SE 4485: Software Engineering Projects

Fall 2024

Final Report

|  |  |
| --- | --- |
| Group Number | 3 |
| Project Title | Knowledge Management Assistant (Team B) |
| Sponsoring Company | The Fellows Consulting Group (FCG) |
| Sponsor(s) | Jeff Buchmiller |
| Students | 1. Blythe Williams  2. Nidhi Prakuzhy  3. Roj Pawig  4. Ashley Primrose  5. Humayl Sheryar  6. Dalton Brua |

# Executive Summary

The Knowledge Management Assistant (KMA) project, sponsored by the Fellows Consulting Group (FCG) is a software development initiative by Group 3 (Team B) in the Fall 2024 semester. This application aims to streamline the process of information management and retrieval by providing an intuitive and robust platform for conducting, storing, and managing search queries.

Key features and capabilities:

* **Intelligent query processing:** Allows users to perform searches, refine queries, and retrieve accurate results from various web and AI-powered sources.
* **Data management:** Stores, organizes, and retrieves past queries and results efficiently while maintain data integrity.
* **User-centric design:** provides an intuitive user interface for seamless interaction and administration features like password management and data cleanup.
* **Scalability**: supports multiple concurrent users and handles large datasets with responsive performance.
* **Reliability and Security:** Ensures data protection through robust architecture and rigorous testing.

Project Objectives

The primary goal is to deliver a fully functional, scalable, and user-friendly application that meets the sponsor’s requirements. The project also emphasizes compliance with professional standards, effective team collaboration, and delivering high-quality documentation and presentations.

Expected Impact

The KMA is expected to improve productivity and efficiency for professionals by reducing the time and effort required for information retrieval and management. By integrating intelligent processing and a reuseable search history, the system has potential applications in consulting, education, and research domains. Its modular design and adherence to industry standards ensure long-term viability and adaptability to evolving needs.

The project serves as an educational experience for the team, allowing members to refine their skills in software engineering, project management, and collaborative development, while delivering a product that provides real-world value to the sponsoring organization and its stakeholders.

Table of Contents

[Executive Summary 2](#_Toc184113768)

[List of Figures 4](#_Toc184113769)

[List of Tables 4](#_Toc184113770)

[1. Introduction 4](#_Toc184113771)

[1.1 Purpose and Scope 4](#_Toc184113772)

[1.2 Product Overview 4](#_Toc184113773)

[1.3 Structure of the Document 5](#_Toc184113774)

[1.4 Terms, Acronyms, and Abbreviations 5](#_Toc184113775)

[2. Project Management Plan 5](#_Toc184113776)

[2.1 Project Organization 5](#_Toc184113777)

[2.2 Lifecycle Model Used 5](#_Toc184113778)

[2.3 Risk Analysis 5](#_Toc184113779)

[2.4 Software and Hardware Resource Requirements 7](#_Toc184113780)

[2.5 Deliverables and Schedule 8](#_Toc184113781)

[2.6 Monitoring, Reporting, and Controlling Mechanisms 10](#_Toc184113782)

[2.7 Professional Standards 11](#_Toc184113783)

[2.8 Evidence all the artifacts have been placed under configuration management 11](#_Toc184113784)

[2.9 Impact of the project on individuals and organizations 11](#_Toc184113785)

[3. Requirement Specifications 12](#_Toc184113786)

[3.1 Stakeholders for the system 12](#_Toc184113787)

[3.2 Use case model for functional requirements 12](#_Toc184113788)

[3.3 Graphic use case model 13](#_Toc184113789)

[3.4 Textual Description for each use case 13](#_Toc184113790)

[3.5 Rationale for your use case model 15](#_Toc184113791)

[3.6 Non-functional requirements 16](#_Toc184113792)

[4. Architecture 16](#_Toc184113793)

[4.1 Architectural style(s) used 16](#_Toc184113794)

[4.2 Architectural model 16](#_Toc184113795)

[4.3 Technology, software, and hardware used 17](#_Toc184113796)

[4.4 Rationale for your architectural style and model 17](#_Toc184113797)

[4.5 Traceability from requirements to architecture 19](#_Toc184113798)

[5. Design 21](#_Toc184113799)

[5.1 GUI (Graphical User Interface) design 21](#_Toc184113800)

[5.2 Static model – class diagrams 22](#_Toc184113801)

[5.3 Dynamic model – sequence diagrams 22](#_Toc184113802)

[5.4 Rationale for your detailed design model 23](#_Toc184113803)

[5.5 Traceability from requirements to detailed design model 23](#_Toc184113804)

[6. Test Plan 25](#_Toc184113805)

[6.1 Requirements/specifications-based system level test cases 25](#_Toc184113806)

[6.2 Techniques used for test generation 27](#_Toc184113807)

[6.3 Assessment of the goodness of your test suite 27](#_Toc184113808)

[6.4 Traceability of test cases to use cases 27](#_Toc184113809)

[7. Evidence the Document Has Been Placed under Configuration Management 29](#_Toc184113810)

[8. Engineering Standards and Multiple Constraints 29](#_Toc184113811)

[9. Additional References 29](#_Toc184113812)

[Acknowledgment 30](#_Toc184113813)

# List of Figures

Figure 1.0 Graphic Use Case Model

Figure 1.1 Architecture Model

Figure 1.2 GUI Design

Figure 1.3 Class Diagram

Figure 1.4 Query Submission Sequence Diagram

Figure 1.5 Admin Stored Query Deletion Sequence Diagram

Figure 1.6 History Sequence Diagram

Figure 1.7 Change Password Sequence Diagram

Figure 1.8 Save Result Sequence Diagram

Figure 1.9 Discard Result Sequence Diagram

# List of Tables

Table 1.0 Requirements to Architecture Traceability

Table 1.1 Requirements to Detailed Design Model Traceability

Table 1.2 Requirements/Specification-based Test Cases

Table 1.3 Test cases to Use Cases Traceability

# Introduction

## Purpose and Scope

The purpose of this project is to design and develop a Knowledge Management Assistant. Which is a software application aimed at enhancing knowledge retrieval and management. Sponsored by The Fellows Consulting Group (FCG), this project addresses the need for efficient handling of information searches and reuse by integrating user-driven query management, a sophisticated backend for data processing, and user-friendly interaction interfaces. The scope includes developing a scalable and maintainable system capable of processing user queries, aggregating results from various sources, and maintaining a historical record for reuse.

## Product Overview

The Knowledge Management Assistant is built on a three-tier client-server architecture. Which leverages technologies such as Python, ReactJS, MongoDB, and cloud-based solutions. Key capabilities of the application include:

* Conducting and refining queries to retrieve accurate responses.
* Managing previously stored queries and results.
* User-friendly interfaces for searching, saving and managing information.
* Supporting administrative tasks like password management and service control.

The system targets scenarios such as:

* Supporting consultants in preparing detailed reports through quick and reliable information retrieval
* Assisting teams in collaborative environments by maintaining consistent records of queries and responses.
* Enabling efficient analysis and synthesis of information sourced from external services like AI and web scrapers.

## Structure of the Document

This document outlines the project details in the following sections:

* Introduction
* Project Management plan
* Requirements specification
* Architecture
* Design
* Test plan
* Configuration management
* References

## Terms, Acronyms, and Abbreviations

KMA: Knowledge Management Assistant

FCG: Fellows Consulting Group

UI: User interface

NLP: Natural language processing

API: Application Programming interface

# Project Management Plan

## 2.1 Project Organization

* The development team will consist of a Project Manager, Front End developers, Back End Developers, and QA based off interests and experiences of the group members.
* Sponsor- Jeff, Team Leader- Blythe, Group Members- Nidhi, Dalton, Roj, Ashley, Humayl

## 2.2 Lifecycle Model Used

* AGILE

## 2.3 Risk Analysis

* Project risks:
  + Performance
    - There is the risk that the program of the application will run into performance degradation. Meaning the possibility of the software processing slower response times or having difficulty taking in a multitude of requests of different sizes.
    - Likelihood: There is a moderate chance of this risk occurring.
    - Risk reduction: To avoid having performance issues, we can create a strategy to handle all possible scenarios that might affect the runtime of the program.
    - Mitigation: We can optimize our data.
    - Rationale: Design the program with scalability in mind.
  + Timeline and Constraints
    - There is the risk that the project will face delays due to unforeseen challenges or resource limitations. Whether we run out of time to complete a finished project within the timeline or limited access to necessary tools.
    - Likelihood: There is a moderate chance of this risk occurring.
    - Risk reduction: Our team can break the project down into manageable tasks throughout each meeting, along with consulting with our sponsor with further questions. In doing so, we can slowly progress with our project to come to a finished product within the timeline.
    - Mitigation: Using agile methodologies and regularly assessing our timeline. Keeping the scope realistic.
    - Rationale: Regularly assess the project scope and schedule.
  + User engagement
    - There is the risk that the design does not meet the user’s expectations or the chance that the application itself is difficult to use for new users.
    - Likelihood: There is a low to moderate chance of this risk occurring.
    - Risk reduction: Our team will take priority in creating this application with an appealing and easy to use UI. Considering any scenarios that a future user might have issues with and creating a solution throughout the planning process.
    - Mitigation: Conducting early user testing as well as gathering feedback to ensure a more intuitive design.
    - Rationale: Engaging our sponsor in the early stages of software development. As well as testing the application to align with the user’s expectations and needs.
  + Team skill gaps
    - There is a risk that some of the members on the team may lack or not have much experience in the skills needed to develop the project.
    - Likelihood: There is a low to moderate chance of this risk occurring.
    - Mitigation: Invest time in practicing software related skills for the project outside of class if a member of the team feels that they need it.
    - Rationale: Building skills needed will help ensure that our team can handle the complex aspects of this project.
  + Inadequate Testing
    - There is a risk that we will run into failures identifying bugs due to not testing accurately.
    - Likelihood: There is a moderate to high chance of this risk occurring.
    - Mitigation: We can implement rigorous manual testing, as well as ensuring that we test at every phase of our project.
    - Rationale: Testing will help to reduce the risk of creating a final product with bugs or performance issues.
  + Hardware/Software Incompatibility
    - There is a risk that there could be compatibility issues with different software versions or certain devices.
    - Likelihood: There is a low to moderate chance of this risk occurring.
    - Mitigation: We can test the application across different devices and platforms to ensure that the application is useable across different versions of software or devices.
    - Rationale: Ensuring compatibility and avoiding the chance of alienating users.
  + Inadequate Documentation
    - There is the risk that one of the members or us as a group could have poor internal or external documentation. Which could lead to user confusion when it comes to the application.
    - Likelihood: There is a low to moderate chance of this risk occurring.
    - Mitigation: We can keep thorough internal documentation and possibly create user-friendly guides or FAQs
    - Rationale: Proper documentation will help our team stay organized and assist the users in understanding our application.
  + Scope Creep
    - There is the risk that our project scope is continually increasing beyond its original plan.
    - Likelihood: There is a high chance of this risk occurring.
    - Mitigation: We need to clearly define the scope and obtain the stakeholder agreement. Also, continually go over the scope and prioritize features that fall within said scope.
    - Rationale: Managing the scope from the start will prevent our application from becoming overwhelming and it will help our team meet deadlines.
  + Feature Overload
    - There is the risk that our team will get too ambitious and add too many features which could overwhelm the users.
    - Likelihood: There is a moderate chance of this risk occurring.
    - Mitigation: We can prioritize the must-need features.
    - Rationale: Focusing on the core functions of the application will make it easier to use and more focused on the goal of the application and user needs.
  + User data loss
    - There is the risk that our application may accidentally lose stored information, either to user error or technical malfunction.
    - Likelihood: There is a low to moderate chance of this risk occurring.
    - Mitigation: We can implement a data recovery option or a backup system.
    - Rationale: Data reliability is something that will ensure user trust and continued use.

## 2.4 Software and Hardware Resource Requirements

Software requirements:

* Operating System
  + Windows 10/11 or macOS 10.15 and above.
* Development Environment
  + Integrated Development Environment (IDE) - Visual Studio Code
* Programming Languages
  + Python 3.8 or later
  + JavaScript (Node.js 14.0 or later)
* Web Frameworks and Libraries
  + Flask
  + ReactJS
* Database
  + MongoDB
* Version Control
  + Git

Hardware requirements:

* Development Workstations
* Server Hardware
* Networking Equipment

New Software learned during the project:

* MongoDB: Team members gained experience using MongoDB for database management, including data modeling, CRUD operations, and integration with Python applications.
* ReactJS: Front-end developers explored ReactJS to create dynamic and responsive user interfaces for the presentation layer of the application.
* Flask: Backend developers honed their skills with Flask to handle application logic, routing, and API integration efficiently.
* Git: The team enhanced proficiency with version control, using Git for collaborative coding, and change tracking across distributed team members.

## 2.5 Deliverables and Schedule

The deliverables are as follows:

* **September 6, Project Management Plan**
  + This deliverable is expected to take ~8 hours to complete, split up between the 6 members of the group.
    - Each member of the group was assigned one or two points on the project management document, and each member should spend 1-2 hours working on this document.
  + The activities of this deliverable include:
    - Documenting the plans for the management of our project
    - Getting the sponsor’s feedback and changing any aspect that seems unreasonable
* **September 20, Requirements Documentation**
  + This deliverable is expected to take ~20 hours to complete, split up between the 6 members of the group
    - Most of the time in this phase will be used to talk with the sponsor to determine what the project should look like, and the overarching ideas about the project.
    - The rest of the time will be writing it all down, gathering and combining the ideas, and producing a professional and readable requirements document.
  + The activities of this deliverable include:
    - Talking with the sponsor and determining the exact specifications for the application
    - Determining which platform the application will operate on.
    - Creating prototype user-interfaces.
    - Documenting all requirements composed for the project
* **October 18, Architecture Documentation**
  + This deliverable is expected to take ~30 hours to complete, split up between the 6 members of the group
    - Architecture is a very technical portion of the project and is often difficult to create efficiently. Many aspects of architecture require a very good understanding of the project and the many tools and platforms the project may use. Therefore, this portion of the project will require research to be done, as well as setting up the environments for the impending parts of the project.
    - Assuming we can split up most of the work between the members, each student should be expected to spend 4-6 hours researching and documenting their contributions.
  + The activities of this deliverable include:
    - Researching platform environments
    - Setting up environments for the planned platform(s)
    - Setting up version control and coding environments
    - Determining language(s), database(s), and protocol(s) that will be used
    - Documenting all architecture specifications
* **November 1, Detailed Design Documentation**
  + This deliverable is expected to take ~20 hours to complete, split up between the 6 members of the group
    - The detailed design documentation will outline the entire project and the way in which we will approach creating it. This should be extremely well documented and as such will require different diagrams and explanations.
    - We will assign one or more diagrams to each member, and we will allocate 3-4 hours for each member to make their diagrams and explanations.
  + The activities for this deliverable include:
    - Creating system-wide vs individual component diagrams
    - Creating class diagrams
    - Creating sequence and use-case diagrams
    - Writing documents describing each diagram and its use
* **November 15, Test Plan**
  + This deliverable is expected to take ~15 hours to complete, split up between the 6 members of the group
    - The test plan must be a very detailed document that takes a lot of time to design. This is because we must think of all the ways we can break our project and try to fix those problems. Because of this, we have allocated 2-3 hours for each member to think about the project and come up with tests for all the functionality of the project.
    - The last few hours will be for a member to collect all the tests and test ideas and compile them into one document which we can use as our test plan.
  + The activities of this deliverable include:
    - Designing tests
    - Documenting all tests and test ideas
* **November 30, Final Project Presentation**
  + This deliverable is expected to take ~10 hours to complete, split up between the 6 members of the group
    - The final project presentation is very important to the project’s success. The project may work perfectly and to all specifications, however if we cannot explain and demo the project well, no one will know.
    - Therefore, each member will work on a specific part of the presentation, then every member will check the others’ parts of the presentation to ensure it is of the highest quality. We will allocate each member 1-2 hours to ensure their slides are accurate and check the group’s work.
  + The activities of this deliverable include:
    - Preparing presentation slides
    - Preparing project demo
    - Practicing the presentation and demo
    - Gathering feedback from the sponsor on our presentation
* **December 2, Final Project Report**
  + This deliverable is expected to take ~20 hours to complete, split up between the 6 members of the group
    - The final project report is the most important document of the project. This means our group must ensure that this document is of a professional standard.
    - The work associated with this portion of the project includes mainly gathering information, however we need to collect all documents, code, and diagrams to be outlined in the last report. This will take a lot of time to collect and organize, so each member will be given around 4 hours to gather any information they can find.
  + The activities of this deliverable include:
    - Gathering and collecting all information used and created during the project
    - Organizing information gathered into one cohesive and understandable document.

## 2.6 Monitoring, Reporting, and Controlling Mechanisms

* Monitoring, reporting, and controlling are essential processes that must be carried out for the successful execution of any project. These processes include tracking and assessing the project’s progress, identifying any areas where adjustments to the settled plan and schedule are needed, and implementing the proposed changes as needed. Accordingly, corresponding reports on these important processes will be made to ensure satisfactory results, reduce project risks, and stay on scope.
* Status Reports: Continuous communication among project team members is expected as well as engagement with the sponsor, requiring members to provide status reports on current project tasks and/or performance. Monitoring the project through status reports provides necessary insight into how the team and stakeholders are satisfied with the quality and performance while also identifying any problem areas that may require more attention. Status reports are expected by team members orally or written through online communication during sponsor meetings and near important deadlines (See Deliverables and Schedule). Status reports must include one’s current standing on their assigned task and should also contain any other concerns. Although it is a critical process, status reports can and will be typically informal (due to time constraints and other responsibilities) as it is focused on members being transparent about their personal and project concerns, allowing the team to understand each other and be proactive on any issues that arise.
* Change Management: In tandem with status reports, managing necessary changes to the project is a natural and expected process that all project members are responsible for due to the nature of how projects are executed. Therefore, it is significant to have an appropriate procedure to manage change. As per the course syllabus, the ability to use a configuration management system and develop CM processes is expected for students to learn and be able to execute during this course. Configuration management tools are essential for tracking deliverables and synchronizing any changes made by multiple team members. CM tools that will be used include cloud storage (OneDrive/Google Drive), word processing software (Word/Google Docs), and code hosting platforms (GitHub). These tools allow the project team to see different versions when changes to a document are made, providing an important control measure for reviewing changes and comparing differences between two different versions. Accordingly, when making changes it is up to the members’ discretion to do it before discussing with the team as some changes are miniscule in terms of importance, e.g. fixing grammar mistakes. However, project members are expected to discuss and propose any significant changes with other team members, possibly requiring feedback from the sponsor. Changes made to the project not only include document and code changes but also possible scheduling changes that may need to be arranged for certain team members. Similar to status reports, formal change management reports are not required due to the primary focus on executing project deliverables and objectives, where formal change reports could induce time constraints and increase project risks. Since CM tools will be used, changes will be tracked, easily identifying the member who made a change to the document or source code. Simple comments can be made to further communicate minute changes to other team members. Moreover, when proposing more significant changes, members must make note of the change and discuss amongst the team, which may result in an impromptu meeting. It is of utmost importance that major changes are also communicated to the sponsor through email or other forms of communication. As previously underlined, communication in any shape or form is essential for the success of the project, making change management a more streamlined process that would not give rise to any more risks. Overall, management reports regarding changes are not expected formally, but members are expected to discuss, communicate, and be transparent about changes made/proposed to any aspect of the project.
* Time Management: Time management is an encompassing project control that covers not only general time management but also schedule management and resource management, important measures that ensures satisfactory project performance To ensure the project remains on track and on scope, it is essential to produce a project schedule report at the start of the project (See Deliverables and Schedule), which would be used as an outline to manage the team’s time and schedule. This report helps members identify possible risks and delays early, allowing for corrective actions to be taken. Additionally, it is important that weekly schedule reports are made to ensure everyone is on the same page, minimizing any possible miscommunication and misunderstanding about what tasks must be finished and what deliverable is due. Again, these reports are not required to be formal documents prepared by the team but must simply be conveyed through accepted forms of communication. Unitedly, resource management is necessary for appropriate time management. Managing and monitoring project resources, otherwise known as team members, are supervised by the project leader under the guidance of the project sponsor. The team, as a whole, is responsible for managing who is assigned what task and when it should be done. During weekly meetings, resource management should always be discussed, referring to the assignment of sub deliverables to specific team members. Reports regarding resource management are not required, but like other forms of project controls, they must be appropriately discussed by the team during sponsor meetings or through online communication. Provided that this course is about 16 weeks (about 3 and a half months), time management is crucial for the success of the project. It all comes down to competent communication, managing each members’ time appropriately.

## Professional Standards

* Students are expected to act with a high-level of responsibility and academic honesty, demonstrating a high standard of individual honor in scholastic work as per the University’s policies. Regular class participation and engagement in group activities is expected, as explained in the course syllabus.
* refer to Appendix A for more details

## 2.8 Evidence all the artifacts have been placed under configuration management

## 2.9 Impact of the project on individuals and organizations

The KMA project significantly impacts individuals and organizations by enhancing knowledge management practices through innovative software design.

Impact on public health, safety, and welfare

The KMA indirectly contributes to public welfare by supporting informed decision-making and reducing cognitive workload for professionals. For example, consultants, researchers, and educators can efficiently retrieve and manage relevant information, enabling them to focus on delivering value in their respective fields.

Social and Economic impacts

* **Global Accessibility:** The system is designed to operate across multiple platforms and devices, promoting inclusivity and accessibility for users worldwide.
* **Enhanced Productivity:** By automating and streamlining the process of query management, KMA reduces time spent on repetitive tasks, resulting in cost savings and increased productivity for organizations.
* **Cost Efficiency:** The use of open source technologies and cloud based services ensures that organizations can implement KMA without significant financial investment, making it accessible even for small businesses and non-profits.

Environmental Considerations

The adoption of cloud-based technologies minimized the environmental footprint by reducing the need for physical infrastructure. Additionally, the efficient design of the KMA ensures that computational resources are used judiciously, further contributing to environmental sustainability.

Integration of Global, Cultural, and Social Factors

The design of the KMA considers diverse user needs by prioritizing a user-friendly interface and accessibility. The system supports multiple languages for query input and output, ensuring usability in a variety of cultural contexts. The inclusion of collaborative features enhances teamwork across geographically distributed teams, fostering global connectivity and cooperation.

By addressing the dynamic needs of modern organizations and aligning with ethical development practices, the KMA project exemplifies how technology can enhance individual productivity and organizational efficiency while contributing positively to societal and global goals.

# Requirement Specifications

## 3.1 Stakeholders for the system

Primary stakeholders:

* Knowledge Managers: Individuals interacting with the application to create, retrieve, and manage knowledge queries and results.
* Administrative users: Those with permission to manage system configurations, such as setting or changing passwords and deleting stored queries or results.

Secondary stakeholders

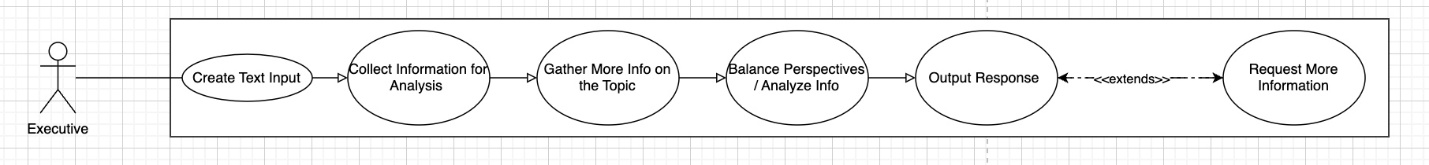
* Sponsor company: The Fellows Consulting group
* Sponsor: Jeff Buchmiller
* Development Team: Blythe Williams, Nidhi Prakuzhy, Roj Pawig, Ashley Primrose, Humayl Sheryar, and Dalton Brua

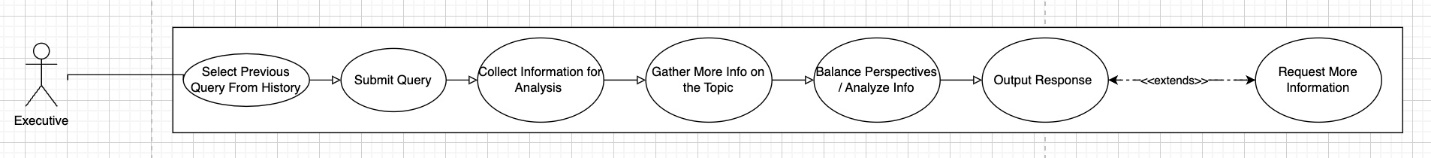
## 3.2 Use case model for functional requirements

The functional requirements are captured through the following use cases:

* Knowledge manager creates a query
* User searches history for a previous query
* User logs into the system
* Option to change password
* Save or discord Query Results
* Administrative deletion of stored data
* Retrieve results from internet and AI services

## 3.3 Graphic use case model





A diagram of a type of war

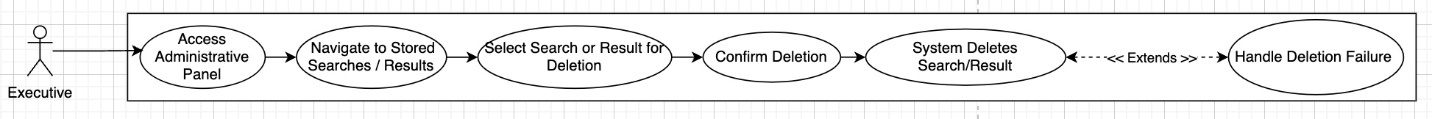
Description automatically generated

A diagram of a diagram

Description automatically generated with medium confidence

A diagram of a diagram

Description automatically generated with medium confidence



A diagram of a diagram

Description automatically generated with medium confidence

Figure 1.0 Graphic Use Case Model

## 3.4 Textual Description for each use case

* Knowledge manager creates a query
  + Participating Actors: Knowledge Manager
  + Entry Condition(s): User creates text input
  + Normal Flow of Events: text input -> collect info for analysis/gather more info on the topic (multiple iterations) -> balance perspectives/analyze info -> output response
  + Exit Condition(s): User is satisfied with the response
  + Exceptions (Alternate Flow of Events): text input -> collect info for analysis/gather more info on the topic (multiple iterations) -> balance perspectives/analyze info -> Executive wants to see more information on a specific perspective -> collect info for analysis/gather more info on the topic (multiple iterations) -> output response
  + Special Requirements: friendly easy to use design for the user
* User searches history for a previous query
  + Participating Actors – Knowledge Manager
  + Entry Condition(s) - User selects a previous query from history; History is not empty.
  + Normal Flow of Events: User selects query from history -> change/submit query -> collect information for analysis/gather more info (multiple iterations) -> balance perspectives/analyze info -> output response -> user saves response/requests more information
  + Exit Condition(s) - User is satisfied with the response
  + Exceptions (Alternate Flow of Events)
* User logs into the system
  + Participating Actors – Knowledge Manager
  + Entry Condition(s) - Knowledge manager would like to use the application
  + Normal Flow of Events: Knowledge manager starts application -> application prompts user for login/password -> User types in login/password -> application verifies login/password -> application allows user access
  + Exit Condition(s) - User gains access to application OR User closes application
  + Exceptions (Alternate Flow of Events) - User starts application -> application prompts user for login/password -> User types in login/password -> application rejects login/password -> application does not allow user access -> prompts user for login/password again
* Option to change password
  + Participating Actors – Knowledge manager
  + Entry Condition(s) – Knowledge manager would like to change the password
  + Normal Flow of Events: user starts application -> application prompts user for login/password -> user selects to change password -> user successfully changes password -> application updates database to save new password -> application prompts user to login with new password -> user enters login information with new password -> application verifies login/password -> application allows user access.
  + Exit Condition(s) – user gains access to application OR user closes application
  + Exceptions (Alternate Flow of Events)
* Save or discord Query Results
  + Participating Actors – Knowledge Manager
  + Entry Condition(s) – Knowledge manager has executed a query and the result is presented to the user.
  + Normal Flow of Events: user views current response -> application displays save or discard options to user -> user choose to save or discard current response -> application saves query to database if user chooses to save OR application discards the query -> application returns to main interface after action is completed
  + Exit Condition(s) – If query is saved or discarded, the user returns to main interface
  + Exceptions (Alternate Flow of Events) – user is unable to save query -> application prompts error message and prompts the user to try saving or discarding the query again -> user selects to save, discard or cancel option -> application saves if user selects to save OR application returns to main interface
* Administrative deletion of stored data
  + Participating Actors: Knowledge Manager
  + Entry Condition(s): Knowledge manager decides to remove a stored search or result
  + Normal Flow of Events: Knowledge manager accesses the administrative panel of the system -> Knowledge manager navigates to the section that stores searches/results -> Knowledge manager selects the search or result to be deleted -> system prompts the knowledge manager for confirmation -> Knowledge manager confirms the deletion -> System permanently removes the search or result from the database.
  + Exit Condition(s): The stored search or result is successfully deleted, or the knowledge manager cancels the operation.
  + Exceptions (Alternate Flow of Events): If deletion fails (due to system failure or restricted access) an error message will display asking the Knowledge manager if they would like to retry or cancel the operation.
  1. Special Requirements: The system should ensure that only authorized administrators can delete stored data. Confirmation prompts should prevent accidental deletions.
  2. Retrieve results from internet and AI services
  3. Participating Actors: Knowledge Manager
  4. Entry Condition(s): Knowledge manager submits a new query that requires gathering information from both external internet sources and AI services.
  5. Normal Flow of Events: Knowledge manager creates and submits a new query through the user interface -> The system connects to various internet search engines and AI services -> The system retrieves information relevant to the query from these sources -> The system processes and aggregates the data to formulate a comprehensive response -> The result is displayed for the knowledge manager to review.
  6. Exit Condition(s): The knowledge manager reviews and either accepts or modifies the selected response.
  7. Exceptions (Alternate Flow of Events): If the system fails to connect to external internet sources or AI services, it should notify the knowledge manager and either offer an option to retry or provide the choice to use internally stored results only. -> If the AI services return conflicting or incomplete data, the system should present this information to the knowledge manager for further clarification or search refinement.
  8. Special Requirements: The system must be able to handle different formats and structures of data retrieved from multiple internet sources and AI services. The response time should be optimized to ensure the system is usable in real time.

## Rationale for your use case model

The rationale comes from understanding the scope and functional requirements of the project including the following:

* an interface (input/question & output/response) for the knowledge manager
* the knowledge manager can submit searches and receive results in response
* the knowledge manager can peruse through results from previous searches
* the knowledge manager can select part of a previous response and submit a word or body of text as a new search
* after a result is presented to the user, user can choose whether it gets saved in the system or not
* the back-end portion of the software knows how to retrieve results that match the search in some way from the internally stored results of previous searches
* the back-end portion of the software knows how to retrieve results that match the search in some way from some number of Internet search engines and Internet AI services
* the back-end portion of the software knows how to process results from multiple sources and formulate a single response from them - perhaps this is NLP (natural language processing) based and/or AI (artificial intelligence) based, but neither is required
* administrative feature to set or change the password
  1. administrative feature to delete any stored search and/or result deemed not to be worth keeping around

## 3.6 Non-functional requirements

* Performance (response time)
  1. Response time in 10 seconds or less.
* Usability (User-friendly)
  + Multiple users can act as a knowledge manager.
  1. Number of concurrent users up to 10.
  2. Each user has their own account/password, which can be changed by administrators
* Availability (Always able to access)
  + Utilize servers and software to be available to the team via the university (publicly available for free).
* Scalability (With more information, still produces accurate responses)

# Architecture

## 4.1 Architectural style(s) used

* Generic Layering
  + Three-Tier Client-Server Architecture
* How does the architecture support various features of your application?
  + For the first tier, the presentation tier, this will handle some examples use cases such as user input/output, logging in, changing passwords, and viewing history. Second tier would be the application tier where the features supported will handle processing tasks such as web scraping, gathering and processing information from previous searches, and so on. For the last tier, which is the data tier, the features supported would include saving or retrieving previous queries as well as ensuring that our application can hold data/queries.

## 4.2 Architectural model

A computer screen with white text

Description automatically generated

Figure 1.1 Architecture Model

What the model represents:

* Presentation subsystem: (input/outputs) -> user login, user interaction, password management, output to user. Tech used -> ReactJS.
* Application subsystem: web scraping, data processing, search history analysis. Tech used -> Python with API integration.
* Data subsystem: stores previous search queries, handles retrieval, data validation, storage. Tech used -> MongoDB, Github.

How the architecture interacts with each other:

* Presentation subsystem interacts with Application subsystem.
* Application subsystem interacts with both data and presentation subsystems.
* Data subsystem interacts with Application subsystem.

What the interactions are for:

* The presentation subsystem interacts with the Application subsystem because the presentation will send requests to the application.
* With the interactions between the application subsystem, it takes the requests from the presentation subsystem. After the application processes the data, it sends over the query to the data subsystem and waits for the data subsystem to send back data validation. Once data validation has been completed, the application processes the recent history and sends back the query to the presentation subsystem for the user to see the output.

The data subsystem interactions between the application is just a transfer of data. The application requests data validation and the data layers requests data to store and validate.

## 4.3 Technology, software, and hardware used

* Python 3.8
* ReactJS
* MongoDB
* GitHub

## 4.4 Rationale for your architectural style and model

A three-tier architecture is highly suitable for implementing the proposed knowledge management system. This architecture offers several key benefits, including separation of concerns, scalability, performance optimization, and security as detailed below:

1. **Separation of Concerns**

The three-tier architecture provides a clear division between the presentation layer, application layer, and data layer, ensuring each layer can be independently managed and optimized. This separation aligns with the system’s functional complexity and modular design.

* **Presentation Layer:** Handles interactions with the Knowledge Manager, i.e. user input, password management, and output to the user. This layer isolates user-facing functionality from business logic and data management, enhancing usability and maintainability. Changes to the UI can be made without affecting other layers.
* **Application Layer (Knowledge Management Assistant):** Contains core business logic, such as web scraping, data processing and search history analysis. This layer processes user requests, retrieves data from external sources, and abstracts complex operations (i.e. data aggregation and validation) from both the user and data storage, improving performance and scalability.
* **Data Layer:** Manages data storage and retrieval, including storing queries, handling data validation, and maintaining data integrity. This layer ensures secure and isolated data management, enhancing data security and integrity.

1. **Scalability**

The three-tier architecture allows for easy scalability. As the system grows, each layer can be scaled independently to handle increased demands:

* The Data Layer can be optimized for faster data retrieval, supporting the requirement for scalability with large datasets.
* The Application Layer can be enhanced to handle more complex data processing and AI services, ensuring the system remains responsive under heavy loads.

1. **Performance and Exception Handling**

This architecture supports efficient performance management, with the Application Layer responsible for data aggregation and analysis. Real-time response requirements can be met by optimizing this middle layer. In case of failures (i.e. a failed connection to external AI services), the application layer can handle retries or fallback procedures without affecting the user experience in the presentation layer.

1. **Security and Data Management**

Sensitive operations, such as storing queries and managing user passwords, are isolated within the data layer. This architecture provides tighter control over access to stored data, preventing unauthorized modification or accidental deletions. The application layer ensures business logic processing is secure and protected from direct user interaction.

**Client-Server Relationship in the Three-Tier Architecture**

1. **Client (Presentation Layer)**

The user interface acts as the client, initiating requests such as submitting queries or logging in. These requests are sent to the application layer for processing.

1. **Middle-Tier Server (Application Layer)**

Acts as the intermediary between the client and the data layer. It processes user requests, retrieves data from external sources, and transforms it into usable results which are then sent back to the client. It also functions as a client to the Data Layer.

1. **Backend Server (Data Layer)**

This layer stores and manages data. It responds to requests from the application layer for data storage and retrieval, ensuring data integrity and security.

**Benefits of the three-tier client-server architecture**

* **Modularization:** Each layer is modular, allowing for independent development, maintenance, and scaling.
* **Scalability:** The system can scale at each layer (presentation, application, or data) independently to handle increased load without affecting other layers
* **Security:** Sensitive operations like data validation and storage are isolated in the Data Layer, while user-facing functions remain in the Presentation Layer. The Application Layer ensures business logic is protected from direct user access.
* **Maintainability:** Changes or updates can be made to each tier without impacting the others, improving long-term maintainability and reducing system downtime.

## 4.5 Traceability from requirements to architecture

|  |  |  |
| --- | --- | --- |
| REQUIREMENT | ARCHITECTURE | DESCRIPTION |
| Users must be able to input text. (UC01) | Presentation Tier | Presentation subsystem shall provide an interface for users to input text. |
| K.M.A. must be able to process text. (UC01) | Application Tier | Application subsystem shall receive user input and process the text to perform the task appropriate to the input. |
| K.M.A. must be able to gather information from the Web. (UC01) | Application Tier | Application subsystem shall be able to perform web scraping/data extraction. |
| K.M.A. must be able to analyze information gathered. (UC01) | Application Tier | Application subsystem shall be able to analyze information based on user input, choosing the most balanced perspective. |
| K.M.A. must output its response to users. (UC01) | Application/Presentation Tiers | Application subsystem shall be able to produce a sufficient response to user input and have it output on the presentation subsystem. |
| K.M.A. must save user queries and responses to a database. (UC01) | Application/Data Tiers | Application subsystem shall be able to save queries submitted by user and its appropriate response to the data subsystem. |
| Users must be able to access and view a history of queries and responses. (UC02) | Presentation/Data Tiers | Presentation subsystem shall have an interface for users to view a history of previous queries and responses that are saved in the data subsystem. |
| Users must be able to select previous queries and modify or re-submit to the application. (UC02) | Presentation/Data Tiers | Presentation subsystem shall have an option for users to select and re-submit previous queries (which are saved in the data subsystem). |
| Users must be able to search through previous queries. (UC02) | Presentation/Data Tiers | Presentation subsystem shall have an option for users to search using input as keywords, retrieving results that match the search from the data subsystem. |
| Users must be able to login. (UC03) | Presentation/Application /Data Tiers | Presentation subsystem shall have a user login interface when first launching the application. Application subsystem retrieves login information, validating user with the data subsystem. |
| Users must be able to change passwords. (UC04) | Presentation/Application /Data Tiers | Presentation subsystem shall have an interface to allow users to change passwords. Application must verify if user is authorized to change passwords. Application must delete old password and save new password to the data subsystem. |
| Users must be able to save queries. (UC05) | Presentation/Application /Data Tiers | Presentation subsystem shall have an interface allowing users to have an option to save queries. Application subsystem must save query to the data subsystem when user prompts it. |
| Users must be able to discard queries. (UC05) | Presentation/Application Tiers | Presentation subsystem shall have an interface allowing users to have an option to discard queries. Application subsystem must delete query. |
| Users must be able to delete stored results. (UC06) | Presentation/Application/Data Tiers | Presentation subsystem shall have an interface allowing users to view and delete stored results. Application subsystem must delete result from data subsystem when user prompts it. |
| K.M.A. must be able to authorize users. (UC06) | Application/Data Tiers | Application subsystem shall be able to verify users based on login information, validating with information from data subsystem. |
| K.M.A. must be able to gather information from AI services. (UC07) | Application Tier | Application subsystem shall be able to connect AI APIs and generate results based on user input. |

Table 1.0 Requirements to Architecture Traceability

\* K.M.A. - Knowledge Management Assistant

# Design

## 5.1 GUI (Graphical User Interface) design

A login screen with a blue and red dot pattern

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure 1.2 GUI Design

## 5.2 Static model – class diagrams

A diagram of a company

Description automatically generated

Figure 1.3 Class Diagram

## 5.3 Dynamic model – sequence diagrams

Query Submission Sequence Diagram:

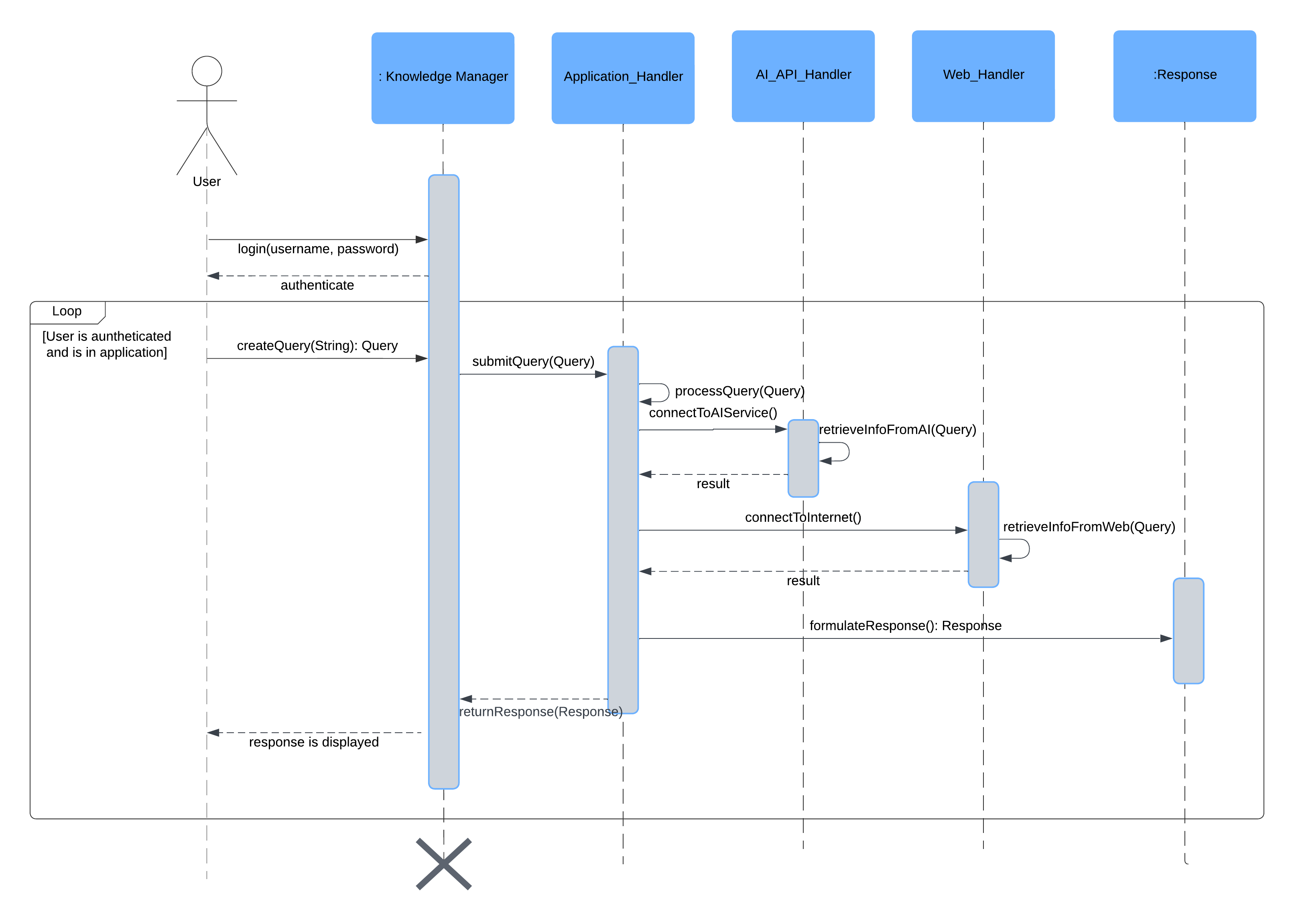


Figure 1.4 Query Submission Sequence Diagram

Admin Stored Query Deletion Sequence Diagram:

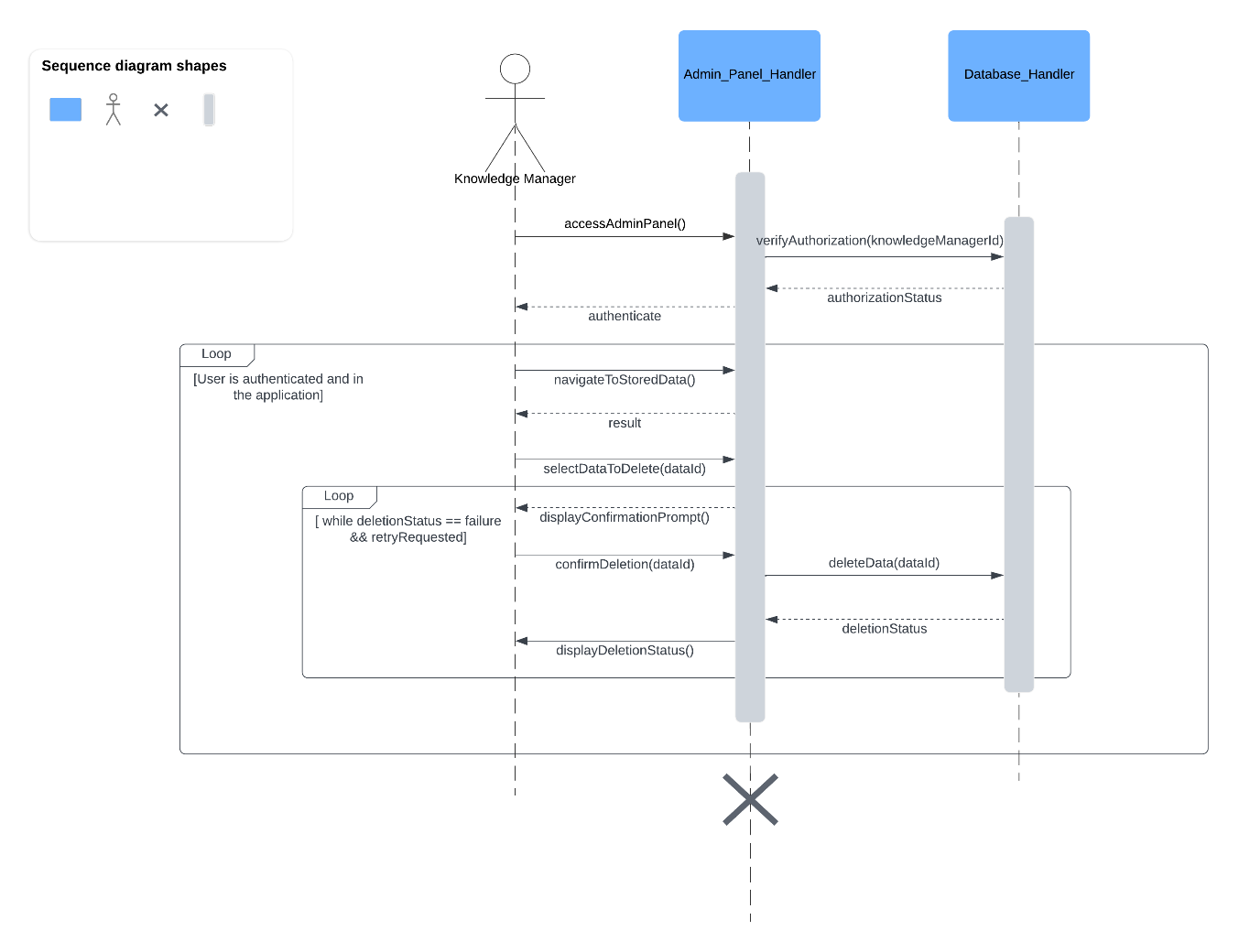


Figure 1.5 Admin Stored Query Deletion Sequence Diagram

Goes through history sequence diagram:

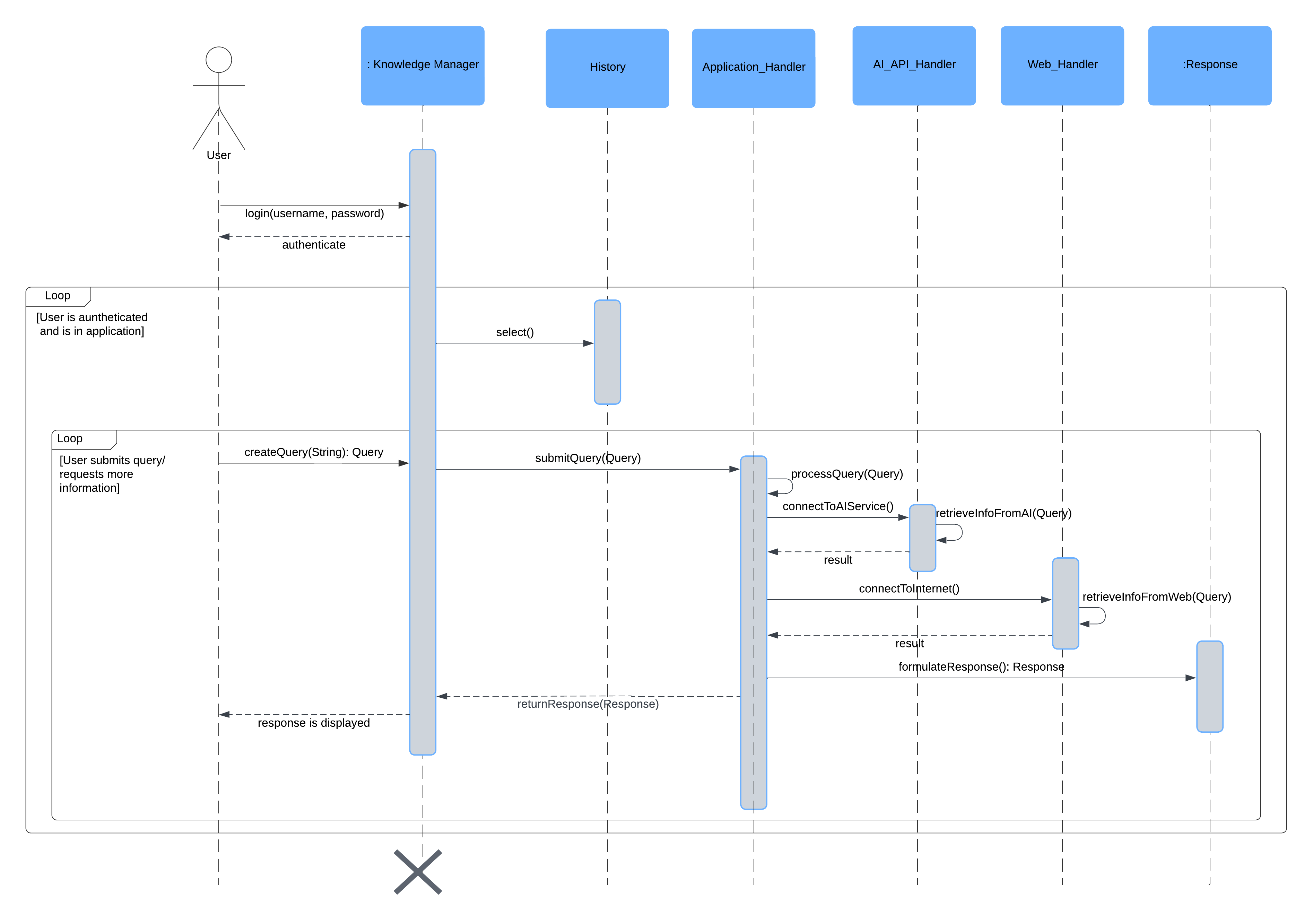


Figure 1.6 History Sequence Diagram

Users option to change password sequence diagram:

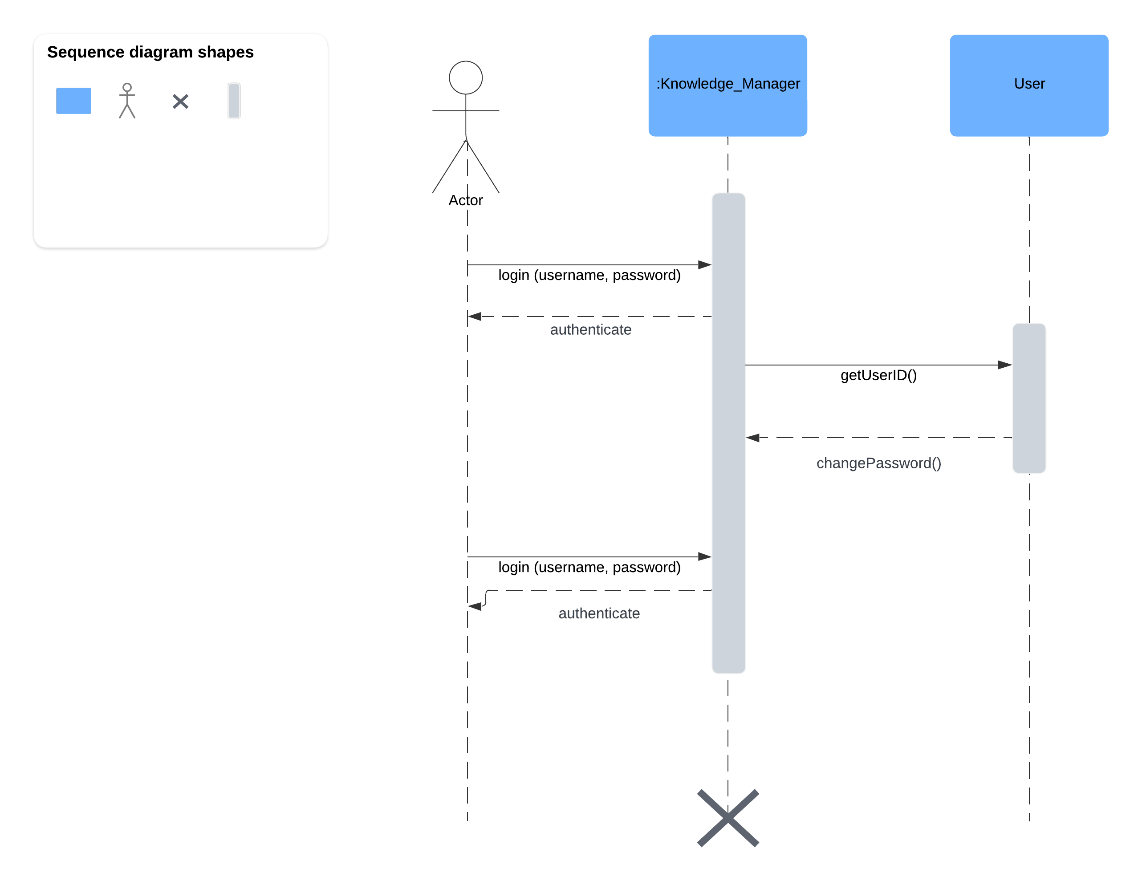


Figure 1.7 Change Password Sequence Diagram

User chooses to save result sequence diagram:

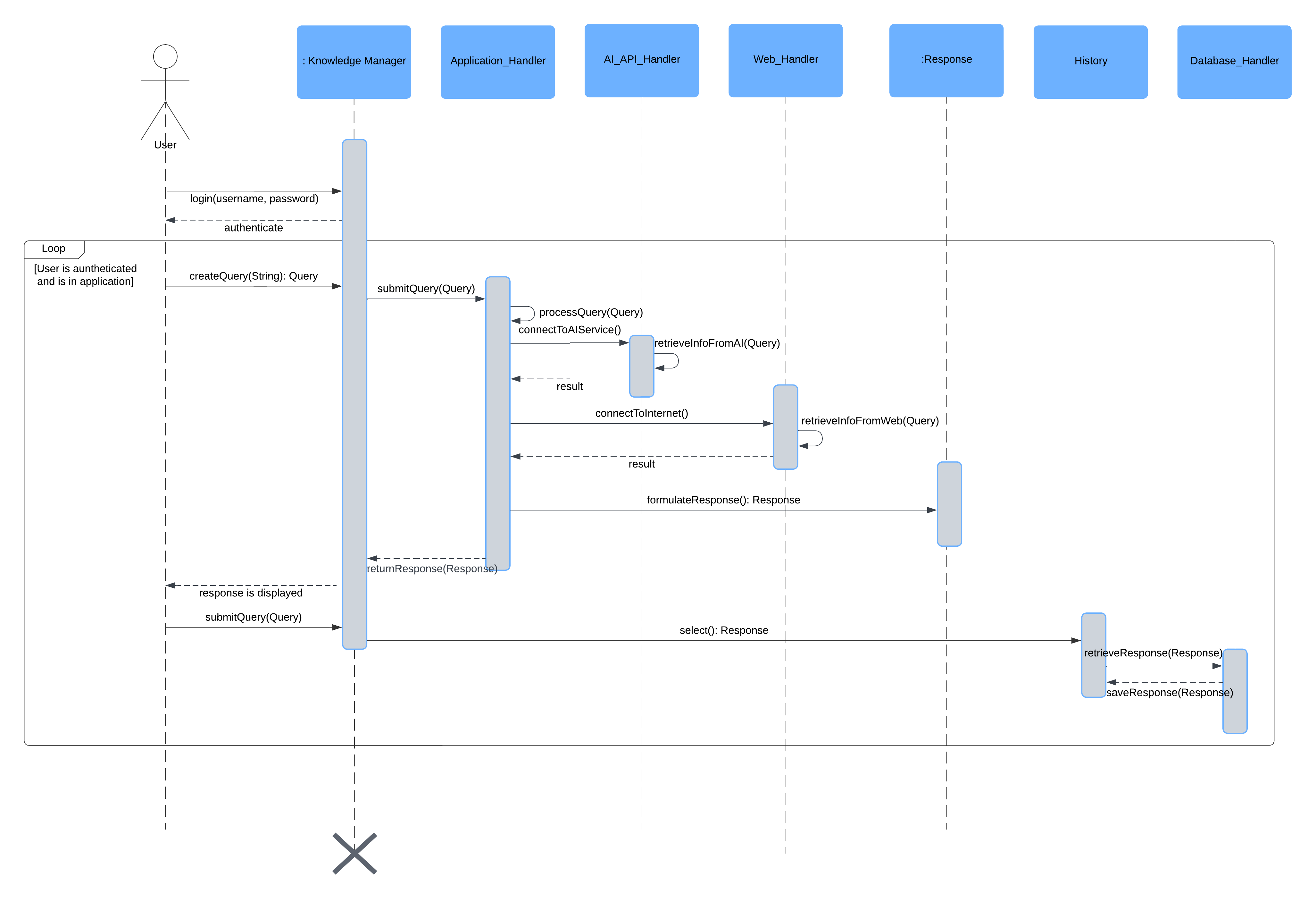


Figure 1.8 Save Result Sequence Diagram

User chooses to discard result sequence diagram:

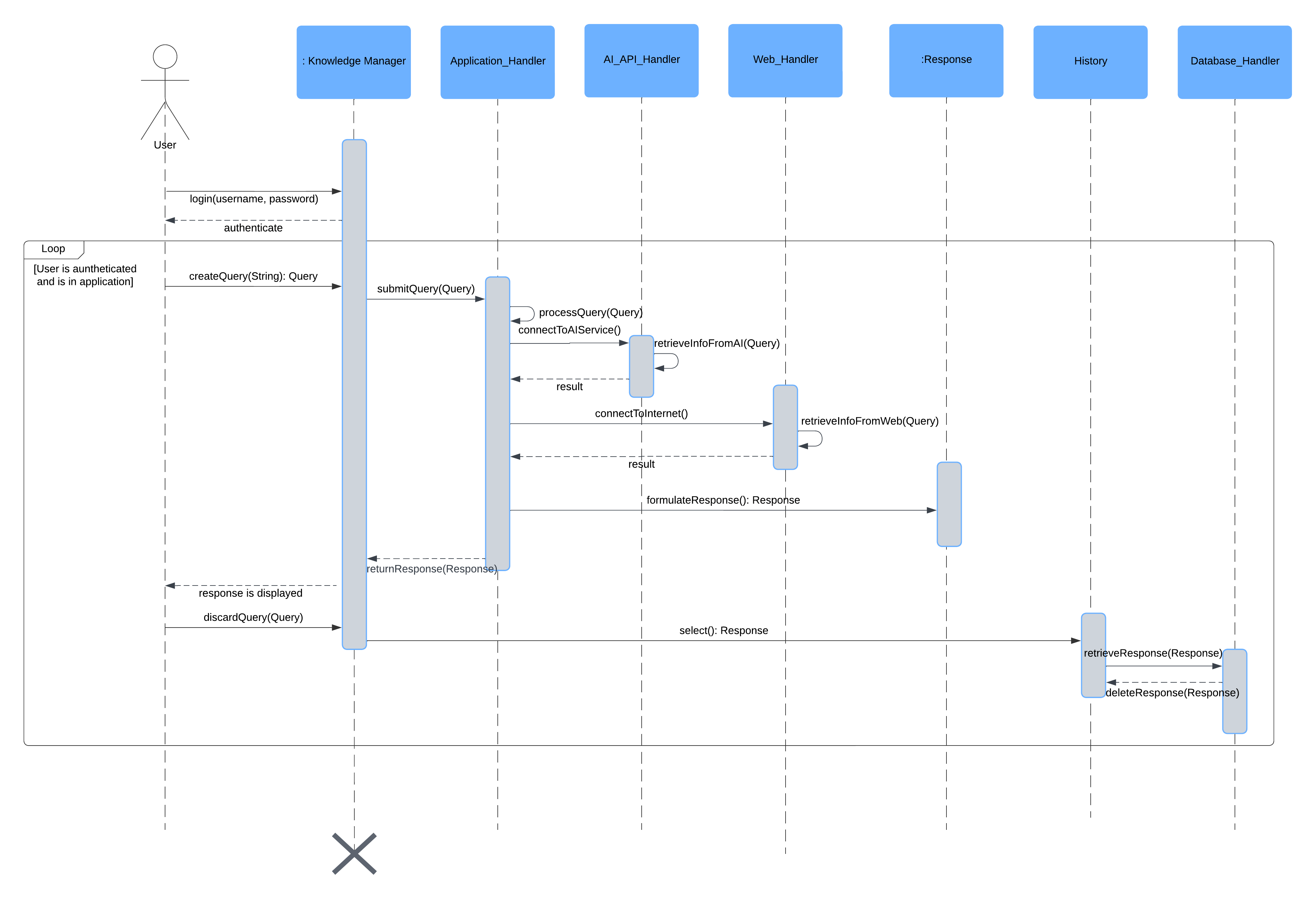


Figure 1.9 Discard Result Sequence Diagram

## 5.4 Rationale for your detailed design model

The detailed design model of the application has been developed to ensure a robust, scalable, and maintainable solution. The design decisions are driven by requirements, project constraints, and best practices in software engineering. Here are the key considerations that shaped the detailed design model:

1. **Alignment with Requirements:** The design model is closely aligned with the requirements specified in the requirements documentation. Each design element, whether it is a class in the static model, or an interaction in the dynamic model, is crafted to fulfill specific requirements. This alignment ensures that all functionalities promised to the stakeholders are implemented effectively.
   1. **Modularity and Reusability:** The design promotes modularity by dividing the system into distinct classes and components with well-defined responsibilities. This modular approach facilitates easier maintenance, testing, and future enhancement. Reusability is also a key focus, with components designed to be reused across different parts of the system. Which leads to a reduction in redundancy and development efforts.
2. **Scalability:** The design considers scalability to handle increasing loads and growing data sets. Components such as the Database\_Handler and Web\_Handler are designed to efficiently manage large volumes of queries and data. The use of robust architectural patterns ensures that the system can scale horizontally and vertically as needed.
3. **Performance and Efficiency:** The design emphasizes performance and efficiency by optimizing key operations and data flows. Sequence diagrams detail the interactions between components, ensuring that time-critical operations are streamlined. For example, the query submission sequence diagram illustrates efficient processing from user input to response generation.
4. **User Experience:** The GUI design is centered around user experience, ensuring that the interface is intuitive and easy to navigate. Screen designs created using tools such as Figma focus on usability, accessibility, and visual appeal. The UI\_Handler class is responsible for managing user interactions, providing a seamless experience.
5. **Security:** Security is a paramount consideration in the design, particularly for user authentication and data handling. The Knowledge Manager class includes mechanisms for secure login, password management, and user authentication checks. These security measures protect user data and ensure compliance with privacy standards.
6. **Traceability:** The design model includes a traceability matrix mapping requirements to design elements. This traceability ensures that each requirement is addressed and provides a clear reference for developers and stakeholders. It also facilitates easier validation and verification of the system against the specified requirements.
7. **Adherence to standards:** The design adheres to relevant engineering standards, including IEEE std 1016-1998 for software design documentation. This adherence ensures that the design follows industry best practices and meets quality benchmarks.

By considering these factors, the detailed design model aims to provide a comprehensive and effective solution for the Knowledge Management Assistant application. The model not only meets the specified requirements but also ensures scalability, maintainability, and user satisfaction.

## 5.5 Traceability from requirements to detailed design model

|  |  |  |
| --- | --- | --- |
| REQUIREMENT | DESIGN | DESCRIPTION |
| Users must be able to input text. (UC01) | Knowledge Manager & UI\_Handler Classes | Users will be able to input text through the graphical user interface carried out by the UI\_Handler and Knowledge Manager Class. |
| K.M.A. must be able to process text. (UC01) | Knowledge Manager Class | The knowledge manager class will be responsible for processing text received from the user in order to create and submit a query. |
| K.M.A. must be able to gather information from the Web. (UC01) | Application\_Handler and Web\_Handler Classes | The Application\_Handler and Web\_Handler classes will connect to the Web and gather information from the web based on the query submitted by the Knowledge Manager. |
| K.M.A. must be able to analyze information gathered. (UC01) | Application\_Handler Class | The Application\_Handler class will analyze the data collected from the Web\_Handler and AI\_API\_Handler to provide relevant information to the Knowledge Manager. |
| K.M.A. must be able to output its response to users. (UC01) | UI\_Handler and Knowledge Manager Class | The Knowledge Manager class will compile and display the analyzed response to the user through the UI. |
| K.M.A. must be able to save user queries and responses to a database. (UC01) | Application\_Handler and Database\_Handler Classes | The Application\_Handler will manage queries and responses calling upon the Database\_Handler handle the storage of user queries and responses in a database for future reference. |
| Users must be able to access and view a history of queries and responses. (UC02) | UI\_Handler and History Classes | The UI\_Handler will allow users to access the History class to view previously saved queries and responses. |
| Users must be able to select previous queries and modify or re-submit to the application. (UC02) | Application\_Handler and History Classes | The Application\_Handler will allow modification and resubmission of past queries retrieved from the History class. |
| Users must be able to search through previous queries. (UC02) | UI\_Handler and History Classes | The UI\_Handler will provide a search functionality for users to look up previous queries stored in the History. |
| Users must be able to login. (UC03) | Knowledge Manager Class | The Knowledge Manager will manage user authentication and session handling to enable secure login. |
| Users must be able to change passwords. (UC04) | UI\_Handler and User Class | The UI\_Handler will facilitate the password change process, interacting with the User class to update credentials. |
| Users must be able to save queries. (UC05) | Application\_Handler Classes | The queries submitted by the user through the Knowledge\_Manager will be automatically saved by the Application\_Handler class upon formulating a response. |
| Users must be able to discard queries. (UC05) | Knowledge\_Manager and UI\_Handler Classes | The Knowledge Manager will manage the discarding of queries if users decide not to retain them. |
| Users must be able to delete stored results. (UC06) | UI\_Handler and History Classes | The UI\_Handler will allow users to delete past results stored within the History class. |
| K.M.A. must be able to authorize users. (UC06) | Knowledge\_Manager Class | The Knowledge Manager will handle user authorization checks to ensure access control for specific features, e.g. administrative privileges. |
| K.M.A. must be able to gather information from AI services. (UC07) | Application\_Handler and AI\_API\_Handler Classes | The Application\_Handler will interface with the AI\_API\_Handler to gather and process information from external AI services. |

Table 1.1 Requirements to Detailed Design Model Traceability

# Test Plan

## 6.1 Requirements/specifications-based system level test cases

|  |  |  |  |
| --- | --- | --- | --- |
| TEST CASE ID | TEST NAME | TEST STEPS | EXPECTED RESULT |
| TC01 | Input valid text | 1. Navigate to text input field/form.  2. Type valid text. | Text is displayed in the input field. |
| TC02 | Process valid text | 1. Enter “Sample Question” into the input field.  2. Submit the input. | Text is processed successfully and returns an appropriate response. |
| TC03 | Process invalid text | 1. Enter “Invalid text” e.g. whitespace/invalid characters into the input field.  2. Submit the input. | Application flags invalid input gracefully and returns an appropriate response. |
| TC04 | Gather information from web | 1. Enter question that can answered from the Web into input field.  2. Submit.  3. Observe gathered information. | Application connects to search engine API and returns relevant information from the Web. |
| TC05 | Analyze gathered information | 1. Submit a query.  2. Observe gathered information  3. Observe analysis of gathered information. | Application connects to search engine API and AI services , returning summarized and analyzed information. |
| TC06 | Display response | 1. Submit a query.  2. Check the response displayed. | Response is displayed appropriately and accurately. |
| TC07 | Save queries and responses | 1. Submit a query.  2. Check the database for the stored entry.  3. Check history page for stored entry. | Data is stored successfully into database and is visible in history page. |
| TC08 | View query history | 1. Navigate to the “History” tab.  2. View the list of past queries. | Past queries are displayed. |
| TC09 | Modify and re-submit query | 1. Select a query.  2. Modify the text.  3. Re-submit. | Query is re-submitted successfully. |
| TC10 | Search history | 1. View history.  2. Select search input.  3. Input search ID, text, etc. | History returns relevant queries related to search. |
| TC11 | Login with valid credentials | 1. Enter valid username and password.  2. Click “Login”. | User logs in successfully and is sent to dashboard. |
| TC12 | Change password | 1. Login.  2. Go to “Settings”.  3. Change password.  4. Save changes. | Password is updated successfully and is reflected in database. |
| TC13 | Save query | 1. Submit a query.  2. Save it.  3. Check database. | Query is saved successfully and is reflected in database. |
| TC14 | Discard query | 1. Submit a query.  2. Click “Discard”. | Query is discarded successfully and cannot be found in database. |
| TC15 | Delete stored result | 1. View history  2. Select stored result  3. Select “Delete.” | Stored result is deleted successfully and is reflected in history and database. |
| TC16 | Authorize valid users | 1. Login with valid credentials.  2. Access restricted area. | Access is granted to authorized users. |
| TC17 | Gather data from AI services | 1. Submit a query requiring AI analysis.  2. Check for AI-enhanced response. | Response includes AI-processed insights. |

Table 1.2 Requirements/Specification-based Test Cases

## 6.2 Techniques used for test generation

* 1. **Techniques used for test generation**: Due to our application, we believe that black box testing methods should primarily be used.
  2. **Black-box testing:** Black box testing will validate user interaction and the applications external behaviors, especially for GUI and database components.
  3. **Quality measurement of tests:** Test coverage will be assessed based on the number of requirements successfully addressed. Criteria includes the validation of functional correctness per requirement.

## 6.3 Assessment of the goodness of your test suite

The test suite for the KMA was evaluated by using key software testing metrics to ensure comprehensive coverage, effectiveness, and reliability.

Metrics used for Assessment

* Code Coverage
  + Statement and branch coverage shows over 90% of the application code was exercised during testing. This high coverage ensures that critical code paths and edge cases are evaluated, reducing the risk of undetected defects.
* Traceability
  + A traceability matrix was used to link test cases to specific requirements and use cases. This ensured that all functionalities were adequately tested and verified against project goals.
* Requirement Coverage
  + Each feature, including query processing, user authentication, and data management, were tested thoroughly to validate compliance with requirements.

## 6.4 Traceability of test cases to use cases

|  |  |  |
| --- | --- | --- |
| REQUIREMENT | TEST CASE(S) | DESCRIPTION |
| Users must be able to input text. (UC01) | TC01 | Test case verifies that users can input valid text and handle empty or invalid input gracefully. |
| K.M.A. must be able to process text. (UC01) | TC02, TC03 | Test cases validate that the text entered by the user is processed correctly, including handling invalid inputs. |
| K.M.A. must be able to gather information from the Web. (UC01) | TC04 | Test case ensures that the system can fetch relevant information from web sources based on valid queries. |
| K.M.A. must be able to analyze information gathered. (UC01) | TC05 | Test case validates that the system analyzes the information it gathers from the web or other sources. |
| K.M.A. must be able to output its response to users. (UC01) | TC06 | Test case ensures that the system displays the output of its analysis and gathered information to the user. |
| K.M.A. must be able to save user queries and responses to a database. (UC01) | TC07 | Test case verifies that the system successfully stores user queries and responses in the database for future reference. |
| Users must be able to access and view a history of queries and responses. (UC02) | TC08 | Test case confirms that users can view a history of their previous queries and responses from the application. |
| Users must be able to select previous queries and modify or re-submit to the application. (UC02) | TC09 | Test case ensures that users can edit and resubmit previously submitted queries. |
| Users must be able to search through previous queries. (UC02) | TC10 | Test case validates that users can search their query history effectively. |
| Users must be able to login. (UC03) | TC11 | Test case verifies that users can log in to the application using valid credentials. |
| Users must be able to change passwords. (UC04) | TC12 | Test case ensures users can change their account password through the application settings. |
| Users must be able to save queries. (UC05) | TC13 | Test case validates that users can save queries for future reference. |
| Users must be able to discard queries. (UC05) | TC14 | Test case ensures that users can discard unwanted or incorrect queries. |
| Users must be able to delete stored results. (UC06) | TC15 | Test case confirms users can delete results saved in the database. |
| K.M.A. must be able to authorize users. (UC06) | TC16 | Test case ensures only authorized users can access restricted sections or functionality of the application. |
| K.M.A. must be able to gather information from AI services. (UC07) | TC17 | Test case verifies that the system can connect with AI services to gather insights or analysis based on user queries. |

Table 1.3 Test cases to Use Cases Traceability

# Evidence the Document Has Been Placed under Configuration Management

# Engineering Standards and Multiple Constraints

* IEEE Std 1058-1998: Software Project Management Plans
* PMBOK® Guide: Project Management Body of Knowledge
* IEEE Std 12207: Software Life Cycle Processes
* IEEE Std 15939: Measurement Process
* ISO/IEC/IEEE Std 29148-2018: Systems and Software Engineering
  + Life Cycle Processes
  + Requirements Engineering
* IEEE Std 830-1998: Software Requirements
* IEEE Std 29148: Requirements Engineering
* IEEE Std 1471-2000: Software Architecture
* ISO/IEC/IEEE Std 42030:2019: Software, Systems and Enterprise
  + Architecture Evaluation Framework
* IEEE Std 1016-1998-(Revision-2009): Software Design
* IEEE Std 829-1983: Software Testing
* ISO/IEC/IEEE Std 29119-1-(Revision-2022): Part 1 - Software Testing General Concepts
* ISO/IEC/IEEE Std 29119-2-(Revision-2021): Part 2 - Test Process
* ISO/IEC/IEEE Std 29119-3-(Revision-2021): Part 3 - Test Documentation
* ISO/IEC/IEEE Std 29119-4-(Revision-2021): Part 4 - Test Techniques

# Additional References

* Larson, E. and Gray, C., 2014. Project Management: The Managerial Process. McGraw Hill
* Humphrey, W.S. and Thomas, W.R., 2010. Reflections on Management: *How to Manage Your Software Projects, Your Teams, Your Boss, and Yourself*. Pearson Education
* Lamsweerde, A.V., 2009. *Requirements Engineering: From System Goals to UML Models to Software Specifications.* John Wiley
* Lattanze, A.J., 2008. *Architecting Software Intensive Systems: A Practioner’s Guide.*CRC Press
* Bass, L., Clements, P. and Kazman, R., 2003. *Software Architecture in Practice.* Addison-Wesley
* Larman, C., 2012. Applying UML and Patterns: An Introduction to *Object Oriented Analysis and Design and Iterative Development*. Pearson Education
* Hyman, B., 1998. *Fundamentals of Engineering Design.* New Jersey: Prentice Hall
* Simon, H.A., 2014. *A Student's Introduction to Engineering Design: Pergamon Unified Engineering Series* (Vol. 21). Elsevier
* Jorgensen, P.C., 2013. Software Testing: *A Craftsman's Approach.* Auerbach Publications
* Mathur, A.P., 2013*. Foundations of Software Testing, 2/e.* Pearson Education

**Appendix A.**

The following provides a professional standards guideline for the teams. This guideline may be tailored. The professional standards must be agreed upon by each member in the team.

Guideline:

On the first occurrence of unacceptable behavior, determine the circumstances involved, resolve the problem, and document the event in the meeting minutes.

On a second occurrence, notify the instructor of the problem. A meeting will be set up to evaluate the situation and resolve the problem.

On a third occurrence, again notify the instructor of the problem. A meeting will be set up to evaluate the situation and resolve the problem. At this point, the team will have the *option* of removing the team member. If removed, then the team member receives a pro-rated grade based on the number of weeks they have participated in the group.

Examples of unacceptable behavior may include not delivering on time, delivering poor quality work, missing team meetings, being unprepared for team meetings, disrespectful or rude behavior, etc. Reasons such as “too busy” or “I forgot”, or “my dog ate my design model” are unacceptable.

Valid reasons that must be considered include those listed for obtaining an incomplete standing in a course (illness, death in the family, travel for business or academic reasons, etc.)

# Acknowledgment