Test Plan for hack_iu Repository

1. Objective

The objective of this test plan is to validate the functionality, stability, and reliability of the hack_iu repository. This includes testing the core features, data management, logging, text editing, Bash command execution, and the user interface. The goal is to ensure all parts of the system are working as expected and handle edge cases gracefully.

2. Test Categories

The tests will be organized into the following categories:

1. **Core Functionality Tests**: Validate the key-value store operations (CRUD).
2. **Text Editing Tests**: Test file viewing, creation, string replacement, and insertion.
3. **Bash Command Execution Tests**: Test Bash command execution capabilities.
4. **Logging Tests**: Ensure proper logging is happening for API calls, errors, and tool operations.
5. **Web Interface Tests**: Validate Streamlit UI components, including key-value interactions and log analysis.
6. **Performance and Edge Case Tests**: Test for system performance and robustness under edge cases and stress scenarios.

3. Test Scenarios

3.1 Core Functionality Tests

- **Test Case 1**: **Count the Total Key-Value Pairs in data.jsonl**
  - **Input**: Request to count the total key-value pairs in the data.jsonl file.
  - **Expected Output**: The system should return the correct count based on the number of lines in the data.jsonl file multiplied by 4 (each line contains 4 key-value pairs).
  - **Pass Criteria**: The output should correctly compute and return the total number of key-value pairs.
- **Test Case 2**: **Insert a New Entry in output.jsonl**
  - **Input**: Insert an entry with Key: "server.config", Value: "production" into output.jsonl.
  - **Expected Output**: The system should correctly insert the entry and update the file.
  - **Pass Criteria**: The file should be updated with the new entry, and no errors should be returned.
- **Test Case 3**: **Retrieve Key-Value Pair**
  - **Input**: Retrieve a key-value pair by Key from data.jsonl.
  - **Expected Output**: The system should return the value associated with the provided key.
  - **Pass Criteria**: The correct value for the provided key should be returned.
- **Test Case 4**: **Update a Key-Value Pair**
  - **Input**: Update the Value for an existing Key in data.jsonl.
  - **Expected Output**: The system should update the value and save it in the file.
  - **Pass Criteria**: The value for the key is successfully updated, and the changes are saved correctly in the file.

- **Test Case 5**: **Delete a Key-Value Pair**
    - **Input**: Remove a key-value pair based on Key.
    - **Expected Output**: The system should remove the specified key-value pair from the data.jsonl file.
    - **Pass Criteria**: The specified key-value pair should be deleted, and the file should be updated correctly.

3.2 Text Editing Tests

- **Test Case 6**: **View File Content**
    - **Input**: Request to view the content of a file.
    - **Expected Output**: The system should display the content of the specified file.
    - **Pass Criteria**: The content of the file should be correctly returned.
- **Test Case 7**: **Create a New File**
    - **Input**: Create a new file with specified text.
    - **Expected Output**: The system should create the file at the specified path and populate it with the provided text.
    - **Pass Criteria**: The file should be created successfully with the expected content.
- **Test Case 8**: **Replace a String in the File**
    - **Input**: Replace an existing string in the file with a new string.
    - **Expected Output**: The system should replace the old string with the new string.
    - **Pass Criteria**: The file content should reflect the string replacement.
- **Test Case 9**: **Insert Content into a File at a Specific Line**
    - **Input**: Insert new content into a file at a specified line.
    - **Expected Output**: The content should be inserted at the specified line, and the rest of the file should remain intact.
    - **Pass Criteria**: The content should be inserted correctly at the specified line, and the file should be updated.

3.3 Logging Tests

- **Test Case 10**: **Log Key-Value Store Operation**
    - **Input**: Perform a CRUD operation (e.g., insert or update a key-value pair).
    - **Expected Output**: The system should log the operation to the session log.
    - **Pass Criteria**: The operation should be logged with relevant details, such as the action performed and the key-value pair.
- **Test Case 11**: **Log Errors**
    - **Input**: Trigger an error, such as attempting to insert a value at a non-existent line.
    - **Expected Output**: The error should be logged in the system logs.
    - **Pass Criteria**: The error details should be properly logged, including the error message.

3.5 Web Interface Tests

- **Test Case 12**: **Command Input via UI**
  - **Input**: Enter a command via the Streamlit web interface (e.g., insert a key-value pair).
  - **Expected Output**: The system should process the input command and display feedback on the UI.
  - **Pass Criteria**: The correct output should be shown on the UI, reflecting the action performed.
- **Test Case 13**: **View Key-Value Store in Tabular Format**
  - **Input**: Load the web page displaying the key-value store.
  - **Expected Output**: The system should display the key-value store in a clear tabular format.
  - **Pass Criteria**: The key-value store should be displayed correctly, with each entry shown in the table.
- **Test Case 14 Log Histogram Visualization**
  - **Input**: View the histogram of log entries by hour.
  - **Expected Output**: The system should render a histogram showing log entries by the hour.
  - **Pass Criteria**: The histogram should correctly represent the log entry distribution by hour.

3.6 Performance and Edge Case Tests

- **Test Case 15**: **Handle Invalid Inputs**
  - **Input**: Provide invalid input (e.g., malformed JSON, invalid keys).
  - **Expected Output**: The system should validate the input and return an appropriate error message.
  - **Pass Criteria**: Invalid inputs should be handled gracefully with clear error messages.

4. Test Results Documentation

For each test case, the following details should be documented:

All the core, text editing, Logging, Web UI, test cases has been successfully handled.

For, performance and Edge Cases which involve invalid inputs such as incorrect file names, mis-spells, only some cases have been handled correctly by AI agent. There is room of improvement for the case like: Getting different commands as output for bash prompts or the AI agent using different file handlers each time it runs will need fine tuning.

6. Test Tools and Resources

- **Python Version**: Python 3.12.0
- **Dependencies**:

- anthropic
- streamlit
- pandas
- Other dependencies as per requirements.txt

7. Test Execution

The test cases were executed manually, and logs should be collected for every operation performed to verify the system's behavior under normal and edge case conditions.