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

User Stories

US01 - RAG Model Architecture

As a user, I want the system to provide accurate responses based on a knowledge base so that I can get reliable information.

Scenario: Successful query and response retrieval

Given I am on the chat window page
When I type a valid query and click the "Submit" button
Then the system should process my query using the RAG model
And the system should return an accurate response in the chat window.

Scenario: Query with insufficient data

Given I am on the chat window page
When I type a query that the system cannot process due to lack of data
Then the system should notify me that it doesn't have enough information to respond
And suggest I try a different query or ask for additional context.

US02 - Vector Search Implementation

As a user, I want the system to retrieve relevant results using vector search so that I can receive answers based on the most contextually similar information.

Scenario: Successful vector-based query

Given I am on the chat window page
When I type a valid query and click "Submit"
Then the system should perform a vector search to find contextually relevant results
And the system should return a response based on the search.

Scenario: Query that does not return relevant results

Given I am on the chat window page
When I type a query that doesn't return relevant results
Then the system should return a message, indicating it couldn't find relevant information
And suggest that I rephrase my query or provide more specific details.

US03 - Knowledge Graph Integration

As a user, I want the system to understand relationships between entities using a knowledge graph so that I can receive responses with deeper context and meaning.

Scenario: Query with multiple entities

Given I am on the chat window page
When I type a query that involves multiple entities (e.g., "Relationship between protein X and drug Y")
Then the system should use the knowledge graph to understand the connections between those entities
And return an enriched response that explains the relationship.

Scenario: Query that doesn't match entities in the knowledge graph

Given I am on the chat window page
When I type a query that includes entities not recognized in the knowledge graph
Then the system should return a message indicating that it could not find relationships between the entities.

US04 - Query Response Retrieval

As a user, I want the system to retrieve relevant information and provide responses to my queries, so I can access knowledge efficiently.

Scenario: Successful query response retrieval

Given I have submitted a query
When the system processes the query
Then the system should retrieve relevant information using vector search and the knowledge graph
And display a well-formatted response in the chat window.

Scenario: Network error during query submission

Given I have submitted a query
When there is a network error during query retrieval
Then the system should display an error message in the chat window, explaining the network issue
And suggest retrying the query.

US05 - User Registration

As a user, I want to create an account so that I can save my chat sessions and access personalized features.

Scenario: Successful account creation

Given I am on the sign-up page
When I enter valid registration details
And I click the "Create Account" button
Then my account should be successfully created
And I should see a confirmation message.

Scenario: Unsuccessful account creation due to duplicate email

Given I am on the sign-up page
When I enter an email that is already registered
And I click the "Create Account" button
Then I should see an error message indicating the email is already in use.

US06 - Saving and Loading

As a user, I want to save and load chat sessions so that I can revisit chat sessions later.

Scenario: Successful chat session saving

Given I have completed a chat session
When I click the "Save Session" button
Then the system should prompt me to name the session
And save the session under my profile with a confirmation message.

Scenario: Loading a previously saved chat session

Given I have saved chat sessions
When I navigate to the “Saved Sessions” section
And select a saved chat session
Then the system should load the conversation in the chat window.

US07 - Chat Window

As a user, I want to interact with the system through a chat window so that I can ask questions and get responses in a conversational interface.

Scenario: User submits a query through the chat window

Given I am on the chat window page
When I type a message in the input field
And I click the "Submit" button or press "Enter"
Then the system should display my query in the chat window and process it.

Scenario: Empty input submission

Given I am on the chat window page
When I click the "Submit" button with an empty input field
Then the system should display an error message stating that input is required.

US08 - Error Handling

As a user, I want the system to display appropriate error messages so that I can understand when something goes wrong and what to do next.

Scenario: Query with invalid input

Given I have submitted a query
When the query contains invalid characters or exceeds the length limit
Then the system should display an error message near the input field, asking me to revise the query.

Scenario: Network issue during query submission

Given I have submitted a query
When there is a network error
Then the system should display an error message, indicating a connection problem.

US09 - Content Moderation

As a user, I want the system to handle inappropriate content gracefully to maintain a safe environment while still allowing academic or research-based queries.

Scenario: Query containing inappropriate content

Given I have submitted a query
When the query contains harmful or offensive content
Then the system should display a warning message
And prevent the query from being processed.

Scenario: Research-based sensitive query

Given I have submitted a query related to a sensitive topic (e.g., historical event)
When the system processes my query
Then the system should recognize the academic context
And return a response appropriate for research purposes.

Use Cases

Write a Query

Pre-Condition	● The user has opened the web application. ● The query input field is visible and active.
Post-Condition	● The user has successfully entered a query in the input field.
Basic Path	● The user clicks on the query input field. ● The user types a query or message in the input field. ● The system accepts and displays the query in the input field.
Alternative Path	● If the query input field is inactive or unavailable, the system prompts the user to refresh the page or displays a message saying that the input field is unavailable. ● If the user input exceeds a character limit, then the system displays an error message next to the input field asking the user to revise the query.
Related Requirements	● Chat Window

Submit a Query/Message

Pre-Condition	● The user has typed a query into the input field. ● The system is connected to the backend which is connected to the knowledge graph and vector search database.
Post-Condition	● The system successfully receives the query, initiating retrieval from the RAG model. ● The query appears in the chat interface as a sent message.
Basic Path	● The user clicks on the “Submit” button or presses “Enter” to submit the query. ● The system records the query and forwards it to the RAG model for processing. ● The query appears in the chat window as a sent message from the user.
Alternative Path	● If the user submits an empty query, then the system displays a message indicating that input is required. ● If the system encounters an issue with the query submission (e.g., network error), then an error message is displayed in the chat window. ● If the user submits a query that violates content moderation rules, then a warning message is displayed in the chat window and informs the user that the query is invalid.
Related Requirements	● Chat Window

Receive and Display Response

Pre-Condition	● The user has submitted a valid query, and the system has processed it using the RAG model.
Post-Condition	● The system returns a response based on the query and displays it in the chat window.
Basic Path	● The system processes the query using the RAG model. ● The system retrieves relevant information. ● The system generates a response using the RAG model and formats the information in a readable message. ● The system displays the response in the chat window above the user input field.
Alternative Path	● If the user submits an empty query, then the system displays a message indicating that input is required. ● If the system encounters an issue with the query submission (e.g., network error), then an error message is displayed in the chat window.
Related Requirements	● Chat Window ● Query Response Retrieval

Scroll Through the Current Chat Session History

Pre-Condition	● The user has had an ongoing chat session with multiple messages exchanged between the user and the system.
Post-Condition	● The system successfully receives the query, initiating retrieval from the RAG model.

Basic Path	● The user scrolls up in the chat window. ● The system dynamically loads previous queries and responses from the current session as the user scrolls. ● The user can review and scroll back to any message in the session.
Alternative Path	● If the system encounters an issue loading older messages, a loading spinner is shown until the history is fully loaded. ● If the chat session is too long, the system may implement infinite scrolling to manage performance.
Related Requirements	● Chat Window ● Saving and Loading

Save a Chat Session

Pre-Condition	● The user has completed a chat session and wants to save the conversation for future reference.
Post-Condition	● The system saves the chat session and stores it under the user's profile for later access.
Basic Path	● The user clicks the "Save Session" button at the end of the chat. ● The system prompts the user to name or label the chat session. ● The system saves the chat session with the provided name and stores it in the user's session history. ● A confirmation message is displayed to the user indicating the session was successfully saved.

Alternative Path	● If the system encounters an error during the saving process, the user is notified and asked to try saving again later. ● If the user does not provide a session name, the system automatically saves the session with a default timestamp-based name.
Related Requirements	● Saving and Loading

Load a Chat Session

Pre-Condition	● The user has previously saved chat sessions and wants to load a past conversation.
Post-Condition	● The user is able to view and interact with a previously saved chat session.
Basic Path	● The user navigates to the “Saved Sessions” section in the application. ● The system displays a list of saved chat sessions, each labeled with the session name and timestamp. ● The user selects a session to load. ● The system loads the selected chat session in the chat window. ● The user can scroll through and review the saved conversation.
Alternative Path	● If the selected session fails to load, the system shows a pop-up error message and offers the user a chance to retry. ● If the user has no saved sessions, the system displays a message “No saved sessions” in the saved sessions section of the UI.
Related Requirements	● Saving and Loading

User Registration

Pre-Condition	● The user is on the homepage of the RAG web application and is not logged in. The registration form is available to new users.
Post-Condition	● The user successfully created an account, and their details are stored in the system. The user is now logged in and is able to access logged-in features such as saving and loading chat histories.
Basic Path	● The user clicks on the 'Sign Up' or 'Register' button. ● The system displays a registration form requesting information such as username, email, and password. ● The user fills in the required details and submits the form. ● The system validates the provided information (e.g., checking if the email is already in use). ● The system creates a new user account, saves the user details in the database, and logs the user into the application. ● A confirmation message or welcome page is displayed, and the user is redirected to the application's main page.
Alternative Path	● If the user submits incomplete or invalid information (e.g., invalid email format, weak password, etc.), the system highlights the errors and prompts the user to correct them. ● If the username or email is already registered, the system displays an error and asks the user to choose a different email or username. ● If the system encounters a database or server error during registration, a generic error message is shown, and the user is advised to try again later.
Related Requirements	● User Registration and Login

User Login

Pre-Condition	● The user is on the homepage or login page of the RAG web application and already has an account registered.
Post-Condition	● The user successfully logs into the application and is redirected to the main page where they can begin to use the application.
Basic Path	● The user clicks on the "Login" button or navigates to the login page. ● The system displays a login form requesting the user's email/username and password. ● The user enters their email/username and password and submits the form. ● The system validates the credentials by checking the information against the stored user database. ● If the credentials are valid, the system logs the user in and redirects them to the main page. ● A success message is shown briefly, confirming the login.
Alternative Path	● If the user leaves any field blank, the system prompts the user to fill in the missing fields. ● If the entered email/username or password is incorrect, the system displays an error message (e.g., "Invalid username or password") and prompts the user to try again. ● If the user has forgotten their password, the system provides a "Forgot Password" link that directs them to a password recovery process. ● If there is a server/database error during login, a generic error message is shown, and the user is asked to try again later.
Related Requirements	● User Registration

Functional Requirements

RAG Model Architecture

Description	The system will implement RAG architecture, including a receiver and generator. The receiver will search the knowledge base for the most relevant data based on user input, and the generator (LLM) will take the search results and generate an accurate response.
Source	Project scope document provided by the client.
Priority	Priority Level 0: Essential functionality.

Vector Search Implementation

Description	The system will implement vector search to retrieve relevant information based on user queries. This includes generating embeddings, indexing them using vector search libraries (e.g., FAISS or Annoy), and finding similar results based on user input.
Source	Project scope document provided by the client.
Priority	Priority Level 0: Essential functionality.

Knowledge Graph Integration

Description	The system will integrate knowledge graphs (e.g., DBpedia or YAGO) into the RAG pipeline to better retrieve relevant and contextual information for the query.
Source	Project scope document provided by the client.
Priority	Priority Level 0: Essential functionality.

Query Response Retrieval

Description	The system will process user queries, fetch relevant information from the knowledge graph and vector search index, and generate responses using an LLM. If the system does not know the answer to a query, it should tell the user that it doesn't have enough information to respond accurately; it should not lie.
Source	Project scope document provided by the client.
Priority	Priority Level 0: Essential functionality.

User Registration

Description	The system will allow users to create an account by providing a username, password, and email address. This involves a user-friendly interface for registration, input validation, and secure storage of user data.

Source	Internal requirements elicitation among members of the team.
Priority	Priority Level 1: Desirable functionality.

Saving and Loading

Description	The system will automatically save the last 20 chat sessions and display them as clickable titles on the side of the chat window. Users should be able to load previous sessions for reference.
Source	Internal requirements elicitation among members of the team.
Priority	Priority Level 1: Desirable functionality.

Chat Window

Description	The system will display a chat window where users can type messages or queries. Once they press enter or click submit, the system will generate and display a response.
Source	Internal requirements elicitation among members of the team.
Priority	Priority Level 0: Essential functionality.

Error Handling

Description	The system will handle errors like invalid queries, network issues, or app failures by displaying appropriate error messages in the chat window.
Source	Internal requirements elicitation among members of the team.
Priority	Priority Level 1: Desirable functionality.

Content Moderation

Description	The system will prevent responses to harmful or inappropriate queries by showing a warning message. It will moderate content that includes hate speech, threats, violence, or graphic material. However, it will recognize when the query is research-based (e.g., questions about sensitive historical events like the Holocaust) and respond appropriately.
Source	Internal requirements elicitation among members of the team.
Priority	Priority Level 1: Desirable functionality.