Code Editor with Stack Overflow and Expert Recommendation

Technical Documentation Report

Shreyansh Pathak (D24CSA006)
Ratnesh Dubey (M24CSA0027)
Somesh Joshi (M24CSA0031)
Dhiraj Raj (M24CSA009)
Neha Sharma (M24CSE014)

November 16, 2024

Contents

1 Abstract

This report presents a comprehensive analysis of a web-based application that integrates Stack Overflow's API with an expert recommendation system. The system provides users with both relevant Stack Overflow answers and expert recommendations based on their queries, while also offering an integrated Python code editor with execution capabilities. The platform serves as a unified development environment that combines search functionality, expert networking, and code testing capabilities.

2 Introduction

2.1 Project Overview

The application addresses common challenges faced by developers by providing:

- Integrated Stack Overflow search functionality
- Expert recommendation system based on query topics
- Interactive Python code editor with execution capabilities
- Real-time code testing environment

2.2 Objectives

The primary objectives of this project include:

- Streamlining the development workflow
- Providing immediate access to relevant programming solutions
- Facilitating connections with domain experts
- Offering an integrated development and testing environment

3 System Architecture

3.1 Backend Components

3.1.1 Core Application (app.py)

The Flask-based backend implements the following key endpoints:

Key Endpoints

- `/`: Main application interface
- `/search_stack_overflow`: Stack Overflow API integration
- `/run_code`: Code execution handler
- `/recommend_expert`: Expert recommendation processor

```python
@app.route('/search_stack_overflow', methods=['POST'])
def search_stack_overflow():
    query = request.json.get('query')
    try:
        keywords = find_keywords(query)
        tags = "; ".join(keywords)
        tags = tags + '; ' + 'python'
        response = requests.get(
            STACK_OVERFLOW_API_URL,
            params={
                'intitle': query,
                'tagged': tags,
                'site': 'stackoverflow',
                'order': 'desc',
                'sort': 'votes'
            }
        )
        return jsonify(response.json())
    except requests.exceptions.RequestException as e:
        return jsonify({'error': str(e)}), 500
```

Listing 1: Core Route Implementation

3.1.2 Utilities Module (utilities.py)

The utilities module contains two primary components:

```python
def find_keywords(query):
    kw_extractor = yake.KeywordExtractor()
    keywords = kw_extractor.extract_keywords(query)
    return [kw[0] for kw in keywords][:5]
```

Listing 2: Keyword Extraction Implementation

2. Expert Recommendation System The recommendation system utilizes a graph-based approach implemented using NetworkX:

```python
class RecommendExpert:
    def __init__(self, save_file='recommend_expert_graph.pkl'):
        self.save_file = save_file
        self.graph = self.load_graph()

    def add_user(self, user_id, user_name, link, expertise):
        self.graph.add_node(user_id,
                            user_name=user_name,
                            link=link,
                            expertise=expertise)
        self.graph.add_edge(user_id, expertise)
        self.save_graph()
```

Listing 3: Expert Recommendation Class

3.2 Frontend Architecture

3.2.1 Code Editor Implementation

The application utilizes CodeMirror with the following configuration:

```
editor = CodeMirror.fromTextArea(codeEditor, {
    lineNumbers: true,
    mode: "python",
    theme: "material-darker",
    hintOptions: {
        completeSingle: false
    },
    extraKeys: {
        'Tab': function(cm) {
            cm.showHint({ hint: CodeMirror.hint.python });
        }
    }
});
```

Listing 4: CodeMirror Configuration

3.2.2 Search Interface

The search interface implements:

- Real-time button state management
- Loading indicators
- Grid-based result display
- Comprehensive error handling

4 Key Features Analysis

4.1 Stack Overflow Integration

The system integrates with Stack Overflow's API providing:

- Keyword-based search functionality
- Python-focused result filtering
- Vote-based result sorting
- Rich metadata display

4.2 Expert Recommendation System

Key features include:

- Graph-based expert-topic relationships
- Persistent profile storage
- Topic-based matching algorithm
- Interactive UI for recommendations

4.3 Code Editor Features

- File import/export capabilities
- Integrated execution environment
- Output capture and display
- Comprehensive error handling

5 User Interface Design

5.1 Main Components

The UI implements several key design patterns:

```
.results-container {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
    padding: 20px;
    background-color: #f9f9f9;
    border: 2px solid #ddd;
}

.sidebar {
    position: fixed;
    width: 300px;
    height: 100%;
    transition: right 0.3s ease-in-out;
}
```

Listing 5: Key UI Styles

5.2 Interactive Elements

The interface features:

- Hover effects for search results
- Loading animations
- Smooth transitions
- Responsive layouts

6 Implementation Details

6.1 Search Functionality

```
function searchStackOverflow(query) {
    const resultsContainer = document.getElementById("searchResults");
    const loadingSpinner = document.getElementById("loadingSpinner");

    resultsContainer.style.display = "none";
    loadingSpinner.style.display = "block";

    fetch('/search_stack_overflow', {
```

```
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ query: query })
    })
    .then(response => response.json())
    .then(data => {
        // Process and display results
    })
    .catch(error => {
        console.error('Error:', error);
    });
}
```

Listing 6: Search Implementation

6.2 Code Execution

```
@app.route('/run_code', methods=['POST'])
def run_code():
    code = request.json.get('code')
    output = ""
    old_stdout = sys.stdout
    redirected_output = io.StringIO()
    sys.stdout = redirected_output

    try:
        exec(code, {})
        output = redirected_output.getvalue()
    except Exception as e:
        output = f"Error: {str(e)}"
    finally:
        sys.stdout = old_stdout

    return jsonify({"output": output})
```

Listing 7: Code Execution Handler

7 Performance Considerations

7.1 Frontend Optimizations

- Debounced search inputs
- Lazy loading implementation
- Efficient DOM updates
- Resource management

7.2 Backend Optimizations

- Caching mechanisms
- Efficient graph traversal

- Error handling strategies
- Resource cleanup procedures

8 Security Considerations

8.1 Code Execution Security

- Sandboxed execution environment
- Input validation and sanitization
- Output limitation controls
- Resource usage monitoring

8.2 API Security

- Rate limiting implementation
- Request validation
- Error handling procedures
- Data sanitization

9 Future Enhancements

Potential improvements include:

1. Authentication system integration
2. Extended programming language support
3. Collaborative coding features
4. Enhanced code analysis tools
5. Improved recommendation algorithms
6. Real-time collaboration capabilities
7. Advanced code debugging features
8. Integration with additional developer resources

10 Conclusion

The Stack Overflow Integration and Expert Recommendation System successfully combines multiple complex components to create a unified development assistance platform. Its modular architecture ensures maintainability and extensibility while providing a robust and user-friendly experience. The system effectively addresses common developer challenges by providing immediate access to relevant solutions, expert guidance, and integrated testing capabilities.

11 References

1. Stack Overflow API Documentation
   `https://api.stackexchange.com/docs`
2. Flask Documentation
   `https://flask.palletsprojects.com/`
3. CodeMirror Documentation
   `https://codemirror.net/docs/`
4. NetworkX Documentation
   `https://networkx.org/documentation/stable/`
5. YAKE Keyword Extraction Documentation
   `https://github.com/LIAAD/yake`

A Installation Guide

To set up the development environment:

```
# Install required Python packages
pip install flask requests networkx yake

# Clone the repository
git clone <repository-url>

# Navigate to project directory
cd project-directory

# Run the application
python app.py
```

B Configuration Parameters

Key configuration parameters include:

- Stack Overflow API endpoint configuration
- Server configuration settings
- Graph storage location specifications
- API rate limit settings
- Development environment variables