Deel QA Automation/Manual home exercise

Full name: Rodolfo Pacheco Navarro.
Start date: 3/19/2022

TEST PLAN

Individual Client Sign Up Test Plan

Product: Deel Client Sign Up Page.
Prepared by: Rodolfo Pacheco.
Date: 3/19/2022

1.0 INTRODUCTION

This test approach document describes the appropriate strategies, process, workflows and methodologies used to plan, organize, execute and manage testing of software projects within Deel.

1.1 Objective

Assure that the system meets the full requirements, including quality requirements and fit metrics for each quality requirement and satisfies the use case scenarios and maintain the quality of the product.

1.2 Team Members

Resource Name	Role
Rodolfo Pacheco	QA Engineer
Deel Dev 1	Developer
Deel Dev 2	Developer

2.0 SCOPE

2.1 In Scope

The Individual Client Sign Up Test Plan defines the unit, integration, system, regression and client acceptance testing approach. The test scope includes the following:

- Testing of all the functional, performance, security and use cases requirements listed in the Use Case document.
- Quality requirements and fit metrics.

- End-to-end testing and testing of interfaces of all systems that interact with Deel Client Sign Up page.

2.2 Out of Scope

The following are considered out of scope for The Individual Client Sign Up system Test Plan and testing scope:

- Functional requirements testing for systems outside The Individual Client Sign Up application.

3.0 ASSUMPTIONS/RISKS

3.1 Assumptions

This section lists assumptions that are made specific to this project:

- For User Acceptance testing, the development team has completed unit, integration and system and met all the requirements based on Requirement Traceability Matrix.
- User Acceptance testing will be conducted by End-users.
- Test results will be reported using the Deel Bug Tracker Tool. Failed scripts and defects will be sent to the developers team directly.
- Test scripts are developed, code-reviewed and approved.

3.2 Risks

Risk	Mitigation
Team members lack the required skills for website testing.	Plan training course to skill up your members
The project schedule is too tight; it's hard to complete this project on time.	Set Test Priority for each of the test activities.
Wrong budget estimate and cost overruns.	Establish the scope before beginning work, pay a lot of attention to project planning and constantly track and measure the progress

4.0 TEST APPROACH

The project is using an agile approach, with iterations every three weeks. At the end of the iteration the requirements identified for that iteration will be delivered to the team and will be tested.

Functional testing approach:

Test	Input	Expected Output

Validate profile cannot be completed without filling required all fields.	All required fields are blank or using the default values.	Validation saying to fill all the required fields is displayed.
Validate SQL injection instructions cannot be entered into alphanumeric fields.	SQL instructions entered into alphanumeric fields.	Entered fields don't affect database integration.
Validate all field lengths are consistent with the database.	Number of characters on fields exceeding the database lengths.	System prompts an error message saying to enter less characters on the fields is displayed / Fields are locked to enter only the maximum length of the field on the database.
Validate **Date of birth** cannot be a future date.	Date of birth field with a future date.	System prompts an error message saying to select a past date.
Validate **Date of birth's** correct date format.	**Date of birth** field with a different date format.	System prompts an error message saying to enter a correct format date.
Validate alphanumeric characters cannot be entered into the **Date of Birth** field.	Alphanumeric characters entered in the **Date of Birth** field.	System prompts an error message saying to enter a correct format date.
Validate entering a correct past date into the **Date of Birth** field.	A past date with correct format entered into the **Date of Birth** field.	Date is correctly saved.
Validate alphanumeric cannot be entered into the Phone **number** field.	Alphanumeric characters entered in the **Phone number** field.	System prompts an error message saying to enter only numbers into the **Phone number** field. / A phone mask validation to prevent entering alphanumeric characters.
Validate negative numbers cannot be entered into the **Phone number** field.	Negative numbers entered into the **Phone number** field.	System prompts an error message saying to not enter negative numbers into the **Phone number** field. / A phone mask validation to prevent entering alphanumeric characters.
Validate **Phone number** field length.	Enter more than 10 numbers into the **Phone number** field.	System prompts an error saying to enter a 10 digit phone number / A Phone number mask validation to enter only 10 numbers.
Validate **Zip code / Post code** exists/is valid.	An invalid/non-existent zip code entered on the **Zip code** field.	System prompts an error message saying the zip code is invalid.
Validate **Zip code / Post code** format is valid according to the selected country.	A zip code with a different format than the selected country is entered. I.e: select	System prompts an error saying the zip code does not correspond with the selected

	United States as country and enter a Canadian zip code.	country.
Validate a valid **Zip code / Post code** is correctly saved.	A correct **zip code** entered according to the selected country.	Zip code is correctly saved.

4.1 Test Automation

Automation tests will take part on the project and will be divided into three sections:

- Sanity (smoke) tests: will be executed upon each commit.
- Regression tests: will be executed every night.
- End2end test: will run on weekends using heavy data.

5.0 TEST ENVIRONMENT

Include the minimum hardware requirements that will be used to test the application. Testers will have access to one or more application/database servers separate from any used by non-test members of the project team. Testers will also have access to a number of 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.

6.0 MILESTONES / DELIVERABLES

6.1 Test Schedule

Task name	Start	Finish	Effort	Comments
Test planning	3/19/2022	3/21/2022	2d	
Staff and train new resources	3/21/2022	4/4/2022	2w	
First deploy to QA test environment	4/4/2022			
Functional testing iteration 1	4/4/2022	8/4/2022	1w	
Second deploy to QA environment	11/4/2022			
Functional testing iteration 2	11/4/2022	15/4/2022	1w	
System testing	18/4/2022	22/4/2022	1w	
Regression testing	25/4/2022	29/4/2022	2d	

UAT	2/5/2022	3/5/2022	1d	
Resolution of final defects and final build testing	4/5/2022	11/5/2022	1w	
Deploy to staging environment	12/5/2022			
Performance testing	12/5/2022	13/5/2022	1d	
Release to production	13/5/2022			

6.2 Test Deliverables

Deliverable	For	Date / Milestone
Test Plan	Project Manager, QA Lead, QA Team	3/23/2022
Traceability Matrix	Project Manager, QA Lead	3/24/2022
Test Results	Project Manager	13/5/2022
Test Status Report	QA Lead	16/5/2022
Metrics	All Team Members	16/5/2022

AUTOMATION

Pleases specify how you are going to automate the test plan in terms of:

1. **Tools:**
   - **Automation test framework:** Selenium.
   - **Language:** Typescript.
   - **Test runner:** Protractor.
   - **Reporter:** Cucumber Reports.
   - **Source control:** GitHub.
2. **Processes:**
   Automation Features Backlog:
   We need to have a backlog for the functionalities that we want to automate taking in consideration this criteria:
   - Repetitive tests that run for multiple builds.
   - Tests that tend to cause human error.
   - Tests that require multiple data sets.
   - Frequently used functionality that introduces high risk conditions.
   - Tests that are impossible to perform manually.
   - Tests that run on several different hardware or software platforms and configurations.

- Tests that take a lot of effort and time when manual testing.

Review the feature to automate and review existing pages, steps and features.
After we have the functionality we want to automate, we need to review the existing code in order to avoid code duplication.
Documentation on code:
Automation code is usually written, added and updated by multiple individuals to a version control system. In such an environment, proper documentation, comments and consistent naming conventions are of the utmost importance. It helps organize every developers' code and helps their peers keep track of the entire code base. If one coder leaves the team or wants to add new functionality by using existing code, they can debug, update, test and analyze results with greater results.
Version control:
It's very important to have a version control system to reduce possibilities of errors and conflicts, meanwhile project development through traceability to every small change and to help us in recovery in case of any disaster or contingent situation.
Code review:
A very good practice to add in the automation process is the code review, since it's always good to have a second pair of eyes to check the code for mistakes and possible improvements.
Continuous integration:
It is the practice of integrating changes from different QAs in the team into a mainline as early as possible, in best cases several times a day. This makes sure the code individual QAs work on doesn't divert too much. When you combine the process with automated testing, continuous integration can enable your code to be dependable.

3. **Stages:**

Step 1: Defining the Scope of Automation
We need to define the area of your Application Under Test that will be automated. In this case is the Fixed Contract creation. We need to know precisely the team's test state, the amount of test data, also the environment where tests take place. This are the criterias that can help to determine the scope:

- Technical feasibility.
- The complexity of test cases.
- The features or functions that are important for the business.
- The extent to which business components are reused.
- The ability to use the same test cases for cross-browser testing.

Step 2: Selecting a Testing Tool
After determining the scope, it is now the time to pick up a tool for automation testing. Yet, it solely depends on the technology on which the application tests are built. Each type of tool or framework may serve different demands, therefore having a thorough understanding of multiple tool types is also a prominent factor in choosing the best tool.

Step 3: Planning, Designing, and Development

At this stage, we need to create an automation strategy and plan. This plan can include the following items:

- The selected automation testing tool.
- Framework design and its features.
- A detailed timeline for scripting and executing test cases.
- In-scope and Out-of-scope items of automation.
- Goals and deliverables of automation testing process.

Step 4: Executing Test Cases and Build the reports
Once finishing all of the preceding steps, it's time to write the scripts, run the test automatically, either by running the code directly or by calling an application's API or user interface. After the execution, the test report provides a consolidated summary of the testing performed so far for the project.

Step 5: Maintaining previous test cases
Test maintenance is extremely important if we want to expand the collection of reusable test scripts. Once the automated tests have been scripted and running, they still need updating if the application changes the next time.

4. **Reporting:**
   Reporting is an important element of a Test automation framework as it plays a crucial component in a well-designed test automation tool. The success of the designed test automation framework and its survival also depends on how effectively the reporting mechanism is implemented.
   Test automation report is the blueprint that shows how the script was executed. It must show the scripts that were executed, steps in the test script which were executed, execution time, checkpoints that were passed, failed, skipped, and broken, if failed, skipped or broken what was the actual results and failure reason.
   Test execution reports should be detailed enough when there is a test execution pass as well as for other reasons. The report should be detailed enough to identify whether it's a script failure or an application failure.
   The test execution report also should be detailed enough even for other SDET to open a defect utilizing the information provided in the report.

TEST TYPES

We now want to divide the automation test plan into:
1. Sanity (smoke) test - upon each commit.
2. Regression test - running every night.
3. End2end test - running on weekends with heavy data.

Please describe how you recommend doing that.

I'll recommend using Jenkins, an open-source automation tool written in Java with plugins built for Continuous Integration purposes.Jenkins is used to build and test your software projects continuously making it easier for developers and QAs to integrate changes to the

project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

For the Sanity test we need to use a webhook to let Jenkins know then the application repository is pushed, so it can trigger our sanity testing build.

For the Regression test we need to schedule jobs in Jenkins, so we can set the job to be built every night. In the case of End2 we may use the same case that regression and also use Data-Driven Tests, which allows us to run the same test case with many varying inputs, therefore increasing coverage from a single test. In addition to increasing test coverage, data driven testing allows the ability to build both positive and negative test cases into a single test.

We need to take in consideration these points too:
Keep Records for Better Debugging: When tests fail, it's important to keep records of the failure as well as text and video logs of the failed scenario so that testers can identify reasons for test failure. If possible, choose a testing tool with an in-built mechanism for automatically saving browser screenshots in each step of the test. This makes it easy to detect the step at which the error occurs.
Early and Frequent Testing: To get the most out of automation testing, start testing early in the sprint development lifecycle. By doing so, testers can start detecting bugs as they appear and resolve them immediately. Needless to say, doing this saves much of the time and money that would have to be spent to fix bugs in a later development stage or even in production.
Prioritize Detailed & Quality Test Reporting: Automation should serve to reduce the amount of time QA teams have to spend verifying test results. Set up adequate reporting infrastructure with the right tools which generate detailed and high-quality reports for every test.