

Pulse - Help Website Q&A Agent

Technical Interview Assignment

Overview

Create an AI-powered question-answering agent that can process documentation from help websites and accurately answer user queries about product features, integrations, and functionality.

Requirements

Core Functionality

1. The agent should accept a help website URL as input (e.g., help.zluri.com, help.slack.com)
2. Process and index the content for efficient querying
3. Accept natural language questions via terminal interface
4. Provide accurate answers based on the processed documentation
5. Clearly indicate when information is not available in the documentation

Technical Requirements

1. **Input Processing**
 - Handle different help website URL formats
 - Implement proper error handling for invalid URLs
 - Support recursive crawling of documentation pages
2. **Content Processing**
 - Extract meaningful content while filtering out navigation elements, headers, footers
 - Maintain proper context hierarchy of the documentation
 - Handle different content types (text, lists, tables)
3. **Question Answering**
 - Process natural language questions
 - Implement semantic search or similar technology for finding relevant content
 - Format answers in a clear, readable manner
 - Include source references (URLs) for answers when possible
4. **Error Handling**
 - Graceful handling of network issues

- Clear feedback for unsupported websites
- Proper indication when questions cannot be answered

Example Usage

bash

Copy

```
# Starting the agent
python qa_agent.py --url https://help.example.com

# Example interaction
> What integrations are available?
[Agent responds with integration information from documentation]

> How do I enable feature X?
[Agent provides step-by-step instructions if available]

> Does the product support feature Y?
Sorry, I couldn't find any information about feature Y in the
documentation.
```

Evaluation Criteria

1. Accuracy (50%)

- Correctness of answers
- Relevance of responses
- Proper handling of "no information" cases
- Consistency in answers
- Context preservation

2. Technical Implementation and Innovation (35%)

- Efficient approach
- Appropriate choice of search/matching algorithm
- Resource usage optimization
- Response time
- Scalability considerations

3. Code Quality (15%)

- Clean, well-organized code structure
- Proper error handling and edge cases

- Use of appropriate design patterns
- Code documentation and comments
- Modular and maintainable architecture
- Proper dependency management

Submission Requirements

1. Code Repository

- Submit via a private GitHub repository
- Include clear README with:
 - Setup instructions
 - Dependencies
 - Usage examples
 - Design decisions
 - Known limitations

2. Documentation

- Technical architecture overview
- Implementation approach
- Future improvement suggestions
- Testing approach

3. Testing

- Include unit tests
- Provide test cases with example questions and expected answers
- Include performance benchmarks

Time Expectation

- 48 hours for a basic implementation

Bonus Points

1. Advanced Features

- Support for multiple documentation sources
- Answer caching mechanism
- Support for different documentation formats
- Confidence scores for answers

2. Technical Improvements

- Docker containerization
- API endpoint addition
- Performance optimizations
- Advanced NLP techniques

Notes

- You may use any programming language, but Python is preferred
- Document any third-party libraries used
- Include any assumptions made during implementation
- Note any limitations of your approach

Submission Process

1. Share repository access with vatsal@pulsegen.io
2. Include a brief video demo (5 minutes max)
3. Be prepared to discuss your implementation decisions in the review call