Videos:

https://drive.google.com/file/d/1C2wegRFf0QRA2LBZdkq01d8-SSOlyxsh/view?usp=sharing
(Begins at 0:00)

Note Book Code

Get code from Steve

10) Summary Remarks (IMG 5238)

I didn't have, I didn't know what expectations we were going to have to throw one of these things together and put everybody in rooms, you know, I was blown away. Honestly, well, and I've been doing this for years, but the things that you all did today were just, like, incredible. You all did a great job, so give yourselves a hand first.

I mean that from the bottom of my heart.

Videos:

https://drive.google.com/file/d/18yDQTsU5zlid-lVEB2Tr5Tf48802T7v4/view?usp=sharing
(Begins at 0:00)