Project Title: Enhancing Automated Web Scraping and Analysis Tool with Python

Student Information

Github: https://github.com/AggQ

Timezone: (GMT+8) Singapore

Project Abstract

This project aims to enhance the capabilities of a Python-based automated web scraping and analysis tool. The tool will be further developed to improve web data extraction, analysis, and visualisation, providing users with comprehensive insights from diverse sources. Additionally, the project will involve creating a tutorial to onboard new contributors, focusing on unit testing and contributing to the tool's development.

Project Description

The current Python-based automated web scraping and analysis tool provides basic functionalities for gathering data from websites, performing analysis, and presenting insights. However, there is room for improvement in terms of efficiency, scalability, and user-friendliness. This project will address these areas and introduce new features to enhance the tool's utility and accessibility.

Phase I

- Conduct a survey with the user community to identify existing limitations and areas for improvement in the tool.
- Analyse feedback and prioritise enhancements based on user needs and feasibility.
- Develop a plan for measuring and improving unit test coverage to ensure the reliability and robustness of the tool's codebase.

Phase II

```python
class TestWebScraper(unittest.TestCase):
    def setUp(self):
        self.scraper = WebScraper()

    def test_fetch_website(self):
        # Positive Test Case: Ensure website is fetched successfully
        website_content = self.scraper.fetch_website('https://example.com')
        self.assertIsNotNone(website_content)

        # Negative Test Case: Ensure invalid website returns None
        invalid_website_content = self.scraper.fetch_website('https://invalidwebsite.com')
        self.assertIsNone(invalid_website_content)


class TestDataAnalyzer(unittest.TestCase):
    def setUp(self):
        self.analyzer = DataAnalyzer()

    def test_analyze_data(self):
        # Positive Test Case: Ensure data analysis returns expected results
        data = [1, 2, 3, 4, 5]
        result = self.analyzer.analyze_data(data)
        self.assertEqual(result, {'mean': 3.0, 'median': 3, 'max': 5, 'min': 1})

        # Negative Test Case: Ensure empty data returns None
        empty_data = []
        result = self.analyzer.analyze_data(empty_data)
        self.assertIsNone(result)
```

Two test classes, `TestWebScraper` and `TestDataAnalyzer`, were created to assess the functionality of the `WebScraper` and `DataAnalyzer` classes, respectively.

The `TestWebScraper` class evaluates the `fetch_website` method's behaviour with both valid and invalid URLs. Meanwhile, the `TestDataAnalyzer` class verifies the correctness of the `analyze_data` method's output, including its handling of empty data.

These tests ensure robustness and accuracy in the web scraping and data analysis functionalities of the tool.

- Implement enhancements to the web scraping pipeline to improve efficiency and scalability.
- Introduce new data analysis modules to provide more advanced insights and visualisation options.

- Extend the tool's functionality to support automated scheduling and reporting for recurring tasks.
- Collaborate with the community to address any identified issues and encourage contributions to improve unit test coverage.

Phase III

- Create a comprehensive tutorial for new contributors, covering the basics of unit testing and contributing to the tool's development.
- Document the development process and guidelines for adding new features or improving existing functionalities.
- Integrate the tutorial into the tool's documentation and developer resources to facilitate onboarding of new contributors.
- Solicit feedback from users and contributors to iterate on the tutorial and improve its effectiveness over time.

Development Process

The development process will follow an iterative approach, with regular feedback loops and milestone reviews to track progress and ensure alignment with project goals. Continuous integration and automated testing will be employed to maintain code quality and reliability throughout the development lifecycle.

Deliverables

- Enhanced web scraping and analysis tool with improved efficiency, scalability, and functionality.
- Increased unit test coverage and improved code quality to enhance reliability and maintainability.
- Comprehensive tutorial for new contributors, covering unit testing and contributing guidelines.
- Documentation updates and developer resources to support ongoing development and community engagement.

By the end of the project, the aim is to provide users with an enhanced and more accessible web scraping and analysis tool, while also fostering a collaborative and inclusive community around its development.