SSDL Systems and Software Development Lab

Cognitivity

Test Plan

Submitted by: Daniel Lyubin

Mark Erlich

Ben Hunter

Guy Shemesh

Ophir Katz

Peer Sagiv

Supervised by: Tanya Brokhman

Table of Contents

1. Introduction

1.1 Purpose

This test plan describes the testing approach and overall framework that will drive the testing of the Cognitivity project. The document introduces:

- Test Strategy: rules the test will be based on, including the givens of the project (e.g.: start / end dates, objectives, assumptions); description of the process to set up a valid test (e.g.: entry / exit criteria, creation of test cases, specific tasks to perform, scheduling, data strategy).
- Execution Strategy: describes how the test will be performed.

2. Test strategy

2.1 Test objectives

The objective of the test is to verify that the functionality of the Cognitivity project works according to the specifications. The final product of the test is twofold:

- A production-ready software;
- A set of stable test scripts that can be reused for Functional test execution.

2.2 Test assumptions

Angular was approved by google before use.
MySql approved by oracle before use.

2.3 Scope and Levels of Testing

2.3.1 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 validating the output from the application.

METHOD: The test will be performed according to Functional scripts/Test procedures with a well-defined PASS/FAIL criteria.

2.3.2 User Acceptance Test (UAT)

PURPOSE: this test allows the end users to complete one final review of the system prior to deployment.

METHOD: Will be performed manually by team members according to written test cases.

3. Validation and Defect Management

It is the responsibility of the tester to open the defects, link them to the corresponding script, assign an initial severity and status,

It is the responsibility of the developer to retest after a fix is provided and close the defect.

Defects will be categorized according to the following severity status:

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.

4. TEST ENVIRONMENT

The test environment of the backend MVC logic will be Maven test running. We will check all logic of the backend using automated tests, with code coverage.

The test environment of the frontend (and GUI) will be only UAT since it is incredibly difficult and time consuming to create automated tests for GUI.

The test environment of the data server will be using a docker container to make the data manipulation abstract for the MVC layer.

5. Test cases

5.1 Function tests

5.1.1 Backend MVC Tests:

- Controller Tests:

  Objective: To assert that the controller classes are receiving, transferring and manipulating data properly.
  Defects categorization:
- Critical – The data of the http message was not transferred correctly, or wasn't returned correctly.
- High – Some formats was not understood in the transferred data.
  Input: A HTTP Message
  Test logic:
  1. Send a HTTP message from
  2. Check in the backend that the data is received correctly with no errors.
- Controller Tests:

  Objective: To assert that the data conversion from JSON representation to a Java object is done correctly.
  Defects categorization:
- Critical – The conversion crashes because the data was not in the correct format.
- High – The conversion is faulty because some fields of the objects were not initialized properly.
  Input: JSON representation of an object (Question, TestID etc...)
  Test logic:
  1. Call to a conversion function on the JSON data.
  2. Check that for every field, the value it holds is the correct one according to the mapping in the JSON object. i.e., if in the JSON object there was a mapping $questionID : 111$ then the object will hold the value 111 in the field questionID.

  Objective: To assert that the data is being pushed and pulled correctly from the data server by the defined methods.
  Enter criteria: The data server is initialized and ready.
  Exit criteria: The data was pulled from the server correctly with no errors.
  Defects categorization:

- Critical – The data was not pushed/pulled from the server at all.
- High – Some of the data was pushed/pulled, and some wasn't.
- Low – The data was successfully pulled from the server, but some entries are missing.

Input: Some data to push to the server (including different types of data, to different table, with different fields etc…)

Test logic:

1. Push some data to the server (doesn't require use of function, just MySQL queries on some info).
2. For each method that requests data from the server (whether it requests a single entry, or a list of entries by some criteria), call it and receive the data.
3. Test that the received data is the same as the data that was pushed in stage 1 of the test, according to the criteria.

Objective: To assert that the data on the server is being updated correctly.
Enter criteria: The data server is initialized with data (tables) and ready.
Exit criteria: The data was updated as expected.
Defects categorization:

- Critical – The data was not updated at all.
- High – The data was updated, but the old data was not erased from the data server.
- Low – The data was updated but the values are not as expected.

Input: Some data to update/delete.
Test logic:

1. Call to an update/delete method on the data.
2. assert that the data was updated correctly by pulling it and checking the fields. Or (if delete), assert that the data was deleted and is not in the server by querying it.

5.1.2 Hibernate coherence with DB:

Objective: To make sure we can read data from db

Enter criteria: There is a db up and running
Exit criteria: The reading has finished
Defects categorization:
High – if there isn't an option to read data, the managers won't be able to analyze tests
Test logic:

1. enter data to the db, to each of the tables
2. read data from the db using hibernate
3. check that the data is the same as we entered

Objective: To make sure we can write data to db

Enter criteria: There is a db up and running
Exit criteria: The writing has finished
Defects categorization:
Critical – if there is a defect at writing, we will lose data
Test logic:

1. write data to the db using hibernate, to each of the tables
2. read data from db
3. check that the data is the same as we entered

5.2 UAT tests

Those are GUI components so there will be only UAT.

5.2.1 Manager's Preview Mode:

- Preview Validation Test:
  - **Objective:** check if the data of the project isn't lost when exiting the preview mode.
  - **Test Logic**:
    - Fill the project with questions, blocks, etc.
    - Step into preview mode.
    - Exit preview mode.
    - Check that data that was shown in preview mode is the same as the data in edit mode.
  - **Defect categorization**
    - **High:** Data isn't consistent
  - **Enter Criteria:** There is data filled for the project.
  - **Exit Criteria:** The data in the edit mode is the same as the one in the preview mode.

5.2.2 Subject Registration Form:

- Preview Valid Test
  - **Objective:** to check that the registration screen behaves as expected when given valid values.
  - **Test Logic:**
    - Fill all the fields in the form.
    - make sure submission is possible.
  - **Defect categorization**
    - **High:** the subjects must be able to submit valid values.
  - **Enter criteria:** the registration form is empty.
  - **Exit criteria:** all the fields are filled with valid values, and submission is possible.
- Preview Invalid Test
  - **Objective:** to check that the registration screen behaves as expected when given invalid values.
  - **Test Logic:**
    - Fill all the fields in the form with invalid values.
    - make sure submission is not possible and that the wanted alerts are shown.
  - **Defect categorization**
    - **High:** the subjects must submit only valid values so we can show it in the analysis view.
  - **Enter criteria:** the registration form is empty.

- **Exit criteria:** all the fields are filled with invalid values, and submission is not possible. for every field there an alert.

5.2.3 Manager's Dashboard Tests/ Analysis View:

The dashboard is GUI, so there will be only UAT.

- Test List test:
  - **Objective**: check if the tests shown in the list are the tests of the manager.
  - **Test Logic**:
    - Log the tests returned from the service.
    - Click on the 'Tests' label on the navbar.
    - Check that every test in the logged tests is in the table, and no other tests are shown.
  - **Defect categorization**
    - **High**: Data isn't consistent
  - **Enter Criteria:** There is test information in the DB
  - **Exit Criteria:** Tests are shown in the table
- Navigation Test:
  - **Objective**: Insure that navigating between pages is correct and doesn't disrupt the design of the page.
  - **Test Logic:** Navigate to different pages using the labels and links available
  - **Enter Criteria:** The dashboard page has loaded
  - **Exit Criteria:** The dashboard is intact. Specifically, the navbar is in its right place and components outside the dashboard are not appearing.
  - **Defect categorization**
    - **Critical:** There are components not related to the dashboard, shown in the dashboard.
    - **High:** Wrong components are router or routing is not working.
    - **Medium:** The navbar is not showing.
- **Filtering Test**:
  - **Objective:** Check that project filtering (by date, #questions, tags) works.
  - **Test Logic**:
    - Choose a filter type.
    - Check that the projects shown are the ones with the specified type.
  - **Enter Criteria:** There are projects in the database.
  - **Exit Criteria:** Shown projects are with the specified type.
  - **Defect Categorization:**
    - **High:** Wrong projects are shown.

5.2.3 Questions creation and view tests:

- **Creating a question test:**

- **Objective:** check by the manager if he can add a new question to a test
- **Enter criteria:** The manager is logged in and already created a test.
- **Exit criteria:** The manager can view the question in the test.
- **Defects categorization:**
  - Critical - the question didn't added at all to the test.
  - High - the question wasn't added correctly.
  - Low - the view isn't satisfying enough.
- **Test logic:**
  - **Entering the form for creating a question -** The manager should enter the dashboard, and to access a specific test that he wishes to add a question, after that he needs to edit the test and to access the form for creating a question.
  - **Filling the form for creating a question -** The manager needs to fill all relevant details in the form - The question itself, type of question (multiple choice and the required organization of the answers, open question, rating question), the answers (depending on the type of the question).
  - **Submitting the form -** After finishing filling the form the manager will submit the form.

- **Editing the settings of a question:**
  - **Objective:** check if the manager can edit the question, and change its settings.
  - **Enter criteria:** The manager is already logged in and created the question that he would like to edit.
  - **Exit criteria:** The manager can view the new edited question in preview mode.
  - **Defects categorization:**
    - Critical - The question didn't changed
    - High - The question wasn't edited correctly
    - Low - The editing isn't satisfying enough
  - **Test Logic:**
    - **Entering the test:** The manager enters the test that he would like to change.
    - **Accessing the required block:** The manager scrolls down to the block of question which consists the question that the manager wants to edit.
    - **Entering to edit mode:** The manager enters to the edit question, which being showed after pushing the settings button in the required question.
    - **Editing the question:** The manager edits the question according to the allowed changes as described in the HLD (changing the type of question, text of question, text of answers and so forth).

        - **Submitting the changes:** The manager submits the changes the question.
- **Changing position of question:**
    - **Objective:** To check if the manager is able to change the question position inside a block
    - **Enter criteria:** The manager is logged in and already created the question the he would like to move.
    - **Exit criteria:** The manager can view the question in its new position at the preview mode.
    - **Detects categorization:**
        - High - The question didn't move at all or to the wrong position
    - **Test Logic:**
        - **Entering the test:** The manager enters to the test that he would like to change.
        - **Accessing the required block:** The manager scrolls down to the block that hold the question that he would like to move.
        - **Entering edit mode:** The manager enters to the edit question, which being showed after pushing the settings button in the required question.
        - **Moving the question:** The manager moves the question by clicking up or down, to the required position.
- **Deleting a question:**
    - **Objective:** check if the manager can delete a question.
    - **Enter criteria:** The manager is already logged in and created the question that he would like to delete.
    - **Exit criteria:** The manager can't view the question that he deleted.
    - **Defects categorization:**
        - Critical - The question didn't deleted
        - High - The question deleted but the test didn't changed properly (creating a hole in the test and so forth)
    - **Test Logic:**
        - **Entering the test:** The manager enters the test that he would like to change.
        - **Accessing the required block:** The manager scrolls down to the block of question which consists the question that the manager wants to delete.
        - **Entering to edit mode:** The manager enters to the edit question, which being showed after pushing the settings button in the required question.
        - **Deleting the question:** The manager pushed to delete the question.
- **Adding a tag to a question:**
    - **Objective:** check if the manager can add a tag to a question.
    - **Enter criteria:** The manager is already logged in and created the question that he would like to a add to tag to it.

- **Exit criteria:** The manager can view the tag that was added to the question.
- **Defects categorization:**
  - Critical - The tag wasn't added
- **Test Logic:**
  - **Entering the test:** The manager enters the test that he would like to change.
  - **Accessing the required block:** The manager scrolls down to the block of question which consists the question that the manager want to add a tag to.
  - **Adding a tag:** The manager adds a tag to the question he wants.

- **TIme triggers**
  - **Objective:** To be able as manager to add time triggers.
  - **Enter criteria:** The manager is logged in and already created the question that he would like to add a time trigger to.
  - **Exit criteria:** The time trigger works when activated in a test
  - **Defects categorization:**
    - Critical - The time trigger wasn't added at all
    - High - The time trigger doesn't works properly, doesn't count the time right.
  - **Test Logic:**
    - **Entering the test:** The manager enters the test that he would like to change.
    - **Accessing the required block:** The manager scrolls down to the block of question which consists the question that the manager want to add a time trigger to.
    - **Entering to edit mode:** The manager enters to the edit question, which being showed after pushing the settings button in the required question.
    - **Adding a time trigger:** The manager adds a time trigger to the question he wants.
- Present Block's questions
  - **objective:** to make sure the questions are visible inside the block.
  - **test logic**:
    - open a collapsed block.
    - make sure all the questions are fully visible.
  - **enter criteria:** the block is collapsed, and contains at least 1 question.
  - **exit criteria:** the block is not collapsed and all the question are visible.
  - **defect categorization:**
    - **critical** - the user must be able to view the question from the block, because that's the only way to see them.
- Rename a block
  - **objective:** make sure the user can rename a block.
  - **test logic:**

        - try and rename a block.
    - **defect categorization:**
        - **low**: it's only important to the eye and not for the system functionality.
    - **enter criteria:** there is a block , doesn't matter how many question it has.
    - **exit criteria:** the block's name is changed.
    - **test logic:**
        - press the "rename" button and change the blocks name.
        - make sure it's saved.
- change blocks order
    - **objective:** make sure the block's order is dynamic and changeable.
    - **enter criteria:** have more than 2 blocks.
    - **exit criteria:** the blocks order (including their questions) have changed.
    - **test logic**
        - change the order of the blocks randomly.
        - check the "preview mode" as well as the regular view to make sure the order has been changed.
    - **defect categorization:**
        - **medium** - it hurts the user ability to make to tests as different as possible.

- import a block
    - **objective:** make sure that importing a block works as it should.
    - **enter criteria:** have a saved block in the DB.
    - **exit criteria:** have the saved block imported and shown correctly in the current view.
    - **test logic:**
        - create a block by importing it from a saved block.
        - make sure all the content of the block is shown in a valid way.
        - make sure all the content is shown.
    - **defect categorization:**
        - **high** - it prevents from the user to reuse existing blocks.

- delete a block
    - **objective:** make sure that the deletion of a block works.
    - **enter criteria:** have an existing block, and two other, previous and next.
    - **exit criteria**: the block is erased successfully.
    - **defect categorization:**

- **critical -** the user must be able to dynamically erase blocks from the test.

- **test logic:**
  - click on the deletion button.
  - make sure the block is erased.
  - make sure the other blocks are show in a valid way.