1. Introduction

The overall purpose of testing is to ensure the SOFTWARE meets all of its techinical, functional, and bysiness requirements. The purpose of this document is to describe the overall test plan and strategy of testing the SOFTWARE. The approach described in this document provides the framework for all testing related to this application.

1.1 Objectives

The quality objectives of testing SOFTWARE are to ensure complete validation of the business and software requirements.

- Verify software requirements are complete and accurate
- Perform detailed test planning
- Identify testing standards and procedures that will be used on the project
- Prepare and document test scenarios and test cases
- Regresion testing to validate that unchanged functionality has not been affected by changes
- Manage defect traching SOFTWARE's
- Provide test metrics/testing summary reports

1.2 Team Members

Name	Role	Email	GitHub
Fedeles Daniel Andrei	Automation Tester (Front-end)	daniel.fedeles@info.uaic.ro	qazwer1
Sparhat Cosmin Mihai	Automation Tester (Front-end)	cosmin.sparhat@gmail.com	Sparhat
Carnaru Tudor	Automation Tester (Back-end)	tudor.carnaru@yahoo.com	genius252
Dobos Stefan	Automation Tester (Back-end)	alexandru.dobos@info.uaic.ro	stfn12
Stanciu Emilian	Manual Tester	emi_stanciu2002@yahoo.com	Emilian28
Bordeianu Andreea	Manual Tester	andreea.bordeianu@info.uaic.ro	AndreeaBorde
Gavrin Flaviana	Manual Tester	flaviana.gavril@info.uaic.ro	Flavinia
Palasanu George	Manual Tester	george.palasanu@info.uaic.ro	Palasanu
Ciobanu Bogdan	Manual Tester		bogdan-ciobanu

1.3 Goals

The goals of testing this application include validating the quality, usability, relaiability and performance of the application. Testing will be performed from a black-box approach and a part of white-box. Tests will be designed arround requirements and functionality.

Another goal is to make the tests repeatable for use in integration and regression testing during the project lifecycle, and for future application upgrades. Repeated tests (Smoke and Regression) should be automated.

Quality

First a verification SOFTWARE's will be undertaken involving reviews and meetings to evaluate documents, plans, requirements and specifications to ensure that the end result of the application is testable and the requirements are covered. The overall goal is to ensure that the requirements are clear, complete, detailed, cohesive, attainable and testable. In addition, this helps to ensure that requirements are agreed to by all stakeholders.

Second, actual testing will be performed to ensure that the requirements are met. The standard by wich the application meets quality expectation will be based upon the requirements test matrix, use cases and test cases to ensure test case coverage of the requirements. This testing will also help to ensure the utility of the application.

Reliability

Reliability is both the consistency and repeatability of the application. A large part of testing an application involves validating its reliability in its functions, data, and system availability. To ensure reliability, the test approach will include positive and negative (break-it) functional tests. In addition, to ensure reliability throughout the iterative software development cycle, regression tests will be performed on all iterations of the application.

2. Scope

1)Functional testing are fully covered with testcases in Github and partially by Automation Tester in Java.

2)System testing provided by QA on Formal stages. During thet stage QA check all the features correct working.

Cross Browser Testing:

- User interface testing. Every acceptance/full yesy run QA of course check the main things of Main GUI. They should be clear to understand their workflow.
- Usability-testing. In the main usability should be thought on the architecture/design stage but QA also work with it. We provide results as improvement tickets(Issues) in github.
- Integration testing done by everiday activity.
- Performance testing was done by QA team.
- Unit testing was done by QA team.
- Automation Tester was done by QA team

- Manual Tester was done by QA team.

Portability testing:

Opera	Firefox	Chrome	Internet Explorer	Safari	Microsoft EDGE
	OK	OK			

Test modules:

- GUI
- Controller
- View1
- View2

Testing types:

- Functional testing: Unit testing, Integration testing, Regresion Testing, Acceptance Testing
- Non-Functional testing: Performance testing, Load testing, Stress testing, Portability testing

Stages of testing:

- Exploratory testing
- Create testcases

3. Test Strategy

The project is using an agile approach, with 3 weeks iterations. At the end of each 3 week the requirements identified for that iteration should be delivered to the team and all functionality will be tested.

Smoke test will be executed by manual after each build sent to qa . If there is a release, full regression test will be executed by manual and automation.

Black Box Testing: It is some time called behaviour testing or Partition testing. This kind of testing focuses on the functional requirements of the software.

GUI Testing: GUI testing will includes testing the UI part of report. It covers users Report format, look and feel, error messages, spelling mistakes, GUI guideline violations.

Functional Testing: Functional testing is carried out in order to find out unexpected behaviour of the report. The characteristic of functional testing are to provide correctness, reliability, testability and accuracy of the report output/data.

Smoke Testing: Smoke test is executed prior to start actual testing to check critical functionalities of the program is working fine. This set of test cases written such a way that all functionality is verified but not in deep

Regression Testing: Regression testing is the SOFTWARE s of testing changes to program to make sure that the older programming

still works with the new changes

4. Testing Overview

4.1. Preparing test cases

QA will be Preparing Test Cases in Github based on the requirement specifications . This will cover all scenarios for requirements. Test cases should be written before development

4.2. Test Automation

For the test automation it will be used the folowing technologies and tools:

Java and Selenium webdriver and TestNG for the Front-end

Java and Junit for Unit tests.

4.3. QA SOFTWAREs for New Feature

QA SOFTWAREs starts when Feature status is “Ready for QA”, QA tester execute test cases that related to the feature. If there is no defect, QA tester change status of task as “Closed”. If there is defect/s, QA tester change status of feature as “Reopened” and comment to task that includes detail of defect(Steps, expected result and actual result).

4.4. QA SOFTWAREs for fixed Bug

QA SOFTWAREs starts when Bug status is “Ready for QA”, QA tester retests the case. If there is no defect, QA tester change status of task as “Closed”. If there is still defect, QA tester change status of bug as “Reopened” and comment to bug. In retest if bug is closed but developer causes new defect, QA tester opens new bug.

4.5. Unit Testing

All code needs to be unit tested to ensure that the individual unit (class, module or file when valid) performs the required functions and outputs the proper results and data. Proper results are determined by using the design limits of the calling (client) function as specified in the design specification defining the called (server) function. Outer effects and dependencies may be isolated by performing mock tests.

Unit testing is typically white box testing and may require the use of software stubs and symbolic debuggers. This testing helps ensure proper operation of a module because tests are generated with knowledge of the internal workings of the module.

A special form of unit testing is Integration Test, which does not isolate dependencies and external components of the software.

Development/Testing team members are responsible for testing each program produced to demonstrate correct operation of coded modules units.Recommended tools and techniques for unit testing are mockito, Junit.Eclipse moreUnit.

4.6. Load/Performance Testing

Load/performance testing is conducted to demonstrate the correct operation and performance of the software release as well as to identify the operational envelope of the solution, revealing weaknesses and possible bottleneck points. C# tests or Java will be used for testing of SOFTWARE.

4.7. Integration Testing

Version of project must be increased. New version must be deployed to test environment. Smoke test is executed when sufficient time is not available for Regression test. Smoke test cases are executed after code freeze. If QA tester find defect/s, he/she reports the bug in Github Issues.

4.8. User Acceptance Test:

Version of project must be increased. New version must be deployed to test environment. Smoke test is executed when sufficient time is not available for Regression test. Smoke test cases are executed after code freeze. If QA tester find defect/s, he/she reports the bug in Github Issues.

4.9. Smoke Test:

Version of project must be increased. New version must be deployed to test environment. Smoke test is executed when sufficient time is not available for Regression test. Smoke test cases are executed after code freeze. If QA tester find defect/s, he/she reports the bug in Github Issues

4.10. Exploratory test:

Version of project must be increased. New version must be deployed to test environment. Exploratory test is executed when sufficient time is not available for Regression test. Smoke test cases are executed after code freeze. If QA tester find defect/s, he/she reports the bug in Github Issues

4.11. Regression Test:

Version of project must be increased. New version must be deployed to test environment. Regression test is executed when sufficient time is not available for Regression test. Smoke test cases are executed after code freeze. If QA tester find defect/s, he/she reports the bug in Github Issues

4.12. Test Report:

After execution of UAT and Smoke/Regression Test, Test report/s is prepared, Test Report includes; project sprint/version,Test Result Pea chart, Test Result details, Defect details, Date, Environments and versions.

4.13 Defect Tracking:

Defect Detailes:

Title	Screen name and short description about the defect being reported, usually providing key words with which to identify and/or search for the defect.
Description:	Preconditions Environment / Device/ OS /Browsers.

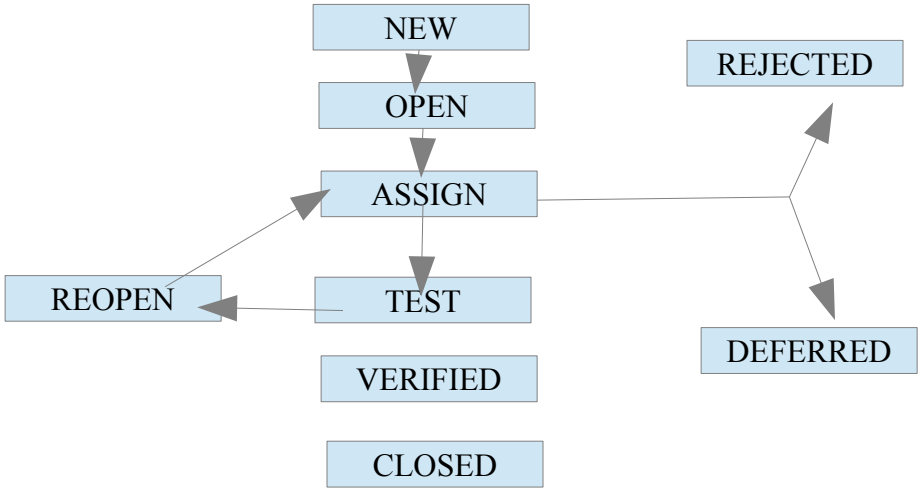
	Test Steps Expected Result Actual Result
Priority	This field describes the impact the defect has on the work in progress and the importance and order in which a bug should be fixed
Component	Component in which the defect was found.

Defect Resolution:
When the defect is opened, it is assigned to the appropriate person, status is changed to “Assigned”
Once the defect is fixed:

1. The developer to whom the defect is assigned will update the defect comments to document the fix that was made. User ID and Date is automatically added to the defect by clicking on “Add Comment”.
2. The developer to whom the defect is assigned will change the status to “Fixed”.
3. The tester will retest the submitted defect.
4. If defect passes the retest, the tester will change Status to “Closed”.
5. If the defect is not fixed, the tester will change the Status to “Reopened” and comment to defect.

Bug Life cycle:

All the issues found while testing will be logged into Github Issues bug tracker. Bug life cycle for this project is as follows:



Definition for Defect Priority:

Priority	Impact	Functionality	Performance	Stability	Graphical/UX
Blocker	Blocks development and/or testing work, production could not run.				
Critical	Significant impact on revenues, customer business operations or the security of the platform where no cost effective work is available.	Development or testing blocked and significantly impacting progress. Any serious regression of previously known/tested functionality Core/critical functionality missing or broken with significant impact on end users	Performance introducing significant overhead to development or test. Critical performance deficiency, system is rendered unusable. Frequent or permanent performance degradation. Easily reproducible.	Crashes or unrecoverable hangs introducing major overhead to development or test.	
Major	Minor impact on revenues, customer satisfaction,customer, business operations or the security of the platform where no cost effective work is available	Critical functionality working but supporting features around it missing or broken with minor impact on end users.	Noticeable decrease in system performance, although system remains usable.	Minor or infrequently problems.	Graphical fault visible to most end users.
Minor	No impact on revenues, customer satisfaction,customer or business operations	Critical functionality working but supporting features around it missing or broken where a cost effective workaround is available with no real impact on end users	Minor or infrequent deterioration in performance.	Stability problem only encountered under conditions not expected to be found during normal use.	Minor graphical fault that would not be noticed by most users.
Trivial					Graphic change or

					issue where there is no impact to users, but business would prefer to align to existing products/marketing/artefects.

5. Test Environment:

NO	Environment	
1	Number of test machines	9
2	Licenced OS	9
3	Browsers	6

6. Test Tools:
6.1 Test Management Tool:
Notepad and Github

6.2. Bug Tracking Tool:
Github Issues

6.3 Automated Test Tools:
Eclipse/Netbeans – Java ID
Java – Technology
Junit – Test Report Tool
TestNG – Test Report Tool
Selenium – Test Framework

6.4 Unit Test Tools
Junit and Java
QA Team will provide test results for executed test.

6.5 Performance Testing Tools:
Performance testing will be performed using free tools and with the support of the dev team.

7 Delivrables:

Delivrable	For	Data
Test Plan	PM, QA M, Test Team	
Test Report	PM, Product Owner/Client	
Test Status Report	PM Product Owner/Client	

8. Entry Criteria

- New features must be functional tested, there is no critical and major defect.
- All test hardware and environments must be in place and free for testing.
- Fixed bugs must be retested and there is no open critical and major bug.
- Overall unit test coverage must be 40%.
- New test coverage must be 80% (for new development).

Total Estimated Test Cases:

View 1	Controller	View2
30	0	11

9. Exit Criteria:

- Test case execution must be 100% complete.
- Passed case rate should be 80%.
- Release cannot be shipped with any outstanding high business priority (critical and major) defects and/or " 9%" of medium priority (minor and trivial) defects.
- Performance of the system must not slow down with the introduction of new features.
- Test cases are executed considering Browsers different versions
- User acceptance testing is completed.
- All the planned requirements must be met

10.Risks:

The following risks have been identified and the appropriate action identified to mitigate their impact on the project.

The impact (or severity) of the risk is based on how the project would be affected if the risk was triggered. The trigger is what milestone or event would cause the risk to become an issue to be dealt with.

#	Description	Probability	Impact
1	Risk of Attrition	HIGH	VERY HIGH
2	Missing SW Licences for development	MEDIUM	HIGH
3	Missing dedicated test environment	MEDIUM	HIGH
4	Distributed team cooperation	HIGH	HIGH
5	Replacing Team Members	HIGH	VERI HIGH