**Functional Test Cases:**

1. **Use a consistent naming convention:** Follow a naming convention that clearly indicates what the test is checking. For example, "TC_Login_01" for a login test case.

```python
# Test Case: TC_Login_01 - Verify successful login with valid credentials
```

2. **Start with a clear test objective:** Begin each test case with a brief description of what you are testing.

```python
# Objective: To verify that a user can log in successfully with valid credentials.
```

3. **Specify preconditions:** Clearly define any prerequisites or initial conditions necessary for the test.

```python
# Preconditions: User must have a registered account.
```

4. **List steps to execute the test:** Provide step-by-step instructions for executing the test case.

```python
# Steps:
# 1. Open the application's login page.
# 2. Enter valid username and password.
# 3. Click the "Login" button.
```

5. **Expected results:** Clearly state what the expected outcome should be after executing the test.

```python
# Expected Result: User should be successfully logged in and directed to the dashboard.
```

6. **Include variations:** Cover different scenarios, including positive and negative test cases.

```python
# Test Case: TC_Login_02 - Verify error message for invalid credentials
# Steps:
# 1. Open the application's login page.
```

# 2. Enter invalid username and password.
# 3. Click the "Login" button.
# Expected Result: An error message should be displayed.

**Non-Functional Test Cases:**

1. **Define the test type:** Indicate the type of non-functional test being performed, such as performance, security, or usability.

```python
# Test Case: TC_Performance_01 - Evaluate application performance under heavy load.
```

2. **Explain the test environment:** Specify the hardware, software, and network conditions under which the test will be conducted.

```python
# Test Environment: Load testing will be conducted on a dedicated test server with simulated user traffic.
```

3. **Identify performance metrics:** Clearly state the performance metrics or benchmarks that need to be met.

```python
# Performance Metrics: The application should maintain a response time of less than 2 seconds under 1000 concurrent users.
```

4. **Describe test data:** Provide details about the test data and its relevance to the non-functional test.

```python
# Test Data: A representative dataset of 100,000 records will be used for load testing.
```

5. **Specify test tools and techniques:** Mention the tools or methodologies used for conducting the non-functional test.

```python
# Test Tools: Apache JMeter will be used for load testing, and OWASP ZAP will be used for security testing.
```

6. **Expected results and acceptance criteria:** Clearly state the acceptable performance or security outcomes.

```python
# Expected Result: The application should handle 1000 concurrent users with a response time of less than 2 seconds, with no security vulnerabilities found.
```

7. **Include error scenarios:** Consider potential failure scenarios and document how they will be tested.

```python
# Test Case: TC_Security_01 - Check for SQL injection vulnerability
# Steps:
# 1. Input a malicious SQL query in the login field.
# 2. Observe the application's response.
# Expected Result: The application should block the malicious input and prevent SQL injection.
```