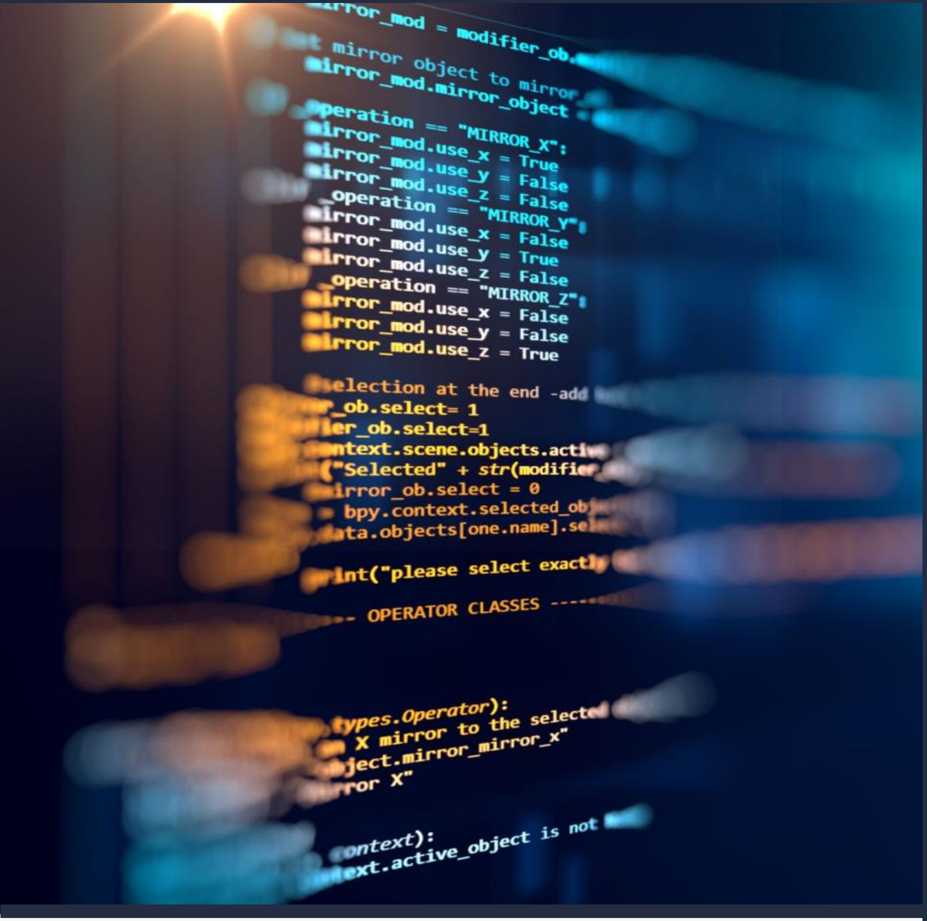


# HACKEREARTH RAG APPLICATION – SPRINT 2

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

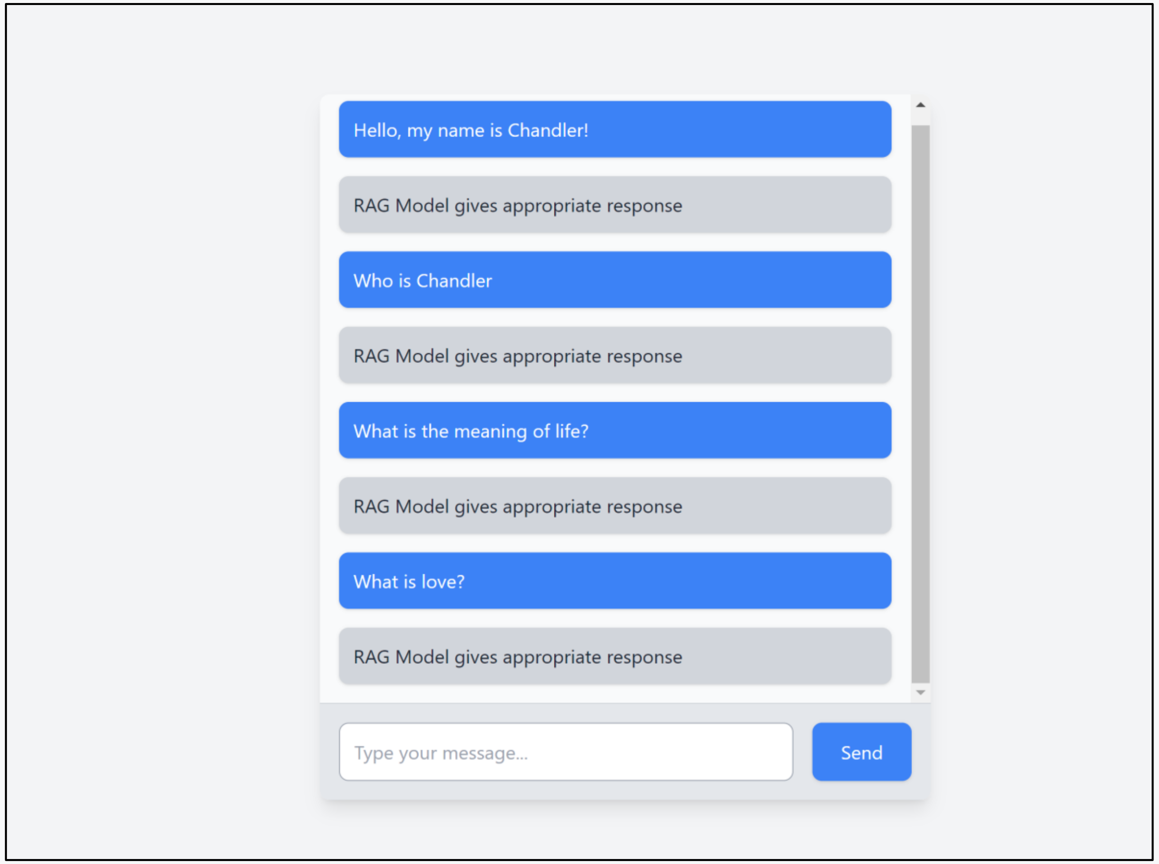
- **Embeddings Creation and Vector Search Preparation**: To implement efficient text embeddings, we researched and tested embedding methods, specifically focusing on BERT-based embeddings for use in the vector search module.
- **Frontend Prototype Development**: Developing a minimalistic chat-style interface to enable seamless interaction between users and the RAG application.
- **Knowledge Graph Querying**: Initial setup and testing of DBpedia integration to enable semantic data retrieval for more accurate responses.
- **System Architecture Refinement**: Documentation updates on the system architecture to ensure a clear breakdown of subsystems, including NLP, embeddings, and data processing workflows.

# Feature Implementation

| Team Member | Feature | Objective |
|---|---|---|
| Ethan | Embeddings | Embeddings Creation and Vector Search Preperation |
| Chandler | Frontend | Frontend Prototype Development |
| Molly | Knowledge Graph | Knowledge Graph Querying |
| Adam | Natural Language Processing | Query Processing |

# Figma Prototype & Client Feedback

- Minimalistic chat/messaging system with the RAG model



Hello, my name is Chandler!

RAG Model gives appropriate response

Who is Chandler

RAG Model gives appropriate response

What is the meaning of life?

RAG Model gives appropriate response

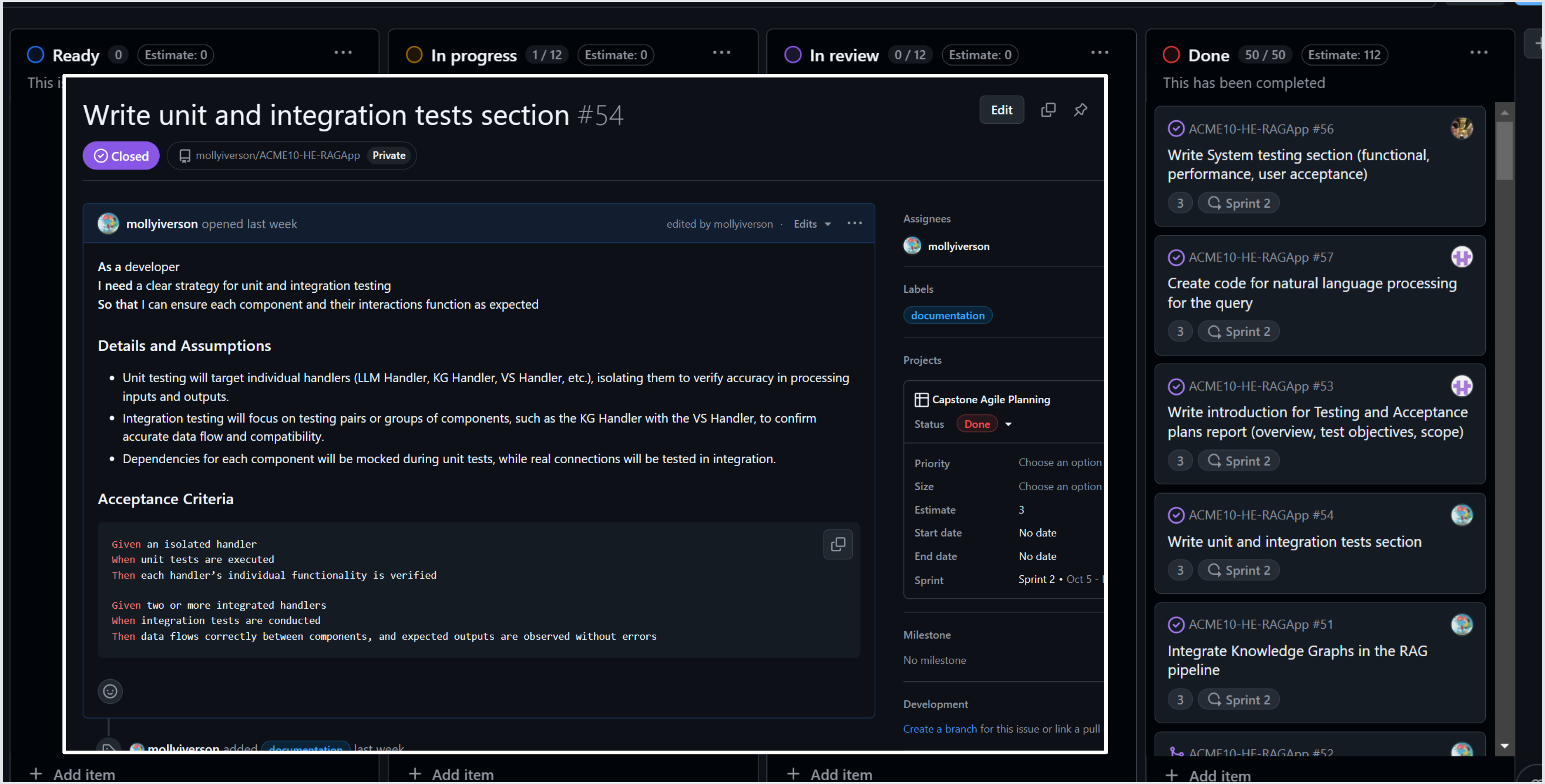
What is love?

RAG Model gives appropriate response

# Kanban Overview & Contributions

**Summary:**

- **Molly**
  - Knowledge Graph code
  - Architecture diagram and subcomponent decomposition report sections
  - Unit and Integration testing report sections
  - Project Milestone document
- **Ethan**
  - Text embeddings research and code
  - Architecture introduction and system overview report sections
  - Testing strategy and environment requirements report sections
  - Sprint report
- **Chandler**
  - React Frontend prototype
  - User Interface design report section
  - System testing report section (functional, performance, and user acceptance testing)
  - Python code for extracting and embedding Wikipedia data
- **Adam**
  - Natural Language query processing code
  - Data Design and some subcomponent report sections
  - Testing overview, test objectives, and scope report sections
  - Python notebook with FAISS tutorial
- **All**
  - Meeting notes for client meetings



Ready 0 Estimate: 0
This i
In progress 1 / 12 Estimate: 0
In review 0 / 12 Estimate: 0
Done 50 / 50 Estimate: 112
This has been completed

# Write unit and integration tests section #54

Edit
Closed
mollyiverson/ACME10-HE-RAGApp Private
mollyiverson opened last week
edited by mollyiverson · Edits

**As a** developer
**I need** a clear strategy for unit and integration testing
**So that** I can ensure each component and their interactions function as expected

## Details and Assumptions

- Unit testing will target individual handlers (LLM Handler, KG Handler, VS Handler, etc.), isolating them to verify accuracy in processing inputs and outputs.
- Integration testing will focus on testing pairs or groups of components, such as the KG Handler with the VS Handler, to confirm accurate data flow and compatibility.
- Dependencies for each component will be mocked during unit tests, while real connections will be tested in integration.

## Acceptance Criteria

```
Given an isolated handler
When unit tests are executed
Then each handler's individual functionality is verified

Given two or more integrated handlers
When integration tests are conducted
Then data flows correctly between components, and expected outputs are observed without errors
```

mollyiverson added documentation last week

| Assignees | mollyiverson |
|---|---|
| Labels | documentation |
| Projects | Capstone Agile Planning |
| Status | Done |
| Priority | Choose an option |
| Size | Choose an option |
| Estimate | 3 |
| Start date | No date |
| End date | No date |
| Sprint | Sprint 2 • Oct 5 - |
| Milestone | No milestone |
| Development | Create a branch for this issue or link a pull |

ACME10-HE-RAGApp #56
Write System testing section (functional, performance, user acceptance)
3 Sprint 2

ACME10-HE-RAGApp #57
Create code for natural language processing for the query
3 Sprint 2

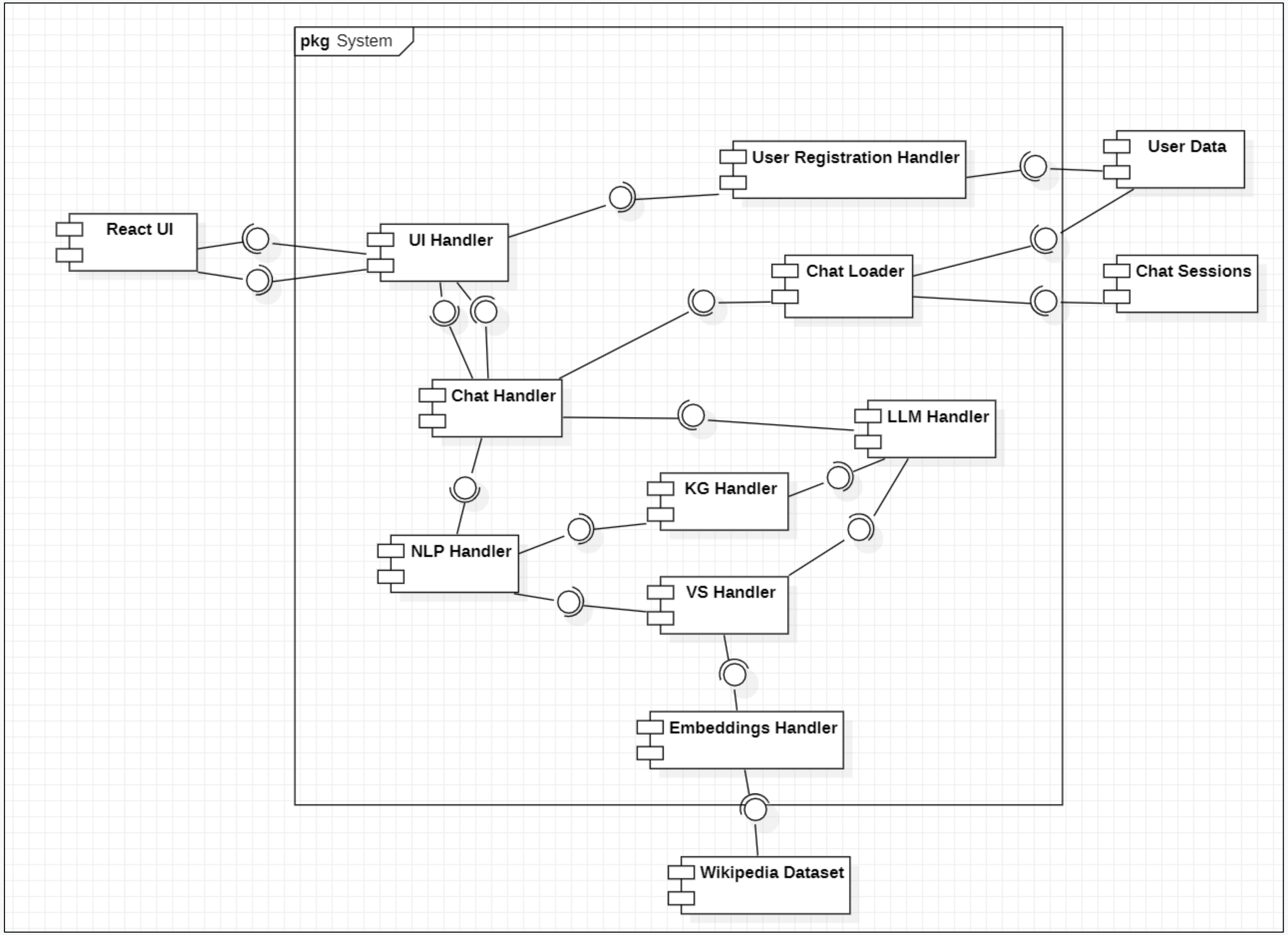
ACME10-HE-RAGApp #53
Write introduction for Testing and Acceptance plans report (overview, test objectives, scope)
3 Sprint 2

ACME10-HE-RAGApp #54
Write unit and integration tests section
3 Sprint 2

ACME10-HE-RAGApp #51
Integrate Knowledge Graphs in the RAG pipeline
3 Sprint 2

ACME10-HE-RAGApp #52

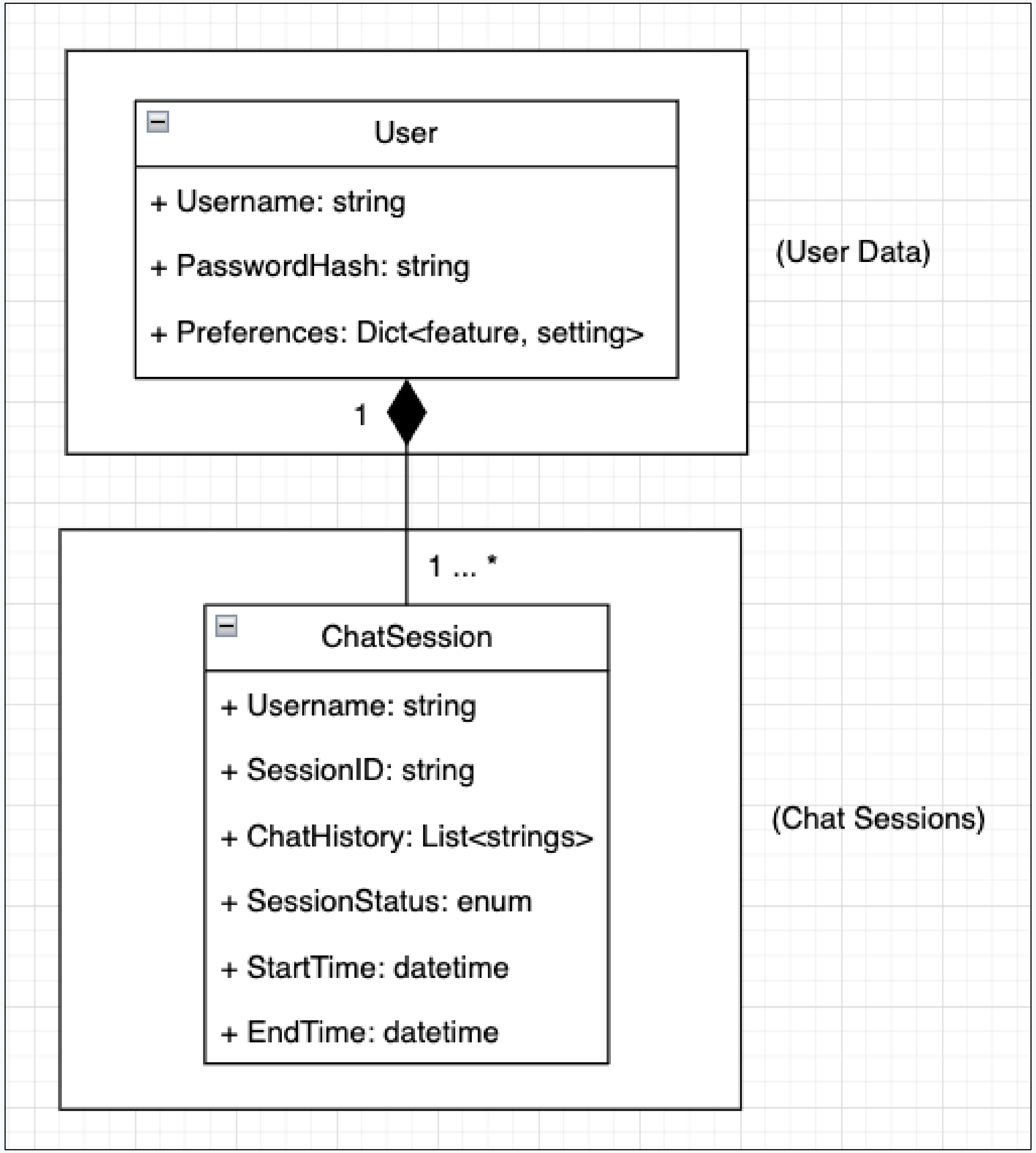
# Documentation of System Architecture, Database, UI, & Testing



# System Architecture

**Technologies chosen:**

- React for UI
- BERT for embedding
- spaCy and NLTK for NLP
- DBpedia for KG
- FAISS for VS
- LLaMA for LLM

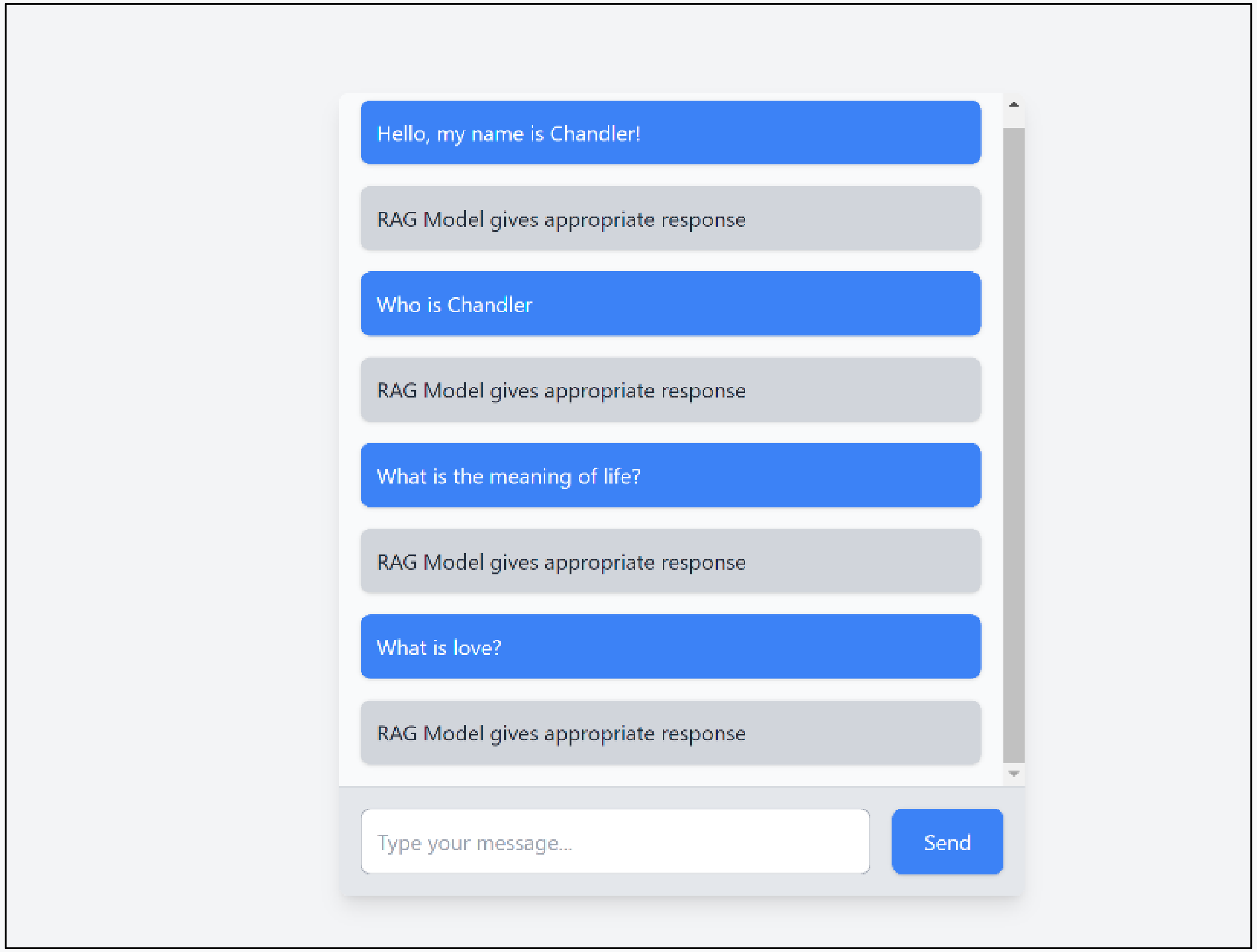


# Database Design

**Optimization strategies:**

- Indexing on Username and SessionID

# User Interface Design



Hello, my name is Chandler!

RAG Model gives appropriate response

Who is Chandler

RAG Model gives appropriate response

What is the meaning of life?

RAG Model gives appropriate response

What is love?

RAG Model gives appropriate response

# Testing Plan

## Unit

- Testing isolated handlers
- Covering edge cases and boundary conditions

## Integration

- Grouping interconnected components to test data flow
- Mocking outside components and dependencies

## System

- Manual functional testing
- Measuring response time
- Stress testing
- Gathering feedback from stakeholders and end users

**Tools:**

- *Unittest* or *PyTest* as a testing framework
- *unittest.mock* for mocking dependencies
- *Coverage.py* to measure code coverage
- *spaCy* and *NLTK* to compare query responses
- *GitHub Actions* for CI/CD
- *Jmeter* or *Locust* to assess response time and scalability

# Sprint Achievements & Challenges

- Looking back on Sprint 2.

- Solution Approach Section
- Testing Approach Section

- Extracting Wikipedia dataset
- NLP processing of a query
- Generating text embeddings for sample of dataset
- DBpedia querying
- Basic frontend prototype of RAG App

# Next Steps & Retrospectives

- Upcoming Tasks:
- Sprint Retrospective
- Client Feedback Consideration
- Generate text embeddings
- Integrate vector search
- Integrate knowledge graph
- LLM processing for responses

# Conclusion

- **Sprint 2 Achievements:**
- **Next Steps:**
  - **Optimize Embedding Generation**: Scale up for the full Wikipedia dataset with GPU acceleration.
  - **Implement FAISS Indexing**: Enable fast vector search for relevant results.
  - **Link to Knowledge Graph**: Enhance response accuracy by integrating vector search with the knowledge graph.
  - **Refine LLM Processing**: Improve query responses by combining vector and knowledge graph data.
  - **Advance Frontend**: Continue developing the chat interface for a smoother user experience.

# Demo