Case Study 1- Development of an E-commerce Sales Chatbot

In the competitive sphere of e-commerce, deploying advanced, interactive, and user-centric sales chatbots is essential for elevating customer experience and streamlining business operations. This case study presents the challenge of developing a comprehensive sales chatbot tailored for an e-commerce platform specialising in a specific product category (such as electronics, books, or textiles). The aim is to facilitate customer interactions from product search to purchase.

Objective:
Design and implement a sales chatbot that enhances the shopping experience by enabling efficient search, exploration, and purchase processes on an e-commerce platform. The deliverables include:
• The chatbot interface and logic.
• A simulated e-commerce server that processes user queries from the chatbot and returns relevant product data. This server should handle a mock inventory via RESTful interactions
(The backend data can be a mock e-commerce inventory)

Requirements:

1. User Interface / Front End:
• Develop a responsive user interface compatible with desktop, tablet, and mobile devices, utilizing modern JavaScript frameworks alongside HTML5 and CSS.
• Implement a login and authentication module to secure user sessions.
• Manage session continuity to maintain user state throughout interactions.
• Design a simple, intuitive chatbot interface with features such as conversation reset buttons and session tracking with timestamps.
• Ensure all chat interactions are stored effectively for later retrieval and analysis.

2. Backend
• Create an API-driven backend system using Python with frameworks like Flask or Django, capable of processing search queries and fetching relevant product data from a database.
• Populate a relational database management system (RDBMS) with at least 100 mock e-commerce product entries.

3. Technical Documentation:
• Document the entire process, including architecture , choice of tools/frameworks, and mock data creation should be documented
• Bonus if the candidate can include a section on potential challenges faced and how they were handled

4. Code Quality and Best Practices:
• The code should be clean, readable, and well-commented, adhering to industry standards for maintainability and scalability. Code should be implemented in python
• Ensure the codebase is modular and fault-tolerant, with clear separation of concerns and robust error handling.
• Provide rational for the choice of frameworks, libraries, and design patterns used in the project.

Evaluation Criteria:

1. UI User Experience:
• Creativity applied while visualising the products that are fetched without compromising the UX
• Innovative ways considered and applied to enable to customer interact , filter and explore the products

2. Technical Implementation:

• Quality of code, including readability, structure, and adherence to best practices for both the client (chatbot and server module)
• Modular architecture and fault tolerance capabilities of the architecture

3. Innovation and Problem-Solving:
• Creativity in approach and solutions to challenges encountered during the project.
• Ability to leverage advanced UI design / web development techniques catering to seamless customer product search experience.

4. Documentation and Presentation:
• Clarity and completeness of technical documentation, including code comments, project setup, and execution instructions.
• Effectiveness in communicating the project's objectives, methodology, results, and learnings.

Deliverables:
• A GitHub repository containing all source code, complete with a detailed README.md outlining project setup, execution instructions, and a comprehensive project summary.
• A detailed project report showcasing the technology stack used, sample queries, and the results obtained.
• A presentation to the recruitment panel, detailing the project approach, technologies utilized, and key learnings.

Option 2: Cloud Engineering Intern

We're also on the hunt for a Cloud Engineer Intern who's looking for the below:
1. Hands-on training with AWS services (bye-bye, boring textbooks!)
2. Deep dive into cloud infrastructure that's actually cool
3. Master server monitoring, logging, and deploying APIs like a boss
4. Level up your tech skills from "meh" to "mind-blowing"

If you are interested please submit the below case study

Background: Build a local cloud-based system that processes CSV files, extracts metadata, and stores information using Localstack (a local AWS cloud stack emulator).

Core Deliverables:

1. CSV Processing Pipeline

• Upload CSV to S3 bucket
• Extract metadata (row count, column count, column names)
• Store metadata in database
• (Optional) Send notification on completion

2. Required Infrastructure

• S3 bucket for file storage
• Lambda function for processing of the file.
• Choice of database:
    a. AWS RDS (MySQL/PostgreSQL) OR
    b. DynamoDB
    c. Below is a sample metadata extracted from csv file.

```
{
'filename': 'example.csv',
'upload_timestamp': '2024-12-14 10:00:00',
'file_size_bytes': 1048576,
```

'row_count': 1000,
'column_count': 5,
'column_names': ['id', 'name', 'age', 'city', 'date']
}

Development Framework:

Localstack is an open-source framework that provides a fully functional local AWS(Amazon Web Services) cloud stack. It allows you to develop and test your cloud applications locally without incurring AWS costs. The community edition supports core AWS services like S3, Lambda, and DynamoDB.

Key Requirements

• Process CSV files up to 10MB. S3 event triggers Lambda function automatically
• Handle basic error scenarios
• Store metadata (rows, columns, timestamps) in a database – DynamoDB or AWS RDS
• Include brief setup documentation

Reference Links:

1. Core Documentation

- Localstack: https://docs.localstack.cloud/getting-started/
- AWS Lambda Python: https://docs.aws.amazon.com/lambda/latest/dg/lambda-python.html
- https://github.com/localstack

2. SDK References

- Boto3: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html
- pandas: https://pandas.pydata.org/docs/