Solution of NETZSCH Tasks

Solution of Task 01: Test Strategy for CSV File Import Functionality

1. Introduction

This document outlines the test strategy for the CSV file import functionality. The goal of this testing is to ensure that the import function accurately reads and stores data from a CSV file into a database while properly handling and reporting errors, such as invalid data formats. Also ensure data integrity, robust error handling, security, and performance. This strategy and approach will guide the testing process, including the types of tests to be performed, resources required, tools to be used, and the overall approach.

2. Objective

The primary objective of the testing is to validate the functionality, reliability, and robustness of the CSV file import process. The testing will ensure:

- The CSV import functionality reads and processes files correctly.
- Data is accurately stored in the database without loss.
- The system effectively handles various error scenarios, including invalid data formats, missing fields, and duplicate records.
- The performance of the import process is within acceptable limits for various file sizes.
- The process is secure against potential threats, such as SQL injection.

3. Scope

- Functional Testing of CSV import (valid and invalid scenarios).
- Data Validation and Integrity Testing.
- Error Handling and Logging.
- Performance Testing with various file sizes.
- Security Testing, including SQL injection and data access control.
- **Scalability Testing**: Ensure the system can handle large-scale imports.
- Regression Testing to ensure new changes do not impact existing functionality.

4. Test Approach

The testing approach will be divided into several phases:

- **Phase 1: Functional Testing**

    - Validate that the import function reads CSV files correctly and stores the data in the database.
    - Test with both valid and invalid CSV files to check error handling and reporting.

- **Phase 2: Data Validation Testing**
    - Ensure that all data is accurately captured, transformed, and stored without any data loss or corruption.
- **Phase 3: Performance Testing**
    - Evaluate the performance of the import function with varying file sizes to ensure it meets performance requirements.
- **Phase 4: Security Testing**
    - Test the import function against SQL injection and ensure that data access is properly controlled.
- **Phase 5: Regression Testing**
    - Conduct regression tests after every significant change to ensure that the CSV import functionality remains intact.

<u>Test Cases</u>

Here are some detailed test cases based on the test strategy:

1. Test Case: Valid CSV Import

Objective: Verify that the function correctly imports valid CSV data into the database.

Preconditions:

The CSV file is correctly formatted with all required columns and valid data types.

The test environment and database are set up and accessible.

<u>Steps:</u>

- Upload the valid_data.csv file to the application.
- Trigger the import function.
- Verify that the import function processes the CSV file without errors.
- Check the database to confirm that all rows from the CSV file are accurately stored.

Expected Results:

- The import function completes successfully.
- All rows from the valid_data.csv file are present in the database with correct data.
- No error messages are displayed or logged.

2. Test Case: Invalid Data Format Handling

Objective: Ensure that the function catches and reports invalid data formats correctly.

Preconditions: The CSV file contains data with incorrect formats (e.g., text in a numerical field).

__**Steps:**__

- Upload the invalid_format.csv file to the application.
- Trigger the import function.
- Observe the system's response to the invalid data format.

Expected Results:

- The import function identifies the invalid data format (e.g., "thirty" instead of a numeric value for Age).
- An error message is displayed or logged indicating the specific issue with data format.
- The invalid row is not imported into the database.

3. Test Case: Missing Fields in CSV

Objective: Validate how the function handles missing fields in the CSV file.

Preconditions: The CSV file is missing some required columns.

__**Steps:**__

- Upload the missing_fields.csv file to the application.
- Trigger the import function.
- Observe the system's response to the missing fields.

Expected Results:

- The import function detects the missing "Age" column.
- An error message is displayed or logged indicating the missing field.
- No data is imported into the database.

4. Test Case: Duplicate Data Handling

Objective: Verify that the function handles duplicate data appropriately.

Preconditions: The CSV file contains duplicate entries.

__**Steps:**__

- Upload the duplicate_data.csv file to the application.
- Trigger the import function.
- Observe how the system handles duplicate entries.

Expected Results:

- The import function identifies the duplicate entry.
- An appropriate message is displayed or logged indicating the duplicate record.
- The duplicate row is either ignored or flagged, based on business rules.

5. Test Case: Large File Performance

Objective: Test the performance of the import function with a large CSV file.

Preconditions: The CSV file is large (e.g., 1 million rows).

Steps:

- Upload the large_data.csv file to the application.
- Trigger the import function.
- Measure the time taken for the import process to complete.

Expected Results:

- The import function completes the process within an acceptable time frame (e.g., less than 5 minutes).
- No errors or performance bottlenecks occur during the import.
- All data is correctly stored in the database.

6. Test Case: Special Character Handling

- **Objective**: Ensure system handles special characters and encodings correctly.
- **Steps**: Upload special_characters.csv, trigger import, verify data.
- **Expected** Results: Special characters are correctly imported.

5. Test Methodology

- **Black-Box Testing**: Used for functional and error handling tests where the tester does not need to know the internal code structure.
- **White-Box Testing**: Applied for security testing, focusing on SQL injection vulnerabilities and potential security flaws.
- **Data-Driven Testing**: Uses various sets of input data to validate the CSV import function under different scenarios.
- **Performance Testing**: Measures the import function's performance using tools that simulate large CSV file uploads.

6. Test Environment

- **Database:** A test database instance will be set up that mirrors the production environment.

- **CSV Files**: Various CSV files will be prepared, including:
  - Valid files with correct data.
  - Invalid files with incorrect data formats, missing fields, and duplicate records.
- **Tools:**
  - **Automated Testing Tools**: Python Pytest framework for running automated test cases.
  - **Performance Testing Tools**: JMeter for simulating large file uploads.
  - **Security Testing Tools**: SQLMap for SQL injection testing.

7. Roles and Responsibilities

- **QA Manager**: Oversee the entire testing process, ensuring the strategy is followed.
- **QA Engineers**: Develop and execute test cases, automate testing scripts, report bugs, and validate fixes.

9. Test Deliverables

- **Test Plan**: Document outlining the overall testing approach and schedule.
- **Test Cases**: Detailed test cases for each functionality.
- **Test Scripts**: Automated scripts for executing test cases.
- **Bug Reports**: Detailed reports of any defects found during testing.
- **Test Summary Report**: A final report summarizing the testing activities, results, and quality of the product.

Solution of Task 02: Create Test Automation Script

The provided test script is designed to validate CSV data processing functions using pytest. It includes five test cases covering various scenarios:

1. Valid CSV Import: Ensures correct data import.
2. Invalid Data Format: Checks for appropriate error handling when data formats are incorrect.
3. Missing Fields: Verifies that the program correctly handles missing data fields.
4. Duplicate Data Handling: Tests the system's capability to detect duplicate entries.
5. SQL Injection Security: Detects potential SQL injection patterns in CSV data.

Explanation of SQL Injection Test Case:

- **Purpose**: To identify potentially harmful SQL injection patterns within data imported from CSV files.
- **Current Implementation**: Detects SQL injection patterns and fail if such patterns are found (Security test). Currently assertion is failing because injection pattern is found

Folder Structure of Source Code:

/NETSZCH

/tests

/data

valid_data.csv

invalid_format.csv

missing_fields.csv

duplicate_data.csv

sql_injection.csv

test_csv_import.py

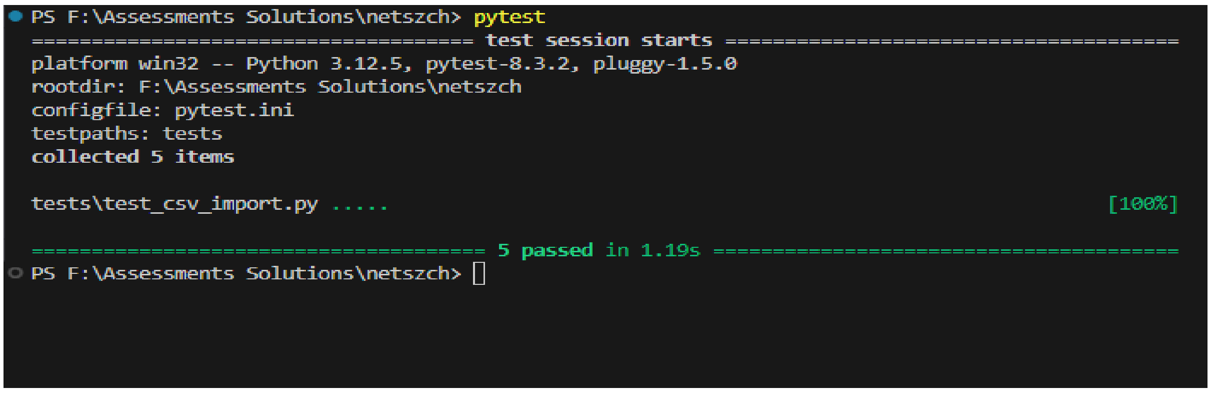
/pytest.ini

(Code Script is attached.)

Execution of Tests:

1. pip install pytest
2. cd /NETSZCH
3. Run this command 'pytest'

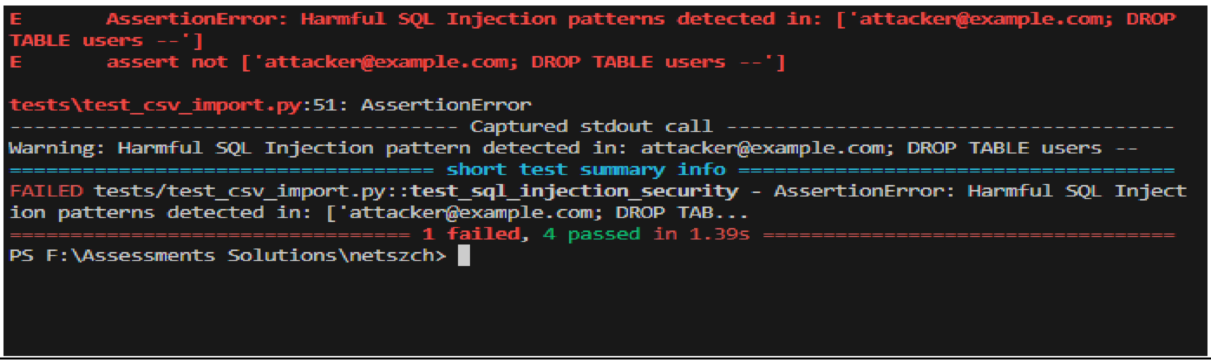
Execution of Result:



```
PS F:\Assessments Solutions\netszch> pytest
===================================== test session starts =====================================
platform win32 -- Python 3.12.5, pytest-8.3.2, pluggy-1.5.0
rootdir: F:\Assessments Solutions\netszch
configfile: pytest.ini
testpaths: tests
collected 5 items

tests\test_csv_import.py .....                                                          [100%]

====================================== 5 passed in 1.19s ======================================
PS F:\Assessments Solutions\netszch> 
```



```
E       AssertionError: Harmful SQL Injection patterns detected in: ['attacker@example.com; DROP
TABLE users --']
E       assert not ['attacker@example.com; DROP TABLE users --']

tests\test_csv_import.py:51: AssertionError
------------------------------------- Captured stdout call -------------------------------------
Warning: Harmful SQL Injection pattern detected in: attacker@example.com; DROP TABLE users --
=================================== short test summary info ====================================
FAILED tests/test_csv_import.py::test_sql_injection_security - AssertionError: Harmful SQL Inject
ion patterns detected in: ['attacker@example.com; DROP TAB...
================================= 1 failed, 4 passed in 1.39s ==================================
PS F:\Assessments Solutions\netszch> 
```

Future Improvements:

- Integration with CI/CD: Plan to Integrate test suite with continuous integration tool like Jenkins, Travis CI, or GitHub Actions for automatic test runs on each commit.
- Enhanced Reporting: Utilize pytest plugins like pytest-html to generate detailed HTML reports of test results.

Task 03: Error Analysis and Report

Summary of Issues

1. **Memory Usage Fluctuations:**
   - **Observation**: The logs indicate consistent scheduled memory usage checks. The working set size and peak working set size fluctuate slightly, with a notable decrease at S2024-07-16 11:47:23.475 where it drops from 262796 kB to 209064 kB.
   - **Impact**: This fluctuation suggest potential memory leak. Further investigation is necessary to determine if this could lead to system instability or performance degradation over time
2. **Hardware Error: SysBus Communication:**
   - **Observation**: Multiple entries in the logs point to a HW error in SysBus communication at different timestamps (e.g., S2024-07-14 00:54:26.849, S2024-07-14 08:16:51.060, S2024-07-15 06:48:02.967).
   - **Impact**: These repeated errors suggest an ongoing issue with system bus communication, which could affect hardware stability and lead to intermittent failures or degraded performance.
3. **Virtual Memory Usage:**
   - Observation: The virtual memory usage steadily increases over time, with slight drops observed intermittently. The peak virtual memory usage recorded was 9436508 kB on S2024-07-17 06:47:30.122.
   - **Impact:** This steady increase indicate inefficient memory management and potential memory leak. It suggests that the system is under stress, possibly leading to slowdowns or crashes if not addressed.
4. **System Temperature Control:**
   - Several entries note the start of temperature programs with the furnace locking and unlocking. These operations seem normal but should be closely monitored in conjunction with the memory and hardware communication errors to ensure they are not contributing to system instability.
5. **System Temperature Control:**

   - The logs show instances where the system identified the mass as invalid with a status of 5. This issue may indicate a calibration problem or sensor malfunction.

6. **Resource Exhaustion Risks:**
   - Physical memory usage reached 7,130,780 kB, suggesting significant memory consumption, which could undermine overall system performance.
7. **User Permissions Issue:**
   - The user "Labor" is not a member of the Administrator’s group, which could limit access to certain functionalities or administrative actions.

Bug Report

1. Issue: HW Error SysBus Communication

- Occurrence:
  - 2024-07-16 06:39:40.981
  - 2024-07-16 11:59:16.924
  - 2024-07-17 01:02:52.870
- Description: This error indicates an issue with the system bus communication at various times. The repeated occurrence suggests a possible underlying hardware or connectivity problem that needs further investigation.
- **Impact:** Potential system instability due to communication failures between critical hardware components.

2. **Issue: Invalid Mass Detection**

- Occurrence:
  - 2024-07-12 11:51:12.560
  - 2024-07-12 11:51:13.573
- **Description**: The system repeatedly identified the mass as invalid, with a status of 5. This could indicate a calibration issue or a sensor malfunction that needs to be addressed to ensure accurate measurements.
- **Impact:** Incorrect mass detection could lead to faulty data processing, compromising the reliability of the system's output.

3**. Issue: ASC Start Dialog Pointer is Null**

- Occurrence:
  - 2024-07-17 11:50:27.379
  - 2024-07-17 11:50:28.411
  - 2024-07-17 11:54:10.970

- Description: The ASC (Automated Sample Changer) start dialog pointer being null indicates a failure in initializing or referencing the ASC's start dialog, potentially leading to failures in subsequent ASC operations.
- **Impact:** This can disrupt automated processes, requiring manual intervention and possibly leading to operational delays.

4. **Issue: ASC Notifications with Repeated FF Values**

- Occurrence: Multiple instances throughout the logs where ASC notifications returned values like FF FF FF or similar.
- Description: These notifications might indicate a default or error state in the ASC communication, possibly due to uninitialized variables or incorrect data being sent/received by the ASC module.
- Impact: If unaddressed, this issue could lead to persistent communication errors and affect the overall reliability of ASC operations.

5. **Issue: Virtual Memory Usage and GDI Objects Increase**

- **Occurrence**:
    - Virtual memory usage and GDI objects slowly increase over time without significant reduction, indicating a potential memory leak.
- **Description**: The logs show consistent increases in virtual memory usage and GDI objects, suggesting a resource leak that could degrade performance or eventually lead to a system crash if not managed.
- **Impact:** Resource leaks can cause system slowdowns and crashes, especially under prolonged or heavy use, affecting the system's reliability and user experience.

Detail Description of Issues Found

1. **High Memory Usage:**
    - **Description**: The application shows signs of high memory consumption with a recorded working set size of 205,132 kB at startup, peaking at 205,168 kB.
    - **Impact**: High memory usage can lead to performance degradation and may cause the application to become unresponsive if demand exceeds available resources.
2. **Resource Exhaustion Risks**:
    - **Description**: Physical memory usage reached 7,130,780 kB, indicating that the application consumes significant amounts of memory, undermining overall system performance.
    - **Impact**: Risk of slowdowns or crashes due to memory exhaustion.

3. **Sensor Communication Issues:**
    - **Description**: Frequent references to command return values of 0, indicating no data or timeout in communication with sensors and instruments during operations.
    - **Impact**: Device configuration and operation reliability are in question, potentially leading to failures in expected sensor performance.
4. **Touchscreen Responsiveness:**
    - **Description**: The touchscreen was confirmed responsive at critical times, but potential delays in communication were noted.
    - **Impact**: Interaction delays could affect user experience and operational efficiency.

5. **Configuration Consistency:**
    - **Description**: The configuration during the startup process appeared consistent; however, the report of 'no change' raises concerns about the verification processes in place.
    - **Impact**: Ongoing validation of configurations is required to ensure miscommunication or misconfigurations do not occur unnoticed.
6. **User Permissions Issue:**
    - **Description**: The user "Labor" is not a member of the Administrator’s group.
    - **Impact**: Possible limitations in accessing certain functionalities or performing administrative actions affecting overall software performance and security.

Recommended Actions

1. **Review and Optimize Memory Management:**
    - Investigate the application's memory usage patterns and optimize to prevent exhaustion.
2. **Investigate Communication Latency:**
    - Analyze touchscreen and sensor communication pathways to address any delays impacting performance.
3. **Enhance Configuration Monitoring:**
    - Implement stronger monitoring tools to detect any discrepancies in configurations during critical operational phases.
4. **Evaluate User Permissions:**

- Review user-level permissions to access necessary features for optimal software operation.

Format of Bug report

Title: SysBus Communication Errors and Memory Management Issues

Environment:

- Hardware: Intel(R) Core(TM) i5-8500 CPU @ 3.00GHz, 16GB RAM, Windows 10 Enterprise 64-bit
- Software: TG 309 Libra Supreme on USBc1, Software Version: 9.4.0, Firmware Version: 3.339.00

Steps to Reproduce:

1. The system operates normally with scheduled temperature programs.
2. The system periodically checks memory usage and monitors hardware communication.
3. Multiple instances of HW error SysBus communication are logged during operation.

Expected Result:

- The system should maintain stable memory usage without significant drops or unexplained increases.
- Hardware communication should be stable without repeated errors.

Actual Result:

- Memory usage drops significantly at specific intervals, suggesting potential memory management issues.
- Repeated HW error SysBus communication errors are logged, indicating a possible hardware communication issue.

Impact:

- Potential system instability due to hardware communication errors.
- Memory management inefficiencies could lead to performance degradation over time.

Thanks!