



Hack the Future: A Gen AI Sprint Powered by Data

Work at the heart of *change*

Data and AI Week



Problem Statement 1: Optimizing Retail Inventory with Multi Agents

Challenge Overview:

In the rapidly evolving retail industry, maintaining an optimal balance between product availability and inventory costs is a key challenge. Retail chains often face issues of stockouts (running out of popular items) or overstocking (leading to higher holding costs). To address these challenges, we invite you to design a multi-agent AI system that collaborates between stores, warehouses, suppliers, and customers to optimize inventory management.

The goal is to create a multi-agent AI system that can predict demand, ensure product availability, reduce inventory holding costs, and improve supply chain efficiency. Your solution should enable seamless collaboration among different agents to manage inventory proactively, avoid stockouts, and minimize the excess holding of inventory, thereby maximizing sales and improving operational efficiency.

Current process:

- **Demand Forecasting:** Retail managers review historical sales data, market trends, and seasonal patterns. Manual calculations or basic forecasting models are used to estimate future demand for various products. These estimates are shared with warehouses and stores to prepare for anticipated demand.
- **Inventory Monitoring:** Retail managers regularly perform physical stock checks at stores and warehouses. Inventory records are manually updated in the system to reflect stock levels. Discrepancies between actual stock levels and the recorded inventory are investigated manually, often requiring manual audits.
- **Pricing Optimization:** Retail managers analyze stock levels and manually adjust prices on slow-moving inventory to encourage sales. Price changes are communicated to all stores and online platforms. Sales trends and customer feedback are periodically analyzed to assess the impact of pricing changes.

Expected Technical Output: Multiagent framework



Problem Statement 2: Smart Shopping: Data and AI for Personalized E-Commerce

Challenge Overview:

In the competitive world of e-commerce, providing personalized and relevant product recommendations is key to improving customer experience, increasing conversion rates, and boosting sales.

In this challenge, your task is to build a multi-agentic AI system that delivers hyper-personalized product recommendations for an e-commerce platform. This system will utilize different agents representing customers, products, and recommendation engines, all collaborating to analyze browsing behavior, predict user preferences, and optimize the overall shopping experience. Your Data & AI system should enable the e-commerce platform to deliver tailored recommendations based on each customer's behavior and interests, ultimately improving engagement, increasing average order value, enhancing customer retention, and driving higher conversion rates.

Current Process:

- **Customer Data Collection and Segmentation:** E-commerce teams manually collect and store customer browsing behavior, purchase history, and demographic data in databases or spreadsheets. Customer profiles are created based on specific attributes like age, gender, location, and purchase history, which are used to manually segment customers into different categories (e.g., frequent buyers, new visitors). Retail managers or marketing teams manually analyze the customer data to identify patterns, trends, and preferences, which can be used to make product recommendations.
- **Product Recommendations :** Based on the customer profile and segmentation, the marketing team manually selects a set of products to recommend to each customer segment. For example, “frequent buyers” might receive recommendations of related products, while “new visitors” could see the most popular products or discounts.

Expected Technical Output: Multiagent framework and SQLite Database for long term memory

Problem Statement 3: Data-Driven AI for Sustainable Farming

Challenge Overview:

Agriculture plays a vital role in sustaining life, but its environmental and economic impact is substantial. With growing challenges like water scarcity, excessive pesticide use, and soil degradation, the need for more sustainable agricultural practices has never been greater. This hackathon aims to leverage AI technologies to create innovative solutions that promote sustainability, optimize resource usage, and improve the livelihoods of farmers.

Develop a multi-agentic AI system that brings together different stakeholders in agriculture—farmers, weather stations, and agricultural experts—to work collaboratively for the optimization of farming practices.

The goal is to reduce environmental impact of farming: Promote practices that lower the carbon footprint, minimize water consumption, and reduce soil erosion.

Current Process:

- **Farmer Advisor:** Provides actionable insights by analyzing input from the farmer about land, crop preferences, and financial goals.
- **Market Researcher :** Analyzes regional market trends, crop pricing, and demand forecasts to suggest the most profitable crops to plant.

Expected Technical Output: Multiagent framework and SQLite Database for long term memory

Problem Statement 4: Empowering Elderly Care with multi agent AI system

Challenge Overview:

As the global population ages, ensuring the well-being of elderly individuals living independently presents a major challenge.

The goal of this hackathon is to develop a multi-agentic AI system that assists elderly individuals by providing real-time monitoring, reminders, and safety alerts, while promoting health management and social engagement. The agents will work together to create a collaborative support system, involving caregivers, healthcare providers, and family members to ensure optimal care and peace of mind.

The system should monitor health, detect unusual behavior, and provide alerts in case of emergencies. It should also provide reminders to manage daily routines, such as medication schedules, appointments, and daily activities.

Current Process:

- **Healthcare Provider:** Monitors health data (e.g., heart rate, blood pressure, glucose levels) through wearable devices and gets alerted if abnormal values are detected.
- **Safety Monitoring System:** Uses sensors or wearables to track movement, activity, and falls. If an elderly person falls or exhibits unusual behavior (e.g., remaining stationary for too long), the system triggers an alert.
- **Daily activity Reminder System:** Sends reminders in form of voice notes to the elderly individual for medication intake, scheduled appointments or daily activities.

Expected Technical Output: Multiagent framework

Problem Statement 5: Enhancing Job Screening with AI and Data Intelligence

Challenge Overview:

The recruitment process often involves manually reviewing numerous job descriptions (JDs) and CVs, which can be time-consuming and prone to human error. The goal of this hackathon is to develop a multi-agent AI system that can automatically read and summarize job descriptions (JDs), match candidate qualifications with the JD, shortlist candidates, and send interview requests based on the match

Current Process:

- **Job Description Summarizer:** Reads and summarizes key elements from the JD, including required skills, experience, qualifications, and job responsibilities.
- **Recruiting agent:** Extracts key data from CVs, such as education, work experience, skills, certifications, and other relevant information. Compares the extracted data from the CV to the summarized JD and calculates a match score based on the relevance of the candidate's qualifications, experience, and skills.
- **Shortlisting Candidates:** Based on the match score, candidates who meet or exceed a defined threshold (e.g., 80% match) are shortlisted for interviews.
- **Interview Scheduler:** Sends personalized interview requests to shortlisted candidates, including potential dates, times, and interview format written in email.

Expected Technical Output: Multiagent framework and SQLite Database for long term memory

Problem Statement 6: Recommendation and design of Customer 360 data product for retail banking customers with agentic solution

Challenge Overview:

Recommending and designing data products involves a structured approach that integrates data engineering and business strategy. This needs a combined effort of data stewards, domain owners, and data engineers. The purpose of the hackathon is to create a multi-agent solution that can – a) understand the specific business use case, b) recommend the target data product structure to fulfill the use case, c) identify the source systems and attributes required to build the data product, d) and create mapping of the source system and target data product attributes. Post the agentic flow, define data product ingress and egress processes and certify the data product against standards (manual certification)

Current Process:

- **Use case:** A detailed use case specifying the data consumer's requirements
- **Target Data Product Design:** Considering the business use case, and existing source systems and attributes, the data designer designs the structure of the target data product(s)
- **Source Attribute Identification:** Identify the source attributes from which values needs to be populated in the target data product(s). Use the existing functional descriptions for the attributes if available in data governance catalog.
- **Mapping Generation:** Create the mapping file covering all the attributes in target data products, having source attribute, target attribute, mapping (direct map or transformation), transformation specifications etc.
- **Data Product Certification :** Define ingress and egress processes, data stores, searchability approach etc. and certify the data product against standards

Problem Statement 7: **AI-Driven Customer Support: Enhancing Efficiency Through Multi-agents**

Challenge Overview:

In modern enterprises, delivering efficient and high-quality customer support is a critical challenge. As businesses scale, customer queries increase in volume and complexity, requiring rapid and accurate resolution. However, customer support teams often struggle with delays in response times, inconsistent resolutions, and inefficient routing of tasks to other teams. To enhance efficiency and customer satisfaction, organizations need an AI-driven system that can summarize conversations, extract actionable insights, recommend resolutions based on historical data, route tasks effectively, and optimize resolution time. The goal is to develop a multi-agent AI system that streamlines customer support operations by automating key processes such as summary generation, action extraction, resolution recommendation, task routing, and resolution time estimation. This system should facilitate seamless collaboration among support agents, technical teams, and business units to ensure quicker, more accurate, and cost-effective customer issue resolution.

Current process:

- **Summary Generation & Action Extraction:** Customer support agents manually review and summarize customer queries, identifying key points and action items like escalations or follow-ups. These actions are then recorded and assigned manually in the ticketing system.
- **Routing Actions for Other Teams:** Manual ticket assignments and follow-ups cause delays due to reliance on agent judgment and predefined rules. Escalations are handled via emails or internal tools, leading to inefficiencies. Misrouted tasks further extend resolution time and require reassignments.
- **Resolution Recommendation through Historical Data:** Search past tickets and knowledge bases for similar cases, recommend solutions based on experience and documentation, and escalate unresolved issues. Inconsistent resolutions arise due to a lack of standardized recommendations.
- **Resolution Time Estimation and Minimization:** Support teams rely on experience rather than data for resolution time estimates, leading to inefficiencies. Manual or SLA-based ticket prioritization creates bottlenecks, while the absence of predictive analytics hinders delay anticipation and workflow optimization. As a result, customers face delays in issue resolution.

Expected Technical Output: Multiagent framework

Technology for AI

- Ollama based on-prem LLMs
- Custom tools for agents– API , web srapper, ML model etc..
- Ollama based embedding models
- SQLite DB
- Multi agent framework

Ollama

Step-by-Step Guide to Using Ollama with Python:

System Requirements

OS	Processor	RAM	Storage	Python Version
Windows	x86_64 (Intel/AMD)	8GB (16GB recommended)	10GB+ free space	3.8 or higher
macOS	Apple Silicon (M1/M2) or Intel x86_64	8GB (16GB recommended)	10GB+ free space	3.8 or higher
Linux	x86_64 or ARM64 CPU	8GB (16GB recommended)	10GB+ free space	3.8 or higher



Ollama

Step1: Install Ollama

Ollama allows you to run AI models locally on your system.

- **Go to the official Ollama website:** <https://ollama.com>
- **Download the installer** for your operating system:
 - **Windows:** Download the .exe file and run it.
 - **Mac:** Download the .dmg file and install it.
 - **Linux:** Open a terminal and run:
curl -fsSL <https://ollama.com/install.sh> | sh
- **Follow the installation steps** to complete the setup.
- **Verify the installation** by running:
 - ollama version

If a version number appears, the installation was successful.

Step2: Download a Model

Now, download a lightweight AI model for your system.

- Open a terminal or command prompt.
- Run the following command to download the model:
- ollama pull mistral

(You can replace "mistral" with another lightweight model from the suggestions below.)

Step3: Install Required Python Package

To use Ollama with Python, install its Python package:

- pip install ollama

Step4: Run a Model Using Python

Now, let's write a simple Python script to interact with the AI model.

- Open a text editor or IDE (like VS Code, PyCharm, or even Notepad++).
- Create a new Python file (e.g., run_model.py).
- Copy and paste the following code:

```
import ollama
# Define the model name (use a lightweight model like "mistral")
model_name = "mistral"
# Define the user input prompt
prompt = "Explain what Agent AI is in simple words."
# Run the model and get a response
response = ollama.chat(model=model_name, messages=[{"role": "user", "content": prompt}])
# Print the response
print("AI Response:", response["message"]["content"])
• Save the file and run it using:
python run_model.py
```

Ollama

Here are a few recommended models:

Model Name	Description
TinyLlama-1.1B	Very lightweight, fast, good for quick demand forecasting.
Gemma-2B	Optimized for efficiency, handles structured data.
Phi-2	Great for reasoning and small-scale predictions.
Flan-T5 Small	Good for fine-tuning and specific NLP tasks.
DistilBERT	Super compact, great for text classification and embeddings.

To explore more models, visit: <https://ollama.com/library>.



We are looking for following from you:

- Agents' interaction design
- Technical approach slide
- Code structure
- Demo Video
- Final presentation which includes Problem statement, Final approach, Team, Solution, Benefit and Impact.

Research Agent Guidelines

Guidelines for Ethical Web Scraping

When using web scraping to gather data for personalizing e-commerce experiences, it is important to follow ethical guidelines to respect the privacy and rights of website owners, users, and the public. Here are some key ethical principles to follow:

1. Respect Website Terms of Service

- Always review and adhere to a website's Terms of Service (ToS) and robots.txt files to ensure that web scraping is allowed. If the site explicitly prohibits scraping in its terms, avoid scraping that site.

2. Data Privacy and User Consent

- Ensure that the data you collect does not violate user privacy or data protection regulations such as GDPR (General Data Protection Regulation) or CCPA (California Consumer Privacy Act).
- If personal data is scraped, avoid using sensitive information (like emails, passwords, etc.) unless explicit consent is given.

3. Scraping Frequency and Load

- Do not overwhelm the website's server by sending too many requests in a short period. Scrape at a reasonable frequency to avoid causing performance issues or downtime.
- Use appropriate delays between requests to avoid overloading the server and to minimize impact on the website's normal operation.

4. Data Anonymization

- When collecting personal data, ensure that any personally identifiable information (PII) is either not scraped or is anonymized to prevent misuse or breaches of privacy.
- Avoid scraping data that can directly identify individuals







5. Fair Use and Avoiding Copyright Violation

- Avoid scraping copyrighted content, especially if it is not in the public domain, without permission. For example, scraping product descriptions or images may violate intellectual property rights.
- Ensure that the data you scrape is used fairly and that it doesn't infringe on copyright laws or harm the website owners' business model.

Synthetic Data link

Access the Synthetic Dataset:

[Click here to access](#)

	[use case 1] Inventory Optimization for Re...
	[Usecase 2] Personalized Recommendatio...
	[Usecase 3] AI for Sustainable Agriculture
	[Usecase 4] AI for Elderly Care and Support
	[Usecase 5] AI-Powered Job Application S...
	[Usecase 7] AI-Driven Customer Support E...

Thank You

