**SE 4485 Software Engineering Project**

**Final Report**

|  |  |
| --- | --- |
| Group Number | 10 |
| Project Title | Internet Research Assistant |
| Sponsoring Company | The Fellows Consulting Group |
| Sponsor | Jeff Miller |
| Students | Bakr Alkayali  Chloe Pascual  Vi Le  Ikraam Rahman  Mohammad Chauhan |

**Executive Summary**

The following final project report is an exhaustive report detailing the Internet Research Assistant’s many facets which starts off with touching on the preliminary project identification to outlining requirements specification. The project overview mentioned at the start of the report will discuss the choice and reasoning of the configuration management system and software development lifecycle chosen which in this instance is GitHub and the Spiral Model respectively. Moreover, the following will deep dive into the architecture and detailed design of the product we have implemented. The architecture will be that of MVC or a model, view, controller type and the reasoning of the decision based on the previous project phases will be highlighted accordingly. Furthermore, the design of various graphs and a Figma conceptual prototype is provided to illustrate the ideal product implementation All the aforementioned will also detail the traceability that was necessitated between the phases and their immediately following phases. The following section discussed will be detailing the development efforts for the actual implementation of the design into a fully working product, including the subset of technologies and development frameworks used as well as the relationships established between them. The last section covered is that of testing and the exhaustive test units to be created using JUnit and their deployment to successfully test the developed product against the use cases formed initially. The report's conclusion will discuss engineering requirements and constraints and a brief acknowledgment. All additional references utilized or inferred from are also provided at the end of this paper.

**Table of Contents**

EXECUTIVE SUMMARY.........................................................................................................................2

LIST OF FIGURES ....................................................................................................................................2

LIST OF TABLES ......................................................................................................................................2  
INTRODUCTION ......................................................................................................................................3

PROJECT MANAGEMENT PLAN...........................................................................................................4

REQUIREMENT SPECIFICATION .......................................................................................................10

ARCHITECTURE ....................................................................................................................................16

DESIGN ....................................................................................................................................................20

EVIDENCE THE DOCUMENT HAS BEEN PLACED UNDER CONFIGURATION MANAGEMENT ....................................................................................................................................................................27

ENGINEERING STANDARDS AND MULTIPLE CONSTRAINTS ....................................................32

ADDITIONAL REFERENCES ................................................................................................................33

ACKNOWLEDGMENT ...........................................................................................................................34

**List of Figures**

Figure 1 Spiral Model ................................................................................................................................5

Figure 2 Overall High-level Use Case Graphic Model.............................................................................10

Figure 3 MVC Diagram.........................................................................................................................19

Figure 4 Main Graphical User Interface of Eyera (Internet Research Assistant) ........................20

Figure 5 Static Model Class.......................................................................................................................21

Figure 6 Dynamic Sequence Diagrams.....................................................................................................22

**List of Tables**

Table 1 Risk Assessment Matrix ...............................................................................................................5

Table 2 Functional Requirement Traceability Matrix ..............................................................................17

Table 3 Traceability From Requirements to Detailed Design Model........................................................23

Table 4 Test Cases to Use Cases Traceability Matrix...........................................................................33

# Introduction

**Purpose and Scope**

* This document provides a comprehensive overview of the project's lifecycle, from inception to completion. It serves as a documented record of the project's objectives, methodologies, outcomes, and key deliverables. The report aims to communicate critical information to stakeholders, including project sponsors, clients, developers, and other relevant parties. Also, it serves as a reference document for future projects and a basis for assessing project success and identifying areas for improvement.
* The final report includes all aspects of the software engineering project:

1. Project Management Plan

2. Requirement Specification

3. Architecture

4. Design

6. Test Plan

7. Configuration Management, Engineering Standards, and Constraints

8. References and Acknowledgment

**Product Overview**

* Purpose of the product:

The primary purpose of Internet Research Assistant is to offer a user-friendly alternative to traditional internet search engines. It aims to simplify the search process and eliminate technical barriers for users who may not be familiar with complex search interfaces. The product is designed to be accessible to individuals with varying levels of technological proficiency. By offering a simplified and intuitive search interface, the assistant empowers users to quickly and confidently find the information they need on the internet. It aims to increase users' sense of control and competence when navigating online resources.

* Capabilities of the product: The product features a clean and intuitive user interface designed to minimize clutter and confusion. It utilizes familiar design elements and straightforward navigation options to enhance usability. It offers basic search functionality like traditional search engines, letting users enter keywords or phrases and receive relevant search results. The product also provides a refined feature for users to process their queries and discover relevant content more effectively.
* Scenarios for using the product: Internet Research Assistant offers users a streamlined search experience with features such as entering search queries, displaying relevant results, refining searches with filters, viewing detailed information, saving and retrieving searches, accessing updated results, and submitting feedback. Users can perform searches on various topics, explore search results, narrow down options with filters, view detailed information about selected results, save searches for future reference, discover updated content, and provide feedback for continuous improvement.

**Structure of the Document**

* Introduction: Provides an overview of the project objectives, scope, and purpose of the report.
* Project Management Plan: Describes the project organization, life cycle, schedule, risk management, and progress tracking mechanisms.
* Requirement Specifications: Details the stakeholders for the system, requirements elicitation process, functional and non-functional requirements, and traceability matrix.
* Architecture: Presents the high-level architectural design of the software system, including system components and interfaces.
* Design: Provides detailed design specifications for individual components and modules, design patterns used, and interface designs.
* Test Plan: Outlines testing objectives, strategies, scenarios, and metrics for evaluating system performance and functionality.
* Configuration Management, Engineering Standards, and Constraints: Documents configuration management processes, compliance with engineering standards, and project constraints.
* References: Lists all citations and references used throughout the document.
* Acknowledgment: Recognizes contributions from team members, stakeholders, and external collaborators.

**Terms, Acronyms, and Abbreviations**

* FR—*Functional Requirement*
* IRA – *Internet Research Assistant*
* MVC — *Model View Controller*
* NFR— *Non-Functional Requirement*
* RTM — *Requirement Traceability Matrix*

# Project Management Plan

**Project Organization**

The Internet Reasearch assistant’s project development was guided by the due dates of each deliverable as a phase, and dividing up each one into tasks that could be completed by each member. Bakr took on extra responsibilities as team leader, in charge of organizing meetings and facilitating communication between the team and Dr. Wong.

Alongside deliverable development, the development of the demo was organized into two teams: front-end consisting of Vi and Chloe, and back-end consisting of Bakr, Ikraam, and Mohammed. Tasks were created based on different feature requirements, created individually and merged using GitHub as a repository.

**Lifecycle Model Used**

Given the deliverable-driven approach to the development of the Internet Reasearch Assistant, making use of the Spiral Model was most appropriate, allowing each development of a deliverable to act as a phase. Beginning with the start of the project, this project made use of eight iterations of the spiral while visiting the four main quadrants (planning, risk analysis, development, and review & analysis) in each phase. This allowed each deliverable to be handled individually with time to plan, assess, create, and review quality of the deliverable before moving onto the next phase.

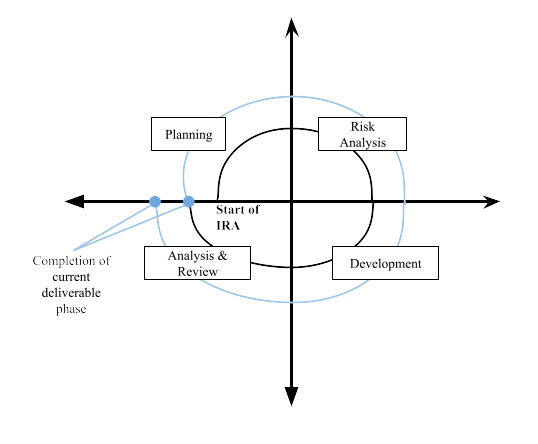


Fig 1. Spiral Lifecycle Model Phase

**Risk Analysis**

Using a risk assessment matrix, most of the risks that could be encountered during the development of IRA are deemed to be low risk. This is mostly due to the likelihood of the risks happening staying low despite their severity being high. The highest risk was determined to be the overload of schoolwork which is more likely than the other risks and can affect multiple team members at once. To mitigate this risk, and the others listed in the table below, the best plan of action for team members to communicate any issues that may arise as soon as possible.

Table 1: Risk Assessment Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk Event** | **Likelihood** | **Impact Severity** | **Calculated Risk Factor** |
| Incompletion of deliverable | 0.1 | 0.9 | 0.09 |
| Severe weather/Power outage | 0.2 | 0.4 | 0.08 |
| Team member sick/out of commission | 0.3 | 0.3 | 0.09 |
| Schoolwork overload | 0.7 | 0.3 | 0.21 |
| External job responsibilities | 0.2 | 0.2 | 0.04 |
| Software crash | 0.1 | 0.6 | 0.06 |
| Loss of data | 0.1 | 1.0 | 0.10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk Factor Ratings** | **0.0 - 0.2** | **0.2 - 0.4** | **0.4 - 0.8** | **0.8 -1.0** |
| Low risk | Moderate to high risk | Serious risk | Catastrophic risk |

**Software and Hardware Resource Requirements**

The development of this project required a variety of software and hardware resources to use as listed below:

Deliverable Development:

* Discord
* Word
* PowerPoint
* ClipChamp
* Diagrams.net

IRA Software Development:

* GitHub
* VSCode
* Figma

Hardware:

* Each member needs access to a computer that can run the aforementioned software without issue

Furthermore, most software and hardware were not new to the members. Though XXXX (Do not forget to describe what *new* software or hardware each team member learned during the project)

* Vi: As a front-end developer, this was my first time using React. I learned how to create functional components, manage state with hooks, and handle routing with React Router. I also gained experience with Node.js and npm for managing project dependencies and improved my skills with Git for version control.
* Bakr: While I’ve dealt with google APIs before, I've never used the google search API. It was a unique experience, learning how to not only call it but also customize the amount of data it query, how the filters would be applied and the logic behind it. How the data is being stored, and what works and what doesn't through trial and error.
* Ikraam: Previously only having taken frontend roles, learning how the google search API returned JSON and how to effectively parse the necessary information for our application use was something new I was fortunate to learn after taking the backend initiative this project. Further learning new version control system techniques and mechanisms were an added plus.

**Deliverables and Schedule**

Project Management Plan - *February 2nd*

* Revised Project Management Plan – *February 23rd*
* 2nd Revised Project Management Plan – *March 18th*

Requirements Documentation – *February 23rd*

* Revised Requirements Documentation – *March 18th*

Architecture Documentation – *March 22nd*

Demo Development Begins – *April 4th*

Detailed Design Documentation – *April 5th*

Testing Plan – *April 19th*

Demo Development Complete – *April 30th*

Final Presentation Slides – *May 1st*

Final Project Report – *May 3rd*

**Monitoring, Reporting, and Controlling Mechanisms**

Communication of the project is handled primarily through Discord. Weekly meetings set-up between the team members and the sponsor are used to discuss the current phase of the project, including risks, progress, and guidance while also leaving time to plan for the next phase. During these meetings, the tasks of each deliverable were assigned to team members who would then be responsible for completing them. Updates are visible though configuration management systems set in place, but supplemental updates are also sent in the Discord to notify others of the status of their tasks. It is also the responsibility of each member to notify the rest of the team if they are experiencing any issues with their tasks wherein a plan could be made to provide help, assuring the task is completed.

With Bakr as team leader, he also oversaw weekly attendance at meetings and summited each deliverable via email to Dr. Wong, the Teaching Assistant, and our sponsor Jeff Miller as another reporting mechanism.

**Professional Standards**

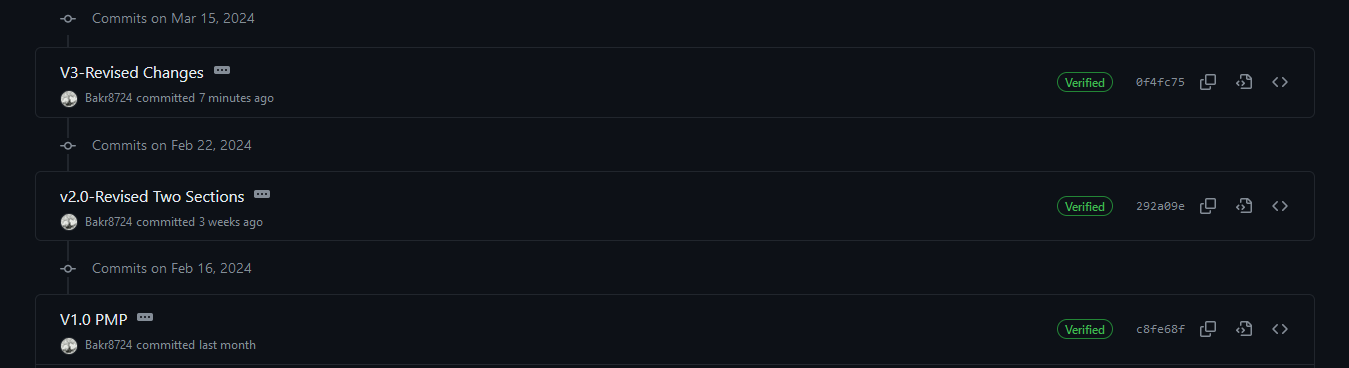
The team followed a general professional code of conduct which emphasized:

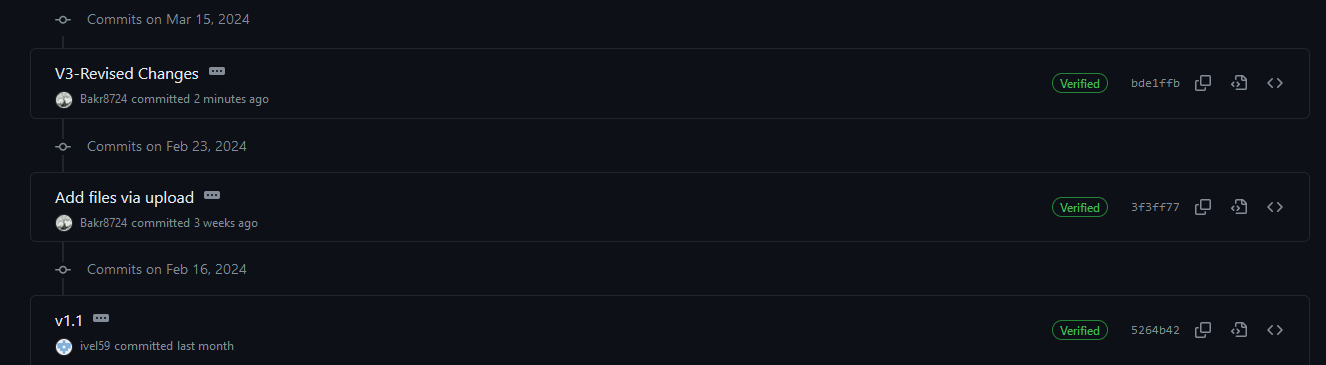
* Communication
* Respect
* Participation
* Growth
* Quality

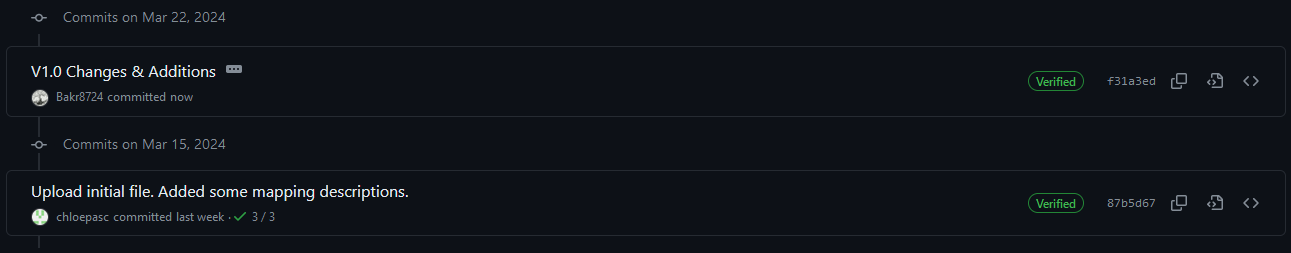
Additional information regarding the standards of the deliverables can be found in the “ENGINEERING STANDARDS AND MULIPLE CONSTRAINTS” section of this document.

**Evidence all the artifacts have been placed under configuration management**

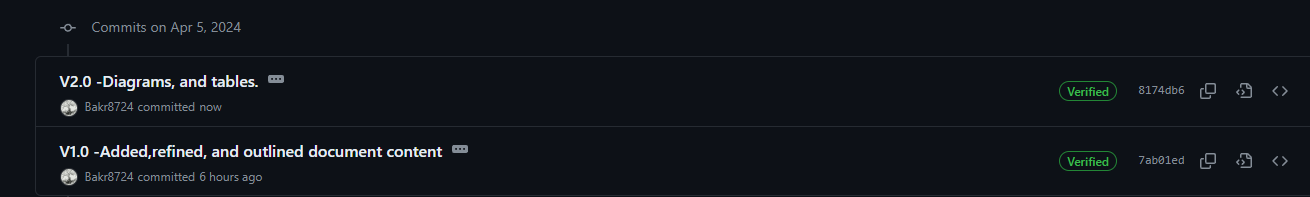
Project Management Plan – *GitHub*

  
Requirements Documentation – *GitHub*

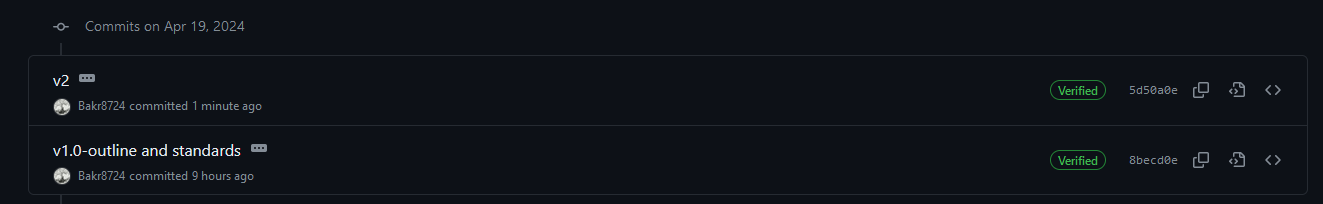
  
Architecture Documentation – *GitHub*



Detailed Design Documentation – *GitHub*

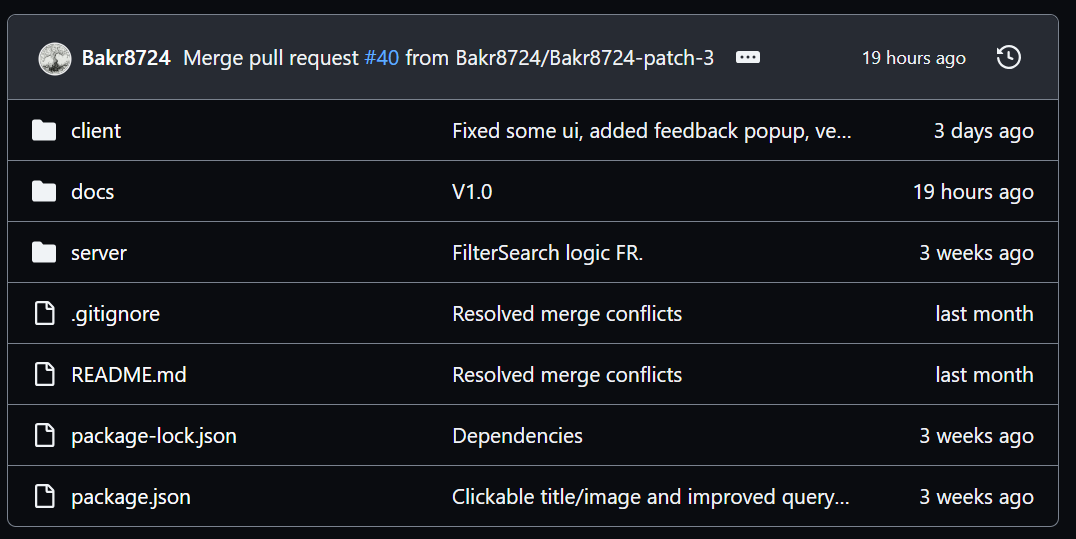


Testing Plan – *GitHub*



Final Project Report – *GitHub*

Demo Development – *GitHub*

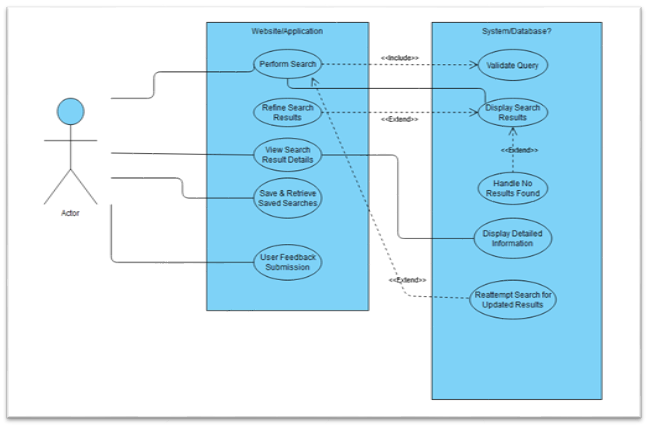


**Impact of the project on individuals and organizations**

The pitch of the Internet Reasearch Assistant was to do what its name implies—aid with internet research. As such the target user domain is anyone who may use the internet for research such as students or working professionals. This is a wide audience that may have a use for the Internet Reasearch Assistant in many fields where, overall, IRA will promote research, development, and innovation in society. Because of this, when defining requirements for the IRA simplicity and customization concepts were prioritized so that its application could be viable for an array of research fields. Additional features other than the base search and refining functionality that were not able to be implemented for the demonstration include being able to save and resume searches, which work to streamline and organize the research process. To further cement the Internet Reasearch Assistant as a standard tool, its use and distribution would be free for individuals and paid for organizations. In closing, IRA was developed by a team of students engaged in their own research to develop the system and hope that its deployment would prove handy to anyone else working on research for a project they feel passionate about.

# Requirement Specifications

Use case model for functional requirements

Fig. 2 Overall High-level Use Case Graphic Model.

**Textual Description**:

* The use case model visually represents the interactions between the users and the search system of the website/application. It consists of the following main functionalities:
  + **Perform Search**: This is the primary action where a user initiates a search by entering a query into the system.
    - Sub Use Case: Validate Query ensures that the input from the user meets the system’s requirements for a valid search term.
  + **Refine Search Results**: Post-search, users can refine the results they receive, filtering them according to various criteria for more precise information.
  + **View Search Result Details**: Users can select a search result to view more detailed information about the item.
  + **Save & Retrieve Saved Searches**: This functionality allows the user to save their searches and access them later.
  + **User Feedback Submission**: Users can submit feedback about the system, which is independent of the search process itself.
* The System’s backend functionalities include:
  + **Display Search Results**: Upon a successful search query, the system processes and displays the results to the user.
  + **Handle No Results Found**: If no results are related to the query, the system handles this event, typically by informing the user.
  + **Display Detailed Information**: Further information about a selected search result is provided to the user upon request.
  + Reattempt Search for Updated Results: Users have the option to re-execute a search to get updated results.
* Note: Each use case is linked to the next logical step in the process with solid lines indicating direct relationships, while dashed lines with extend indicate optional paths that users may take.

# **Rational For Your Use Case Model**

* The rationale behind this use case model is to provide a clear, high-level view of the system’s functionality from the user’s perspective. It is designed to:
  + **Highlight Key Interactions:** The model emphasizes the core functionalities that users will interact with, such as searching, refining results, and viewing details, ensuring that the system’s primary purpose is well understood.
  + **Allow for Scalability**: By separating website and system/database functionalities, the model allows for the system to be scaled and evolved over time, with new features potentially being added as extensions to existing use cases.
  + **Enhance Communication**: It provides a visual tool that can be used to communicate the system’s functionality to stakeholders, developers, and users in a straightforward and accessible manner.
  + **Streamline Testing and Validation**: The model serves as a guide for creating test cases, ensuring that all user pathways are covered and work as intended.

# **Use Case Model For Functional Requirements**

**High-Level Use Case**

***1.*** ***Perform Search***

\*Sub Use Cases:

1.1 Enter Search Query

* **Use Case Name -** Enter Search Query
* **Participating Actors -** User, Database
* **Entry Condition(s) -** 
  + User wants to make a search
  + User clicks on search bar
* **Normal Flow of Events**
  + User enters their search query
  + System processes query keywords
  + System narrows down search results related to query
  + Search results are displayed to user
* **Exit Condition(s)**
  + User clicks off of search bar
* **Exceptions (Alternate Flow of Events)**
  + User clicks on search bar
  + User clicks off search bar
* **Special Requirements**
  + None

***2. Validate Query***

* **Use Case Name -** Validate Query
* **Participating Actors -** User, Database
* **Entry Condition(s) -** 
  + A search query has been entered and submitted by user
* **Normal Flow of Events**
  + Keywords are evaluated by system
  + Relevant pages indexed in database are brought up
* **Exit Condition(s)**
  + Query is validated and results are found
* **Exceptions (Alternate Flow of Events)**
  + Keywords are evaluated by system
  + There is a typo in keywords
  + System suggests replacement for typo in query
  + User selects updated query
  + Relevant pages indexed in database are brought up
* **Special Requirements**
  + Display loading icon or message

***3. Display Search Results***

* **Use Case Name -** Display search results
* **Participating Actors -** User, Database
* **Entry Condition(s) -** 
  + Indexed pages related to search query are found
* **Normal Flow of Events**
  + Indexed pages relevant to query are found
  + Summaries of relevant information on top pages are formed
  + Summaries of information are displayed to user
* **Exit Condition(s)**
  + Information is displayed to user
* **Exceptions (Alternate Flow of Events)**
  + No indexed pages relevant to query are found
  + Error message is displayed to user
* **Special Requirements**
  + Results are displayed in a timely manner

***4. Handle No Results Found***

* **Use Case Name -** Handle no results found
* **Participating Actors -** User, Database
* **Entry Condition(s) -** 
  + No indexed pages related to search query are found
* **Normal Flow of Events**
  + No indexed pages relevant to query are found
  + Display error message to user
  + (Optional): display possible relevant and/or popular results
* **Exit Condition(s)**
  + Error message is displayed to user
* **Exceptions (Alternate Flow of Events)**
* **Special Requirements**
  + Reusable Use Cases:
    - Validate Query
    - Display Error Message

***5****.* ***Refine Search Results-***

\*Sub Use Cases:

2.1 Apply Filter

* **Use Case Name -** Apply filter
* **Participating Actors -** User, Database
* **Entry Conditions -** 
  + User filters search to narrow results
* **Normal Flow of Events -**
  + User enters search query what they’d like to search
  + System will display the search results
  + System will allow the option for the user to filter search results
* **Exit Conditions -**
  + Exits once the system has displayed search results
* **Exceptions -**
  + User decides to search again
  + No search results are found
    - Gives error message

***6. View Search Result Details-***

\*Sub Use Cases:

3.1 Select Search Result

* **Use Case Name -** Select Search Result
* **Participating Actors -** User, Database
* **Entry Conditions -** 
  + User wants to make a search
  + User enters their search into the search bar
  + System displays information regarding the search
* **Normal Flow of Events -**
  + User wants to make a search
  + User enters their search into the search bar
  + System displays information regarding the search
  + System then gives the user relevant topics that correlate to the search
* **Exit Conditions -**
  + The system has displayed the search results
* **Exceptions -**
  + If the user decides to search another topic
  + If the user decides to click on the relevant topics that the system has provided

***7. Display Detailed Information -***

* **Use Case Name -** Display Detailed Information
* **Participating Actors -** Database
* **Entry Conditions -** 
  + User enters search into search bar
  + System displays search results to the user
  + System orders information from most relevant to least
* **Normal Flow of Events -**
  + User enters search into search bar
  + System displays search results to the user
  + System orders information from most relevant to least
  + System displays other topics that can provide more information to the user about the topic
* **Exit Conditions -**
  + The system displays the results of the search query to the user
  + No search results are found
    - Gives error message
* **Exceptions -**
  + No search results found
    - Gives error message to user
  + User continues searching
* **Special Requirements -**
  + Reusable Use Cases:
    - Log User Interaction (for improving search relevance)
    - Display Other Relevant Information
    - Display error message

***8. Save and Retrieve Saved Searches\**** Optional Implementation

● Use Case Name - Save Search Query

● Participating Actors: User, Database

● Entry Conditions:

○ Users want to save their search for future reference.

● Normal Flow of Events:

○ User saves a search query.

○ System confirms the search has been saved.

● Exit Conditions:

○ Users can access saved searches later.

● Exceptions:

○ Issues with saving due to system error or connectivity issues.

● Special Requirements:

○ Provide a user-friendly interface for managing saved searches.

***9. Reattempt Search for Updated Results\**** Optional Implementation -maybe take out at end

● Use Case Name: Reattempt Search for Updated Results

* Participating Actors: User, Database
* Entry Conditions:
  + User has previously executed a search and wants to re-execute it for a potentially better response/answer.
* Normal Flow of Events:
  + User selects a previously executed search query to reattempt it.
  + System processes the search query as a new request.
  + System displays the search results updated since the last search was made.
* Exit Conditions:
  + User views the updated search results.
* Exceptions:
  + There are no new or better results since the last search then the system informs the user that there is no update to the former answer.
* Special Requirements:

***10. User Feedback Submission***

* Use Case Name: User Feedback Submission
* Participating Actors: User, System
* Entry Condition:

The user has feedback to submit.

* Normal Flow if Events:

1. The user initiates the feedback by clicking a button.
2. System prompts the user to enter their feedback.
3. The user enters their feedback to the chat interface.
4. System confirms receipt of the feedback and thanks the user.
5. System sends feedback to database for review and action.

* Exit Conditions:
  + The user’s feedback has been successfully submitted and acknowledged by the system.
* Exceptions (Alternate Flow of Events):
  + The user cancels the feedback submission process before completing it. In this case, the system acknowledges the cancellation and returns to the Home interface.
  + The system fails to submit the feedback due to a system error. The system informs the user about the error and asks them to try again later.
* Special Requirements:
  + The submission process should be simple and straightforward to ensure users can easily submit their feedback.

# Architecture

**Architectural style(s) used**

* MVC
* This architecture supports various features of Internet Research Assistant application by separating into Model- Controller- View, this allows for efficient code organization, modular components, and greater flexibility. Each component can be developed and tested independently, which can improve the quality and reliability of the application. It also makes the application more scalable and maintainable, as new features or changes can be implemented in their respective components without affecting the others.

**Architectural model**

* Our architectural model is defined by the following subsystems:
  + **Model<<Subsystem>>**: Manages data and business logic, including search algorithms and data storage. This involves querying database and APIs for research data, processing and analyzing data, and managing user data.
  + **View<<Subsystem>>**: Handles the presentation layer, responsible for displaying the user interface and presenting search results. This includes displaying search results, handling user input, and updating the display based on changes in the Model.
  + **Controller<<Subsystem>>**: Acts as an intermediary between the Model and View, processing user input, and executing search queries. This includes processing user input, updating the Model based on that input, and updating the View based on changes in the Model.

**Technology, software, and hardware used**

* Front-End: HTML, CSS, JavaScript (React framework)
* Back-End: Node.js for server-side logic. Google Search Engine API as the search engine
* Database: mySQL for data structure.
* Infrastructure: Proxmox or Docker ensuring scalability and deployment efficiency.

**Rationale for your architectural style and model**

* The justification for choosing the MVC architecture is that the model offers and allows for ease of code creation, modifiability, and maintainability. The architecture style being a product development architecture focuses on separating the different varying parts of an application from business logic to UI logic, allowing the developers the ability to separate the software project into smaller modules, each of which can be developed, tested, and integrated separately. This is especially valuable when creating an Internet research assistant as the breadth and volume of requests can necessitate complex code that must work in cohesion. While all the aforementioned features and processes of the MVC architecture aid in development, it also significantly improves the success of maintainability as individual layers can be modified or updated easily. Moreover, the model allows for ease of scalability if required in the future based on the volume of requests directed to the internet research assistant.

**Traceability from requirements to architecture**

* Using the Requirements Traceability Matrix (RTM) to link each requirement with the architectural components or subsystems responsible for fulfilling it, ensuring that all requirements are covered by the system’s design.
* Functional RTM

Table 2: Functional Requirement Traceability Matrix

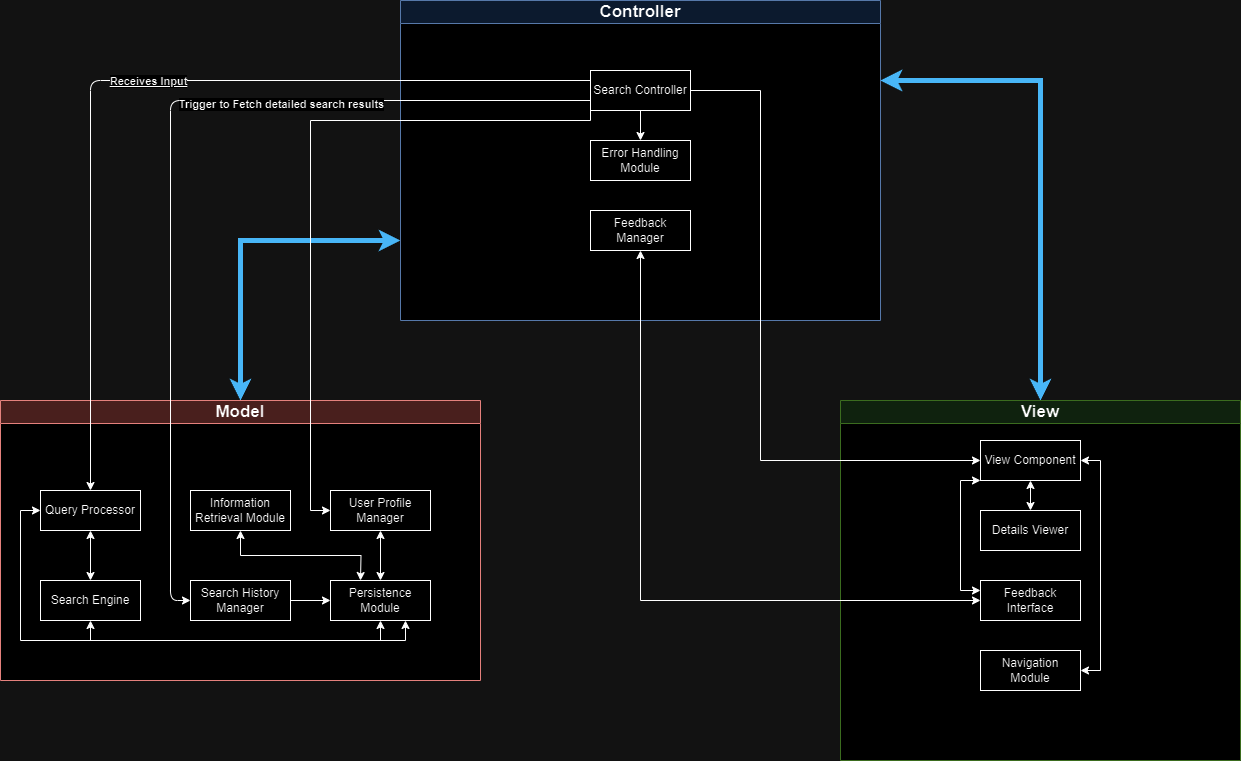
|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Requirement Description | Architectural Component(s) | Rationale |
| FR1 | Perform Search | Search Controller, Query Processor | Initiates search process, handling user inputs and processing queries using Google Search Engine API. |
| FR2 | Validate Query | Query Process | Validates query against system criteria, ensuring only viable searches are conducted. Will offer suggestions for invalid queries. |
| FR3 | Display Search Results | View Component, Search Engine | Renders search results in UI, where Search Engine retrieves relevant data from the database. |
| FR4 | Handle No Results Found | View Component, Error Handling Module | Informs users when no results match their query, enhancing UX by providing feedback and suggestions. |
| FR5 | Refine Search Results | Filter Module, View Component | Allows users to narrow down search results, improving result relevance. |
| FR6 | View Search Result Details | Details Viewer <<Subsystem>> | Displays detailed information about a selected search result. |
| FR7 | Display Detailed Information | Information Retrieval Module | Fetches and presents extensive details on user-selected search results |
| FR8 | Save & Retrieve Saved Searches | User Profile Manager, Persistence Module | Manages user data for saved searches, offering personalization and convenience. |
| FR9 | Reattempt Search for updated Results | Search History Manager, Query Processor | Facilitates re-execution of searches, ensuring users have access to the most current information. |
| FR10 | User Feedback Submission | Feedback Manager <<Subsystem>> | Collects user feedback for system improvements, enhancing the overall quality. |

* Non-Functional RTM

Table 3: Non-Functional Requirement Traceability Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | **Requirement Description** | Architectural Component(s) | Rationale |
| NFR1 | Usability-User Interface | View Component (MVC) | Ensures an intuitive and accessible interface for all users. |
| NFR2 | Usability-History & Navigation | User Profile Manager, Navigation Module | Facilitates easy access to previous searches and smooth navigation through search history. |
| NFR3 | Performance-Response Time | Caching Mechanism, Search Engine | Reduces search response times, providing a smooth user experience. |
| NFR4 | Performance-Scalability | Load Balancer, Database Replication | Supports system scalability to handle increased user loads efficiently. |
| NFR5 | Portability | Cross-Platform Compatibility Layer | Ensures the application functions across various platforms and devices. |
| NFR6 | Reliability | Database Replication, Error Handling Module | Ensures system reliability and consistent user-acceptable behavior. |
| NFR7 | Efficiency | Resource Management Module | Optimizes resource usage, ensuring the system runs efficiently. |
| NFR8 | Accessibility | Accessibility Features | Incorporates accessibility standards to ensure usability for all users, including those with disabilities. |

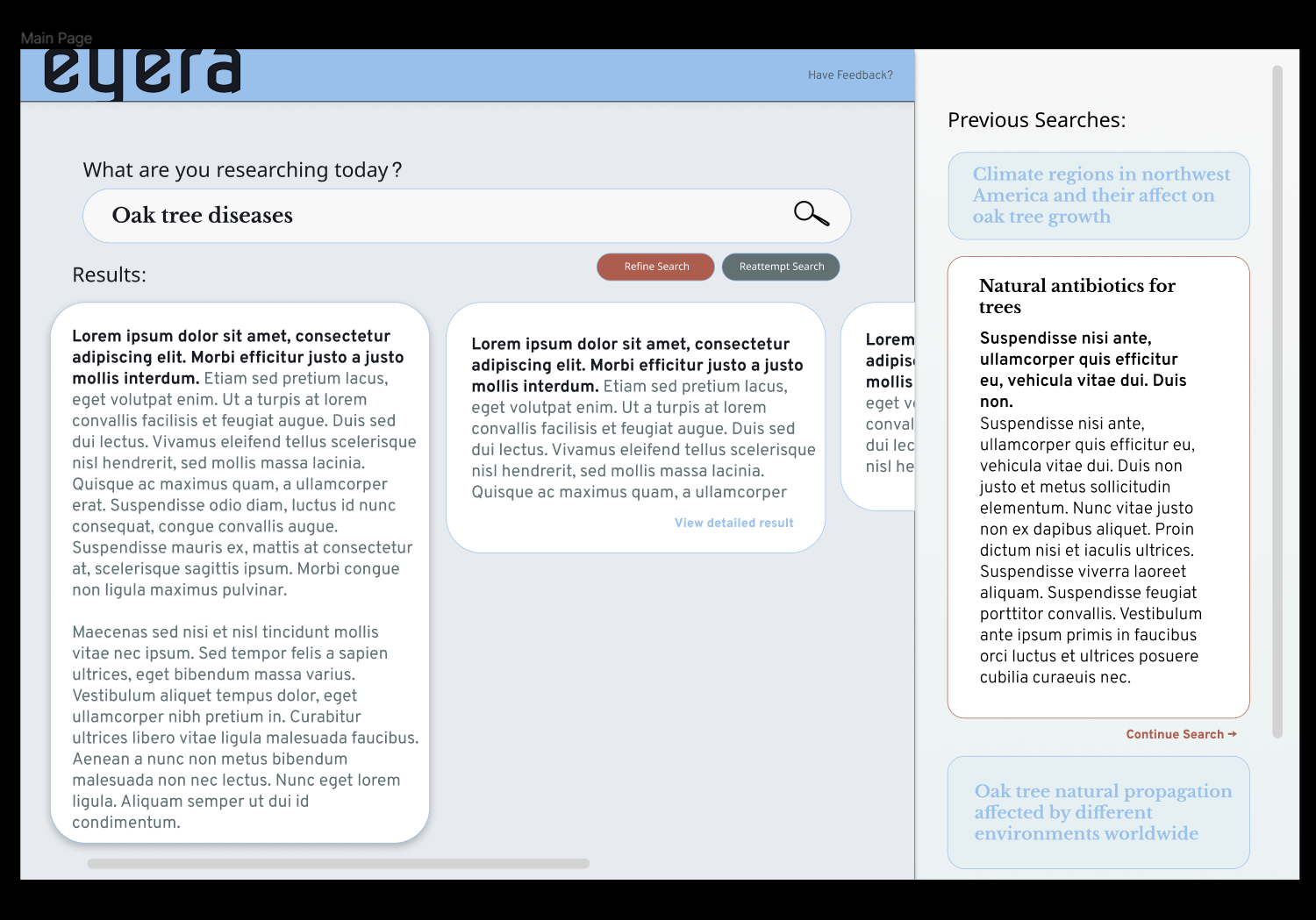
# **MVC Diagram:**

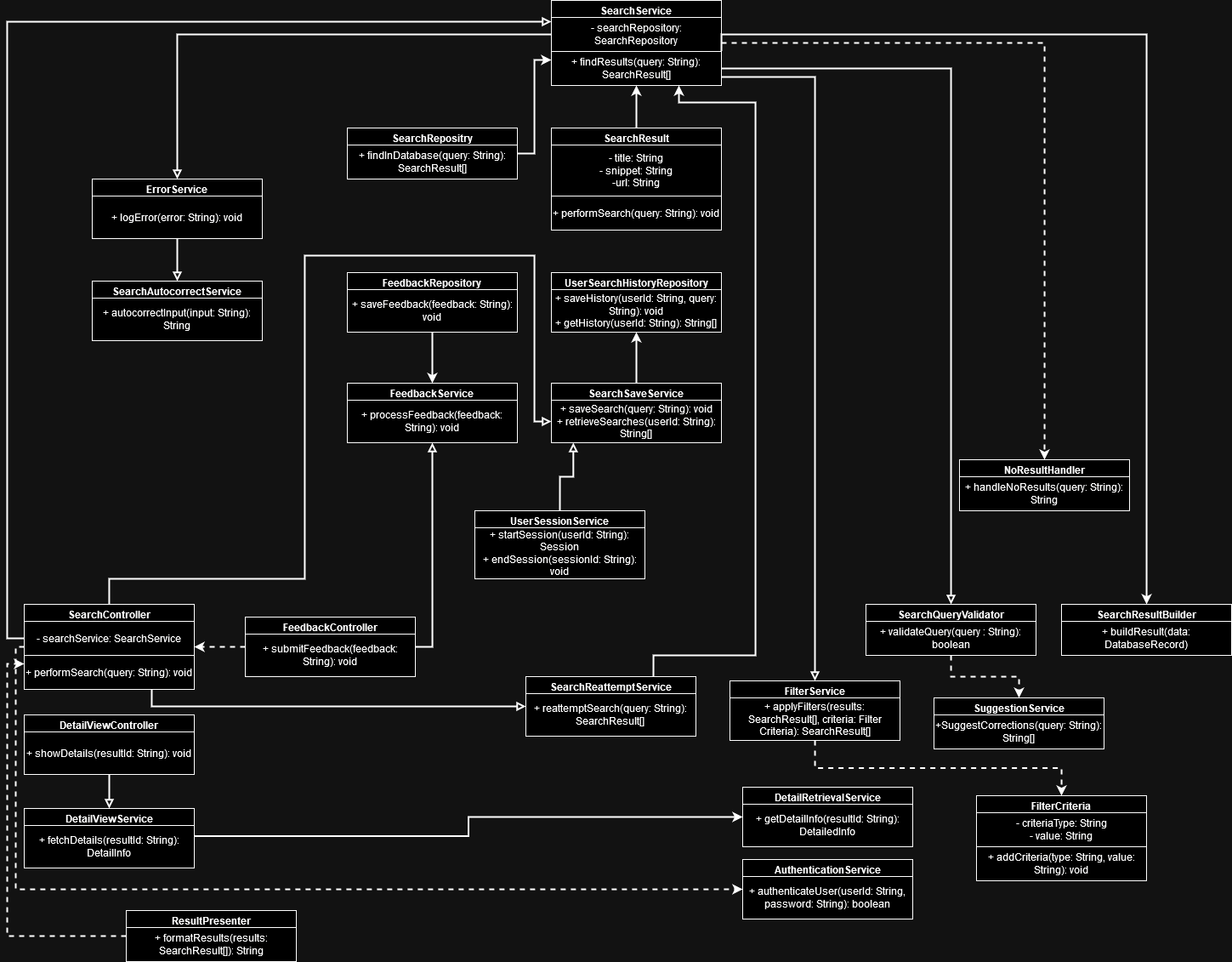
Fig. 3. MVC Diagram

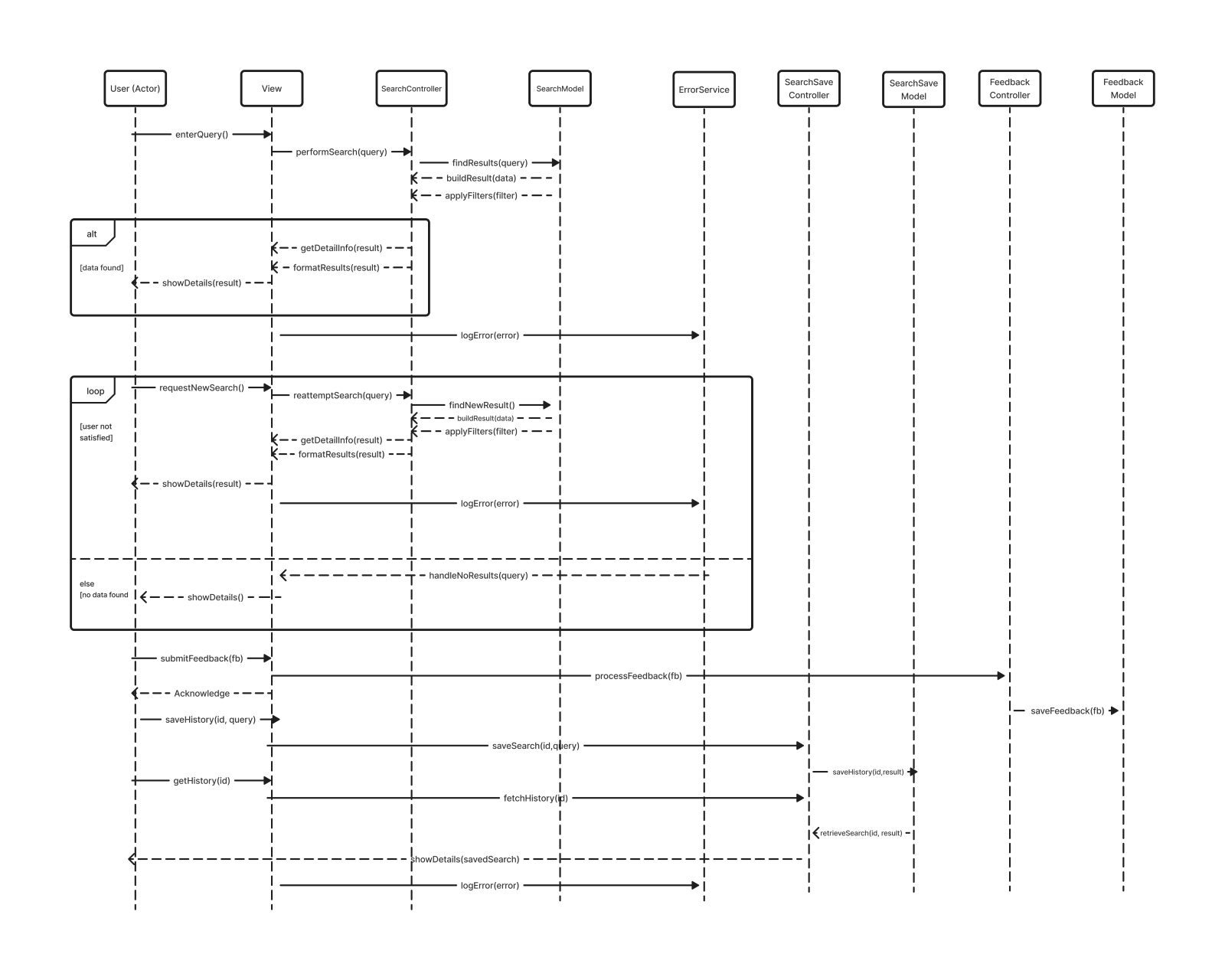
# 

# Design

**GUI (Graphical User Interface) design**

 Fig. 4. Main Graphical User Interface of Eyera

 Fig. 5. Static Model Class Diagram

Fig. 6. Dynamic model – sequence diagrams

**Rationale for your detailed design model**

Our detailed design model covers both dynamic and static versions of the software with a class diagram and a sequential diagram. This provides a full view of the digital architecture in application and action that will help with the development process. These diagrams consider the found requirements and establish classes, objects, and methods depicted in the class diagram and show how these things communicate in the sequential diagram. With this detailed design model, the architectural factors that determine how requirements will be met have been fully identified and documented.

Table 4: Traceability From Requirements to Detailed Design Model

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Examples/Scenarios** | **Design Element** | **Design Description** |
| **FR1:**  **Perform**  **Search/Enter**  **Search Query** | By shifting the focus towards utilizing the Google Search Engine API while also retaining the flexibility to enhance and customize search results according to our projects scope. | SearchController  SearchService  SearchQueryValidator  APIResultProcessor  ResultPresenter | Captures user input from the UI. Validates the input for basic checks before sending it to the SearchService. Handles user feedback based on the search outcome (result found, no results, suggestions.)  Prepares the query for the Google API, including additional parameters to refine the search as needed. Communicates with the Google Search Engine API to perform the search. Processes the API response to format or filter the results as needed.  Validates the search query against both basic criteria and any specific requirements or optimizations for the Google API. Ensures the query is not empty, overall broad, or in a format likely to lead to unhelpful results.  Takes the raw results from the Google API and processes them according to the application’s requirements such as filtering and augmenting with additional data.  Formats the processed results into a user-friendly format for display. Handles the logic for presenting the results to the user through the UI, including any interactive elements like links or action buttons. |
| **FR2:**  **Validate**  **Query** | If a user types “javacsrpit” into the search box and attempts to search. Then the SearchQueryValidator checks the query and approves it, but the SuggestionService will pick up the misspelling and suggests the correct term, the user will then be prompted with something like “Did you mean: javascript”, this process allows the user to correct the query before it proceed, thus saving API requests. | 1. SearchQueryValidator  2. SuggestionService | 1.Performs client-side validation of the search query. Checks for non-empty, properly formatted queries, and any additional parameters such as checking against ineffective queries, a refinement process of sorts.  2. Activated if SearchQueryValidator finds the query likely to return poor results. Offers suggestions or autocorrects the query based on common misspellings or search optimizations. |
| **FR3:**  **Display**  **Search**  **Results** | Say for instance a user looks up “latest tech news” the SearchService fetches the results from the Google API, which includes the various articles and sources. The ResultPresenter goes over the results, and converting each into an HTML snippet using something like toHTML method of SearchResult . This makes the results readable with a nice UI touch to them showcasing summaries with clickable titles, making it easier for the user to browse through the info and select topics of interest. Which means we won’t depend on how Google API presents its data, since we’ll be adjusting it to our/users needs. | 1. ResultPresenter    2. SearchResult | 1. Formats the raw API search results into a structure suitable for the web UI. Handles pagination or scrolling mechanisms if the API returns many results.  2. Encapsulates individual search result data, making it easy to manage and display each result. May include methods for rendering HTML or other web-friendly formats. |
| **FR4:**  **Handle No**  **Results**  **Found** | User looks for something that's not common, leading to zero results. Instead of saying “no result found” the application would call NoResultHandler to provide a better tip, be it different keywords, related topics, etc. in a friendly and helpful way. | 1. NoResultHandler | 1.Determines the appropriate response when no results are found, which could involve providing search tips, suggesting related queries, or simply informing the user in a friendly manner. |
| **FR5:**  **Refine**  **Search**  **Results / Apply Filter** | A user searches for “potatoes” and receives a wide array of results. But the user is interested in results that are related to the “History”. The user then applies a filter with the source keyword “History”. FilterService uses FilterCriteria to sift through the results, returning only those that match the source criteria. Then the UI updates to display the search results. | 1. FilterService  2. FilterCriteria | 1. Applies filters to the search results based on the define criteria(categories). Could adjust results in response to user-selected filtering options.    2. Encapsulates the parameters for filtering, which might include keywords, data ranges, source preferences, etc. |
| **FR6:**  **View Search**  **Result Detail/**  **Select Search Result** | If a user is interested in an article titled “Renewable Energy.” Upon clicking this search result, DetailViewController is triggered, which in turn invokes DetailViewService to fetch more comprehensive information about this article. DetailViewService retrieves the article’s full text, images, and any other details. DetailViewController then ensures this content is nicely formatted and displayed to the user, providing an in-depth view of the article directly within the application. | 1. DetailViewController  2. DetailViewService | 1.Handles user actions for selecting a search result and requests detailed information for display. Manages the UI aspects of presenting detailed information to the user.    2. Responsible for fetching detailed information about a selected search result. This might involve making an additional API call to Google or querying another service/database for enriched data. |
| **FR7:**  **Display**  **Detailed**  **Information** | A user may display extra results information using the DetailViewController where DetailViewService will be prompted to display addtional information other than the initial description of a search result upon first view. | 1. DetailRetrievalService | 1. Retrieves detailed information about a particular search result from the database. |
| **FR8:**  **Save and**  **Retrieve**  **Saved**  **Searches** | A user looks up an article. They decide to save this search result to revisit these articles later. Upon clicking the “Save Search” button, SearchSaveService.saveSearch is invoked, storing the query and the corresponding results. The user can retrieve their saved searches through a dedicated section of the application, where SearchSaveService.retrieveSavedSearches provides a list of their previously saved searches, allowing easy access to past queries and results. | 1. SearchSaveService    2.UserSearchHistoryRepository | 1. Manages saving and retrieving searches for users.  2. Interfaces with storage to save and retrieve user search history. |
| **FR9:**  **Reattempt**  **Search for**  **Updated**  **Results** | Say a user saved a query from a while back, regarding some tech company and their progress. Now the user wants to see how far they’ve gotten. So they opt to reattempt the search. Using SearchReattemptService, the system retrieves the original query from their saved searches and re-executes it. The service then displays to the user whether any new results are available since their last search. | SearchReattemptService | Handles logic for re-executing past searches to fetch updated results. |
| **FR10:**  **User Feedback**  **Submission** | A user wants to submit feedback, a simple form in the UI is to be present. Once the feedback is submitted, FeedbackController takes over, passing the feedback to FeedackService for processing. A simple acknowledgement that their feedback has been received. In the backend the feedback is stored by FeedbackRepository ready to be accessed by the development team. | 1. FeedbackController    2. FeedbackService    3. FeedbackRepository | 1. Accepts feedback from the user through the UI  2. Processes and persists feedback to the database.  3. Handles storage and retrieval of user feedback. |

# Test Plan

**Requirements/specifications-based system level test cases**

* **Test Case ID: 1** 
  + Requirements: User must be able to search for content using keywords.
  + Test Steps:
    - Navigate to the search bar: Open the web page and navigate to the search bar.
    - Input search results: Enter ‘example’ into the search bar.
    - Execute Search: click on the search icon or press enter to initiate the search.
    - Observe Results: View the results displayed on the results page.
  + Expected Results:
    - The results will be displayed on the webpage.
  + Actual Result:
    - Their results are displayed on the webpage.
* **Test Case ID: 2** 
  + Requirements: Validate Query
  + Test Steps:
    - Launch the application and navigate to the search bar.
    - Enter and submit a search query.
    - The system evaluates the query.
    - The system then brings up relevant pages indexed in the database
  + Expected Results:
    - All test steps demonstrate that the system validates the search query and brings up the relevant information from the database that is then displayed to the user.
  + Actual Result:
    - The actual results produced the user’s search query back to the user.
* **Test Case ID: 3** 
  + Requirements: Display search results
  + Test Steps:
    - Launch the application
    - Enter a valid search query
    - Verify result validity
    - Verify result format
    - Verify scroll ability
    - Check page formatting
    - Check ability button interactions
  + Expected Results:
    - All test steps demonstrate that the correct search results are displayed and formatted correctly with the ability to scroll through them, while leaving other components of the page undisturbed.
  + Actual Result:
    - The actual results produced a working and correct looking page when displaying search results.
* **Test Case ID: 4** 
  + Requirements: Handle no results found
  + Test Steps:
    - Launch the application
    - Enter an invalid search query
    - Verify display no result found message
    - Check page formatting
    - Check ability of button interactions
  + Expected Results:
    - All test steps demonstrate that the no results message is displayed, and no actual results are displayed. While the rest of the page and its components remain intact and orderly.
  + Actual Result:
    - The no results found message was displayed to the user while the and other components remained intact.
* **Test Case ID: 5** 
  + Requirements: The user must be able to filter/refine their search
  + Test Steps:
    - Input search in the search bar
    - Click on the “filter button”
    - Select keyword or any other filter
    - Click apply
  + Expected Results:
    - The results will be displayed based on those filters.
  + Actual Result:
    - The results are displayed based on those filters.
* **Test Case ID: 6** 
  + Requirements: Search Result Details / Select Search Result
  + Test Steps:
    - Launch the application
    - Enter “Renewable Energy” in the search bar and press the magnifier/ or hit Enter.
    - From the list of search results, observe if several search results pop out.
  + Expected Results:
    - The application launches successfully.
    - The search results for “Renewable Energy” are displayed.
    - All the articles from the list of search results are relevant to “Renewable Energy”.
  + Actual Result: Same as Expected Results.
* **Test Case ID: 7** 
  + Requirements: Display Detailed Information
  + Test Steps:
    - Click on one of the suggested articles.
    - Go back to the tab Search Results Page
    - Click on another article.
    - Go back to the tab Search Results Page.
  + Expected Results:  
    7.2.1. It navigates to the clicked article. Return to the Search Results Page
  + Actual Result: Same as Expected Results.
* **Test Case ID: 8** 
  + Requirements: Save and Retrieve Saved Searches
  + Test Steps:
    - Navigate to search bar.
    - Enter search query.
    - Save search result to database.
    - Display previous search results to user.
    - Previous queries are saved so user can easily access them.
  + Expected Results:
    - All test steps demonstrate that the user can access all previous search queries easily.
  + Actual Result: Didn’t implement
* **Test Case ID: 9** 
  + Requirements: Re-attempt Search
  + Test Steps:
    - Iterates through the former list of results with the new list of results
    - Fail if duplicate founds and pass if all new display results are different
  + Expected Results:
    - The aforementioned tests should pass if the results displayed are unique results based on the search query parameters.
  + Actual Result:
    - A new set of search results with their respective content are displayed after the user interacts with the re-attempt Seach feature.
* **Test Case ID: 10** 
  + Requirements: User Feedback Submission
  + Test Steps:
    - A review button should become available to the user after a search is conducted.
    - When the button is interacted with it should bring a panel that allows the user to enter feedback.
    - The feedback should be saved appropriately in the backend.
  + Expected Results:
    - The tests should pass if a user feedback panel is made available and feedback is saved.
  + Actual Result:
    - After the user performs a search, a button appears to the user
    - After clicking on the panel, a feedback panel appears
    - The user is returned to the search engine while user feedback is saved

**Techniques for test generation**

* **Black Box Testing**
  + Used For: Testing FR1 through FR9
  + Description: We are using this technique to validate behaviors such as search functionality (FR1), query validation (FR2), result display (FR3), no-results handling (FR4), and search refinement (FR5). For instance, when testing FR1, testers would input various search queries to ensure the system retrieves and displays correct results without knowing how the search algorithm works internally.
  + Applicability: Suitable for validating user-interface interactions and system outputs without requiring knowledge of the underlying code, making it ideal for front-end features and integrated system tests.
* **White Box Testing**:
  + Used For: Testing FR10, focusing on feedback handling mechanisms.
  + Description: This testing technique involves detailed examination of the logical flow of the software and is used to ensure that all internal operations perform as intended. For FR10, this could include testing the feedback submission process, ensuring that the feedback is processed, categorized, and stored correctly, examining error handling and security validations during feedback submission.
  + Applicability: Essential for validating backend processes, security measures, and data handling procedures that are not visible through the UI but crucial for the applications reliability and integrity.
* **Criteria for Measuring the Quality of Tests**:
  + **Coverage**: Measures how much of the application functionality the test case exercises.
    - High: The test exercises all paths related to the requirement.
    - Medium: The test exercises the most critical paths related to the requirement.
    - Low: The test exercises only some paths related to the requirement.
  + **Effectiveness**: Measures the test's ability to identify defects.
    - High: Frequently identifies defects when they are present.
    - Medium: Sometimes identifies defects when they are present.
    - Low: Rarely identifies defects when they are present.
  + **Maintainability**: Measures how easy it is to update the test when changes are made to the application.
    - High: Test can be easily updated with minimal effort.
    - Medium: Some effort required to update the test.
    - Low: Significant effort required to update the test.
  + **Precision**: Measures the accuracy of the test in targeting specific conditions.
    - High: Accurately targets and tests the specific conditions.
    - Medium: Generally accurate but may include some irrelevant conditions.
    - Low: Poor accuracy, often testing irrelevant conditions.
  + **Readability**: Measures how easily other team members can understand the test.
    - High: Very clear and easy to understand.
    - Medium: Somewhat clear but could be improved.
    - Low: Difficult to understand without detailed explanation.

**Assessment of the goodness of your test suite**

**Black Box Testing**

Used For: Testing FR1 through FR9

Description: We are using this technique to validate behaviors such as search functionality (FR1), query validation (FR2), result display (FR3), no-results handling (FR4), and search refinement (FR5). For instance, when testing FR1, testers would input various search queries to ensure the system retrieves and displays correct results without knowing how the search algorithm works internally.

Applicability: Suitable for validating user-interface interactions and system outputs without requiring knowledge of the underlying code, making it ideal for front-end features and integrated system tests.

**White Box Testing**:

Used For: Testing FR10, focusing on feedback handling mechanisms.

Description: This testing technique involves detailed examination of the logical flow of the software and is used to ensure that all internal operations perform as intended. For FR10, this could include testing the feedback submission process, ensuring that the feedback is processed, categorized, and stored correctly, examining error handling and security validations during feedback submission.

Applicability: Essential for validating backend processes, security measures, and data handling procedures that are not visible through the UI but crucial for the applications reliability and integrity.

**Criteria for Measuring the Quality of Tests**:

**Coverage**: Measures how much of the application functionality the test case exercises.

* + - High: The test exercises all paths related to the requirement.
    - Medium: The test exercises the most critical paths related to the requirement.
    - Low: The test exercises only some paths related to the requirement.
  + **Effectiveness**: Measures the test's ability to identify defects.
    - High: Frequently identifies defects when they are present.
    - Medium: Sometimes identifies defects when they are present.
    - Low: Rarely identifies defects when they are present.
  + **Maintainability**: Measures how easy it is to update the test when changes are made to the application.
    - High: Test can be easily updated with minimal effort.
    - Medium: Some effort required to update the test.
    - Low: Significant effort required to update the test.
  + **Precision**: Measures the accuracy of the test in targeting specific conditions.
    - High: Accurately targets and tests the specific conditions.
    - Medium: Generally accurate but may include some irrelevant conditions.
    - Low: Poor accuracy, often testing irrelevant conditions.
  + **Readability**: Measures how easily other team members can understand the test.
    - High: Very clear and easy to understand.
    - Medium: Somewhat clear but could be improved.
    - Low: Difficult to understand without detailed explanation.

**Traceability Of Test Cases To Use Cases**

Table 5: Test Cases to Use Cases Traceability Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Requirement ID | Test Description | Expected Outcome |
| TC1 | **FR1:**  **Perform Search/Enter Search Query** | Ensure the user can enter a query in the search bar and receive the correct results. | The correct search results are displayed on the webpage. |
| TC3 | **FR2:**  **Validate Query** | Ensure the user’s search query is valid. | Valid search queries allow the system to search for what the user has inputted. |
| TC3 | **FR3:**  **Display Search Results** | Test the display of search results after a submitted valid query. | User can view and scroll through formatted result information with the rest of the website UI remaining intact. |
| TC4 | **FR4:**  **Handle No Results Found** | Test the display of an error message after the submission of an invalid query. | Error message is displayed to user in place of search results with the rest of the website UI remaining intact. |
| TC5 | **FR5:**  **Refine Search Results / Apply Filter** | Test the functionality of the filter options by applying various filters (like keywords) after an initial search to see if the search results are appropriately refined based on the filters. | Only results that match the specified filters are displayed, correctly refining the search. |
| TC6 | **FR6:**  **View Search Result Details / Select Search Result** | Test that a user can successfully view the search result after entering a search query. | User successfully sees the results under the search bar. |
| TC7 | **FR7:**  **Display Detailed Information** | Test that a user can successfully see the whole article after clicking on one of the results. | User is successfully navigated to the clicked article. |
| TC8 | **FR8:**  **Save and Retrieve Saved Searches** | Ensure the user’s previous search queries are saved and able to be retrieved for the user. | User can access previous search queries without hindrance. |
| TC9 | **FR9:**  **Reattempt Search for Updated Results** | Ensure That a user can re-attempt a search and be provided a unique set of search results | User is provided a unique set of search results after interacting with the re-attempt search feature |
| TC10 | **FR10:**  **User Feedback Submission** | Ensure that the user is able to view a user submission feedback panel and enter feedback | User is provided a panel to input feedback which is adequately saved |

# Evidence the Document Has Been Placed under Configuration Management

* Name of the CM tool: GitHub
* Version number of before: [ebbac3c](https://github.com/Bakr8724/CapstoneSearchTool/commit/ebbac3c028e9bd17569250e13f81ab3027c1aeb8)
* Version number after:
* Difference between the two: Added additional info required, for each section, added tables, and finalized document.
* Review of each change:
  + Before: Added outline, structured document, and formatting, along with standards and references.
  + After: Added info for all sections, outlined and formatted the document. Revised some errors, etc.
* Other info:

# Engineering Standards and Multiple Constraints

# IEEE Std 830-1998: Software Requirements Specification.

* ISO/IEC/IEEE Std 29148-2018: Systems and Software Engineering
  + Life Cycle Processes
  + Requirement Engineering
* 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
* ISO/IEC 27001:2022- Information security, cybersecurity and privacy protection
* ISO 9241-210:2019- Ergonomics of human-system interaction – Part210: Human-centred design for interactive systems.
* IEEE Std 1016-1998-(Revision-2009): Software Design
* ISO/IEC 12207:2017: Systems and software engineering – Software life cycle processes.
* ISO/IEC 15288:2023: Systems and software engineering – System life cycle process.
* IEEE Std 1471-2000: Software Architecture
* ISO/IEC/IEEE Std 42030:2019: Software, Systems and Enterprise
  + — Architecture Evaluation Framework
* ISO/IEC 25010:2024: Systems and Software Engineering
  + — Systems & software Quality Requirements & Evaluation
  + — System & software quality models
* W3C Guidelines
* Document Version Control:
  + Method: Use of a version control system (like Github) for the PMP document.
  + Purpose: To track changes, revisions, and updates to the document over time.
  + Rationale: Ensures that there is always a clear, up-to-date version of the PMP available to all team members and stakeholders.
* Change Management Procedure:
  + Steps:
    - Request for changes submitted by team members.
    - Review of change requests in team meetings.
    - Approval or rejection of changes by the Project Manager or the entire team.
    - Documentation of approved changes in the PMP.
  + Rationale: Provides a structured process for modifying the PMP, ensuring that changes are considered and agreed upon by the team.
* Access Control:
  + Method: Restricting editing access to authorized team members onl.y
  + Purpose: To prevent unauthorized changes or accidental modifications.
  + Rationale: Maintains the integrity and consistency of the PMP.
* Section Rationale:
  + Configuration management is critical for maintaing the integrity and utility of the PMP. It provides a clear history of changes and ensures that all team membersa re working from the most current version of the plan. This process is essential for effective project management, particularly in a team environment where multiple individuals are collaborating on the same document.
* IEEE Std 1058-1998: Software project Management Plans
* PMBOK Guide: Project Management Body of Knowledge
* ISO 10006:2017 – Quality management in projects
* IEEE Std 15939: Measurement Process

# **Additional references**

# Lamsweerde, A.V, 2009, Requirements Engineering: From System Goals to UML Models to Software Specification

* Requirements Engineering - A Good Practice Guide, I. Sommerville and P. Sawyer, Wiley
* Non-Functional Requirements in Software Engineering, L. Chung, B. Nixon, E. Yu and J. Mylopoulos, Kluwer Academic Publishing, 2000
* Jorgensen, P.C., 2013. Software Testing: A Craftsman's Approach. Auerbach Publications
* Mathur, A.P., 2013. Foundations of Software Testing, 2/e. Pearson Education
* Beizer, B., 2003. Black-Box Testing: Techniques for Functional Testing of Software and Systems. Wiley.
* Fewster, M., and Graham, D., 1999. Software Test Automation: Effective Use of Test Execution Tools. Addison-Wesley Professional.
* Copeland, L., 2004. A Practitioner’s Guide to Software Test Design. Artech House.
* Myers, G.J., Sandler, C., and Badgett, T., 2011. The Art of Software Testing 3rd Edition.
* Boehm, B., 1988: A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes.
* Cockburn, A., 2000: Writing Effective Use Cases. Addison-Wesley.
* Sommerville, l., 2015: Software Engineering. 10th Edition, Addison-Wesley.
* Martin, R.C., 2003: Agile Software Deveolpment: Principles, Patterns, and Practices. PRentice Hall
* Lattanze, A.J., 2008. Architecting Software Intensive Systems: A Practitioner’s Guide. CRC Press
* Bass, L., Clements, P. and Kazman, R., 2003. Software Architecture in Practice. Addison-Wesley
* Perforce, “A GUIDE TO TRACEABILITY IN PRODUCT DEVELOPMENT Requirements Traceability Matrix — Everything You Need to Know,” <https://www.perforce.com/resources/alm/requirements-traceability-matrix> (accessed Mar. 15, 2024)
* IEEE Computer Society, “1471-2000- IEEE Recommended Practice for Architectural Description of Software-Intensive Systems,” <https://standards.ieee.org/ieee/1471/2187/>
* [The\_Mythical\_Man-Month](https://en.wikipedia.org/wiki/The_Mythical_Man-Month)
* PMBOK Guide: Project Management Body of Knowledge
* Larson, E., and Gray, C., 2014. Project Management: The Managerial Process
* 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.

# Acknowledgment

The Internet Research Assistant project and all its inclusive deliverables including the final product implementation could not have been possible without the following mentioned individuals. A thank you to Dr. Wong and the course TA’s for providing structured project organization and constructively critiquing deliverables. As well as a special thanks to the invaluable and consistent guidance provided by the project sponsor Jeff. Lastly, but certainly not least project lead Bakr who handled all communications with stakeholders and led the project successfully as well as all the other project team members (Vi, Chloe, Ikraam, Mohammad) without whom the project would not have been a success. Thank you.