Technical Assistance for "A Smart Network for Technology Transfer and Commercialisation with Funnel Model (SMARTNET)"

Contract No: TR14C2.2.05-04/001

EUROPEAID/140284/IH/SER/TR

TEST SPECIFICATIONS (TS) DOCUMENT

15.05.2023

REPUBLIC OF TURKEY
MINISTRY OF INDUSTRY
AND TECHNOLOGY

YTÜ

İTÜ

GEBZE

COMPETITIVE
SECTORS
PROGRAMME

Technical Assistance for “A Smart Network for Technology Transfer and Commercialisation with Funnel Model (SMARTNET)”

Contract No: TR14C2.2.05-04/001

EUROPEAID/140284/IH/SER/TR

TEST SPECIFICATIONS (TS) DOCUMENT

15.05.2023

Document Control and Approval Sheet

Project Name	Technical Assistance for "A Smart Network for Technology Transfer and Commercialisation with Funnel Model (SMARTNET)"
Contract Number	TR14C2.2.05-04/001
Reference Number	EUROPEAID/140284/IH/SER/TR
End Recipient of Assistance	Yıldız Technical University
Contract Signed	08.03.2022
Commencement Date	25.05.2022
Duration	36 Months
Prepared By	Technical Assistance Team
Date of Document	15.05.2023
Contractor	SwanLeuco Danışmanlık AŞ Kiliçdede Mah. Kastamonu Sok. No:6 11/54 55060 İlkadım Samsun 0362 503 55 64 okan.gumus@swanleuco.com

Revision Information

Date	Revision	Notes and Changes
15.05.2023	1.0	Initial Revision

DISCLAIMER

This document has been produced with the technical assistance of the European Union under Technical Assistance for "A Smart Network for Technology Transfer and Commercialisation with Funnel Model (SMARTNET)" Project, Türkiye. Service Contract Number: "EUROPEAID/140284/IH/SER/TR".

Table of Contents

LIST of ABBREVIATIONS 5

1. INTRODUCTION 6

2. OUR APPROACH TO TESTING 6

2.1 ALPHA TESTING 6

2.2 BETA TESTING 8

3. TEST TOOLS 8

4. TEST LEVELS 9

4.1 UNIT TESTS 9

4.2 INTEGRATION TESTS 10

4.3 USER INTERFACE TESTS 10

4.4 SYSTEM ACCEPTANCE TESTS 10

5. TEST METHODOLOGIES 10

5.1 REGRESSION TESTS 10

5.2 BLACK BOX TESTS 11

5.3 WHITE BOX TESTS 12

5.4 SMOKE TESTS 14

5.5 SANITY TESTS / LOGIC TESTS 15

LIST of ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CA	Contracting Authority
CISOP	Competitiveness and Innovation Sector Operational Programme
DS	Decision Support
ERA	End Recipient of Assistance (Beneficiary)
EUD	Delegation of the European Union to Türkiye
GTU	Gebze Technical University
HKU	Hasan Kalyoncu University
ICT	Information and Communication Technologies
IP	Intellectual Property
IT	Information Technologies
ITU	Istanbul Technical University
KE	Key Expert
MIS	Management Information System
ML	Machine Learning
MoIT/DoEUFP	Ministry of Industry and Technology Directorate of EU Financial Programmes
OCU	Operation Coordination Unit
OCUD	Operation Coordination Unit Director
OS	Operating Structure
OIZ	Organized Industrial Zone
QA	Quality Assurance
R&D	Research and Development
RCOP	Regional Competitiveness Operational Programme
SME	Small and Medium Sized Enterprise
TA	Technical Assistance
TAT	Technical Assistance Team
TCD	Test Cases Document
TDZ	Technology Development Zone
TNA	Training Needs Assessment
ToR	Terms of Reference
TRL	Technology Readiness Level
TS	Test Specifications / Test Specialist
TTI	Technology Transfer Intermediary
TTO	Technology Transfer Office
YTU	Yıldız Technical University
WIPO	World Intellectual Property Organisation

1. INTRODUCTION

This Test Specifications (TS) document contains and explains the contractor development teams strategy, approach and methodology that will be employed to effectively test the developed **Smartnet MIS** software platform. This document also includes the scope of the **Alpha Testing** done by the development team per the requirements of the Terms of Reference (ToR) and the Technical Proposal (TP).

2. OUR APPROACH TO TESTING

Software testing is defined as the process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related technical environment to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects. It also includes to verify that a product continues to work when the technical environment is changed (patches, upgrades etc.).

Software testing involves a continuous process that includes requirements analysis, test planning, case development, environment setup, execution and analysis.

In Smartnet MIS Platform we would be employing two cycles of testing, the first cycle named **Alpha Testing** will be conducted by in-house Quality Assurance (QA) teams to ensure that the developed code confirms with the functional and non-functional requirements.

The second cycle is **Beta Testing** which is conducted by a select group of final users and stakeholders to ensure that non-technical eyes can also verify that the developed features and functionality runs as intended.

2.1 ALPHA TESTING

Alpha Testing is a form of internal acceptance testing performed mainly by the in-house software QA and testing teams, while Alpha testing can also be done by the potential users or customers of the application, it is still considered a form of in-house acceptance testing. Alpha testing is usually the last testing done by the test teams at the development site after the acceptance testing and before releasing the software for the beta test.

Alpha Testing is a methodology to assess the quality and stability of a product under test in the testing environment, by focusing on:

- Uncover showstopper bugs/major bugs (specifically in edge cases and corner areas)
- Usability Issues
- Feature gaps
- Compatibility/Inter-operability Issues

Stakeholders of Alpha Testing phase will be the Consultant's Development Team and Quality Assurance Team including appointed ERA members and when required, selected stakeholders who have subject matter expertise in domain and knows the desired attributes of the end-product well.

Procedures for Alpha Testing:

As a first step, we will gather and review the Functional Requirements from all possible documents available which includes meeting notes and expert interviews. Then we will clear ambiguities, queries at the earliest on the functional requirements. The functional requirements as well as any user stories will be entered into **JIRA backlog**, following the development of the functionality, the code is then pushed to the **Test Environment** where the first stages the alpha testing will be carried out: **Test Environment Control Testing** and **Test Environment UAT Testing**.

In the **Test Environment Control Testing** phase the Test Specialists (TS) from the development team will check on the Test Environment if the developed code is functional and fit for purpose comparing the developed features of the code with the specifications and user story entries in **JIRA**.

The next phase is the **Test Environment UAT Testing** phase in which the product owners (lead by ERA and Key Experts of the project) checks the functionality in the Test Environment and marks the test item as **Done** or sends it to **Bugfix** along with explanations to the development team on the corrections required.

After the alpha tests in the Test Environment are completed and accepted the code is then deployed to the **Production Environment** where the Production environment tests are done in a similar manner: **Production/Live Environment Control Testing** and **Production/Live Environment UAT Testing**.

In the **Production/Live Environment Control Testing** phase the Test Specialists (TS) from the development team will verify on the Production Environment if the developed code is functional in the Production Environment by comparing the developed features of the code with the specifications and user story entries in **JIRA**.

The final phase is the **Production/Live Environment UAT Testing** phase the product owners (lead by ERA and Key Experts of the project) verifies the functionality in the Test Environment and marks the test item as **Done** or sends it to **Bugfix** along with explanations to the development team on the corrections required at which time the feature is retracted from the Production Environment for further development and testing.

Testing activities will indicatively start, after the completion of each spring and all testing will be indicatively finalized by the end of the 12th Month of the project. However, the Consultant with its proposed approach to continuous integration and development foresees that further improvement and testing activities will continue during the finalization and software assurance phase of the project.

2.2 BETA TESTING

Beta Testing stage will follow the internal alpha test cycle. This is the final testing phase where the Consultant will release the Smartnet MIS Platform to a selected group of external users outside the dedicated test teams, ERA or immediate stakeholders.

This initial software version is known as the beta version. Though the Consultant will perform rigorous in-house testing with its dedicated quality assurance test team, it's practically impossible to test a platform as big as the Smartnet MIS for each and every combination of the test environment. Therefore, a selection of a Beta Test Group will be performed under full coordination of the ERA to test the beta version of the Smartnet MIS Platform for an extended period of time (indicatively one week) to detect and fix the minor glitches before the final deployment.

The procedure of the Beta Testing is similar to Alpha Testing, but the Consultant will ensure that the Beta Test users are notified of the objective, scope and approach to be used for testing and the feedback is collected by the TAT team and entered into JIRA.

3. TEST TOOLS

This section will provide detailed information on the tools that will be employed for testing the Smartnet MIS Platform to be developed.

Postman is a tool and collaboration platform that enables the use of services for API development. Tests are automated by creating test groups that can be run repeatedly. Postman; unit tests, functional tests, integration tests, end-to-end tests, regression tests, etc. It can be used to automate many types of testing, including.

Postman in the scope of the Smartnet MIS platform:

- Used for possible service and external APIs,
- Sending requests to APIs,
- Testing APIs,
- Create and manage APIs/monitors the performance of APIs,
- Publish and share APIs.

JMeter can be used for testing static files such as HTML, image, CSS, and javascript (JS) on demand, as well as testing web services that generate dynamically generated content based on SOAP (Simple Object Access Protocol) and REST (REpresentational State Transfer) in web applications.

Resource requests (web requests) made by real users to servers while using a web application are simulated with the help of JMeter as if real users were requesting these resources. User scenarios (the way users use the web application) simulated with JMeter can be configured as if more than one user is

running the same scenario at the same time by differentiating the inputs (input) of the web application, and a load of the desired size can be created on the system.

Within the scope of the Smartnet MIS, we will employ JMeter while performing the load tests to check with page speeds and any performance bottlenecks.

For example, within the scope of Smartnet MIS, we make multiple user logins to the at the same time.

- Does the page slow down after user logins?
- Is the page exploding? Checks are made.
- How's the speed situation? Is it running slowly?

Excel is an industry standard software to create spreadsheets. Thanks to Excel, computer users can use various formulas through a spreadsheet systematically and effortlessly perform functions such as organizing, formatting, calculation, and reporting. In the scope of Smartnet MIS, Excel is used to prepare test cases.

Acunetix is a software that tests known vulnerabilities and injection attacks and provides notifications and in the scope of Smartnet MIS is used in vulnerability tests.

Katalon Studio is an test automation suite where you can create test suites and run your test cases in the order you want. In this way, you can follow how many of the test scenarios you have run were successful and how many failed. Within the project scope, **Regression Tests** are performed with Katalon.

4. TEST LEVELS

Smartnet MIS Platform will be subject to four different levels of testing, that include **Unit Tests** run on the code; **Integration Tests** that focus on interaction between modules, units, and systems; **User Interface Tests** checks that automates and mimics user-behaviour checking the frontend of the software and **System Acceptance Tests** focusing on the behaviour and capabilities of the entire platform.

4.1 UNIT TESTS

Unit tests are coded and run by the programmer who writes the code. It reduces the risk of errors in software, catches functional and non-functional errors, finds errors in the unit, and prevents them from moving to higher levels.

4.2 INTEGRATION TESTS

Integration Tests focus on interactions between units or systems. The focus of these tests is not on “how the module or system works”, but on “how it integrates with other systems and modules”. Integration tests usually result in identifying Data Mismatch Errors, Interface Errors, and Inter-System/Inter-Module Communication Errors.

4.3 USER INTERFACE TESTS

User Interface Tests automatically checks the application’s front end is working correctly. While coding such tests, the steps of the user are analysed and mimicked, and these coded steps are followed in the application.

These tests take more time and requires constant maintenance as the user interface or expected user behaviour changes, therefore they are quite resource intensive.

4.4 SYSTEM ACCEPTANCE TESTS

System Acceptance Testing focuses on the behavior and capabilities of an entire system or product. It generally deals with the end-to-end tasks that the system can perform and the non-functional behaviors it displays while performing these tasks.

System Acceptance Testing reduces the risk by performing the controls before they are communicated to the end user by verifying that the system is complete and will function as expected, thereby building confidence in the quality of the system.

Typical errors found in system tests are:

- Incorrect calculations.
- Incorrect or unexpected system functional or non-functional behavior
- Faulty control and data flows within the system
- Inability to perform functional end-to-end jobs as they should be and completely.

System Acceptance Tests build confidence in the quality of the system as a whole verifying that the system is complete and will function as expected and by verifying whether the functional and non-functional behavior of the system is as specified in the requirements.

5. TEST METHODOLOGIES

5.1 REGRESSION TESTS

Regression tests are done after upgrades or any other system maintenance to check that new code does not affect or alter the behaviour of existing code. It is a type of test used to test that after any change made to the software, the newly introduced changes do not affect the other areas of the software.

Regression tests are used to control the changes made to the live code. These changes may be a new function, bug fix, or performance enhancement.

Regression tests’ primary purpose is to check that critical areas of the application still work as expected.

In practice, testers create a set of regressions based on the critique of the functions. (20, 30 steps) and ensure that the steps produce the same results when run following a new version of the software.

How to Perform Regression Test?

To perform Regression Testing, we first need to debug the code to detect errors. Once the bugs are identified, necessary changes are made to fix them, then regression testing is performed by selecting the relevant test cases from the test suite that covers both the modified and affected parts of the code.

When Should Regression Testing Be Performed?

Typically, regression testing is performed under the following conditions.

- A new requirement for an existing feature
- A new feature or functionality
- To fix code base flaws
- Source code optimized to improve performance.
- Patch fixes
- Changes in configuration

Within the scope of the Smartnet MIS Platform after each 2-week sprint plans, the development team applies pre-defined regression tests before taking the code live. Each test must perform 100% successfully on all steps defined before code deployments are performed.

5.2 BLACK BOX TESTS

Black Box Testing is an ethical hacking action that mimics the behavior of a malicious user to check the security level of an external environment or a website. In simple words, the “Black Box” test does not require any knowledge of the system. Such tests have a limitation that is described in the scope and negotiated with a client. The application of social engineering to gain knowledge is also discussed. Also known as the “Behavior Test”.

The black box testing technique is often associated with requirements-based testing. It helps to decide whether a system design meets the requirements and to avoid errors in implementation.

TYPES OF BLACK BOX TESTS

Functional Testing contains only the input and the corresponding valid output analysis. It is mainly done to determine the functionality and behavior of the system. Software testers ask, “Does this system do what it claims?” It focuses on answering this question. The tester checks how the system works.

Functional Tests are based on User Requirements and previously completed System Tests and contains behavioural flows of the Application and may cover other tests such as Smoke Testing, and Regression Testing.

Dysfunctional Testing checks how efficiently the system is performing its functions. It focuses on the non-functional requirements of the system such as performance, scalability, and usability that are not related to black box testing.

Dysfunctional testing controls how the system responds, based on the expectations of the users and contains controls for the Application's Performance, covering tests such as Load Tests, Performance Tests, and Security Tests.

BLACK BOX TESTING TECHNIQUES

Equivalence Class Tests are used to optimize the number of possible test cases while maintaining reasonable test coverage. Equivalence Class Tests are used by the team of testers for grouping and partitioning of the test input data, which is then used for the purpose of testing the software product into a number of different classes.

These different classes resemble the specified requirements and common behavior or attribute(s) of the aggregated inputs. Thereafter, the test cases are designed and created based on each class attribute(s) and one element or input is used from each class for the test execution to validate the software functioning, and simultaneously validates the similar working of the software product for all the other inputs present in their respective classes.

Limit Value Tests, also called **Boundary Value Tests**, focus on values at the limits. This technique determines whether a given range of values is acceptable to the system. It is very useful in reducing the number of test cases. It is best suited for controls where input is within certain ranges.

Decision Table Tests employ a decision chart that places certain causes and effects in a matrix. Each column has a unique combination that tests a business-case or user scenario and an expected result.

ADVANTAGES OF BLACK BOX TESTING

- Black Box Tests are done from the user's perspective and help reveal inconsistencies in specifications. The tester does not need to know any programming languages or how the software is coded and implemented.
- Testing can be performed by an organization independent of the developers, allowing for an objective perspective and avoiding developer bias.
- As soon as the specifications are complete, test cases can be designed.

DISADVANTAGES OF BLACK BOX TESTING

- Only a small number of possible inputs can be tested, and many program paths are left untested.
- Without clear specifications, which is the case with many projects, test cases will be difficult to design.
- If the software designer/developer has already run a test case, the tests may be unnecessary.

5.3 WHITE BOX TESTS

White box testing is a technique that uses the internal or source code of a program to design different test cases to check the quality of the program. In this technique, the internal structure and implementation of how an application works are known to the tester. It is also known by many other names such as White box testing, Glass Box Testing, Clear Box Testing, and Open Box Testing.

TYPES OF WHITE BOX TESTS

Path Testing is a white box testing approach based on a program's control structure. A control flow chart is created using the structure and the different paths in the chart are tested as part of the process. This test involves a thorough understanding of the structure of the program, as it depends on the control structure of the program.

Loop Testing covers loops that are one of the core concepts applied in a large number of algorithms. Loop Testing is about determining the loop validity of these algorithms. The purpose of this test is to uncover vulnerabilities that may exist in any given cycle. An example of a vulnerability that can be found in loop testing is incorrect indexes in loops. When indexes in an iterative loop are not programmed correctly, it can cause more bytes to be copied than necessary.

Conditional Testing tests the logical conditions for each value, regardless of whether it is true or false. This means that both if and else conditions are validated in the case of an IF-ELSE conditional statement.

Unit Testing is a method of testing a unit, the smallest logically separable piece of code in a system. Unit testing ensures that every component works as intended.

Mutation Testing is a type of test based on changes or mutations. Minute changes are made to the source code to see if the provided test cases can find bugs in the code. The ideal situation would be for none of the test cases to pass. If the test is successful, it indicates an error in the code. The mutant (modified version of the code) is said to have “survived”. If the test failed, there was no error in the code and the mutant was eliminated. Our goal is to eliminate all mutations.

Integration Testing is done to check that the modules/components work as intended when combined, that is, to ensure that modules that perform well independently will not experience difficulties when combined.

White-box **Penetration Testing**, also known as crystalline or slanted-box pen testing, provides the tester with complete network and system data, including network maps and passwords. This saves time and lowers the overall cost of an engagement. We use the participation model in software testing. The engagement model is a strategy that defines the basis of cooperation between the software development company and the customer.

The focus of the engagement model is the customer’s demands, needs, and interests. It also provides a level of flexibility, responsibility, and control. White box penetration testing can be used to simulate a particular attack on a particular system using as many attack paths as possible.

WHITE BOX TESTING TECHNIQUES

Statement coverage is a testing technique where the tester passes all statements and ensures that every statement of the code is executed at least once. So, every line of code is validated. This technique ensures that all statements are executed without errors or problems. In the flowchart, all nodes are covered and crossed at least once, and it also helps to focus on the code where the error is found.

Branch Coverage is a technique for testing in which test cases are designed in such a way that each branch is tested at least once. This technique analyzes every possible path of the system. In the case of a flowchart, each edge must be traversed at least once to ensure that no branches behave abnormally in the application.

Road Coverage Testing involves defining all possible roads and covering them. It is a software testing technique where paths are executable statements from input to output points in the system. This unfortunately is also a very time-consuming method.

Loop Testing strategy involves testing independent and dependent code loops, and their values. Errors often occur at the beginning and end of cycles. This strategy is about testing cycles.

Basic Path Testing is a technique in which control flow charts are made from the code and then their cyclic-complexity is calculated. Complexity defines the number of individual paths so that the minimum number of test cases can be designed.

ADVANTAGES OF WHITE BOX TESTING

- All code and structures are tested in white box testing.
- It results in the optimization of code removal errors and helps to remove extra lines of code.
- This test can start at an earlier stage.
- White box testing is easy to automate.
- Identify the Dead Code or other issues.
- This test covers all possible test scenarios because the tester has technical and programming knowledge.

DISADVANTAGES OF WHITE BOX TESTING

- These tests are quite expensive and time-consuming.
- Code redesign requires test cases to be rewritten.
- Missing functions cannot be detected.
- This technique can be very complex and sometimes unrealistic.
- White box testing requires a highly knowledgeable programmer due to the complexity of the level of testing that needs to be done. Sometimes it is unrealistic to be able to individually test every available state of the application.

5.4 SMOKE TESTS

Smoke Testing is a software testing process that determines whether the deployed software build is stable. It is a non-detailed test to understand whether the most important functions of an application are working. It is usually performed to check critical functions after the release of a new version of the application. It is a test of the 'main functions'.

Purpose of Smoke Test

The main purpose of the smoke test is to detect important problems at an early stage. Smoke tests are designed to demonstrate system stability and compliance with requirements.

For example, A new registration button was added to the login window, and the build was distributed with the new code. We do a smoke test on a new build.

- In the smoke test, a simple checklist should be prepared first.
- With the smoke test, it cannot be checked that the application is 100% error-free.

When do we do the Smoke Test?

When new functions of the software are developed, it is checked that a high-importance error does not occur in the new version or that a previously occurring error has been resolved.

How is the Smoke Test done?

Smoke Testing is usually done manually, but there is a possibility to do the same via automation. It can vary from organization to organization.

What happens if we don't do a Smoke Test?

If we don't do smoke testing in the early stages, defects can be encountered in later stages, which can be costly, and showstoppers where the defect found in later stages can affect the release of deliveries.

5.5 SANITY TESTS / LOGIC TESTS

Sanity Tests determine that the suggested functionality is working roughly as expected. If the plausibility test fails, the build is rejected to save the time and costs involved in a more rigorous test. The goal is not to thoroughly validate the new functionality, but to determine that the developer has applied some logic in producing the software. It is to verify the "rationality" of the system to move forward with more rigorous testing.

When do we do the Sanity Test?

- The areas where the code is changed on the application are checked.
- It is usually the last stage of the application, and no errors are expected.

How is the Sanity Test done?

Sanity Testing is usually done manually, but there is a possibility to do the same via automation. In the project scope, after the previous errors in the system are resolved and the live update is made, these errors are emphasized, and error-based, control tests are made to see whether the errors are corrected or not.

The contents of this publication are the sole responsibility of SwanLeuco in consortium with Evoluxer, Asturex, RRDA, StartUp Division, Inycon and can in no way taken as the view of the European Union and the Ministry of Industry and Technology of the Republic of Türkiye.