Test Plan

Productivity Web Application

Prepared by: Lyutfi Ismail

01/11/2021

Table of Contents

[1. Introduction 5](#_Toc86958270)

[2. Objectives and tasks 6](#_Toc86958271)

[2.1. Objectives 6](#_Toc86958272)

[2.2. Tasks 6](#_Toc86958273)

[3. Scope 7](#_Toc86958274)

[3.1. General 7](#_Toc86958275)

[3.2. Tactics 7](#_Toc86958276)

[4. Testing strategy 8](#_Toc86958277)

[4.1. Unit Testing 8](#_Toc86958278)

[4.2. System and Integration Testing 8](#_Toc86958279)

[4.3. Software quality assurance testing 9](#_Toc86958280)

[4.4. User Acceptance Testing 11](#_Toc86958281)

[4.5. Automated Regression Testing 12](#_Toc86958282)

[5. Hardware Requirements 12](#_Toc86958283)

[6. Test Schedule 13](#_Toc86958284)

[7. Control Procedures 14](#_Toc86958285)

[7.1. Problem Reporting 14](#_Toc86958286)

[8. Features to be tested 15](#_Toc86958287)

[9. Features not to be tested 15](#_Toc86958288)

[10. Recourses & responsibilities 16](#_Toc86958289)

[11. Dependencies 17](#_Toc86958290)

[12. Risks and assumption 18](#_Toc86958291)

[13. Tools 19](#_Toc86958292)

# Introduction

This project is part of the Individual Track of semester 3 in Fontys UAS. The project aims to display the abilities of the developer to create a full-stack web application that include a predefined set of requirements at its final stage.

The goal of this project is to develop a web-application, which user will use to improve their productivity. By setting goals and to-do items, they will be given a chance to organize their work so that they can improve their time management and overall productivity.

# Objectives and tasks

## Objectives

For creating a well-rounded and feature rich application, while using an agile methodology of development, continuous changes will have to be made to the product. In 3 week stretches (called sprints) a new version of the project will be supplied to the client. For a smooth improvement between the versions of the application and also a general control of functionality, needed parts of the project will be tested. This will be done in multiple ways to ensure coverage of all possible problems that may arise.

## Tasks

Task for implementing testing to the project will include:

1. Researching the best way to utilize testing methods

2. Deciding on exact methods that will be used

3. Managing when to implement which testing methods

# Scope

## General

While the project is in a development phase, we will test the connectivity of the web-app and the API that supplies its data. Also, the API itself has to be tested to ensure that the different units of code that are responsible for certain functionality are working as intended.

## Tactics

The tactics we will use for the testing are pretty simple. We will first use unit testing for the methods and objects found in our classes and after the unit testing is completed, we will use GitLab runners for the integration testing to take place when the code is pushed to the GitLab repository. If an integration test fails, then the repository maintainer will be notified that his commit was flawed.

# Testing strategy

## Unit Testing

Definition:

For the unit testing phase, we plan to test the code and logic of every class found in our project, so as a result every class will have its own test class. Because of this practice we will have a better understanding of why something went wrong and find a solution faster. For each class we will unit test every feature and the process will be declared complete when all the tests have passed.

Methodology:

The unit test scripts will be written in the back-end where the IDE that compiles the code will be used to run the unit tests and check the main functionality of the given code.

## System and Integration Testing

Definition:

System Integration Testing is defined as a type of software testing carried out in an integrated hardware and software environment to verify the behavior of the complete system. System and Integration Testing is performed to verify the interactions between the modules of a software system.

For Example, software and/or hardware components are combined and tested progressively until the entire system has been integrated.

Methodology:

The System & Integration testing will take place when the developer pushes something in the project repository, the test scripts will be written by the developer that pushes his code to the repository and the tests will be run on a GitLab runner. We will have a branch for every feature of the project and separate GitLab runners to test those features. This way we have a more specific testing that will help the developer find and fix errors faster but also the code will be more maintainable and avoid merging errors.

## Software quality assurance testing

Definition:

Software quality assurance is a means and practice of monitoring the software engineering processes and methods used in a project to ensure proper quality of the software.

This will guarantee that the code that is being written will fallow guidelines for best software practices. It includes standards and procedures that managers, administrators or even developers may use to review and audit software products and activities to verify that the software meets quality criteria.

Methodology:

This test will be done using the SonarQube software, that can be linked with GitLab, so that everytime new changes are pushed, the new code generated meets the standards that SonarQube checks for.

## User Acceptance Testing

Definition:

The purpose of the acceptance test is to confirm that the system is ready for operational use. During acceptance tests, end-users (customers) of the system compare the system to its initial requirements.

Participants:

A group of people (end-users) will be selected. This will include people:

1. Past users of the client’s other products

2. Of different social groups

3. With different background

Methodology:

Firstly, a state of the product, ready for user testing, needs to be achieved. From there, a test script will have been prepared. This will guide users, unfamiliar with the application, through its features. The testing will be monitored and recorded. This will provide data needed to further understand the usability of the application.

Needed insights will be the following:

1. Flow of most frequent usage

2. Coherence

3. Ease of use

4. Bugs that may appear

5. Rating of the app itself

## Automated Regression Testing

No automated regression tools like Selenium or Katalon Studio will be used in the project for the moment, as CI pipeline in GitLab would be created. This pipeline would automatically build the application and run the tests each time a new feature is added to the main branch, ensuring that the new functionality has not caused any unintended effects and that the system or component still works as specified in the requirements.

# Hardware Requirements

For the different methods of testing, a list of hardware components will be required.

For the continuous integration, a computer capable of handling an uninterrupted deployment of the GitLab-runner will be needed

For the user acceptance testing, a computer that will run both the front-end and the back-end of the application will be needed

# Test Schedule

The following testing methods will be initialized with the first software components that are added:

1. Unit testing - all new software changes will be accompanied with corresponding Unit tests

2. Continuous integration testing - with each change in the project online repository, a pipeline will be initiated to ensure functionality remains in the application

User acceptance testing will begin with the development of a version that has a responsive front-end and a minimum amount of features that can be presented to a test audience.

# Control Procedures

## Problem Reporting

Problems will undoubtedly arise. Most reported problems are expected to be received from either User testing. This testing method provides an “error message” containing information of where, what, and why a test has failed. That way these issues will be almost immediately dealt with. If or when an instance of Continuous integration testing fails, again an “error message” accompanies the raised problem that gets recorded and saved for when the problem needs to be fixed. In this case, the developer will have to troubleshoot the issue and resolve it as soon as possible.

# Features to be tested

● Communication between our API and the front end client.

● The web-app receiving information from the API.

● The app displaying information as intended.

● The web-app being able to take user input.

● The web-app sends the user input to the API.

# Features not to be tested

The following functions will not be tested:

- this list currently empty, as we are dealing only with software components that to the most part can be tested. However, as the project grows, there can be parts where testing cannot be done in a sensible way and would the use of too many 3rd party tools to complete.

# Recourses & responsibilities

The developer will be responsible for their code and the unit testing that comes with it. If a pushed code fails, the integration test then the developer who pushed it will be notified so he can start fixing it. The unit testing environment will be the IntelliJ IDE and for the integration testing we will use GitLab Runner and the GitLab version that the Fontys institution provides.

# Dependencies

The main dependency would be the 3rd party tools that will be used to for the testing.

# Risks and assumption

The main risks for the testing of the project:

- The introduction of new tools that have to be used when testing. Since every new tool has a learning curve, this means that the developer cannot immediately implements some testing procedures before fully understanding the proper way to use them

- Time management. Because the project follows the agile/scrum methodology, each sprint new improvements of the project have to be delivered. Ideally, no new features will be delivered before they are appropriately tested and ensure their expected functionality.

New risk may arise during the development to the project that will be added to this list.

# Tools

The developer would use a number of automation tools for testing. If needed, new tools could be added to the list at a later date.

● CI/CD GitLab pipeline - it would automate the integration and deployment and would ensure that each time a new change has been made and added to the main application, no unexpected behavior would occur. By using GitLab’s pipeline, issues would be caught early and the number of integration problems would be reduced, allowing the developer to deliver software more rapidly.

● SonarQube – it will ensure the quality of the software, which will help the developer of the project to abide by the guidelines for writing clean and understandable code.