SE 4485: Software Engineering Projects

Spring 2025

Detailed Design Documentation

|  |  |
| --- | --- |
| 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

This document provides a detailed outline of the Ontology Generation system, which uses WordPress, PHP, and Python to enable users to input a domain and receive an ontology generated by Llama 3.2. The design elaborates on the architectural components that were previously outlined. Specific classes, methods, UI elements, and their interactions are all covered in detail within the document. The document will serve as an implementation blueprint, and will ensure all requirements, functional and non-functional, are met by the design.

TABLE OF CONTENTS

1. Introduction………………………………………………………………………………………….
2. GUI (Graphical User Interface) Design……………………………………………………………...
3. Static Model…………………………………………………………………………………………
4. Dynamic Model……………………………………………………………………………………...
5. Rationale for Design Model………………………………………………………………………….
6. Traceability from Requirements……………………………………………………………………..
7. Configuration Management Evidence……………………………………………………………….

LIST OF FIGURES

1. Figure 1: Domain Input Form………………………………………………………………………..
2. Figure 2: Loading Indicator appears after Form submission…………………………………………
3. Figure 3: Results being displayed……………………………………………………………………
4. Figure 4: Admin Page Content………………………………………………………………………
5. Figure 5: Class Diagram…………………………………………………………………….
6. Figure 6: Zoomed out view of BMPN………………………………………………………………..
7. Figure 7: Left side of BPMN………………………………………………………………..
8. Figure 8: Right side of BPMN………………………………………………………………

LIST OF TABLES

1. Table 1: Functional Requirements Traceability……………………………………………………...
2. Table 2: Performance Requirements Traceability…………………………………………………...
3. Table 3: Reliability and Usability Requirements Traceability……………………………………….
4. Table 4: CM Tool Evidence………………………………………………………………………….

INTRODUCTION

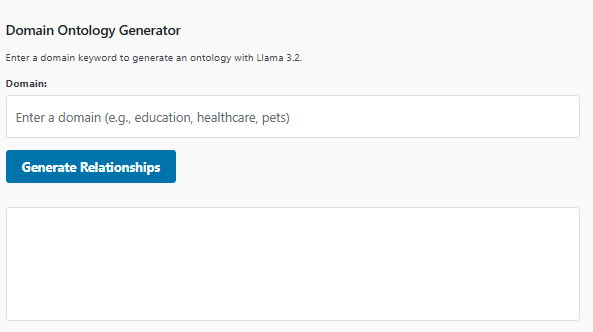
This detailed design document helps bridge the architectural model described before, and the actual implementation of the ontology generating LLM. The document defines specific components, classes, and interfaces required to implement the system across all layers: WordPress Frontend, PHP Backend, Python Processing, and Ollama API interaction. The document is organized into 5 main sections: GUI Design, Static Models (Class diagrams), rationale for design choices, and traceability from requirements to detailed design.

**GUI (Graphical User Interface) Design**

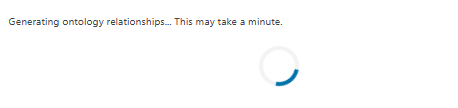
**Main Interface**

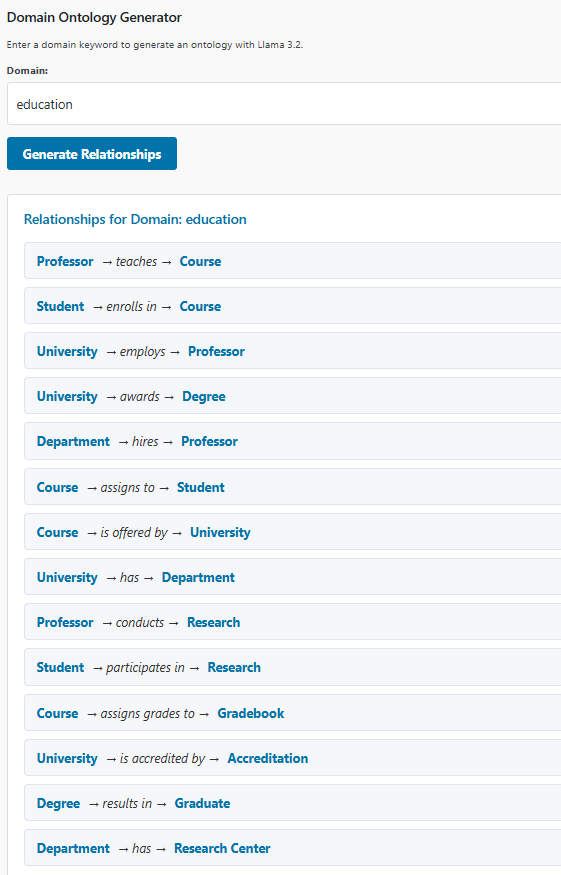
The main interface is implemented through a WordPress short code that renders a form for domain input and displays the resulting ontology. The interface consists of three primary components:

1. Domain Input Form
2. Loading Indicator
3. Results Display



**Figure 1: Domain Input Form**



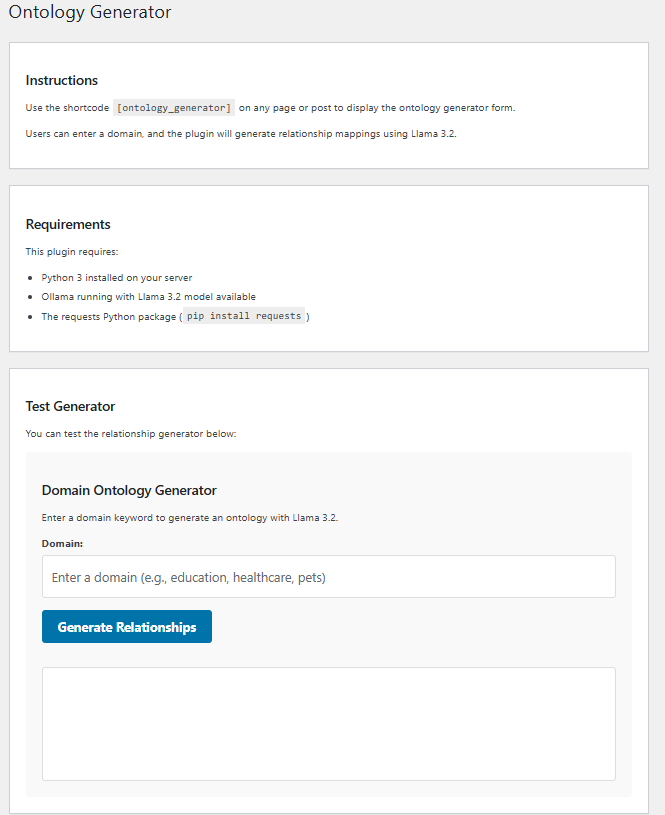
**Figure 2: Loading Indicator appears after Form submission**  
  


**Figure 3: Results being displayed**

**Admin Configuration Interface**

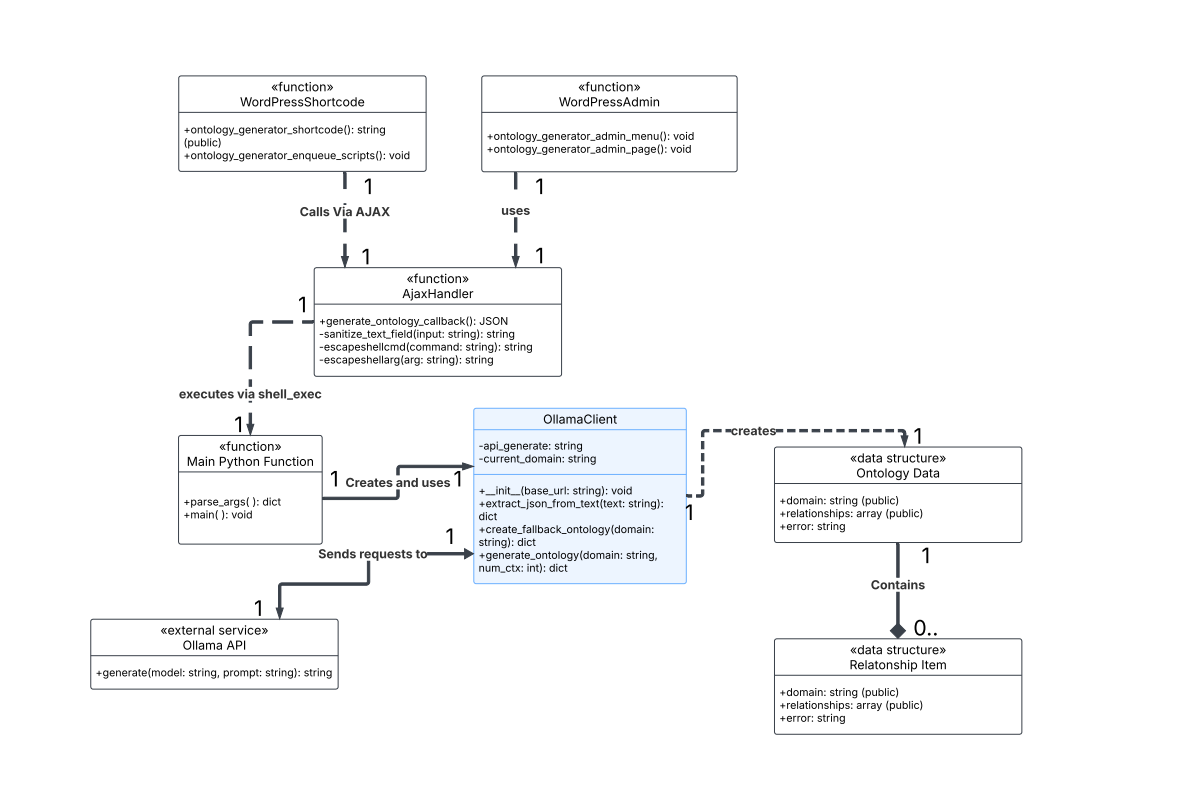
The admin interface provides both system configuration and documentation. It consists of:

1. Instructions Panel
2. Requirements Panel
3. Test Generator Panel



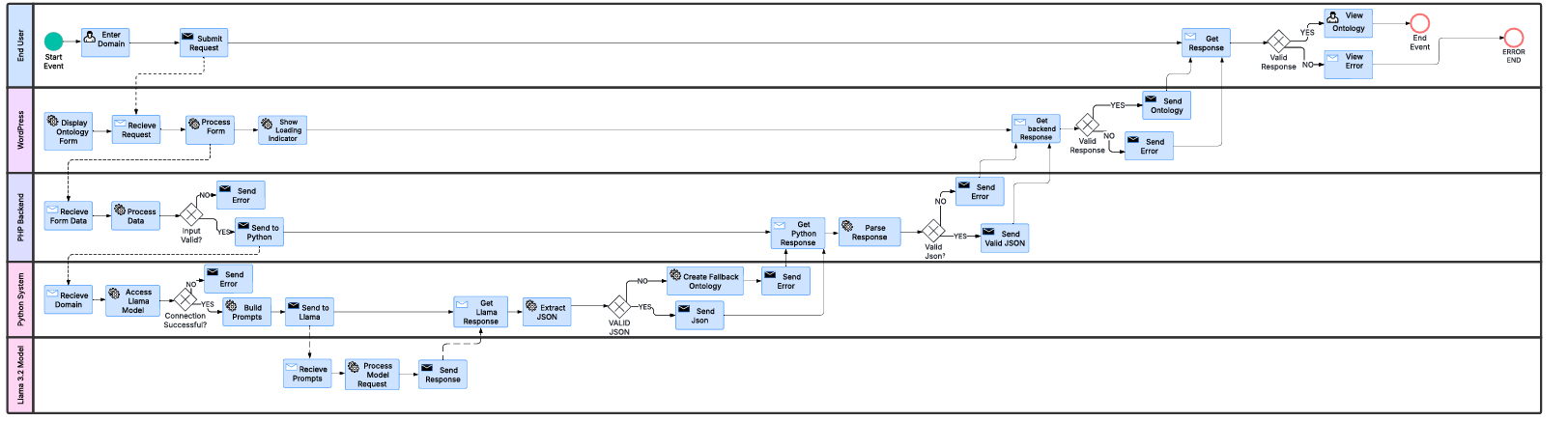
**Figure 4: Admin Page Content**

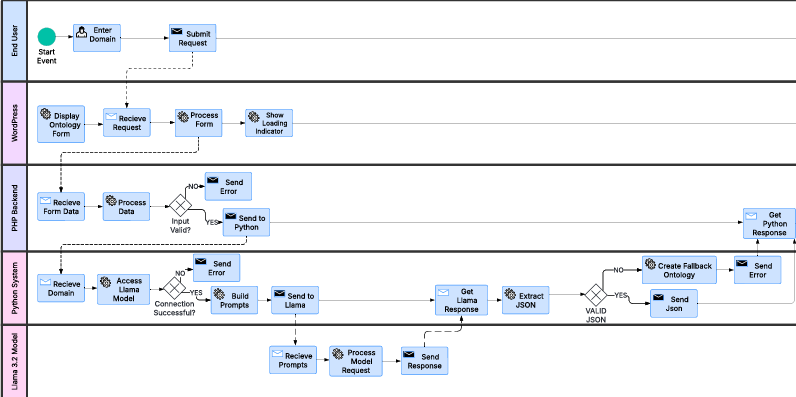
STATIC MODEL

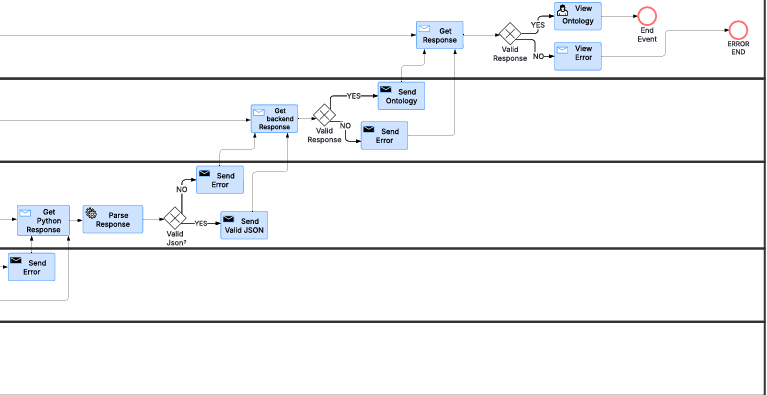
**Figure 5: Class Diagram**

DYNAMIC MODEL

The system's dynamic behavior is comprehensively represented through a BPMN diagram that illustrates the end-to-end flow across all system layers. The model captures both the happy path and error handling scenarios in a unified way.

**Figure 6: Zoomed out view of BMPN**

**Figure 7: Left side of BPMN**

**Figure 8: Right side of BPMN**

RATIONALE FOR YOUR DETAILED DESIGN MODEL

The set of key principles that were followed when designing the system are described below. The decisions were made to balance the performance, reliability, and maintainability of the system. The layered organization of classes follows the layered architectural approach described before, and to help create a clear separation of concerns throughout the system. The WordPress front-end layer focuses entirely on user interaction and display logic, implementing a responsive interface that can handle user input and is able to visualize relationships. The separation of concerns also ensures that UI components remain lightweight and responsive, allowing them to meet the established performance targets.

The PHP backend layer helps with communication between the user interface and processing logic. By implementing comprehensive input sanitization through WordPress's sanitize\_text\_field() function and command escaping via escapeshellcmd() and escapeshellarg(), the design addresses potential security vulnerabilities at this critical boundary. The use of WordPress nonce checks further strengthens security by preventing cross-site request forgery attacks. These security measures are implemented without compromising the performance targets specified in the requirements.

At the core of the system, the Python processing layer implements the domain specific logic for ontology generation. The OllamaClient class encapsulates all interaction with the Llama model. The extract\_json,\_from\_text() method is incredibly robust due to the unpredictable nature of LLMS; implementing multiple parsing strategies helps maximize the likelihood of extracting valid JSON, while still being able to gracefully handle errors.

Comprehensive error handling throughout the system was imperative. Each component implements the appropriate error detection and recovery mechanisms tailored to its specific role. In the WordPress layer, AJAX error callbacks provide immediate feedback to users. The PHP layer implements structured error responses with descriptive messages. The Python layer employs comprehensive try-except blocks to catch and categorize various failure modes, from API connection issues to JSON parsing problems. This multi-layered approach to error handling ensures that problems are caught and addressed at the appropriate level of the system. The overall design prioritizes simplicity while still meeting all functional and non-functional requirements. Rather than implementing complex hierarchies or abstractions, the system uses straightforward functions, and classes with responsibilities. This approach makes the code much more accessible and maintainable over time.

TRACEABILITY FROM REQUIREMENTS TO DETAILED DESIGN MODEL

**Functional Requirements Traceability**

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Design Components** | **Implementation Details** |
| Display Homepage | ontology\_generator\_shortcode() function | WordPress shortcode that renders the form interface on the site with UTD styling patterns |
| Input Domain Keyword | Form HTML in ontology\_generator\_shortcode() | Text input field with id="domain" and appropriate placeholder text |
| Validate Input | jQuery validation, sanitize\_text\_field() function | Client-side validation with jQuery on form submission; Server-side sanitization in generate\_ontology\_callback() |
| Generate Ontolgogy | OllamaClient.generate\_ontology() method | Python method that builds system and user prompts for Llama 3.2 and processes the response |
| Display Ontology Graph | displayRelationships() jQuery function | JavaScript function that renders styled relationship items using jQuery DOM manipulation |
| Display Error Message | Error handling in all layers | CSS error classes, PHP error JSON responses, Python exception handling, jQuery error display functions |

**Table 1: Functional Requirements Traceability**

**Performance Requirements Traceability**

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Design Components** | **Implementation Details** |
| Input validation < 500ms | jQuery form validation | Client-side validation is lightweight and operated independently of backend |
| LLM processing < 15s | OllamaClient.generate\_ontology() | Optimized prompts and context window management with timeout settings |
| UI interactions < 100ms | jQuery event handlers | Asynchronous jQuery event handling with immediate UI feedback |
| Graph rendering < 2s | displayRelationships() function | Optimized HTML/CSS rendering and efficient DOM generation |
| Homepage load < 3s | ontology\_generator\_shortcode() | Lightweight template design with minimal initial HTML |

**Table 2: Performance Requirements Traceability**

**Reliability and Usability Requirements Traceability**

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Design Components** | **Implementation Details** |
| 99.9% uptime | Multi-layer error handling, create\_fallback\_ontology() | Multi-layer error handling, fallback ontology generation, and input validation |
| Learn in < 5 minutes | Form interface in ontology\_generator\_shortcode() | Intuitive interface with clear labeling and simple single-field design |

**Table 3: Reliability and Usability Requirements Traceability**

EVIDENCE THE DESIGN MODEL HAS BEEN PLACED UNDER CONFIGURATION MANAGEMENT

|  |  |  |
| --- | --- | --- |
| **Version** | **Author** | **Change** |
| 1.0 | Alberto Escobar | Initial version of Design Documentation |
| 1.1 | Alberto Escobar | Added class diagrams and BPMN as instructed by sponsor |
| 1.2 | Alberto Escobar | Completed traceability matrix and rationale |

**Table 4: CM Tool Evidence**

ENGINEERING STANDARDS AND MULTIPLE CONSTRAINTS

* + IEEE Std 1016-1998-(Revision-2009): Software Design [[pdf](https://course.techconf.org/se4485/IEEE/IEEE-Std-1016-1998-(Revision-2009)-Software-Design.pdf)]

ADDITIONAL REFERENCES

* + Larman, C., 2012. *Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development.* Pearson Education
  + Hyman, B., 1998. *Fundamentals of Engineering Design.* New Jersey: Prentice Hall
  + Simon, H.A., 2014. *A Student's Introduction to Engineering Design: Pergamon Unified Engineering Series* (Vol. 21). Elsevier