Documentation

System Design

Requirement Analysis:

Perform an analysis of typical user queries and interactions across a range of blogs to gather diverse requirements.

Design user interaction flows that guide users through their queries using a series of contextually relevant questions, enhanced by a logical chain of thought.

Architecture Design:

Build a scalable and efficient system capable of real-time data retrieval, processing, and dynamic response generation.

Components:

Data Retrieval: Utilize WordPress APIs to fetch real-time content updates.

Embedding Generator: Convert textual content into vector embeddings using models like Sentence-BERT.

Vector Database: Employ a system like Faiss to store and retrieve embeddings efficiently.

RAG Processor: Integrate RAG to generate responses based on retrieved information.

Chain of Thought Module: Develop this module to enhance the RAG outputs with logical progression and context continuity.

User Interface: Design an interactive chat interface that can dynamically display the chatbot's thought process.

Implementation

WordPress Data Retrieval and Embedding Generation:

Real-Time Data Fetching: Implement hooks and REST API calls within WordPress to fetch new and updated content.

Pseudocode for Embedding Update:

```
function update_embeddings_on_new_post(post):
text = extract_text(post)
embeddings = generate_embeddings(text)
update_vector_database(post.id, embeddings)
```

RAG Setup and Chain of Thought Integration:

RAG Configuration: Utilize Hugging Face's Transformers to configure the RAG system.

Chain of Thought Implementation:

Integrate a CoT strategy to process queries in a stepwise manner, improving the logical flow and relevance of responses.

Pseudocode for Chain of Thought Processing:

```
function process_query_with_chain_of_thought(user_query,
previous_context):
    initial_response = rag_generate_response(user_query)
    thought_steps = develop_reasoning_steps(initial_response, previous_context)
    final_response = refine_response_based_on_thought_steps(thought_steps)
    return final_response
```

Integration with WordPress

Plugin Development: Create a WordPress plugin that allows easy integration and configuration of the chatbot across sites.

API Implementation: Develop secure REST APIs using Flask or FastAPI for backend communication between the WordPress plugin and the AI system.

Testing and Evaluation

Functional Testing: Test the complete system functionality including data fetching, response generation, and UI interaction.

Performance Testing: Measure response times, accuracy, and scalability.

Chain of Thought Testing: Specifically evaluate the logic and coherence of the responses generated by the CoT strategy.

Documentation and Reporting

System Documentation: Provide detailed documentation covering system architecture, codebases, integration methods, and usage instructions.

Operational Manual: Include setup guides, configuration details, and troubleshooting instructions for end-users and system administrators.

Project Report: Outline challenges encountered, solutions implemented, and performance metrics, along with future improvement recommendations.