CSCI3100: Software Engineering

Test Plan Template

Au Yeung Sai Hui, Lui Ka Ho, Cheng Shui Kwong

April 16, 2025

0.1 Document Revision History

Version	Revised By	Revision Date	Comments
1.0	Au Yeung Sai Hui	9 May 2025	Initial draft

1 Test Plan for POSTer

This is a sample test plan for a Website named "POSTer". The most important sections include:

1. Scope and Objectives — Outlining what will be tested and what is excluded from testing.

2. Test Cases and Scenarios — Expanding on the "Scope and Objectives" by providing a detailed breakdown of the scope, along with testing procedures and pass/fail criteria. Similar test cases and scenarios can be merged together.
3. Team Roles and Responsibilities — Defining who is responsible for each task.

4. Testing Approach & Timeline and Schedule — Detailing how and when the tests will be carried out.

1.1 1. Scope and Objectives

This section outlines what will and will not be tested, helping to avoid scope creep and maintain focus throughout the testing process. It also establishes clear boundaries for the testing efforts.

1.1.1 Scope:

The test plan focuses on ensuring the functionality, performance, and user experience of the Pokeman website. The following areas will be tested:

- Core website mechanics (e.g., User Following, Account creation, Posting).
- User interface (UI) and user experience (UX).
- Multiuser functionality (if applicable).
- Compatibility across supported platforms (i.e. PC).

1.1.2 Out of Scope:

- Security issues.
- Testing of third-party integrations not directly related to the website (e.g., external payment systems).
- Hardware-specific performance testing beyond the officially supported devices.
- Modding or hacking scenarios (basically security issues).
- Some hard-to-test aspects of the software (please see Risk assessment and mitigation).

1.1.3 Objectives:

- Verify that the websitee meets functional and non-functional requirements, and provides a bug-free experience.
- Ensure the website is free of critical design flaws.
- Confirm that the website performs well under various conditions.
- Validate that the website is ready for release and meets quality standards.

1.2 2. Test Cases and Scenarios

1.2.1 Test Cases for Functional Requirements:

1.2.1.1 Backend

1.2.1.1.1 Tweet

Test code: GitHub - Test Code for Tweet

- Setup:
  1. Create 2 users (testuser, testuser2)
  2. Create 3 tweets (id = 1, 2, 3)

3. Check if user exists

- Test Create Function:
  - Steps:

1. Create a tweet 'forth tweet' as testuser
2. check if the tweet id, content and user are correct

  - Expected Result: New tweet is created
  - Pass/Fail Criteria: if "tweet id = 4, content = 'forth tweet', username = testuser" on the new tweet then pass else fail

- Create Tweet After Login:
  - Steps:

1. login as testuser
2. Create a tweet 'This is my test tweet'
3. Check if the tweet ID is correct

  - Expected Result: New tweet is created
  - Pass/Fail Criteria: if new tweet id = 5 then pass else fail

- List Tweet:
  - Steps:

1. Login as testuser
2. Check the response

  - Expected Result: Successfully login and grab tweets
  - Pass/Fail Criteria: if "Status code = 200, length of the response = 4" then pass else fail

- Tweet Detail:
  - Steps:

1. Login as testuser
2. Check thee detail of tweet with id 1

  - Expected Result: Tweet id and the content is correct
  - Pass/Fail Criteria: if the content of tweet with id = 1 is 'Hello world, this is my first tweet' then pass else fail

- Like Tweet:
  - Steps:

1. Login as testuser
2. Like the tweet with id 1

3. Check if the tweet is liked

- Expected Result: tweet with id 1 is liked
- Pass/Fail Criteria: if the like count of tweet with id 1 = 1 then pass else fail

- Unlike Tweet:
  - Steps:

1. Login as testuser
2. Like the tweet with id 2
3. Check if the tweet with id 2 is liked
4. Unlike the tweet
5. Check if the like is removed on tweet with id 2

- Expected Result: Like count of tweet with id 2 becomes 1 then 0
- Pass/Fail Criteria: if like count of tweet with id 2 = 0 then pass else fail

- Retweet:
  - Steps:

1. Login as testuser
2. Retweet tweet with id 3
3. Check if new tweet is made and if the original tweet is nested in the retweet and unchanged.

- Expected Result: New tweet is made and the original tweet is nested and unchanged
- Pass/Fail Criteria: if expected result is met then pass else fail

1.2.1.1.2 Account

Test code: GitHub - Test Code for Account

- Setup:

1. Create 2 users with password (user1, user2)

- Profile:
  - Steps:

1. Check if both users have profile

- Expected Result: Both users have profile
- Pass/Fail Criteria: Number of profiles exist = 2

- Test Follow Function:
  - Steps:

1. Follow user1 as user2
2. check follower count of user1 and user2

   - Expected Result: Only user2 follows user1
   - Pass/Fail Criteria: if "user2 is following user1, user1 is not following anyone" then pass else fail

- Follow After Login:

  - Steps:

1. Login as user1
2. follow user2 as user 1
3. check follower count of user2

   - Expected Result: user1 follows user2
   - Pass/Fail Criteria: if "Follower count of user2 = 1" then pass else fail

- Unfollow:

  - Steps:

1. login as user1
2. Follow user2 as user1
3. unfollow user2
4. check follower count of user2

   - Expected Result: "user1 follows then unfollows user2"
   - Pass/Fail Criteria: if "Follower count of user2 = 0" then pass else fail

- Follow self:

  - Steps:

1. login as user1
2. follow user1 as user1

   - Expected Result: User1 cannot follow himself
   - Pass/Fail Criteria: if "Follower count of user1 = 0" then pass else fail

1.2.1.2 Frontend

1.2.1.2.1 Before login

POSTer

- View:

  - Steps:

1. Run manage.py runserver on the terminal
2. open http://localhost:8000

- Expected Result: Navigation bar, create post, tweet list are shown
- Pass/Fail Criteria: Pass if all of the above is shown, else fail

- Not Authenticated:

  - Steps:

  0. DO NOT login as any user
  1. Perform Like action
  2. Perform Unlike action
  3. Perform retweet action
  4. Attempt to post

  - Expected Result: All actions Performed redirects to login page
  - Pass/Fail Criteria: Pass if expected result is observed, else fail

- Navigation bar:

  - Steps:

  0. DO NOT login as any user
  1. Click on the Home Button
  2. Click on "Login"
  3. Click "Register

  - Expected Result: "Home" redirects to home page, "Login" redirects to the login page, Register redirects to the register page
  - Pass/Fail Criteria: Pass if expected result is observed, else fail

1.2.1.2.2 Register

POSTer Register

- View:

  - Steps:

  1. Click Register in the navigation bar

  - Expected Result: Username, password, password confimation
  - Pass/Fail Criteria: Pass if all of the above is shown

2. Register:

   - Steps:

   1. Enter username, password and password confirmation

2. Click register

- Expected Result: Username cannot be empty, password need to follow all the requirement mentioned
- Pass/Fail Criteria: Pass if successfully register an account

1.2.1.2.3 Home Page

POSTer

- View:
  - Steps:

1. Check if the home page is shown after login.

  - Expected Result: Navigation bar, posting textbox is shown
  - Pass/Fail Criteria: Pass if all of the above is shown else fail

- Create Post:
  - Steps:

1. Enter message inside post block
2. Click post

  - Expected Result: Corresponding tweet is created and list below post block
  - Pass/Fail Criteria: Pass if expected result is observed, else fail

- Edit Profile:
  - Steps:

1. Click Edit in the navigation bar to enter update profile page
2. Enter location, bio, first name, last name, and email
3. Click save
4. Click home to return to home page

  - Expected Result: First name and last name is shown in the tweet
  - Pass/Fail Criteria: Pass if expected result is observed, else fail

- Like and Unlike:
  - Steps:

1. Like the first post
2. Unlike the first post

  - Expected Result: Like count increases by 1 when user clicks like, decreases by 1 when user clicks unlike

  - Pass/Fail Criteria: Pass the expected result is observed and the like count cannot be increased and decreased by over 1 else fail

- Retweet:
  - Steps:

  1. Retweet the first tweet

  - Expected Result: New tweet is created which includes the first tweet
  - Pass/Fail Criteria: Pass if expected result is observed, else fail

- Explore:
  - Steps:

  1. Click explore in the navigation bar

  - Expected Result: All tweets including tweets from other users are shown
  - Pass/Fail Criteria: Pass if expected result is observed, else fail

- Profile:
  - Steps:

  1. Click other user's icon or @username

  - Expected Result: Redirect into corresponding user's profile, showing first name, last name, username,follower numbers, following number, location, bio, follow button and user's tweets
  - Pass/Fail Criteria: Pass if expected result is observed, else fail

- Follow/Unfollow:
  - Steps:

  1. Click on the Follow button on another user
  2. return to the home page.
  3. return to profile page, click on the unfollow button on the user followed
  4. return to home page.

  - Expected Result: Following user's tweet is shown after step 2 and disappears after step 4
  - Pass/Fail Criteria: Pass if expected result is observed, else fail

- Load next:
  - Steps:

  1. Create 20 tweets
  2. Scroll down to the bottom
  3. click the load next button

  - Expected Result: More tweets is loaded
  - Pass/Fail Criteria: Pass if expected result is observed, else fail

- Logout:
  - Steps:

1. Click logout button in the navigation bar
2. click to logout

  - Expected Result: Redirect into logout page after clicking logout button, redirect into login page after logout
  - Pass/Fail Criteria: Pass if expected result is observed, else fail

1.2.2 Non-Functional Requirements

1.2.2.1 1. Performance Testing

1. Website Load Time

- Objective: Verify that the website loads within the acceptable time limit.
- Steps:
  1. Launch the website on a supported platform.
  2. Measure the time taken to load the website using Firefox Devtools/Chrome Lighhouse.
- Expected Result: The website loades within 1 second on both high-end devices and low-end devices.
- Pass/Fail Criteria: Pass if the load time is within the acceptable range; fail otherwise.

1.2.2.2 2. Usability Testing

1. Navigation Intuitiveness

- Objective: Verify that the website's menus and controls are intuitive for new users.
- Steps:
  1. Ask a new user to navigate through the main menu, inventory, and settings without prior instructions.
  2. Observe their interactions and note any confusion or errors.
- Expected Result: The user can navigate the menus and controls without difficulty.
- Pass/Fail Criteria: Pass if the user completes tasks without significant confusion; fail otherwise.

1.2.2.3 3. Reliability Testing

1. Save/Load Consistency

- Objective: Verify that the website saves and loads data reliably.
- Steps:
    1. Log out the website at various points.
    2. Log back in.
- Expected Result: The website and its posts are loaded correctly without data corruption.
- Pass/Fail Criteria: Pass if the save/load functionality works without issues; fail otherwise.

1.2.2.4 4. Compatibility Testing

1. Cross-Platform Compatibility

- Objective: Ensure the website functions correctly on all supported platforms (i.e. PC).
- Steps:
    1. Use the Website on PC.
    2. Test core features.
- Expected Result: The website performs consistently on PC.
- Pass/Fail Criteria: Pass if there are no issues; fail otherwise.

1.3 3. Resource AllocationK

1.3.1 1. Team Roles and Responsibilities

1.3.1.1 Key Team Members:

1. Test Lead/Manager:
    - Oversees the entire testing process.
    - Develops the test strategy and ensures alignment with project goals.
    - Assigns tasks to team members and monitors progress.
    - Communicates testing progress, risks, and results to stakeholders.
2. Test Engineers/Quality Assurance (QA) Testers:
    - Write, execute, and maintain test cases.
    - Perform functional, non-functional, and exploratory testing.
    - Log and track defects in the bug tracking system.
    - Collaborate with developers to verify bug fixes.

3. Developers (for Unit and Integration Testing):
    - Perform unit testing on individual components.
    - Collaborate with testers during integration testing to resolve issues.
    - Provide technical support for debugging and fixing defects.
4. UI/UX Designers:
    - Validate the user interface and user experience.
    - Ensure that the website meets accessibility and usability standards.
5. Development and operations (DevOps)/IT Support:
    - Set up and maintain the testing environments (e.g., servers, databases, tools).
    - Ensure continuous integration/continuous deployment (CI/CD) pipelines are functioning.
6. Product Owner/Business Analyst:
    - Provide clarification on requirements and acceptance criteria.
    - Validate that the website meets business objectives during User Acceptance Testing (UAT).

1.3.1.2 Resource Allocation Table:

Role	Name	Responsibilities	Time Commitment
Test Lead	Cheng Shui Kwong	Oversee testing, manage team, report progress.	4 Days
QA Tester	Cheng Shui Kwong	Execute test cases, log bugs, perform regression.	4 Days
Developer	Cheng Shui Kwong	Perform unit testing, assist with integration issues.	4 Days
UI/UX Designer	Cheng Shui Kwong	Validate UI/UX and accessibility.	4 Days
Product Owner	Cheng Shui Kwong	Validate requirements and acceptance criteria.	4 Days

1.3.2 2. Tools and Software

1.3.2.1 Testing Tools:

- Github, Firefox Devtools, Chrome Lighouse, IDE

1.3.3 3. Testing Environments

1.3.3.1 Environment Types:

1. Development Environment:
   - Used by developers for unit testing and initial debugging.
   - Includes incomplete or partially implemented features.
2. Testing/Staging Environment:
   - A replica of the production environment used for system, integration, and regression testing.
   - Includes all features and configurations required for testing.
3. Production Environment (for UAT):
   - Used for final validation during User Acceptance Testing.
   - Mimics the live environment to ensure the website is ready for release.

1.3.3.2 Environment Setup:

- Hardware Requirements:
  - Devices for compatibility testing (e.g., PC, consoles, mobile devices).
  - High-performance machines for performance and stress testing.
- Software Requirements:
  - website builds, test tools, and debugging utilities.
  - Access to databases, APIs, and other backend systems.

1.3.3.3 Example Environment Allocation Table:

Environment	Purpose	Tools/Resources	Owner
Development	Unit testing, debugging.	IDEs, debugging tools	Developers
Staging/Testing	System and regression testing.strategies	IDE	QA Team
Production (UAT)	User Acceptance Testing.	Final website build, production-like setup	Product Owner

1.3.4 4. Time Allocation

1.3.4.1 Effort Estimation:

- Test Planning and Preparation: 10–15% of the total testing effort.
- Test Case Execution: 50–60% of the total testing effort.
- Bug Reporting and Retesting: 20–25% of the total testing effort.
- Regression Testing: 10–15% of the total testing effort.

1.3.4.2 Time Allocation Table:

Activity	Effort (%)	Responsible Team Member(s)
Test Planning	10%	Test Lead
Test Case Creation	15%	QA Testers
Test Execution	50%	QA Testers, Automation Engineers
Bug Reporting/Retesting	20%	QA Testers, Developers
Regression Testing	15%	QA Testers, Automation Engineers

1.3.5 5. Budget Allocation

- It is $0.

1.4 4. Testing Approach

1.4.1 Types of Testing:

- Unit Testing: Validate individual components (e.g., followers, user bio, login/logout system).
- Integration Testing: Test interactions between different website modules (e.g., multiuser functionality, db module and inventory system), from developers' point of view.
- System Testing: Verify the website as a whole from the user's point of view, including UI, and performance.
- Regression Testing: Ensure that new updates or fixes do not introduce new bugs.
- User Acceptance Testing (UAT): Gather feedback from users to ensure the website meets expectations.

1.4.2 Methodologies:

- Manual testing for the website and user experience.
- Automated testing for repetitive tasks (e.g., regression testing).
- Exploratory testing to uncover edge cases and unexpected issues.

1.5 5. Timeline and Schedule

1.5.1 1. Waterfall Model

1.5.1.1 Timeline:

Phase	Duration	Activities
Test Planning & Preparation	May 8 - May 11	Create test plan, write test cases, set up environment.
Unit Testing	May 8 - May 11	Test individual modules.
Integration Testing	May 8 - May 11	Test interactions between modules.
System Testing	May 8 - May 11	Test the entire system for functionality and performance.
User Acceptance Testing	May 8 - May 11	Validate the system with end-users.
Bug Fixing & Regression	May 8 - May 11	Fix bugs, perform regression testing.
Final Validation & Release	May 8 - May 11	Conduct exhaustive testing for core features, prepare for deployment.

1.6 6. Risk Assessment and Mitigation

1.6.1 Potential Risks:

1. Delays in Development: Could impact the testing schedule.
    - Mitigation: Regular communication with the development team to adjust timelines as needed.
2. High Bug Volume: May overwhelm the testing team.
    - Mitigation: Prioritize critical bugs and allocate additional resources if necessary.
3. Insufficient Resources:

- Risk: Lack of testers, tools, or devices.
- Mitigation: Prioritize critical test cases and borrow resources from other teams if needed.

1.7 7. Success Criteria

- All critical and high-priority bugs are resolved.
- Website meets performance benchmarks on all supported platforms.
- Positive feedback from UAT participants.
- All test cases pass with no major issues.
- Website is approved for release by stakeholders.

1.8 8. Reporting Requirements

1.8.1 Documentation:

- Description of the test cases in the code repository or dedicated test repository.
- Test case execution reports.
- Bug reports with detailed descriptions, steps to reproduce, and screenshots/videos.
- Weekly progress updates to stakeholders.

1.8.2 Communication:

- Regular meetings with the development team to discuss testing progress and blockers.
- Final test summary report to stakeholders before release.

This test plan provides a structured approach to ensure the website is thoroughly tested and ready for release.