SE 4485: Software Engineering Projects

Spring 2025

Test Plan

Group Number	11
Project Title	Develop an Ontology-Generation LLM Application
Sponsoring Company	The Fellows Consulting Group (FCG)
Sponsor(s)	Tom Hill
Students	1. Alberto Escobar 2. Dara Moheimani 3. Lee Rafael Filomeno 4. Brandon Bailey 5. Blanca Berrios Henriquez 6. Saishrey Bhandare

ABSTRACT

The test plan document outlines the testing approach used for the ONtology Generation LLM application. The document details test cases derived from functional and non-functional requirements, specifies test generation techniques using both black-box and white-box approaches, and it provides traceability between test cases and use cases. The testing approach is encompassing of all components the WordPress interface, input validation, LLM integration with Ollama, ontology generation, and visualization capabilities. The test plan will serve as a framework to ensure that all requirements are met.

TABLE OF CONTENTS

LIST OF TABLES

INTRODUCTION

This document specifies the test plan for the Ontology-Generation LLM application that leverages advanced LLMs to generate domain ontologies. The system is being developed in two different parts: the initial local Windows implementation and the eventual deployment to a public web server. The system utilizes Ollama for local LLM implementation with the DeepSeek R1 model. The core components being tested include the domain keyword processing system, the ontology generation engine, and the web-based user interface.

The document is organized in a way to provide a comprehensive understanding of the testing approach through a few key sections. The Requirements/Specification's-Based System Level Test Cases section details specific test cases derived directly from system requirements. The Techniques for Test Generation section explains the testing methodologies that were used. The document also includes a traceability matrix mapping test cases to use cases, engineering standards that were followed, and any sources that were referenced when developing the test plan.

REQUIREMENTS/SPECIFICATIONS-BASED SYSTEM LEVEL TEST CASES

Test Case 1: Homepage Display
Description: Verifies that the homepage loads correctly and displays all required elements
Preconditions: WordPress site is operational
Steps and Expected Results:

1. Access WordPress site URL
2. Verify navigation menu is displayed
3. Verify the ontology interface link is accessible
4. Verify project description displayed
5. Measure the page load time

Homepage displays correctly with all elements, navigation menu includes interface access, page loads within 3 seconds (per NFR)

Test Case 2: Domain Keyword Input
Description: Verifies that the user can access and interact with the domain keyword input form
Preconditions: Interface is accessible
Steps and Expected Results:

1. **Verify the domain input field is present**
2. **Enter a test domain**
3. **Submit the form**

Interface is accessible, input field accepts text entry, form submitted correctly, UI interactions complete within 100 ms (per NFR).

Test Case 3: Input Validation - Valid Input
Description: Verify that valid input is correctly processed.
Preconditions: Ontology generation interface is accessible.
Steps and Expected Results:

1. Enter a valid domain keyword (e.g., "healthcare", within 1-50 characters).
2. Submit the form.
3. Observe system response.

System accepts the input, no validation errors are displayed, processing begins for ontology generation, and validation completes within 500ms (per NFR).

Test Case 4: Input Validation - Invalid Input
Description: Verify that invalid input is correctly rejected.
Preconditions: Ontology generation interface is accessible Steps and Expected Results:

1. Submit empty input.

2. Submit input exceeding 50 characters.
3. Submit input with invalid characters.
4. Verify system stability during invalid input tests.

System displays appropriate validation error messages for each invalid input type, form is not submitted for processing, validation completes within 500ms, system doesn't crash during validation (99.9% reliability per NFR).

Test Case 5: Ontology Generation
Description: Verify that the system correctly generates an ontology from a valid domain.
Preconditions: Valid domain keyword submitted, Ollama service running with DeepSeek R1 model.
Steps and Expected Results:

1. Submit a valid domain keyword (e.g., "pets").
2. Wait for ontology generation to complete.
3. Verify the response structure.

Processing indicator displays during generation, ontology generation completes within 15 seconds, responses contain valid JSON with domain and relationships, relationships include proper cardinality and category information, at least 10 meaningful relationships are generated.

Test Case 6: Relationship Connectivity Validation
Description: Verify that the generated ontology maintains connectivity between all entities.
Preconditions: System has generated an ontology with multiple entities.
Steps and Expected Results:

1. Generate an ontology for a complex domain.
2. Inspect the relationship graph structure.
3. Test with intentionally disconnected entities.

All entities in the ontology are connected, no isolated entities or subgraphs exist, the system detects disconnected components and adds appropriate connecting relationships.

Test Case 7: Ontology Visualization - List View
Description: Verify that the generated ontology is correctly displayed in list view.
Preconditions: Ontology has been successfully generated.
Steps and Expected Results:

1. Wait for list view to render.
2. Verify relationship presentation.
3. Verify list view performance.

List view displays by default after generation with all relationships properly formatted, each relationship shows source entity, relationship type, target entity, relationship categories are color-coded correctly, cardinality information is displayed, list renders within expected timeframe, and UI interactions complete within 100ms.

Test Case 8: Ontology Visualization - Graph View
Description: Verify that the generated ontology is correctly visualized as a graph
Preconditions: Ontology has been successfully generated
Steps and Expected Results:

1. Click the "Visual View" button.
2. Wait for Mermaid diagram to render.
3. Verify visualization elements.
4. Test interactive functionality.

System switches from list view to graph view, Mermaid diagram renders correctly within 2 seconds, all entities and relationships appear with proper styling and labeling, zoom and pan controls function correctly.

Test Case 9: View Toggle Functionality
Description: Verify that users can toggle between list and visual views.
Preconditions: Ontology has been successfully generated.
Steps and Expected Results:

1. Verify default view.
2. Click the "Visual View" button.
3. Click the "List View" button.

Verify performance of view toggling Results: List view is displayed by default, clicking Visual View button displays graph and hides list view, clicking List View button displays list and hides graph view, UI transitions occur within 100ms.

Test Case 10: Error Handling - Ollama Connection Failure
Description: Verify appropriate error handling when Ollama service is unavailable.
Preconditions: Ollama service stopped or is unreachable.
Steps and Expected Results:

1. Enter a valid domain keyword.
2. Submit the form.
3. Observe error handling.

System displays an appropriate error message about Ollama connection, error message is clear and suggests troubleshooting steps, error details are logged for technical review, system remains stable and doesn't crash (99.9% reliability per Non-functional requirements).

Test Case 11: Error Handling - LLM Processing Failure
Description: Verify error handling when LLM processing fails.
Preconditions: Ollama service is running but configured to return an error.
Steps and Expected Results:

1. Enter a valid domain keyword.
2. Submit the form.
3. Observe error handling.
4. Test retry functionality.

System displays appropriate error message about LLM processing failure, error message provides clear information about the issue, error details are logged, system provides retry option (up to 3 times as specified), system remains stable after multiple retries.

Test Case 12: Performance Testing - Response Time
Description: Verify the system meets all performance requirements.
Preconditions: System is operational under normal load.
Steps and Expected Results:

1. Measure input validation response time.
2. Measure ontology generation time.
3. Measure visualization rendering time.
4. Measure UI interaction response time.
5. Measure homepage load time.

Results: Input validation completes within 500ms, ontology generation completes within 15 seconds, graph visualization renders within 2 seconds, UI interactions complete within 100ms, homepage loads within 3 seconds (all per Non-functional requirements).

Test Case 13: Browser Compatibility
Description: Verify the application functions correctly in Chrome browser.
Preconditions: Chrome browser is installed.
Steps and Expected Results:

1. Access the application in Chrome.
2. Complete an end-to-end workflow.
3. Test interactive features.
4. Verify performance in Chrome.

Application loads correctly with all interface elements displaying as designed, complete workflow functions properly from input to visualization, interactive features (view toggling, zoom, pan) work correctly, all performance metrics meet requirements.

Test Case 14: Usability Testing
Description: Verify that the system meets usability requirements.
Preconditions: Application is fully functional.
Steps and Expected Results:

1. Recruit 5 test users unfamiliar with the system.
2. Provide minimal instructions.
3. Ask users to perform basic operations.
4. Measure learning time.
5. Collect usability feedback.

Users represent target audience, users can complete all basic tasks successfully, users learn basic operations within 5 minutes (per Non-functional requirements), interface follows Stanford NLP site / UTD patterns as specified, users report intuitive experience.

TECHNIQUES FOR TEST GENERATION

The sections outlines testing methodologies that were used to validate the application. Both black-box and white-box approaches were used to ensure comprehensive coverage of the system.

Black-Box Testing Techniques

The following black box techniques will be used:

1. **Equivalence Partitioning**: Input domains are divided into valid and invalid classes:
   - Domain keywords: Valid inputs (1 – 50 char with valid chars) vs. Invalid inputs (empty, exceeding char limit, containing special characters).
   - Relationship validation: Valid objects (containing all required fields vs invalid objects (missing required fields).
2. **Boundary Value Analysis:** Testing boundary conditions for input validation:
   - Domain keywords with 1 character.
   - Domain keywords with exactly 50 characters.
   - Domain keywords with 51 characters.
3. **Decision Table Testing**: Addressing combinations of conditions:
   - Error handling scenarios (Ollama unavailable, LLM processing failure).
   - Ontology connectivity validation with varying ontology structures.
4. **Use Case Testing:** Mapping functional requirements to test scenarios:
   - Use cases are taken from requirements document and converted into test scenarios.
   - Both normal and exception flows will be tested.

White-Box Testing Techniques

The following white-box techniques will be used:

1. Statement Coverage: Ensure all statements in the code are executed at least once:
   - PHP code in the plugin.
   - Python code in the generator.
2. Branch Coverage: Testing all decision points:

- Input validation.
- Error handling paths.
- JSON processing logic.

3. Path coverage: for critical functions
   - Validate_relationship ( ).
   - Extract_json_from_text( ).
   - Validate_ontology_connectivity( ).
4. API integration Testing
   - Integration with Ollama API.
   - WordPress AJAX.
   - Mermaid Diagram rendering.

Test Quality Measurement Criteria

The quality of the tests will be measured using the following criteria:

1. **Requirements Coverage:** the percentage of requirements covered by Test Cases (100% of functional and non-functional requirements will be covered).
2. **Code Coverage:** statement coverage, branch coverage, condition coverage.
3. **Performance Criteria Validation:** response time measurements and reliability/uptime measurements.
4. **Usability Testing Metrics:** time for new users to learn basic operations, and task completion success rate.

TRACEABILITY OF TEST CASES TO USE CASES

Test Case Number	Use Case 1: Display Homepage	Use Case 2: Input Domain Keyword	Use Case 3: Validate Input	Use Case 4: Generate Ontology	Use Case 5: Display Ontology Graph	Use case 6: Display Error Message
TC 1	✓					
TC 2		✓				
TC 3			✓			
TC 4			✓			✓
TC 5				✓		
TC 6				✓		
TC 7					✓	
TC 8					✓	
TC 9					✓	
TC 10				✓		✓
TC 11				✓		✓
TC 12	✓	✓	✓	✓	✓	
TC 13	✓	✓	✓	✓	✓	✓
TC 14	✓	✓	✓	✓	✓	✓

Table 1: Traceability Matrix

Requirements Coverage Analysis
Each requirement from the Requirements Documentations is addressed by the test cases. The section below includes both the functional and non-functional requirements, and the test cases that ensure the requirements are met.

Functional Requirements (Use Cases)
Use Case 1: Display Homepage (TC 1, 12, 13, 14)
Use Case 2: Input Domain **Keyword** (TC 2, 12, 13, 14)
Use Case 3: Validate Input (TC 3, 4, 12, 13)
Use Case 4: Generate Ontology (TC 5, 6, 10, 11, 12, 13)
Use Case 5: Display Ontology Graph (TC 7, 8, 9, 12, 13)
Use Case 6: Display Error Message (TC 4, 10, 11, 13)

Non-Functional Requirements
Input validation within 500 ms (TC 3, 4, 12)
LLM processing within 15 seconds (TC 5, 12)
UI interactions within 100ms (TC 2, 7, 9, 12)
Graph visualization within 2 seconds (TC 8, 12)
Homepage loads within 3 seconds (TC 1, 12)
99.9% uptime/stability (TC 4, 10, 11)
Learning within 5 minutes (TC 14)

EVIDENCE THE TEST PLAN HAS BEEN PLACED UNDER CONFIGURATION MANAGEMENT

Version	Author	Change
1.0	Alberto Escobar	Initial version of Test Plan
1.1	Alberto Escobar	Made changes to traceability matrix
1.2	Alberto Escobar	Final Formatting changes

Table 2: CM Tool Evidence

ENGINEERING STANDARDS AND MULTIPLE CONSTRAINTS
- IEEE Std 829-1983: Software Testing [pdf]
- ISO/IEC/IEEE Std 29119-1-(Revision-2022): Part 1 - Software Testing General Concepts [pdf]
- ISO/IEC/IEEE Std 29119-2-(Revision-2021): Part 2 - Test Process [pdf]
- ISO/IEC/IEEE Std 29119-3-(Revision-2021): Part 3 - Test Documentation [pdf]
- ISO/IEC/IEEE Std 29119-4-(Revision-2021): Part 4 - Test Techniques [pdf]
- Additional standards suggested by the sponsor(s)

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