## INTRODUCTION

## Purpose

This test plan describes the testing approach that will follow the testing of the TrackMe! application. The document introduces:

* Test Strategy: rules the test will rely on (e.g.: objectives, assumptions, notes and motivations); description of the process to set up a valid test (e.g.: creation of functional test cases, definition of the type of the tests and their main structure) as well as how they will be performed. Few lines will describe how the data will be managed in the test phase.
* Execution Strategy: describes how each test summary is organized, the processes to identify and report defects, and how to deal with the success or unsuccess of each test case. It is then described a quick system to assess the quality of the project which can give also a comprehensive view of how the testing phase is going.
* Implementation ???
* Integration ???

# KEY TERM DEFINITIONS

* UAT: User acceptance testing (UAT) is the last phase of the software testing process. During UAT, actual software users test the software to make sure it can handle required tasks in real-world scenarios, according to specifications. UAT is also known as beta-testing.
* Exploratory testing: a style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the quality of his/her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project. [Cem Kaner]
* Performance Testing: in general, it is a [testing](https://en.wikipedia.org/wiki/Software_testing) practice performed to determine how a [system](https://en.wikipedia.org/wiki/System) performs in terms of responsiveness and stability under a particular workload.
* Functional testing: a type of black-box testing that bases its test cases on the specifications of the software component under test.
* Test harness: An artificial environment is necessary for each integration test; the environment consists of driver programs and test data.

# TEST STRATEGY

## Test Objectives

The objective of the test is to verify that the functionality of TrackMe! works according to the specifications.

The test will execute and verify the test cases, with the outcome of each one notified to the development team right away.

The final product of the test is twofold:

* A production-ready software;
* A set of test results to provide quality assurance of the product.

## Test Assumptions

**General**

* Exploratory Testing would be carried out on each module as soon as they are built correctly.
* A number of mocks will come along the modules, in order to follow a top-down testing approach.
* Performance testing is not considered for this estimation.
* All the defects would come along with a detailed format specified in the paragraph related to Test Execution.
* The Test Team assumes that all necessary inputs required during Test design and execution will be supported by Development Team/Ambulance Dispatcher Service appropriately.
* There will be cooperation between the Dev Team and the Test Team, during the execution of the test cases. Detailed instruction will be given on how to report errors during beta-test, as a separate case.
* The testing environment will be always online. (Downtime of the server will allegedly create problems like: Is the functionality not available because of errors in the integration, or the testing environment missed something itself?)
* The system will be treated as a black box; if the information shows correctly online and in the reports, it will be assumed that the database is working properly.

**Functional Testing**

No particular explanation needed here.

**UAT**

* UAT test execution will be performed by end users (U1, U2 and U3) who will be supported by the Dev Team.

## Test Principles

* Testing activities will be publicly tracked and visible (restricted to the Dev, Test Teams and UAT obviously) to avoid redundancy or duplication of effort. Even non-completed test must be tracked.
* Testing environment and data will emulate a production environment as much as possible.
* There will be entrance and exit criteria, namely Pre and Post-Conditions.
* Testing will be a repeatable, quantifiable, and measurable activity.
* The entry criteria refer to the desirable conditions in order to start test execution; only the migration of the code and fixes need to be assessed at the end of each cycle.
* The exit criteria are the desirable conditions that need to be met in order proceed with the implementation.
* Entry and exit criteria are flexible benchmarks. If they are not met, the test team will assess the risk, identify mitigation actions and provide a recommendation. All this is input to the project manager for a final “go-no go” decision.
* Test environment with application installed, configured and ready to use
* The order to test may vary, although it will run paired with the development phase.
* Testing will be focused on meeting the security objectives and quality of service.
* Testing processes will be well defined, yet flexible, with the ability to change as needed.
* Testing will be performed by a top-down approach

## Motivations

It follows a brief description of the strategies adopted and why we have made this choice.

Bottom up integration testing proceeds with the test of each individual lowest level modules first. Lowest modules are combined to form subsystems, the subsystems are tested, and so on.

This approach has several disadvantages, of which:

* One must code and debug test harnesses (extra work).
* There can be difficulty in combining subsystems and then testing them. A critical example: all modules united tested, then combined together (big bang integration testing).

Top-down (which will be used)

Modules at top of structure chart are tested first, starting with the main or control modules, with reduced functionality.

NOTE: For related modules not yet written, it is necessary to use stubs (simple dummy modules used to avoid linker errors).

## Data Approach

* In functional testing, TrackMe! will contain pre-loaded test data and which is used for testing activities. For the other tests it is not needed to have pre-loaded data, since they will be generated (UAT) and they are not needed in the Exploratory testing.

## Scope and Levels of Testing

### Exploratory Test

**PURPOSE**: the purpose of this test is to make sure critical defects are removed before the next levels of testing can start.

**TESTERS**: Testing team.

**METHOD**: this exploratory testing is carried out in the application without any test scripts and documentation

**TIMING**: at the beginning of the cycle of test.

### Functional Test

**PURPOSE:**  Functional testing will be performed to check the functions of application. The functional testing is carried out by feeding the input and validates the output from the application.

**TESTERS**: Testing Team.

**METHOD**: Each case is evaluated in a table exemplified in 4.1.1.

**TIMING**: after Exploratory test is completed.

Please note that since the team is composed by only 2 developers, the approach used before is scalable, for example if the project require more team to cooperate.

### User Acceptance Test (UAT)

**PURPOSE**: this test focuses on validating the system logic. It allows the end users to complete one final review of the system prior to deployment.

**TESTERS**: the UAT is performed by the end users (L1, L2 and L3).

**METHOD**: Simple use of the application. It is however asked to the user to try to use the application with some uncommon pattern.

**TIMING**: After all other levels of testing (Exploratory and Functional) are done. Only after this test is completed the product can be released to production.

## Details about the scope of the test of each kind

### Exploratory Test:

**SCOPE**: First level navigation, check on trivial errors.

### Functional Test

**SCOPE:**

MULTITHREADING SAMPLE CHECK

* + - One by one request of data to one single user by one single third party
    - Multiple requests of data to one single user by one single third party
    - Multiple requests of data to one single user by many third party
    - Multiple requests of data to many user by many third party
    - Multiple tracking of many users concurrent to many requests to many users by many third party.
    - All the functionality running simultaneously.

PRIVACY CHECK

* + - Requests to user violating some of the privacy policy of the system

REACTION TIME GUARANTEE TEST

* + - Each time parameters drop below the threshold the alarm is sent to the server

### User Acceptance Test (UAT)

**SCOPE**: Users are given free-choice on how to use test the application

# EXECUTION STRATEGY

## Test Case summary

Tests will be outlined as follows:

|  |  |
| --- | --- |
| **Pre-Conditions** | These conditions should be met at the start of the test. |
| **Actions** | These are the steps that are to be performed by the tester |
| **Post-Conditions** | These are the conditions that should be met after the test |
| **Outcome** | Success or failure (severity failure to be specified according to 4.1.2.2. table) |

## Transformation of critical points

As specified before the critical points described in the scope section will be translated into Test Case summary, before the test phase starts, and signed together with the Developer team.

## Reporting

Functional testers will keep an excel spread sheet marked with the Pre-Conditions of each test, followed by a row for each step required in the Actions section. The testers will fill out every row of the spread sheet with the outcome of each step. Additionally, in the outcome section it should be defined even if the preconditions don’t apply.

#### Success

Upon success the test, as defined in Post-Conditions, the Developer shall provide a quick one to two sentence descriptions of the state of the program.

#### Unsuccess

If the test should fail, meaning it does not meet the Post-Conditions of the test, then the tester should mark the step at which the test failed, and provide a detailed description of the results. In addition he/she should mark the severity according to the table below. Screenshots should be taken of the failure, and placed in a zip file with the excel spread sheet for the test. This zip file is to be named with the number of the test and the date and time of execution. An issue, specifically a bug report should then be opened. It should be filed with a description of the problem, steps to recreate the problem, and the zip file with appropriate screenshots and test results. High precision is needed here so, every step must be clear and repeatable.

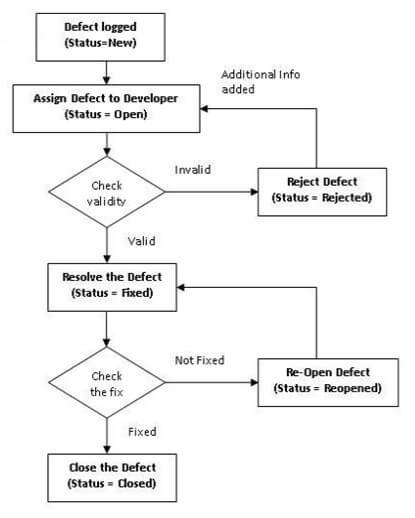
|  |  |
| --- | --- |
| **Severity** | **Impact** |
| 1 (Critical) | * This bug is critical enough to crash the system, cause file corruption, or cause potential data loss * It causes an abnormal return to the operating system (crash or a system failure message appears). * It causes the application to hang and requires re-booting the system. |
| 2 (High) | * It causes a lack of vital program functionality with workaround. |
| 3 (Medium) | * This Bug will degrade the quality of the System. However there is an intelligent workaround for achieving the desired functionality - for example through another screen. * This bug prevents other areas of the product from being tested. However other areas can be independently tested. |
| 4 (Low) | * There is an insufficient or unclear error message, which has minimum impact on product use. |
| 5(Cosmetic) | * There is an insufficient or unclear error message that has no impact on product use. |

#### Unsuccess noticed by the users in beta-test

As soon as bugs are noticed they must write few lines where they specify the error occurred and the error code returned. In addition to the JPEG screenshot as previously mentioned, it is required to specify in a detailed way which steps brought them to error/bug, with a sufficient precision.

## Defect tracking & Reporting

Following flowchart depicts Defect Tracking Process:



## Test Metrics

Test metrics are to measure the progress and level of success of the test executed through development will be produced and shared with the project manager for approval. The following are some of the metrics.

|  |  |  |
| --- | --- | --- |
| Report | Description | Frequency |
| Execution Status | To report on % complete, % Pass, % Fail | Weekly / Daily |
| Code completion | To report on % of class missing, % of methods | Daily |
| Status of the project | Statistics of the number of failed tests each of its kind (high severity, medium severity, …) | Weekly / Monthly |

# TEST ENVIRONMENT

TrackMe! server application will be hosted at the company’s site.   
TrackMe! DB application chosen is MySQL secured with password and listening on port 3306 on the same machine where the application server is running.  
TrackMe! clients will be hosted on three devices: One to host the business party, and the others to host the customer-side user application.

A Windows environment with Internet Explorer 8, 9 and 10, and with Firefox 27.0, as well as Google Chrome 32.0 and later should be available to each tester.