# SE 4485: Software Engineering Projects

Spring 2024

**Test Plan**

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

# ABSTRACT

This document serves to outline the various testing mechanisms which can be used to adequately test the functional requirement detailed in previous documents. It does so by providing the means and methods by which tests can be conducted, whether they are unit tests or tests of other forms. It furthermore outlines the expected and actual results of tests conducted on the implementation in a specified step by step processes. Differing methods of unit tests from white box to black box testing are presented as well as a range of metrics by which testing as a whole is measured in terms of coverage, maintainability, readability, etc. All of the aforementioned culminates in a test case to use case traceability matric.

# TABLE OF CONTENTS

[SE 4485: Software Engineering Projects 1](#_Toc509011016)

[ABSTRACT 1](#_Toc1436037811)

[TABLE OF CONTENTS 2](#_Toc1570695445)

[LIST OF FIGURES 2](#_Toc1067059751)

[LIST OF TABLES 2](#_Toc1957716521)

[INTRODUCTION 2](#_Toc2082290072)

[REQUIREMENTS/SPECIFICATIONS-BASED SYSTEM LEVEL TEST CASES 2](#_Toc179664982)

[TECHNIQUES FOR TEST GENERATION 4](#_Toc1838140534)

[TRACEABILITY OF TEST CASES TO USE CASES 5](#_Toc291395793)

[EVIDENCE THE TEST PLAN HAS BEEN PLACED UNDER CONFIGURATION MANAGEMENT 6](#_Toc1089294476)

[ENGINEERING STANDARDS AND MULTIPLE CONSTRAINTS 6](#_Toc1309545021)

[ADDITIONAL REFERENCES 6](#_Toc1095192356)

# LIST OF FIGURES

There are no figures in this document.

# LIST OF TABLES

Table I - Test Cases to Use Cases Traceability Matrix ........................................................................... 5-6

# INTRODUCTION

This document discusses the test cases, test methods, and traceability between the test cases and requirements that are used to test the implemented features of this internet research tool. The scope of these tests covers the functional requirements previously defined in the architecture document. This document has three core sections that discuss the testing plan: Requirements-Based System Level Test Cases, Techniques for Generation, and a Test Cases to Use Cases Traceability Matrix. The requirements-based system level test cases break down the test case into first defining the use case requirement, then detailing the test steps, and ultimately the expected/actual results of the testing. Moreover, the Techniques for Generation includes a range from black to white box testing and their respective quality criteria metrics. Furthermore, the traceability matrix links use cases to tests with matching ID’s. Alongside these main sections are various standardized inclusions such as an abstract, references, and engineering standards.

# REQUIREMENTS/SPECIFICATIONS-BASED SYSTEM LEVEL TEST CASES

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

# TRACEABILITY OF TEST CASES TO USE CASES

TABLE I

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 is able to re-attempt a seach 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 TEST PLAN HAS BEEN PLACED UNDER CONFIGURATION MANAGEMENT

1. Name of the CM tool: GitHub
2. Version number of before: [8becd0e](https://github.com/Bakr8724/CapstoneSearchTool/commit/8becd0edde6457499d960a3c32cf9083700f294a)
3. Version number after:
4. Difference between the two: Traceability matrix, additional standards and references, test cases, and techniques for test generation
5. Review of each change:
   1. Before: Added standards and references, added cover page, fixed document outline.
   2. After: Added additional standards & references, tables and test cases, outline formatting, and test generation techniques.
6. Other info:

# ENGINEERING STANDARDS AND MULTIPLE CONSTRAINTS

* 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
* IEEE Std 830-1998: Recommended Practice for Software Requirements Specification
* ISO/IEC 25010:2023: Systems and software engineering SQuaRE
* 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.

# ADDITIONAL REFERENCES

* 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.