Test Plan Profiler 2.0

REVISION HISTORY					
Ver.	Description of Change	Author	Date	Approved	
				Name	Effective Date
1	Creation	Andrei Barycheuski, Ekaterina Zavizion	27.11.2022		
2	Update	Andrei Barycheuski, Ekaterina Zavizion	29.11.2022		
3	Update	Andrei Barycheuski	16.12.2022		

Related Artifacts
JIRA Project
QASpace
Confluence

Abbreviations and Acronyms	
TP	Test Plan
CL	Check-list
TC	Test case
BR	Bug report
TRR	Test result report
APP, application	web-application "Profiler 2.0"

1. Introduction

This document describes the approach and methodologies used by the testing team to plan, organize and perform the testing of web-application "Profiler 2.0".

Web-application "Profiler 2.0" is intended to meet the needs of HR specialists, students, trainers, employers.

HR specialists of HR department will be provided with automation in the resume template filling system, data filtering, notification management, document submission control, elimination of manual labor when compiling sets of documents for future employers, implementation of an assessment system for each student, providing potential employers with access to the rating and student's resume with portfolio.

Students will be provided with capabilities to fill out a single resume template, the ability to change the type of text, resume colors, communicate with an HR specialist in one structural space, receiving email notifications about a new message in the APP personal account.

Trainers will be provided with capabilities to view the list of students who participate in the Career Development Program , fill out a student review, view the student's profile.

Employers will be provided with opportunity to close staff shortages by search for an employee who meets criteria that are important for the company.

The Plan additionally details features to be tested, testing risks, resources involved in testing, strategy and schedule.

2. Scope of Work

Below is a list of ‘In Scope’ items that are expected to undergo test.

2.1. Components and Functions to be Tested

#	Application/ component name	Function name	Reference
1	Profiler 2.0/Authorization	User authorization in the system	*VS FE-2*
2	Profiler 2.0/Personal cabinet	Creating a personal cabinet	*VS FE-3*
3	Profiler 2.0/Personal cabinet	Editing a personal cabinet	*VS FE-4*
4	Profiler 2.0/Personal cabinet	Viewing a personal cabinet	*VS FE-5*
5	Profiler 2.0/CV template	Filling out a resume template	*VS FE-6*
6	Profiler 2.0/CV template	Editing the resume template	*VS FE-7*
7	Profiler 2.0/CV template	Viewing a resume template	*VS FE-8*
8	Profiler 2.0/Admin	Admin functions	*VS FE-49 - FE-58*

2.2. Components and Functions Not to be Tested

#	Application/ component name	Function name	Reference/Comment
	Profiler 2.0/Registration		*VS FE-1*
	Profiler 2.0/Link to LinkedIn		*VS FE-9 - FE-11*
	Profiler 2.0/Cover letter		*VS FE-12 - FE-15*
	Profiler 2.0/Video presentation		*VS FE-16 - FE-18*
	Profiler 2.0/Data filter		*VS FE-19 - FE-25*
	Profiler 2.0/Data Archive		*VS FE-26*
	Profiler 2.0/Provide employer with data		*VS FE-27. FE-47*
	Profiler 2.0/Message sending		*VS FE-28*
	Profiler 2.0/Student's review		*VS FE-29 - FE-31*
	Profiler 2.0/Message sending		*VS FE-32*
	Profiler 2.0/Student's notes		*VS FE-33 - FE-44*
	Profiler 2.0/Evaluation system		*VS FE-45*
	Profiler 2.0/Integration		*VS FE-46*
	Profiler 2.0/Student's feedback		*VS FE-48*

3. Quality and Acceptance Criteria

Entry criteria:

-development activities related to Sprint scope are completed

-initial tests are performed by development team.

Suspension criteria:

-hardware / software not available at the time indicated in the project schedule;

-assigned test resources are not available when needed by the test team;

-smoke tests are failed;

-the environment become unstable;

-change of business requirements.

Resumption criteria:

-hardware / software are available again;

-assigned test resources are available when needed by the test team

-all the blockers are fixed;

-the environment become stable;

-necessary changes to test cases have been made

Acceptance criteria:

-all created test cases are executed;

-all found defects are reported to JIRA;

-test artifacts such as Test Plan, Test Result Report, etc. are updated and uploaded to JIRA Project, QASpace;

-the product bug level should reach the acceptance criteria defined in Requirements, or if other is not specified, the product should not have bugs with priority Critical and Major to be released for production.

4. Critical Success Factors

Meet a schedule and complete development and testing of all functionality in term.

Scope of work is defined, functional requirements do not have last minute changes.

5. Risk Assessment

Risk	**Probability**	**Impact**	**Mitigation steps**
The dev team consists of juniors with lack of experience, which may cause increasing time for the development process and decreasing time for the testing	High	High	The scope of work for the first sprints should be small to give the team an understanding of their capabilities
Stuff turnover as team participants can find some job or loose an interest in this project	Medium	High	The created test documentation should be clear and detailed (teat cases, not check lists)
Last minute functional requirement changes from customer/BA may cause increase of the workload for project members	Medium	High	Functional requirements should not be changed after starting the sprint
Delay in fixing bugs. This can lead to a slowdown in project activity	Medium	High	It is necessary to monitor the level of workload of project participants
Missing the deadline	Medium	High	It is recommended to finish all dev tasks one week before the deadline, so that there is reserved time left for testing activities. Set up code-freeze date
Any participant of the project team can get sick	Low	Medium	Depends on the number of sick participants and their role in the project on the particular stage of the sprint
Force-majeure circumstances	Low	Medium	Depends on the kind and impact of the force-majeure circumstances

6. Resources

6.1 Key project resources

#	**Project Role**	**Name**
1	PM/SM	Alexandr Oznobishin

2	PM/SM	Alevtina Polyschuk

6.2 Test Team

#	Project Role	Name	Responsibilities
1	QA engineer	Andrei Barycheuski	TP creation, requirements testing, TC creation, Smoke Test execution, Functional Testing, Regression Test execution, BR, Bugs verification, TRR
2	QA engineer	Ekaterina Zavizion	TP creation, requirements testing, TC creation, Smoke Test execution, Functional Testing, Regression Test execution, BR, Bugs verification, TRR

6.3 Test hardware

#	Role	Resource	Hardware configuration	Software configuration	Owner
1	Test equipment	a computer	laptop Samsung DESKTOP-TBA51RE (CPU - Intel(R) Celeron(R) CPU 1007U 3GHz, RAM - 4GB, GPU - Intel(R) HD Graphics)	Win 10, 64 - bit	Andrei Barycheuski
2	Test equipment	a computer	laptop Acer NITRO AN515-56 (CPU - Intel(R) Core(TM) i5-11300H 3.11 GHz, RAM - 16GB, GPU - NVIDIA GeForce RTX 3050 Laptop GPU)	Win 10, 64 - bit	Ekaterina Zavizion

6.4 Test tools

#	Tool	Comment
1	Jira	Bug Tracking system
2	Confluence	Workspace for teamwork
3	MS Word	Text Editor
4	MS Excel	Spreadsheet Editor
5	Jira Test Management	Test Management
6	Postman API	API testing
7	Oracle VM VirtualBox	Creating macOS virtual machine
8	Xray Test Management for Jira	Test Management
9	MySql 8.0	DataBase management
10	Fiddler Classic	Original Web Capturing Tool for Windows
11	Google Drive	Storing Tracabeality Matrix
12	Google meet	Online meetings with a team
13	Skype	Chat with team

6.5 Testing environment

Recommended browsers for testing

#	Resource	Hardware configuration
1	Google Chrome	mandatory
2	Safari	mandatory
3	Microsoft Edge	mandatory
4	Firefox	mandatory
5	Samsung Browser	mandatory

6	Opera	mandatory
7	Yandex Browser	mandatory
8	Android Browser	mandatory

Operation systems for testing

#	Resource	Hardware configuration
1	Windows 10	mandatory
2	MacOS	mandatory

7. Testing Strategy

Testing Strategy is an evolving document detailing the processes and way we are going to ensure the quality of our product going forward.

Testing is a part of QA. It allows us to determine the level of quality of the feature(s) that we are assessing.

It is not the sole responsibility of testers to carry out QA. The entire team can and should contribute to ensure a high level of quality of the products and services being delivered.

APP will be tested using a “Grey box” approach, which is based on the partial knowledge of internal structure of the application. The purpose of grey box testing is to search and identify the defects due to improper code structure or improper use of application.

In the project there are 3 types of testing that should be conducted.

Unit Testing: Test the smallest piece of verifiable software in the application, performed by developers

System Testing: Conducted on a complete, integrated system to evaluate the system compliance with its specified requirements, performed by QA engineers

Acceptance Testing: Verify whether product meets customer requirements for acceptability, performed by the customer/user

We will be adopting a proactive and risk-based testing approach.

Proactive — This means that the test design process is initiated as early as possible in order to find and fix the defects before the build is created.

Risk-based — This means that we will organize our testing efforts in a way that reduces the level of product risk when the system ships. Risk-based testing uses risk to prioritize and emphasize the appropriate tests during test execution. Risk-based testing involves both mitigations — testing to provide opportunities to reduce the likelihood of defects, especially high-impact defects — and contingency — testing to identify workarounds to make the defects that do get past us less painful. Risk-based testing also involves measuring how well we are doing at finding and removing defects in critical areas.

7.1 Test Methods

Testing is the process of attempting to find discrepancies between the software and its functional specification/ requirements. The goal is to make sure that all functions of the application work as expected.

Manual functional testing – is considered as the main QA method of the application testing. The order of the tests is: simple positive, simple negative, complex positive, complex negative.

7.2 Test Types

Type	Test objective	Frequency
Requirements testing	To eliminate defects in requirements (omissions, contradictions, gaps, ambiguity, etc.)	Every time new requirements are delivered
Feature testing	To verify that each feature (function) of the software operates in accordance with functional specification (user story, use case, requirements specification, etc.)	Every time using test cases, once the feature is delivered by development team
Test Case Based Testing	To ensure the functionalities of software application work as stated in requirements. Testing is based on pre-prepared test cases.	Every time in scope of feature testing and bug verification

Ad-hoc / exploratory testing	To explore the software/system, gain additional knowledge; to break the system and find defects; find unexpected, contradictory functionality which is not covered by specification	Every time in scope of feature testing and bug verification
Graphical User Interface Testing (GUI)	To ensure the functionalities of software application work as stated in specifications by checking screens and controls like menus, buttons, icons, etc	Every time in scope of feature testing. Based on the provided by designers specifications
Usability Interface Testing (UIT)	To verify that user interface meets design guidelines; To find defects in UI design implementation (layouts, colors, fonts, font sizes, graphical elements, labels, etc.); To ensure UI controls, input fields work as expected;	Every time in scope of feature testing. Based on the provided by designers specifications
Compatibility testing	Verify that the software is capable to run: · in different browsers; · on different types of operating system	Performed every time in scope of smoke testing and GIU
Confirmation /verification testing	To verify that bug was fixed	Every time when corrected version of APP is available
Regression testing	To confirm that no new defects were introduced in unchanged areas of the software	Every time when corrected version of APP is available TBD
Integration testing	To verify interactions of different components of the system, interfaces	Performed in scope of feature testing, once the bundle with new feature(s) is provided

7.3 Test Levels

7.3.1 Smoke Test

Smoke Test is performed to quickly assess the readiness of the product for further more deep and thorough testing. It includes testing APP major functions which will be used most often.

If Smoke Test failed, Testing Team sends notification and suspends testing until corrected version of the product is available.

7.3.2 Critical Path Test

Critical Path Test will be performed after Smoke Test is passed. The goal of the Critical Path Test is to find bugs that could affect the major functionality of the application that is most important for the product users. Critical Path Test will be performed manually according to application Test Cases document on all platforms to be certified.

7.3.3 Extended Test

The *Extended Test's* goal to find bugs related to the non-typical but still possible and likely usage scenarios. Extended Test will be performed both according to test cases and using ad hoc testing scenarios.

7.4. Bug and Documentation Tracking

7.4.1 Documentation flow

#	Title	Responsible person(s)	Frequency (delivery time)	Method of delivery
1	Test Plan	All team	Once before the testing start, updates – upon changes	Confluence
2	Checklist/Test Cases	All team	Before the testing start	Jira
3	Bug Reports	All team	Upon finding the bug	Jira
4	Test Result Report	All team	Once per sprint (two weeks)	Confluence

7.4.2 Test case attributes:

#	Parameter	Type of the parameters	Content
1	TC ID/key	Mandatory	Assigned automatically in Jira

2	Project	Mandatory	<Name of the project>
3	Issue type	Mandatory	"Test (Xray)" label
4	Summary	Mandatory	Summary specifies clearly what feature or check TC applies to (e.g. Sign In: API: Sending valid correct values, Sign In: Enter personal cabinet with valid correct credentials)
5	Test type	Mandatory	Manual
6	Priority	Mandatory	Define the order in which a TC should be executed in accordance to business value or other project/ customer needs: **Blocker, Critical, Major, Minor, Trivial**
7	Requirements	Mandatory	Link to requirements (Filled in in field Description)
8	Precondition (s)	Optional	**Required conditions** before executing test steps (In XRAY TM ticket have to be created for pre-condition)
9	Test steps	Mandatory	New action = new step: **1. Step 1 2. Step 2 ...**
10	Test data	Optional	
11	Expected Result(s)	Mandatory	Describe **what should happen** in the results of steps execution.
12	Reporter	Mandatory	
13	Assignee	Optional	
14	Linked Issues	Mandatory	Link to the **User Story/Task** TC applies to
15	Test set	Optional	
16	Test execution	Mandatory	New Test execution ticket have to be created for each Sprint and each invironment
17	Test plan	Mandatory	New Test plan ticket have to be created for each Sprint
18	Linked Bugs	Optional	Link to the **Bug** the TC applies to
19	Attachments	Optional	• Pictures (screenshots) • Files (any kind of logs) • DB query (to get test data) • Documents • Video (just in case of hard reproducible bug)
20	Status	Mandatory	• **Untested** • **Passed** • **Failed** • **Blocked**

7.4.3 Bug report attributes:

#	Parameter	Type of the parameters	Content
1	Bug ID	Mandatory	Assigned automatically in Jira
2	Project	Mandatory	<Name of the project>
3	Issue type	Mandatory	"Bug" label
4	Summary	Mandatory	Short summary of the defect: **What – Where – When**
5	Component/s	Mandatory	"Back-end" or "Front-end" label
6	Environment	Mandatory	
7	Priority	Mandatory	Define the order in which a defect should be fixed in accordance to business value or other project/ customer needs: **Blocker, Critical, Major, Minor, Trivial**
8	Epic Link	Mandatory	Link to the related Epic
9	Reporter	Mandatory	
10	Assignee	Mandatory	Developer who was responsible for the task execution
11	Linked Issues	Optional	Link to the **User Story** the defect applies to
12	Precondition(s)	Optional	**Required conditions** before executing the actual steps (filled in descriprion field)
13	Steps to Reproduce	Mandatory	New action = new step: **1. Step 1 2. Step 2 ...** (filled in description field)
14	Actual Results (s)	Mandatory	Describe **what actually happened**, what currently happens when the bug is present (filled in description field)

15	Expected Result(s)	Mandatory	Describe **what should happen** if the bug was fixed (filled in description field)
16	Attachments	Mandatory	• Pictures (screenshots) • Files (any kind of logs) • DB query (to get test data) • Documents • Video (just in case of hard reproducible bug)
17	Sprint	Mandatory	Sprint name
18	Status	Mandatory	• **To do** • **Ready for Dev** • **In Progress** • **In Review** • **Ready for QA** • **In QA** • **Approved/Done** • **Pre-Production** • **Blocked/On-hold**
19	Comments	Mandatory	When send bug to Done it is obligatory to write comment: "The issue was validated, fixed" "The issue was validated, reopened. The bug is still reproduced using the steps above" "This issue was closed"

7.4.4 Bug Priority Definitions

Priority	Description
Blocker	Bug blocks the story/epic from being worked on during the sprint.
Critical	Bugs that are mission critical to the core functionality. Must be fixed immediately / in the next build (rashes, loss of data, severe memory leak).
Major	Bugs that are related to the core functionality, but don’t have to be fixed before product launch. However, these bugs should be fixed in the first available patch or release after launch.
Minor	Bugs that do not interfere with core functionality and are just annoyances that may or may not ever be fixed. Bugs have workarounds that allow users to accomplish the desired task that the bug may have hindered or the function may still operate but in a degraded fashion. Must be fixed in any of the upcoming builds but should be included in the release.
Trivial	Bug has no impact to the functionality. Cosmetic problem like misspelled words or misaligned text.

7.5 Testing schedule

#	Activity	When	Assignment
1	Test plan creation	To be kept up-to-date when needed	QA engineer
2	Requirements testing	Every time new requirements are delivered	QA engineer
3	Test cases creation	Each time the new functionality is developed	QA engineer
4	Smoke Test execution	Each build (when new version provided)	QA engineer
5	Functional Testing	Once Test case have been written and the feature is in “Ready for QA” status	QA engineer
6	Regression Test execution	Every time new build is delivered on PROD environment	QA engineer
7	Bugs Reporting	Once a defect is found	QA engineer
8	Bugs verification	Each time new build with Bug fixes is deployed on any environment	QA engineer

9	Test Results Report / Sign-Off Report creation	After testing has been completed and the application is ready to be passed to the Customer	QA engineer

8. Project task and estimation

Task	Members	Estimate effort
Create Test Plan	QA engineers	10 man-hours
Requirements testing	QA engineer	50 man-hours
Create Test Cases	QA engineer	200 man-hours
TC Execution + Bug reports	QA engineer	260 man-hours
Bugs Verification	QA engineer	40 man-hours
Regression testing	QA engineer	30 man-hours
TRR	QA engineer	50 man-hours
Total		**640 man-hours**