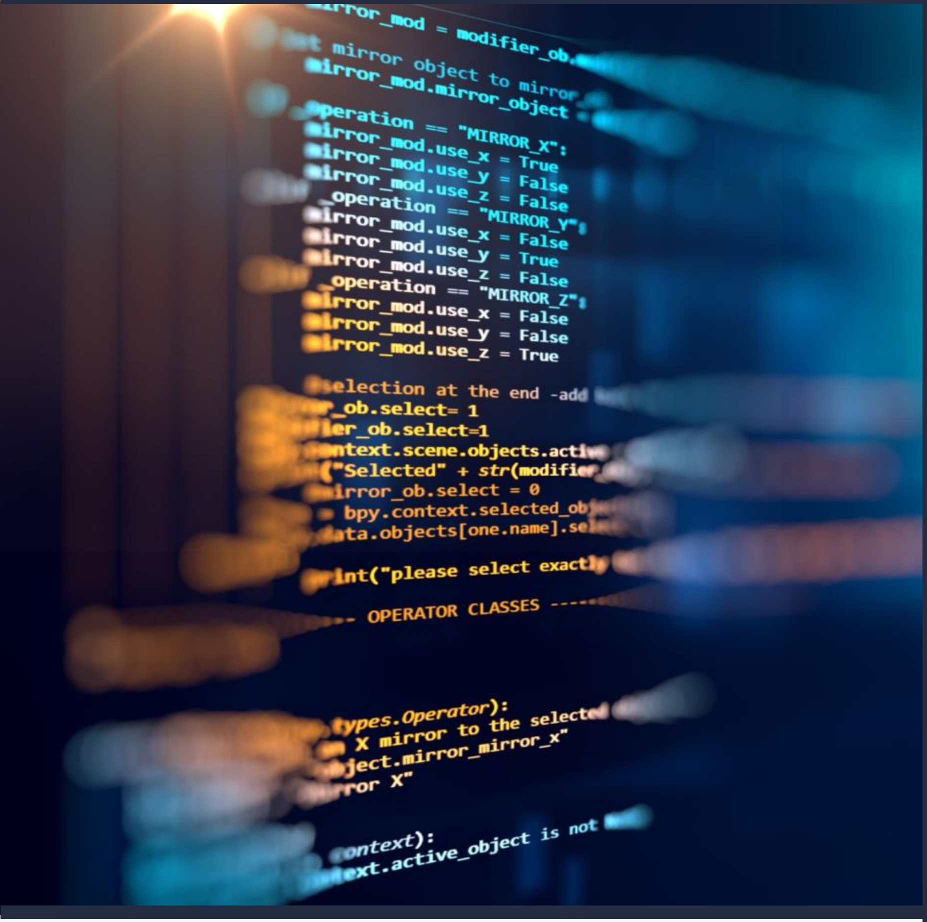


HACKEREARTH RAG APPLICATION – SPRINT 1

Ethan Villalovoz, Molly Iverson, Adam Shtrikman, Chandler Juego

Introduction

We will develop a RAG (Retrieval-Augmented Generation) application for HackerEarth that will utilize vector search, knowledge graphs, and a LLM to answer questions and generate content from a knowledge base of more than 10,000 Wikipedia articles.

Sprint Objectives

1. Collect client requirements
2. Create user stories, use cases, and UML diagrams for the RAG application
3. Identify the necessary RAG application tools and full-stack frameworks, and begin learning

Requirements Gathering Process

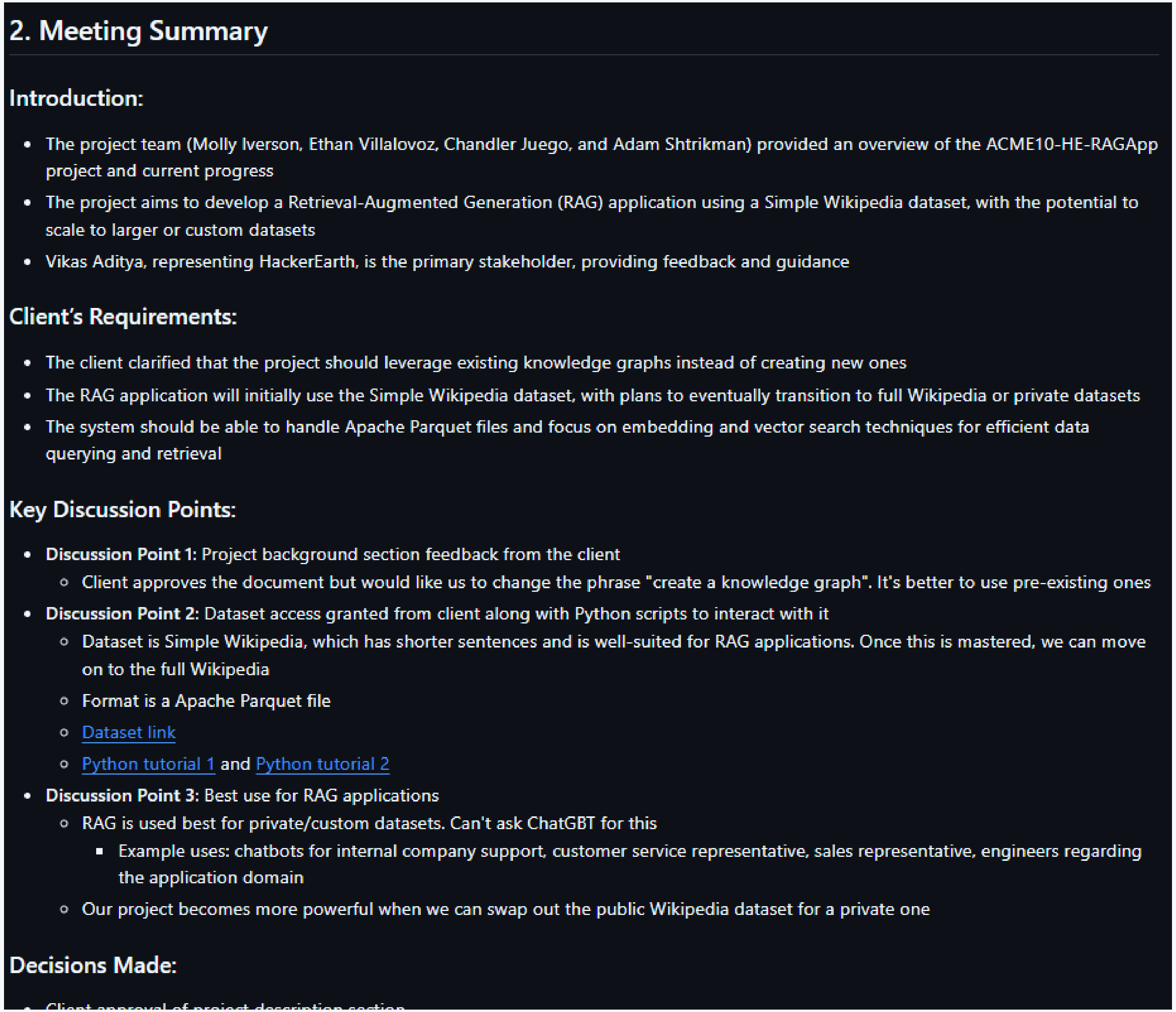
Three client meetings:

- 9/10: Introduction Meeting
- 9/24: Report feedback, dataset questions, best use of RAG apps
- 10/1: Report feedback, tech stack recommendations

Techniques for gathering requirements:

- Stakeholder meetings
- Observation/research of current RAG apps
- Project abstract

Meeting Notes 9/24



2. Meeting Summary

Introduction:

- The project team (Molly Iverson, Ethan Villalovoz, Chandler Juego, and Adam Shtrikman) provided an overview of the ACME10-HE-RAGApp project and current progress
- The project aims to develop a Retrieval-Augmented Generation (RAG) application using a Simple Wikipedia dataset, with the potential to scale to larger or custom datasets
- Vikas Aditya, representing HackerEarth, is the primary stakeholder, providing feedback and guidance

Client's Requirements:

- The client clarified that the project should leverage existing knowledge graphs instead of creating new ones
- The RAG application will initially use the Simple Wikipedia dataset, with plans to eventually transition to full Wikipedia or private datasets
- The system should be able to handle Apache Parquet files and focus on embedding and vector search techniques for efficient data querying and retrieval

Key Discussion Points:

- **Discussion Point 1:** Project background section feedback from the client
  - Client approves the document but would like us to change the phrase "create a knowledge graph". It's better to use pre-existing ones
- **Discussion Point 2:** Dataset access granted from client along with Python scripts to interact with it
  - Dataset is Simple Wikipedia, which has shorter sentences and is well-suited for RAG applications. Once this is mastered, we can move on to the full Wikipedia
  - Format is a Apache Parquet file
  - Dataset link
  - Python tutorial 1 and Python tutorial 2
- **Discussion Point 3:** Best use for RAG applications
  - RAG is used best for private/custom datasets. Can't ask ChatGBT for this
    - Example uses: chatbots for internal company support, customer service representative, sales representative, engineers regarding the application domain
  - Our project becomes more powerful when we can swap out the public Wikipedia dataset for a private one

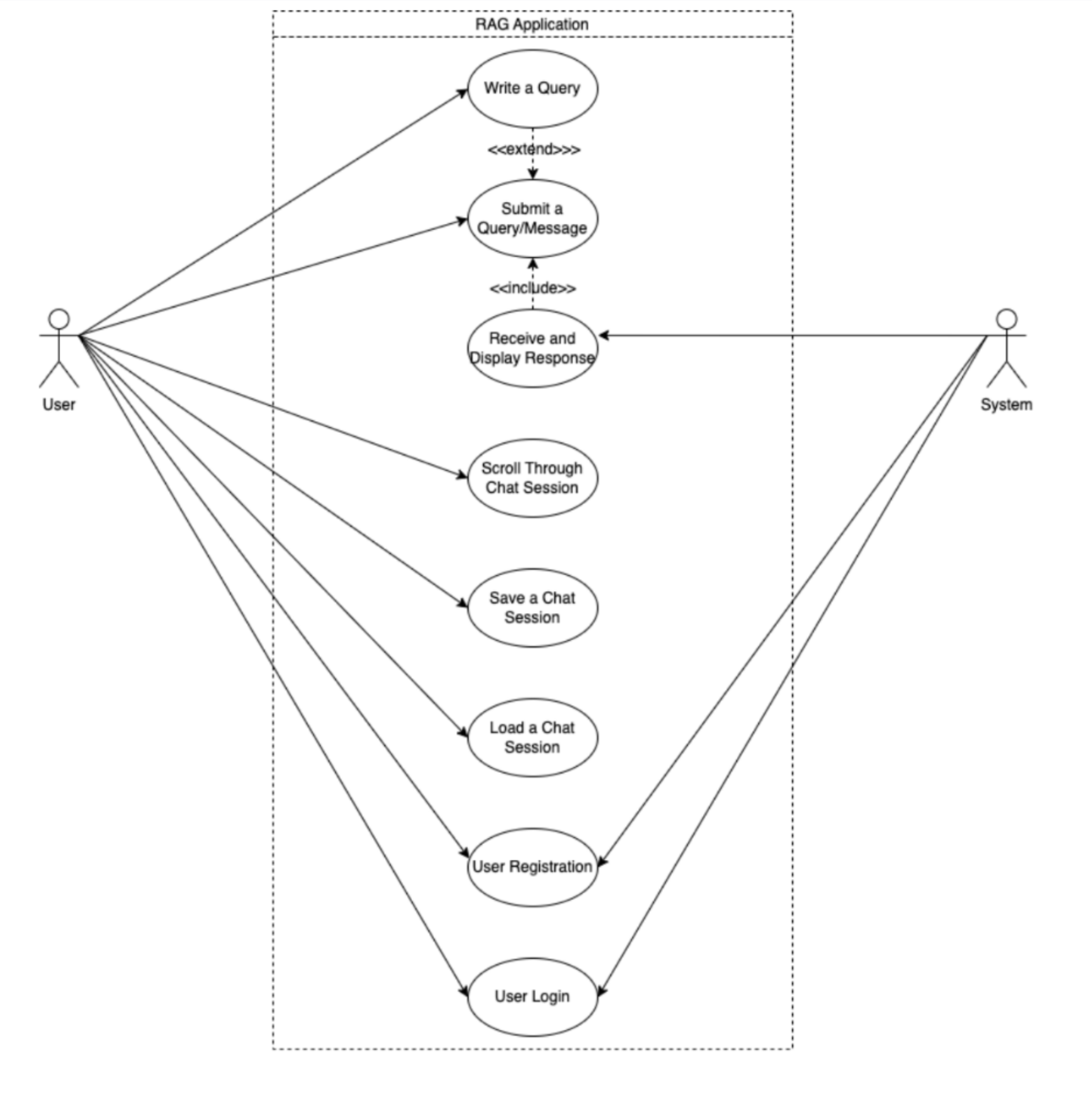
Decisions Made:

- Client approval of project description section

Requirements

- **RAG Architecture**: Combine retrieval and generation to improve how responses are produced.
- **Knowledge Graphs**: Use DBpedia and SPARQL to organize and access structured knowledge.
- **Vector Search**: Implement FAISS to quickly find relevant information.
- **Wikipedia Dataset**: Work with over 10,000 Wikipedia articles to answer queries.
- **LLM Integration**: Use LLaMA to generate responses that are both accurate and relevant.
- **Handling Queries**: Accept user questions, fetch the right information, and generate accurate responses.
- **Final Presentation**: Present the system’s performance and key takeaways.

Documentation of User Stories, Use Cases, and UML Diagrams



Kanban Overview and Contributions

Summary:

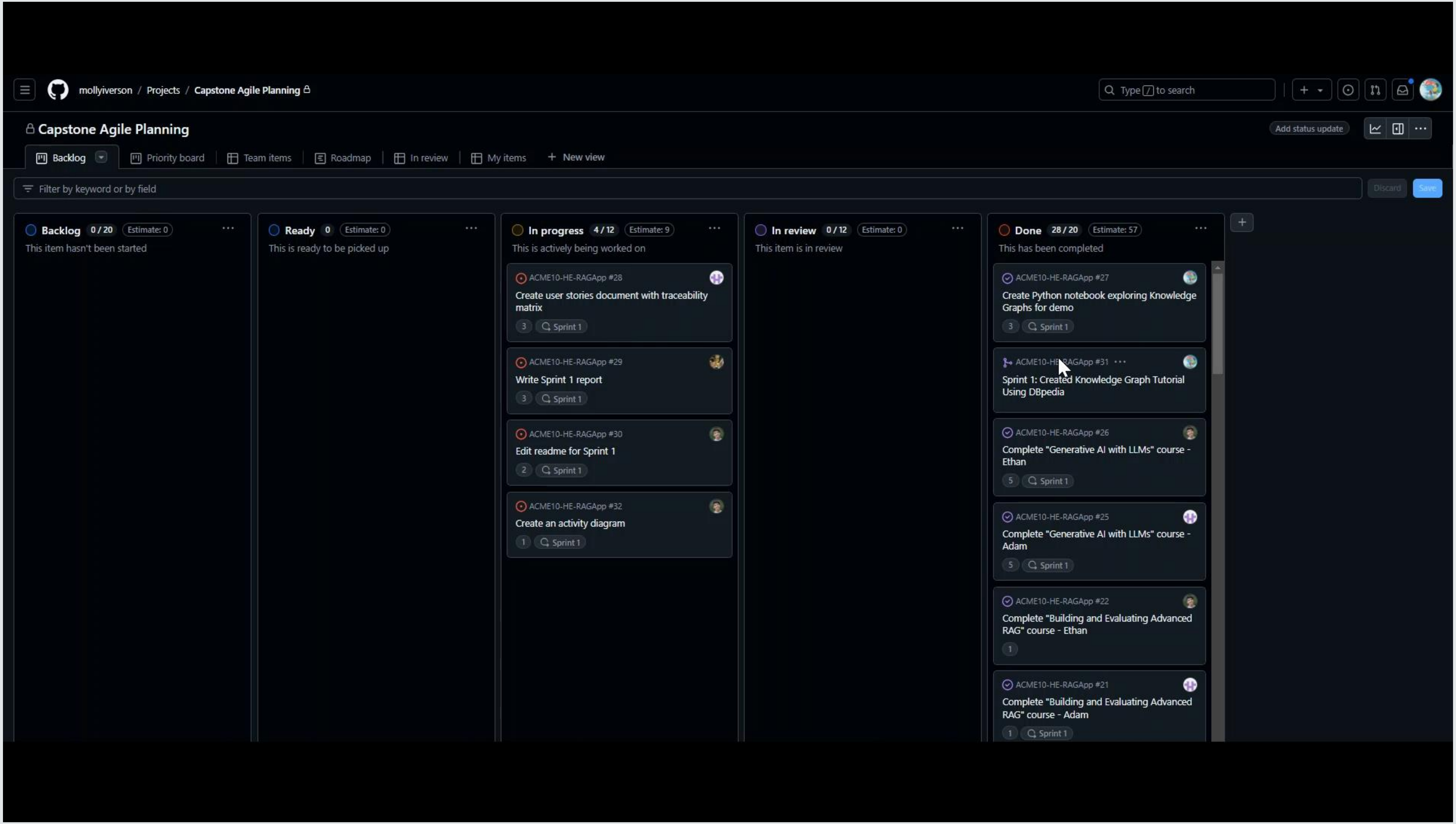
- Molly
  - Background and related work section
  - Functional requirements section
  - Knowledge Graph Python demo code

- Ethan
  - Introduction section
  - Non-functional requirements section
  - Use case and activity diagrams

- Chandler
  - Client and stakeholder information section
  - Use cases section
  - Sprint report

- Adam
  - Project overview section
  - System evolution section
  - User stories document

- All
  - Learning new skills and completing online courses
  - Meeting notes for client meetings



Capstone Agile Planning

Backlog 0 / 20 Estimate: 0
This item hasn't been started

Ready 0 Estimate: 0
This is ready to be picked up

In progress 4 / 12 Estimate: 9
This is actively being worked on

- ACME10-HE-RAGApp #28 — Create user stories document with traceability matrix — 3 — Sprint 1
- ACME10-HE-RAGApp #29 — Write Sprint 1 report — 3 — Sprint 1
- ACME10-HE-RAGApp #30 — Edit readme for Sprint 1 — 2 — Sprint 1
- ACME10-HE-RAGApp #32 — Create an activity diagram — 1 — Sprint 1

In review 0 / 12 Estimate: 0
This item is in review

Done 28 / 20 Estimate: 57
This has been completed

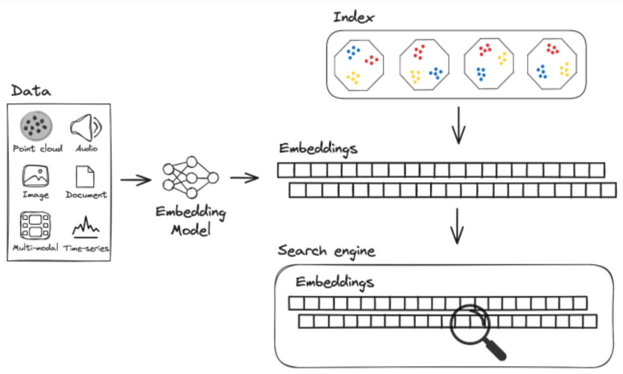
- ACME10-HE-RAGApp #27 — Create Python notebook exploring Knowledge Graphs for demo — 3 — Sprint 1
- ACME10-HE-RAGApp #31 — Sprint 1: Created Knowledge Graph Tutorial Using DBpedia
- ACME10-HE-RAGApp #26 — Complete "Generative AI with LLMs" course - Ethan — 5 — Sprint 1
- ACME10-HE-RAGApp #25 — Complete "Generative AI with LLMs" course - Adam — 5 — Sprint 1
- ACME10-HE-RAGApp #22 — Complete "Building and Evaluating Advanced RAG" course - Ethan — 1
- ACME10-HE-RAGApp #21 — Complete "Building and Evaluating Advanced RAG" course - Adam — 1 — Sprint 1

Skills Identification and Learning

- **Vector Search and Knowledge Graphs**
  - VS: BERT for embedding, FAISS (Facebook AI Similarity Search) for search
  - KG: DBPedia, queried using SPARQL
  - LLM (Response Generation): LLaMA

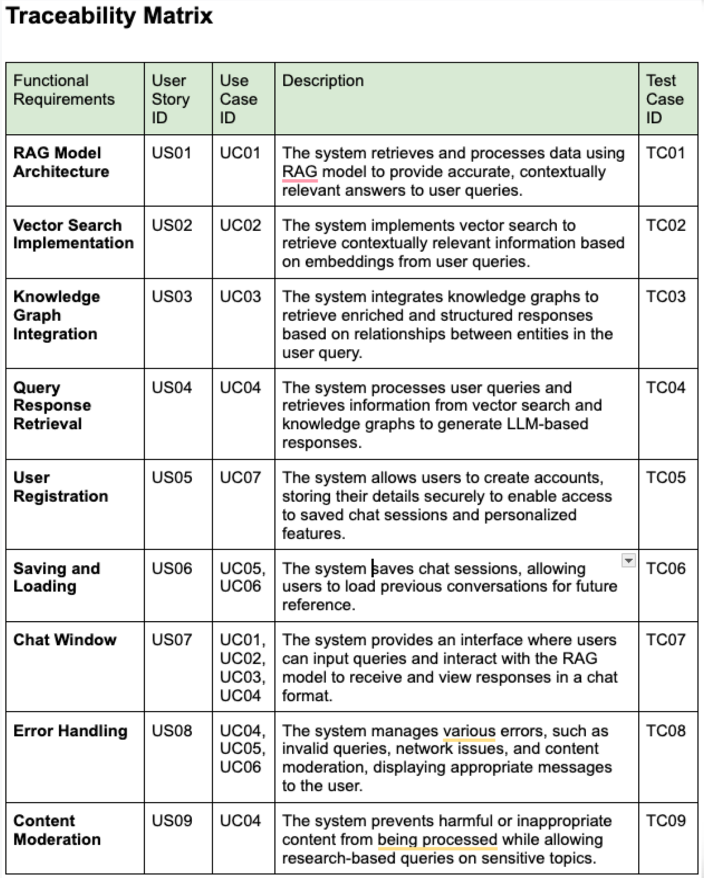
- **Front-End Development**:
  - React library

- **Back-End Development**:
  - FastAPI for RESTful APIs in Python



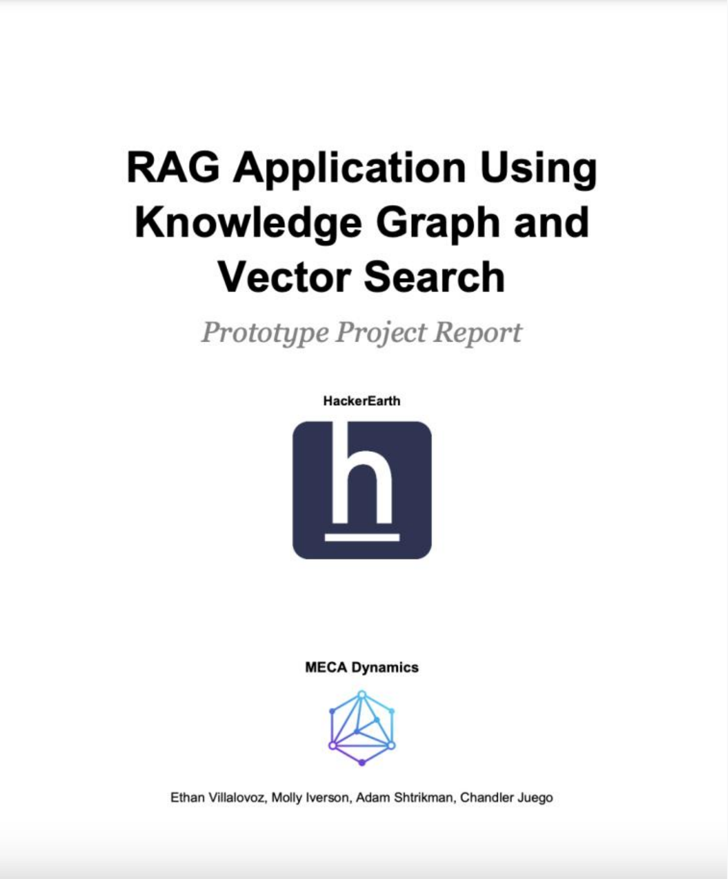
Sprint Achievements and Challenges

- **Looking back on Sprint 1.**



Traceability Matrix

Functional Requirements	User Story ID	Use Case ID	Description	Test Case ID
RAG Model Architecture	US01	UC01	The system retrieves and processes data using RAG model to provide accurate, contextually relevant answers to user queries.	TC01
Vector Search Implementation	US02	UC02	The system implements vector search to retrieve contextually relevant information based on embeddings from user queries.	TC02
Knowledge Graph Integration	US03	UC03	The system integrates knowledge graphs to retrieve enriched and structured responses based on relationships between entities in the user query.	TC03
Query Response Retrieval	US04	UC04	The system processes user queries and retrieves information from vector search and knowledge graphs to generate LLM-based responses.	TC04
User Registration	US05	UC07	The system allows users to create accounts, storing their details securely to enable access to saved chat sessions and personalized features.	TC05
Saving and Loading	US06	UC05, UC06	The system saves chat sessions, allowing users to load previous conversations for future reference.	TC06
Chat Window	US07	UC01, UC02, UC03, UC04	The system provides an interface where users can input queries and interact with the RAG model to receive and view responses in a chat format.	TC07
Error Handling	US08	UC04, UC05, UC06	The system manages various errors, such as invalid queries, network issues, and content moderation, displaying appropriate messages to the user.	TC08
Content Moderation	US09	UC04	The system prevents harmful or inappropriate content from being processed while allowing research-based queries on sensitive topics.	TC09



RAG Application Using Knowledge Graph and Vector Search

Prototype Project Report

HackerEarth

MECA Dynamics

Ethan Villalovoz, Molly Iverson, Adam Shtrikman, Chandler Juego

- Req. Gathering, Client Meetings, Documentation, Technical Learning, Project Report

Next Steps and Retrospectives

- **Looking towards Sprint 2.**

- Develop detailed system overview
- Create architecture, data, and user interface design
- Code a prototype
- Continue client meetings and project report

Conclusion

- **Sprint 1 Achievements:**
- Collected and analyzed client requirements
- Completed user stories, use cases, UML diagrams
- Identified tools: Vector search, knowledge graphs, LLMs
- Began learning necessary tech stack (React, FastAPI, FAISS)
- **Next Steps:**
- Develop system architecture and interface design
- Start coding the prototype
- Continue client meetings and feedback integration

Demo