SOP - Test plan template

HackExpert Penetration Test

Test Plan

Document Change History

Version Number	**Date**	**Contributor**	**Description**
V1.0	Feb 27th 2022	Bastiji, Dave_555, Tyrael	What changes (additions and deletions) were made for this version?

**** Note to Document Author –** Red and blue text (with the exception of the title and document name above) in this document is directed at the template user to describe processes, build standards and help build the document from the template. All such red and blue text should be removed before submitting any formal documentation, including both draft and/or final, deliverables. ********

Updated Feb 27th, 2022

Table of Contents

1 Introduction 3

1.1 Scope 3

1.1.1 In Scope 3

1.1.2 Out of Scope 3

1.2 Quality Objective 3

1.2.1 Primary Objective 3

1.2.2 Secondary Objective 4

1.3 Roles and Responsibilities 4

1.3.1 Developer 4

1.3.2 Adopter 4

1.3.3 Testing Process Management Team 4

1.4 Assumptions for Test Execution 5

1.5 Constraints for Test Execution 5

1.6 Definitions 6

2 Test Methodology 6

2.1 Purpose 6

2.1.1 Overview 6

2.1.2 Usability Testing 6

2.1.3 Unit Testing (Multiple) 7

2.1.4 Iteration/Regression Testing 7

2.1.5 Final release Testing 7

2.1.6 Testing completeness Criteria 8

2.2 Test Levels 8

2.2.1 Build Tests 8

2.2.1.1 Level 1 - Build Acceptance Tests 8

2.2.1.2 Level 2 - Smoke Tests 8

2.2.1.3 Level 2a - Bug Regression Testing 8

2.2.2 Milestone Tests 9

2.2.2.1 Level 3 - Critical Path Tests 9

2.2.3 Release Tests 9

2.2.3.1 Level 4 - Standard Tests 9

2.2.3.2 Level 5 - Suggested Test 9

2.3 Bug Regression 9

2.4 Bug Triage 9

2.5 Suspension Criteria and Resumption Requirements 10

2.6 Test Completeness 10

2.6.1 Standard Conditions: 10

2.6.2 Bug Reporting & Triage Conditions: 10

3 Test Deliverables 11

3.1 Deliverables Matrix 11

3.2 Documents 12

3.2.1 Test Approach Document 12

3.2.2 Test Plan 12

3.2.3 Test Schedule 13

3.2.4 Test Specifications 13

3.2.5 Requirements Traceability Matrix 13

3.3 Defect Tracking & Debugging 13

3.3.1 Testing Workflow 13

3.3.2 Defect reporting using G FORGE 14

3.4 Reports 16

3.4.1 Testing status reports 16

3.4.2 Phase Completion Reports 16

3.4.3 Test Final Report - Sign-Off 16

3.5 Responsibility Matrix 16

4 Resource & Environment Needs 16

4.1 Testing Tools 16

4.1.1 Tracking Tools 16

4.1.1.1 Configuration Management 17

4.2 Test Environment 17

4.2.1 Hardware 17

4.2.2 Software 17

4.3 Bug Severity and Priority Definition 17

4.3.1 Severity List 17

4.3.2 Priority List 18

4.4 Bug Reporting 18

5 Terms/Acronyms 18

1. Introduction

This document describes in detail the Penetration Test exercise being conducted for HackExpert LLC. We detail in this document the various aspects and details of the Penetration Test, being conducted by Team-4 LLC. Team-4 will conduct tests against resources of HackExpert LLC, as described in the scope, and present a report with the findings at the end of the exercise.

1. Scope

Describe the current test approach scope based on your role and project objectives.

1. In Scope

- For this Pen test, Team-4 will only look at testing these two URLs for any deficiencies, weaknesses and vulnerabilities:
  - https://hackexpert.com/cheesebook/
  - https://hackexpert.com/pentest
  - Team-4 will not test any other URLs or URIs for the http://hackexpert.com domain

1. Out of Scope

- Any other subdomains, URIs, network resources or affiliated assets, domains, or resources not explicitly determined by the client (Hackexpert) are considered out of scope for this pentest

1. **Quality Objective**

a. Primary Objective

Team-4's primary objective of testing the aforementioned URLs is to ***identify and expose all issues and associated risks, communicate all known issues to the HackExpert's team, and ensure that all issues are addressed in an appropriate manner before release.*** A comprehensive report including all the findings will be shared with the HackExpert's team at the end of the exercise

1. Roles and Responsibilities

Roles and responsibilities may differ based on the actual SOW. Below listed functions are for testing phase.

1. Developer

An NCI-designated Cancer Center selected and funded by NCICB to participate in a specific Workspace to undertake software or solution development activities. Responsible to:

(a) Develop the system/application

(b) Develop Use cases and requirements in collaboration with the Adopters

(c) Conduct Unit, system, regression and integration testing

(d) Support user acceptance testing

1. Adopter

An NCI-designated Cancer Center selected and funded by NCICB to undertake formal adoption, testing, validation, and application of products or solutions developed by Workspace Developers. Responsible to:

(a) Contribute to Use case, requirement development through review

(b) Contribute to develop and execution of the development test scripts through review

(c) Conduct Full User Acceptance, regression, and end-to-end testing; this includes identifying testing scenarios, building the test scripts, executing scripts and reporting test results

1. Testing Process Management Team

Include NCI, BAH and Cancer Center Leads allocated to the <workspace name>. Group responsible to manage the entire testing process, workflow and quality management with activities and responsibilities to:

(a) Monitor and manage testing integrity and Support testing activities

(b) Coordinate activities across cancer centers

Add more as appropriate to testing scope

1. **Assumptions for Test Execution**

Below are some minimum assumptions (in black) that has be completed with some examples (in red). Any example may be used if deemed appropriate for the particular project. New assumptions may also be added that are reasoned to be suitable to the project.

- For User Acceptance testing, the Developer team has completed unit, system and integration testing and met all the Requirement's (including quality requirements) based on Requirement Traceability Matrix.
- User Acceptance testing will be conducted by End-users
- Test results will be reported on daily basis using Gforge. Failed scripts and defect list from Gforge with evidence will be sent to Developer directly.
- Use cases have been developed by Adopters for User Acceptance testing. Use cases are approved by test lead.
- Test scripts are developed and approved.
- Test Team will support and provide appropriate guidance to Adopters and Developers to conduct testing
- Major dependencies should be reported immediately after the testing kickoff meeting.

1. **Constraints for Test Execution**

Below are some minimum assumptions (in black) followed by example constraints (red). Any example may be used if deemed appropriate for the particular project. New constraints may also be added that are reasoned to be suitable to the project.

- Adopters should clearly understand on test procedures and recording a defect or enhancement. Testing Process Management Team will schedule a teleconference with Developers and Adopters to train and address any testing related issues.
- Developer will receive consolidated list of request for test environment set up, user accounts set up, data set (actual and mock data), defect list, etc. through **GForge** after the initial Adopter testing kick off meeting.
- Developer will support ongoing testing activities based on priorities
- Test scripts must be approved by Test Lead prior test execution
- Test scripts, test environment and dependencies should be addressed during testing kickoff meeting in presence of a SME and request list should be submitted within 3 days of the kickoff meeting
- The Developer cannot execute the User Acceptance and End to End test scripts. After debugging, the developer can conduct their internal test, but no results from that test can be recorded / reported.
- Adopters are responsible to identify dependencies between test scripts and submit clear request to set up test environment

1. **Definitions**

Bugs: Any error or defect that cause the software/application or hardware to malfunction. That is also included in the requirements and does not meet the required workflow, process or function point.

Enhancement:

1. Any alteration or modification to the existing system for better workflow and process.

1. An error or defect that causes the software/application or hardware to malfunction.

Where 1) and 2) is NOT included in the requirements can be categorized as an enhancement.

Enhancement can be added as a new requirement after appropriate Change Management process.

1. **Test Methodology**
   a. **Purpose**
      i. Overview

The below list is not intended to limit the extent of the test plan and can be modified to become suitable for the particular project.

The purpose of the Test Plan is to achieve the following:

1. Define testing strategies for each area and sub-area to include all the functional and quality (non-functional) requirements.
2. Divide Design Spec into testable areas and sub-areas (do not confuse with more detailed test spec). Be sure to also identify and include areas that are to be omitted (not tested) also.
3. Define bug-tracking procedures.
4. Identify testing risks.
5. Identify required resources and related information.
6. Provide testing Schedule.

1. Usability Testing

The purpose of usability testing is to ensure that the new components and features will function in a manner that is acceptable to the customer.

Development will typically create a non-functioning prototype of the UI components to evaluate the proposed design. Usability testing can be coordinated by testing, but actual testing must be performed by non-testers (**as close to end-users as possible).** Testing will review the findings and provide the project team with its evaluation of the impact these changes will have on the testing process and to the project as a whole.

1. Unit Testing (Multiple)

Unit Testing is conducted by the Developer during code development process to ensure that proper functionality and code coverage have been achieved by each developer both during coding and in preparation for acceptance into iterations testing.

The following are the example areas of the project must be unit-tested and signed-off before being passed on to regression Testing:

1. Databases, Stored Procedures, Triggers, Tables, and Indexes
2. NT Services
3. Database conversion
4. .OCX, .DLL, .EXE and other binary formatted executables

1. Iteration/Regression Testing

During the repeated cycles of identifying bugs and taking receipt of new builds (containing bug fix code changes), there are several processes which are common to this phase across all projects. These include the various types of tests: functionality, performance, stress, configuration, etc. There is also the process of communicating results from testing and ensuring that new drops/iterations contain stable fixes (regression). The project should plan for a minimum of 2-3 cycles of testing (drops/iterations of new builds).

At each iteration, a debriefing should be held. Specifically, the report must show that to the best degree achievable during the iteration testing phase, all identified severity 1 and severity 2 bugs have been communicated and addressed. At a minimum, all priority 1 and priority 2 bugs should be resolved prior to entering the beta phase.

Below are examples. Any example may be used if deemed appropriate for the particular project. New content may also be added that are reasoned to be suitable to the project.

Important deliverables required for acceptance into Final Release testing include:

1. Application SETUP.EXE
2. Installation instructions
3. All documentation (beta test scripts, manuals or training guides, etc.)

1. Final release Testing

Testing team with end-users participates in this milestone process as well by providing confirmation feedback on new issues uncovered, and input based on identical or similar issues detected earlier. The intention is to verify that the product is ready for distribution, acceptable to the customer and iron out potential operational issues.

Assuming critical bugs are resolved during previous iterations testing- Throughout the Final Release test cycle, bug fixes will be focused on minor and trivial bugs (severity 3 and 4). Testing will continue its process of verifying the stability of the application through regression testing (existing known bugs, as well as existing test cases).

The milestone target of this phase is to establish that the application under test has reached a level of stability, appropriate for its usage (number users, etc.), that it can be released to the end users and caBIG community.

1. Testing completeness Criteria

Release for production can occur only after the successful completion of the application under test throughout all of the phases and milestones previously discussed above.

The milestone target is to place the release/app (build) into production after it has been shown that the app has reached a level of stability that meets or exceeds the client expectations as defined in the Requirements, Functional Spec., and caBIG Production Standards.

1. **Test Levels**

Testing of an application can be broken down into three primary categories and several sub-levels. The three primary categories include tests conducted every build (Build Tests), tests conducted every major milestone (Milestone Tests), and tests conducted at least once every project release cycle (Release Tests). The test categories and test levels are defined below:

1. Build Tests
   a. ***Level 1 - Build Acceptance Tests***

Build Acceptance Tests should take less than 2-3 hours to complete (15 minutes is typical). These test cases simply ensure that the application can be built and installed successfully. Other related test cases ensure that adopters received the proper Development Release Document plus other build related information (drop point, etc.). The objective is to determine if further testing is possible. If any Level 1 test case fails, the build is returned to developers un-tested.

1. ***Level 2 - Smoke Tests***

Smoke Tests should be automated and take less than 2-3 hours (20 minutes is typical). These tests cases verify the major functionality a high level.

The objective is to determine if further testing is possible. These test cases should emphasize breadth more than depth. All components should be touched, and every major feature should be tested briefly by the Smoke Test. If any Level 2 test case fails, the build is returned to developers un-tested.

1. ***Level 2a - Bug Regression Testing***

Every bug that was “Open” during the previous build, but marked as “Fixed, Needs Re-Testing” for the current build under test, will need to be regressed, or re-tested. Once the smoke test is completed, all resolved bugs need to be regressed. It should take between 5 minutes to 1 hour to regress most bugs.

1. Milestone Tests
   a. ***Level 3 - Critical Path Tests***

Critical Path test cases are targeted on features and functionality that the user will see and use every day.

Critical Path test cases must pass by the end of every 2-3 Build Test Cycles. They do not need to be tested every drop, but must be tested at least once per milestone. Thus, the Critical Path test cases must all be executed at least once during the Iteration cycle, and once during the Final Release cycle.

1. Release Tests
   a. ***Level 4 - Standard Tests***

Test Cases that need to be run at least once during the entire test cycle for this release. These cases are run once, not repeated as are the test cases in previous levels. Functional Testing and Detailed Design Testing (Functional Spec and Design Spec Test Cases, respectively). These can be tested multiple times for each Milestone Test Cycle (Iteration, Final Release, etc.).

Standard test cases usually include Installation, Data, GUI, and other test areas.

1. ***Level 5 - Suggested Test***

These are Test Cases that would be nice to execute, but may be omitted due to time constraints.

Most Performance and Stress Test Cases are classic examples of Suggested test cases (although some should be considered standard test cases). Other examples of suggested test cases include WAN, LAN, Network, and Load testing.

1. Bug Regression

Bug Regression will be a central tenant throughout all testing phases.

All bugs that are resolved as “Fixed, Needs Re-Testing” will be regressed when Testing team is notified of the new drop containing the fixes. When a bug passes regression it will be considered “Closed, Fixed”. If a bug fails regression, adopters testing team will notify development team by entering notes into GForge. **When a Severity 1 bug fails regression, adopters Testing team should also put out an immediate email to development.** The Test Lead will be responsible for tracking and reporting to development and product management the status of regression testing.

1. Bug Triage

Bug Triages will be held throughout all phases of the development cycle. Bug triages will be the responsibility of the Test Lead. Triages will be held on a regular basis with the time frame being determined by the bug find rate and project schedules.

Thus, it would be typical to hold few triages during the **Planning phase**, then maybe one triage per week during the **Design phase**, ramping up to twice per week during the latter **stages of the Development phase**. Then, the Stabilization phase should see a substantial reduction in the number of new bugs found, thus a few triages per week would be the maximum (to deal with status on existing bugs).

The Test Lead, Product Manager, and Development Lead should all be involved in these triage meetings. The Test Lead will provide required documentation and reports on bugs for all attendees. The purpose of the triage is to determine the type of resolution for each bug and to prioritize and determine a schedule for all “To Be Fixed Bugs’. Development will then assign the bugs to the appropriate person for fixing and report the resolution of each bug back into the GForge bug tracker system. The Test Lead will be responsible for tracking and reporting on the status of all bug resolutions.

1. Suspension Criteria and Resumption Requirements

This section should be defined to list criteria’s and resumption requirements should certain degree and pre-defined levels of test objectives and goals are not met.

Please see example below:

- Testing will be suspended on the affected software module when Smoke Test (Level 1) or Critical Path (Level 2) test case bugs are discovered after the 3

rd iteration.

- Testing will be suspended if there is critical scope change that impacts the Critical Path

A bug report should be filed by Development team. After fixing the bug, Development team will follow the drop criteria (described above) to provide its latest drop for additional Testing. At that time, adopters will regress the bug, and if passes, continue testing the module.

1. Test Completeness

Testing will be considered complete when the following conditions have been met:

1. Standard Conditions:

1. When Adopters and Developers, agree that testing is complete, the app is stable, and agree that the application meets functional requirements.
2. Script execution of all test cases in all areas have passed.
3. Automated test cases have in all areas have passed.

4. All priority 1 and 2 bugs have been resolved and closed.
5. NCI approves the test completion
6. Each test area has been signed off as completed by the Test Lead.
7. 50% of all resolved severity 1 and 2 bugs have been successfully re-regressed as final validation.
8. Ad hoc testing in all areas has been completed.

1. Bug Reporting & Triage Conditions:

Please add Bug reporting and triage conditions that will be submitted and evaluated to measure the current status.

1. Bug find rate indicates a decreasing trend prior to Zero Bug Rate (no new Sev. 1/2/3 bugs found).
2. Bug find rate remains at 0 new bugs found (Severity 1/2/3) despite a constant test effort across 3 or more days.
3. Bug severity distribution has changed to a steady decrease in Sev. 1 and 2 bugs discovered.
4. No ‘Must Fix’ bugs remaining prior despite sustained testing.

1. **Test Deliverables**

Testing will provide specific deliverables during the project. These deliverables fall into three basic categories: Documents, Test Cases / Bug Write-ups, and Reports. Here is a diagram indicating the dependencies of the various deliverables:

As the diagram above shows, there is a progression from one deliverable to the next. Each deliverable has its own dependencies, without which it is not possible to fully complete the deliverable.

The following page contains a matrix depicting all of the deliverables that Testing will use.

1. **Deliverables Matrix**

The team will provide the client with the following deliverables:

Deliverable
Documents
Test Approach
Test Plan
• Test schedule
Test Specifications
Test Case / Bug Write-Ups
Test Cases / Results
Test Coverage Reports
GForge Bug tracker for bug reporting
Reports
Test results report
Test Final Report - Sign-Off

1. **Documents**
   a. Test Approach Document

The Test Approach document is derived from the Project Plan, Requirements and Functional Specification documents. This document defines the overall test approach to be taken for the project. The Standard Test Approach document that you are currently reading is a boilerplate from which the more specific project Test Approach document can be extracted.

When this document is completed, the Test Lead will distribute it to the Product Manager, Development Lead, User Representative, Program Manager, and others as needed for review and sign-off.

1. Test Plan

We are planning to perform a series of manual and automatic tests. First an automated test will be performed with specialized pentesting software, that would offer us an starting point to determine if the URLs in-scope follow the most common security practices.

The publicly exposed resources of the website will be tested in a different manner than a any resource behind a login, in the following manner:

Protected resources:

- ACL to ensure the user is limited to the activities that correspond to their role.
- CRUD to identify possible security problems that can affect the validity and integrity of data.
- Code executed in the client machines

Public resources:

- Ensure the correct access and restrictions for non authenticated users
- Audit the code that is executed in the client.

All resources will be tested for:

Complying with standard security practices, within it's functionality and any product exposed by the website interface (Databases, libraries or packages etc…) and are found within scope

The used tools will be the following:

Burp Suite Community Edition v2022.1.1

Member teams:

dave_555 (Pentester)

Bastiji (Pentester)

Tyrael (Pentester)

Risks:

No pentesting is completely safe from triggering functionality that might cause an unexpected malfunction, therefore we identify the following risks:

- Interaction of the automatic or manual tests with the products that are essential for the correct functioning of the website:
  - Web Engines
  - Databases
  - Libraries
  - Any other product or package that can be found within the scope URLs

Mitigation of the risks:

The team is committed to reduce risk or limit the possible impact, therefore the following practices will be mandatory:

- Limit any automatic or manual testing that can interfere with the usual performance, if needed it will be announced to the client.
- Limit any action that can disable the website or any of the products necessary for it's correct functioning.

Any vulnerability will be documented and reported to the client. The team will work the client to resolve it.

?

1. Specify criteria for acceptance of development drops to testing (of builds).

1. Test Schedule

Week 1: The functionality will be tested

Week 2: The team will document their findings and write a report for the client

Week 3: The team will support the client's developer team and ensure that all findings are solved

1. Test Specifications

A Test Specification document is derived from the Test Plan as well as the Requirements, Functional Spec., and Design Spec documents. It provides specifications for the construction of Test Cases and includes list(s) of test case areas and test objectives for each of the components to be tested as identified in the project's Test Plan.

- Unauthenticated users can't access, read, modify or delete any information from the website.
- Authenticated users can't access, read, modify or delete any information not explicitly assigned by the website administrator and/or developer
- The interaction of any user doesn't affect the performance or functionality of the website.

1. Requirements Traceability Matrix

A Requirements Traceability Matrix (RTM) which is used to link the test scenarios to the requirements and use cases is a required part of the Test Plan documentation for all projects. Requirements traceability is defined as the ability to describe and follow the life of a requirement, in both a forward and backward direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases).

Attached is a sample basic RTM which could provide a starting point for this documentation. The important thing is to choose a template or document basis that achieves thorough traceability throughout the life of the project.

1. **Defect Tracking & Debugging**
   a. Testing Workflow

The below workflow illustrates the testing workflow process for Developers and Adopters for User Acceptance and End to End testing.

Pl. note the yellow highlighted process where the Adopter is required to directly send defect list with evidence to the Developer. Similarly, Developer is required to confirm directly with the Adopter after bug fixes along with updating on the Bugzilla.

1. Defect reporting using G FORGE

ALL defects should be logged using 'G FORGE', to address and debug defects. Adopters are also requested to send a daily defect report to the developer. Developers will update the defect list on G Forge and notify the requestor after the defect has been resolved.

Developers and Adopters are required to request an account on G Forge for the relative workspace. Debugging should be based on Priority – High > Medium > Low, these priorities are set by the Adopters and are based on how critical is the test script in terms of dependency and mainly based on use case scenario.

Below screen shot displays 'Add new Defect' screen, fields marked with (*) are mandatory fields and Adopters should also upload the evidence file for all the defects listed.

All **High priority** defects should be addressed within 1 day of the request and resolved/closed within 2 days of the initial request

All **Medium priority** defects should be addressed within 2 days of the request and resolved/closed within 4 days of the initial request

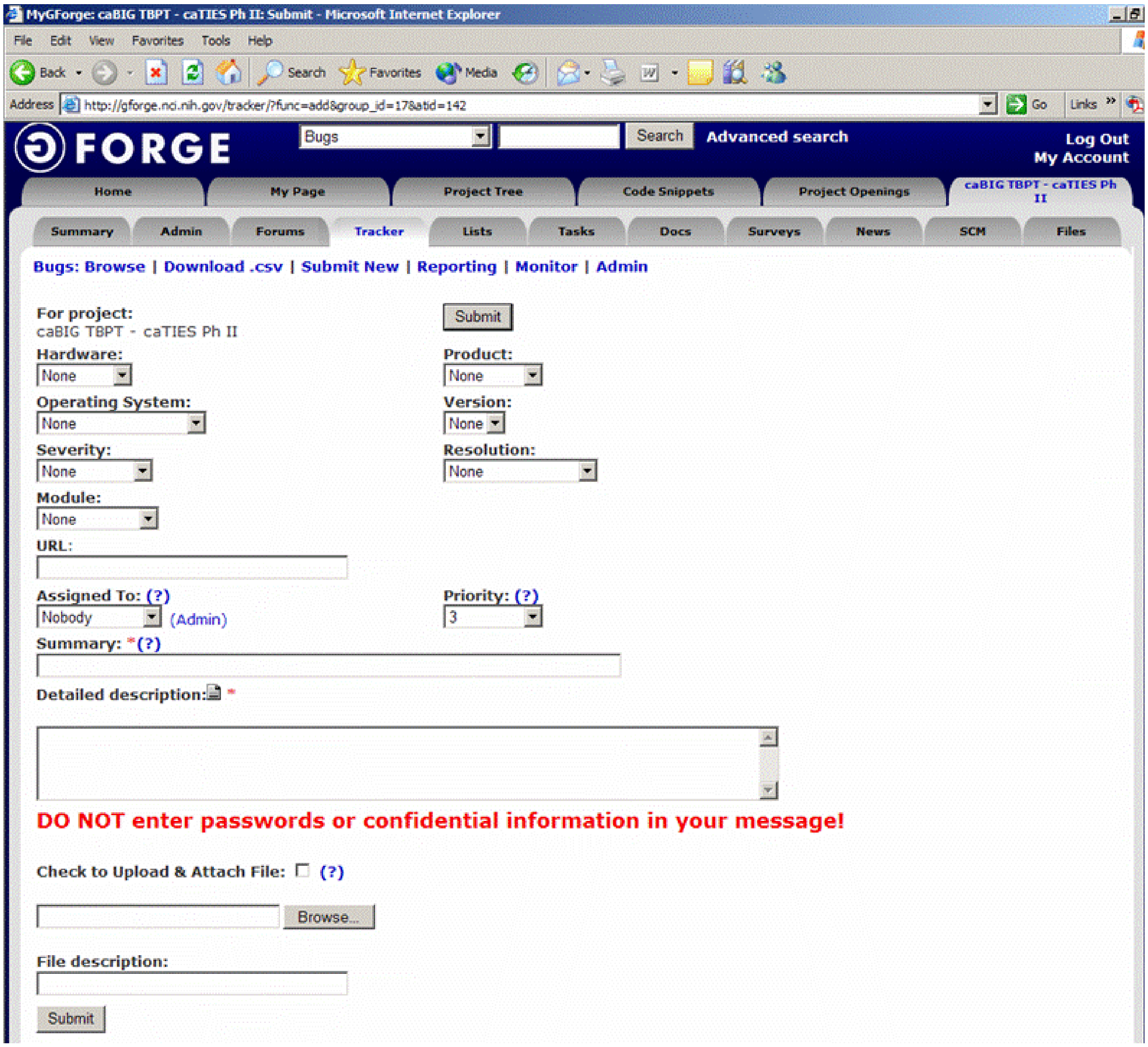
All **Low priority** defects should be resolved/closed no later than 5 days of the initial request.

G Forge URL - http://gforge.nci.nih.gov

User may either search for workspace or select from list of recent project from the bottom right side of the window. E.g. searching for 'caties'.

At the workspace, the user can request Administrators to setup their user account for that workspace.

After login, user can select 'Tracker' tab to 'Submit New' defect. User can add defect info. As shown in below screen.


MyGForge: caBIG TBPT - caTIES Ph II: Submit - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://gforge.nci.nih.gov/tracker/?func=add&group_id=17&atid=142
Go Links
FORGE
Bugs
Search
Advanced search
Log Out
My Account
Home My Page Project Tree Code Snippets Project Openings caBIG TBPT - caTIES Ph II
Summary Admin Forums Tracker Lists Tasks Docs Surveys News SCM Files
Bugs: Browse | Download .csv | Submit New | Reporting | Monitor | Admin
Submit
Browse...
Submit

For project:
caBIG TBPT - caTIES Ph II

Hardware: None

Product: None

Operating System: None

Version: None

Severity: None

Resolution: None

Module: None

URL:

Assigned To: (?) Nobody (Admin)

Priority: (?) 3

Summary: * (?)

Detailed description: *

DO NOT enter passwords or confidential information in your message!

Check to Upload & Attach File: ☐ (?)

File description:

Field Comments

Hardware - List the testing environment hardware. e.g. PC, MAC

Product - Add default to workspace name e.g. caTIES

Operating System - Win 98, Win 2k, Win XP, MAC, etc.

Version - Application version

Severity - If it is an enhancement or a bug with minor to major severity that may not interfere with further testing or completely block any future testing efforts.

Resolution - Only DEVELOPER will update based on the defect

Module - For an application with multiple modules, select appropriate module for the defect reported.

URL -   Add URL if any

Assigned to - To be updated by Developer, for whom the ticket is assigned to

Priority - Set a priority based on severity

Summary - Summary of the defect, bug or issue

Detailed Desc. - Details of the defect, bug or issue

Upload file - Attach evidence file

Submit - Submit the bug ticket

1. **Reports**

The Test Lead will be responsible for writing and disseminating the following reports to appropriate project personnel as required.

1. Testing status reports

A weekly or bi-weekly status report will be provided by the Test Lead to project personnel. This report will summarize weekly testing activities, issues, risks, bug counts, test case coverage, and other relevant metrics.

1. Phase Completion Reports

When each phase of testing is completed, the Test Lead will distribute a Phase Completion Report to the Product manager, Development Lead, and Program Manager for review and sign-off.

The below bullets illustrates an example of what the document may include.

The document must contain the following metrics:

1. Total Test Cases, Number Executed, Number Passes / Fails, Number Yet to Execute
2. Number of Bugs Found to Date, Number Resolved, and Number still Open
3. Breakdown of Bugs by Severity / Priority Matrix
4. Discussion of Unresolved Risks
5. Discussion of Schedule Progress (are we where we are supposed to be?)

1. Test Final Report - Sign-Off

A Final Test Report will be issued by the Test Lead. It will certify as to the extent to which testing has actually completed (test case coverage report suggested), and an assessment of the product's readiness for Release to Production.

1. **Responsibility Matrix**

Insert testing resource matrix – who is involved and what is the role.

1. **Resource & Environment Needs**

1. **Testing Tools**
    a. Tracking Tools

GForge bug tracker is used by caBIG to enter and track all bugs and project issues. The Test Lead is responsible for maintaining the GForge database.

1. ***Configuration Management***

Please include your configuration management plan

1. **Test Environment**

a. Hardware

Include the minimum hardware requirements that will be used to test the Application.

Testing will have access control to one or more application/database servers separate from any used by non-test members of the project team. Testing will also have access control to an adequate number of variously configured PC workstations to assure testing a range from the minimum to the recommended client hardware configurations listed in the project's Requirements, Functional Specification and Design Specification documents.

1. Software

In addition to the application and any other customer specified software, the following list of software should be considered a minimum:

1. NT Workstation 4.0 or Windows 95.
2. MS Office 95 Professional
3. MS Exchange
4. TCM (Testing Tool Server)
5. Task Manager (Testing Tool Server)

1. Bug Severity and Priority Definition

Bug Severity and Priority fields are both very important for categorizing bugs and prioritizing if and when the bugs will be fixed. The bug Severity and Priority levels will be defined as outlined in the following tables below. Testing will assign a severity level to all bugs. The Test Lead will be responsible to see that a correct severity level is assigned to each bug.

The Test Lead, Development Lead and Program Manager will participate in bug review meetings to assign the priority of all currently active bugs. This meeting will be known as "Bug Triage Meetings". The Test Lead is responsible for setting up these meetings on a routine basis to address the current set of new and existing but unresolved bugs.

1. Severity List

The tester entering a bug into GForge is also responsible for entering the bug Severity.

Severity ID	**Severity Level**	**Severity Description**
1	Critical	The module/product crashes or the bug causes non-recoverable conditions. System crashes, GP Faults, or database or file corruption, or potential data loss, program hangs requiring reboot are all examples of a Sev. 1 bug.
2	High	Major system component unusable due to failure or incorrect functionality. Sev. 2 bugs cause serious problems such as a lack of functionality, or insufficient or unclear error messages that can have a major impact to the user, prevents other areas of the app from being tested, etc. Sev. 2 bugs can have a work around, but the work around is inconvenient or difficult.
3	Medium	Incorrect functionality of component or process. There is a simple work around for the bug if it is Sev. 3.
4	Minor	Documentation errors or signed off severity 3 bugs.

1. Priority List

Priority ID	**Priority Level**	**Priority Description**
5	Must Fix	This bug must be fixed immediately; the product cannot ship with this bug.

4	Should Fix	These are important problems that should be fixed as soon as possible. It would be an embarrassment to the company if this bug shipped.
3	Fix When Have Time	The problem should be fixed within the time available. If the bug does not delay shipping date, then fix it.
2	Low Priority	It is not important (at this time) that these bugs be addressed. Fix these bugs after all other bugs have been fixed.
1	Trivial	Enhancements/ Good to have features incorporated- just are out of the current scope.

1. Bug Reporting

Testing team recognizes that the bug reporting process is a critical communication tool within the testing process. Without effective communication of bug information and other issues, the development and release process will be negatively impacted.

The Test Lead will be responsible for managing the bug reporting process. Testing's standard bug reporting tools and processes will be used. GForge is the company-wide standard Bug Logging / Tracking tool. Testing and development will enter their data into the GForge database following the field entry definitions defined below.

1. Terms/Acronyms

The below terms are used as examples, please add/remove any terms relevant to the document.

TERM/ACRONYM	**DEFINITION**
API	Application Program Interface
BAH	Booz Allen Hamilton
BCP	Business Continuity Plan
CAT	Client Acceptance Testing
End-to End Testing	Tests user scenarios and various path conditions by verifying that the system runs and performs tasks accurately with the same set of data from beginning to end, as intended.
ER;ES	Electronic Records; Electronic Signatures
N/A	Not Applicable
NCI	National Cancer Institute
QA	Quality Assurance
RTM	Requirements Traceability Matrix
SME	Subject Matter Expert
SOP	Standard Operating Procedure
TBPT	Tissue Bank and Pathology Tools
TBD	To Be Determined
TSR	Test Summary Report