Professional Bachelor  
Applied Informatics

Test plan

Part of the informatics project   
supported by

Artesis Plantijn College University

FacilitAP

Academic year 2019-2020 2nd semester

*The purpose of a test plan + test execution is to persuade the reader that the built application is robust and meets predefined criteria. A reader should be able to answer two questions based on these documents. The first question being: "What were the criteria set for building the application", that’s what the test plan is for. In other words: "When do we consider the application to be 'good enough'".*  
*The intention is that the reader will receive enough information to be able to judge whether these criteria are valid and sufficiently stringent. You don't want the reader to get the impression here that you are making yourself comfortable by setting the bar very low. On the other hand, you do not want to set the bar too high because then it won’t be feasible. You can persuade the reader of this by devising the right tests and the right criteria and by showing that they were drawn up in advance.*

*The second question to be answered is of course: "Given the criteria set, how well did you end up doing it". This is what the test execution is for and this is all about the results of the tests. But it is not enough to simply paste the results into the test plan. You will have to demonstrate that you are able to interpret those results correctly and that you are following them up correctly. If a test fails, you have to document what went wrong and how you finally managed to solve the problem.*

***(Don't forget to remove this and other italicized template text for the document versions ≥ 1.0)***

Contents

[Version Control 3](#_Toc40459174)

[Terms and abbreviations 3](#_Toc40459175)

[Introduction 4](#_Toc40459176)

[Project description 4](#_Toc40459177)

[Stakeholders 6](#_Toc40459178)

[Risk Analysis 6](#_Toc40459179)

[Test strategy 8](#_Toc40459180)

[Tools 11](#_Toc40459181)

[Literature 11](#_Toc40459182)

# Version Control

| N° | Date | Distribution | Changes |
| --- | --- | --- | --- |
| 0.01 | 2020-05-08 | Kobe De Peuter, Vincent Hertens, Dana Tabatabaie Irani, Carl Geeroms | [draft] Adjust front-page placeholders to our group and fill in project description, stakeholders and test strategy. |
| 0.02 | 2020-05-10 | Dana Tabatabaie Irani, Vincent Hertens, Carl Geeroms | [draft] Add introduction text to version control and terms and abbreviations, grammar check project description and stakeholders. |
| 0.03 | 2020-05-11 | Dana Tabatabaie Irani, Vincent Hertens, Yanu Szapinszky , Carl Geeroms | [draft] Start risk analysis, introduction, correct project description, add more stakeholders. |
| 0.04 | 2020-05-12 | Dana Tabatabaie Irani, Vincent Hertens, Yanu Szapinszky , Carl Geeroms | [draft] Start test strategy and correct risk analysis. |
| 1.00 |  |  | [final version] |

The test plan has been drawn during the first few sprints of our project. To give a good overview of what changes have been made on which dates, the reader can find a summary of the changes for each version in the table down below.

# Terms and abbreviations

| Term | Description |
| --- | --- |
| *FacilitAP* | Name of our application. |
| *UI* | User Interface; the point of human-computer interaction and communication in a device or application. |

Several terms and abbreviations are used throughout the test plan. These are listed in the table down below along with their description.

TODO: sort alphabetically

# Introduction

*The introduction should explain why a test plan is important for your project. So it is not a plea for the use of software tests in general (everyone can find that on Wikipedia), but it should therefore specifically deal with this project. Also indicate for whom this document is intended. At the end of the introduction, in a separate section, also provide a brief summary of what the reader can expect in this document (2-4 sentences). You should also refer to the Blueprint here. The expected number of characters (including spaces): 750 - 1500.*

*\*Why should I write a testplan for MY project\**  
*\*What are the specifics\**  
*\*What CAN go wrong\**  
*\*Why should I test this part of the project and not everything\* (E.g. If we’re using a framework, we won’t be testing the framework, we’ll be testing the logic behind the framework.)*  
*\*What can the reader find in the next pages and in the testdocument\**  
*\*What will the tests be about\**

Our test plan describes our testing approach on our application *FacilitAP*. Test plans are made for multiple reasons. Not only is its purpose for the sake of delivering an optimised and bug-free application on its due date, but also for determining the quality of our deliverables.

A plan like this also describes the testing scope. A clear testing scope helps our team with testing only the most important functionalities and prevents us from putting any efforts in testing out of scope functionalities. But the biggest advantage overall is to give our client insights about our testing process, show him what tests are possible, which functionalities can be tested and what testing resources and equipment are needed. This will help us to gain his confidence.

When we take a closer look at our application, we can see that many things can go wrong. *FacilitAP* can be divided into three sections:

* front-end (what the user sees),
* back-end (database)
* middleware (connection between front- and back-end).

On the front-end, things like not seeing defects, export files of reports not being formatted properly etc. can go wrong. On the back-end everything related to storing files e.g. getting corrupt. The middleware layer can get outdated or new updates could break it.

In this document you will find the follow subjects:

* our stakeholders,
* risk analysis of the functionalities within our application defined in our blueprint,
* our testing strategies and which types of testing we will perform,
* our tools who make all this possible.

# Project description

*This section should roughly correspond to the summary from the Blueprint. It serves to briefly outline what the project is about without the reader having to look up the Blueprint. You can list the functionalities here, but keep it short. You do not have to say anything about testing here, it will follow later. The expected number of characters (including spaces): 750 - 1500.*

Students and the employees of AP encounter various defects daily. Not all these defects get reported to the facility services. Those who do are redirected unstructured and via different channels. They end up in what they call a black box: it’s not clear whether the reports have been received, which ones are getting worked on and which ones aren’t going to be treated. Therefore, defects get reported multiple times and it isn’t immediately clear if the defect has been solved or not.

Last academic year, 2018-2019, the students of applied informatics werechallenged to find a solution for this problem. They developed a mobile application that took care of these issues. However, the application was only available on Android and didn’t work on other platforms. This was a pity because not all the employees and facility services had an Android device at their disposal. Therefore, the application couldn’t be used, and the problem persists.

The problem has been reintroduced in our project and requires a similar approach. We must develop: a web application that is accessible to both computers and smartphones. This way, defects can be reported intuitively. Furthermore, it will be possible to submit tasks on both applications that vary from redesigning classrooms to the installation of projectors. In addition, it will also be possible to retrieve both active and archived reports.

# Stakeholders

*It is very important to mention everyone who was involved in the preparation of this test plan within this project. This, of course, includes developers and testers, including yourself. In addition to yourself, it is always better to have a significant part of the application tested by at least one other tester. If it concerns external people, make sure you have their approval to list them. The contribution can be analysis, sponsor, tester, proofreading, project management,….*

|  |  |
| --- | --- |
| Name | Contribution/Role |
| De Peuter, Kobe | Developer, Tester |
| Hertens, Vincent | Developer, Tester |
| Janssens, Maxim | Developer, Tester |
| Szapinsky, Yanu | Developer, Tester |
| Tabatabaie Irani, Dana | Developer, Tester and proofread test plan |
| Wuyts, Alec | Developer, Tester |
| Van Camp, Vincent | Testing mentor |
| De Pooter, Marijn | SCRUM master |
| Casal Mosteiro, Kelly | Functional mentor |
| Van Eyken, Koen | Soft skills mentor |
| Dils, Sam | Technical mentor |
| Geeroms, Carl | Product Owner |
| *List of roles around our project.* | |

# Risk Analysis

*Describe here the 10-20 most relevant risks specific to the project. These are risks that are linked to the operation of your application and* ***no meta-risks*** *that have to do with, for example, fire, lack of time, internet failure, illness, cooperation with your mentor and the like. There are several techniques and tools that can help you identify risks such as Ishikawa diagrams, a Delphi study, interviews with domain experts, "lessons learned" from projects (from the internship company), etc…. The use of accepting, avoiding, transferring, limiting exploitation (in case of positive risks). The severity of the risk and the corresponding impact are expressed as a number from 0 to 5. The priority can be calculated as rnd (⅕ × risk × impact).* ***Sort from high to low priority.***

*The purpose of the risk analysis is to prioritize the activities within the test strategy. It is often not possible to test everything and the risk analysis helps you to determine which functionalities you must cover first, which parts of the application are most at risk. The expected number of characters (including spaces): 1500 - 5000.*DB Connectivity

| Description | Risk | Impact | Prior. |
| --- | --- | --- | --- |
| 1. Customer suddenly demands more features. | 3 | 5 | 3 |
| 1. Misinterpreting the customer’s product vision. | 3 | 5 | 3 |
| 1. Application not supported on every browser. | 3 | 5 | 3 |
| 1. Users can’t login due to no connection with Azure Portal. | 2 | 5 | 2 |
| 1. Overload of maximum users that the application can handle. | 3 | 3 | 2 |
| 1. Accidentally deleting progress. | 3 | 4 | 2 |
| 1. Application not supported on every operating system. | 2 | 5 | 2 |
| 1. Deliverables not finished in time. | 2 | 5 | 2 |
| 1. Extreme network traffic at user-side by fetching images of defects/attachments of tasks. | 3 | 4 | 2 |
| 1. User uploads an image/attachment that exceeds the file limit. | 1 | 3 | 1 |
| 1. Delivery of necessary information by the customer is overdue. | 2 | 3 | 1 |
| 1. Loss of server credentials. | 1 | 4 | 1 |
| 1. Software licences expire. | 2 | 3 | 1 |
| 1. Downtime of our application because of server hardware failure. | 2 | 3 | 1 |
| 1. Database security breach. | 1 | 5 | 1 |
| 1. Images of defects/attachments of tasks get corrupt. | 0 | 5 | 0 |
| 1. Manual of the application isn’t clear enough for the customer. | 1 | 2 | 0 |
|  |  |  |  |
| Action | Action type | | |
| 1. In order to prevent scope creep, the customer must submit a change request. The change request will be reviewed and added to the sprint backlog for development. | Mitigate/Accept? Meneer Van Camp | | |
| 1. To ensure that the design and the features of the application satisfy the customer, we will work in Scrum. This way we will be able to deliver working code every 2 weeks. The customer can then confirm if he is satisfied with the given result. | Mitigate | | |
| 1. Most of our users will be using browsers that support *FacilitAP*. Otherwise we will inform them that some of the features may not work as intended. We try our best to develop the application, so it is available on all browsers. This is not always possible. Therefore, the users should use a supported browser for the best optimized experience. | Accept | | |
| 1. No steps can be taken to solve this issue. Microsoft must solve this problem. | Accept | | |
| 1. Based on stress tests, we determine the amount of users the application can handle. If the amount of users reaches the threshold, we can use the backup server to increase our capacity. | Mitigate | | |
| 1. In order to keep track of our progress, we use GitLab’s version management. This way, if someone accidentally deletes our progress, we can revert and get the removed progress back. | Mitigate | | |
| 1. We used OS-friendly software in order to eliminate this threat. | Avoid | | |
| 1. If we can’t finish our deliverables in time, we will work overtime to make sure we finish them. | Mitigate | | |
| 1. We will reduce the network traffic by compressing the images. Users will see the compressed images and the attachments will be shown as a preview. | Mitigate | | |
| 1. We will have a file limit so users can’t upload images/attachments larger than 100MB. This way, users only submit the most necessary files. | Mitigate | | |
| 1. We will contact the customer and ask them if they can provide us with the necessary information. If it takes too long, we will contact the people responsible directly. | Transfer/Accept?  Meneer Van Camp | | |
| 1. We will use a password manager to store our passwords securely and to reduce the chance of losing them. | Mitigate | | |
| 1. No steps will be taken to lower the probability. New licences will be bought when they expire. | Accept | | |
| 1. A backup server would be ideal for the application. When the server becomes incapable of running the application due to hardware failure, the backup server will take over its task. This way we can reduce the downtime to a minimum. | Mitigate | | |
| 1. User credentials are not stored in the database. They are handled by Azure Portal. Only information about defects and tasks will get exposed to the intruder. The intruder can’t do anything with the information of defects and tasks, no sensitive information is exposed. | Mitigate | | |
| 1. If the user uploads a corrupt image/attachment and our system doesn’t detect it, it will be stored as it is (as a corrupted file). If the corruption happens on our database, there's a chance we can restore it to its original state using our back-up files. | Accept | | |
| 1. After the development phase of our project, we will hold a presentation of the application for the customer where all his questions will be answered. If our application gets selected, we will hold an appointment with the customer to educate him thoroughly about every functionality of our application. We will also give the customer the opportunity to give feedback about the manual of the application. | Mitigate | | |
| *Project risks and the actions we take to counter them* | | | |

# Test strategy

*Here is a description of how you have integrated testing into the project approach. Have you performed regression tests every sprint? Have you provided a test phase at the end of the project? Etc ... Have you made certain persons responsible within the project for testing or did each team member perform an equivalent part of testing? What is your view on the test share in your project?*  
  
*Afterwards an overview follows of which type of tests have been carried out and which have not. The most common test types (see below) should all be discussed here and for each test type you have to justify why it was used or not. Make sure that the argumentation why you think a certain type of test was not carried out makes sense. If necessary, divide the tests into functional and non-functional tests. The following test types can be considered "current":*

* unit testen
* integratie testen
* systeem testen
* sanity testen
* interface testen
* regressie testen
* beta- of acceptatietesten
* performantie testen (load, spike, stress, …)
* security testen
* cross-browser en cross-systeem testen
* usability testen.

*Under “Criteria” you describe how successful and unsuccessful tests can be distinguished and what the expectations are in terms of coverage (coverage). Remember that criteria are measurable and should not be ambiguous. The expected number of characters (including spaces): 1000 - 3000.*

Maintaining and testing our application takes a lot of time and effort. This will be necessary to successfully complete customer requirements. Therefore, some tests will be automated. The most important part of this is to clearly define what will be tested and when we consider these tests to be successful. The table below shows which tests we perform, what the test contains and what the criteria's are. We will also note the tests we do not perform with the reason.

| Test type | Planned? | Coverage and criteria |
| --- | --- | --- |
| Unit tests | Yes | To ensure the correct functioning of our application on the due date and between each sprint, unit tests of every single map controller will be held. When writing the tests, not only do we cover the critical path, but we strive for having a 100% coverage of each line of code by testing all outcomes of a method. This way, unexpected behaviours can be detected in early phases and be correctly instantly. We define a success in our unit tests when every single test is succeeded, without an exception. |
| Integration tests | Yes | The application gets tested by combining units and testing them as a group. A success means that every unit test functions within the group. |
| System tests | Yes | The purpose of system testing is to test the application in its entirety. This way we get a clear picture of how the application is going to be used and the issues a user can come across in real-time. (TO DO: welke testen?) |
| Sanity tests | Yes | To check if new functionality/bugs have been fixed, we perform sanity tests. This is done manually by our team members whenever new functionality is added or bugs are fixed. Our sanity tests are successful when the newly added or adjusted functionalities work as intended. |
| Interface tests | No |  |
| Regression tests | Yes | To make sure our existing functionalities still work when new code is added, regression tests will be done. Every time a new functionality is added, or bugs are fixed, we rerun our unit tests. On top of that, we also manually test the functionality to see if everything still works properly. |
| Beta- or acceptance tests | Yes | To make sure the system meets the business requirements, our team tests the application and checks whether every requirement is present and functions properly. Every sprint, we have a meeting with our functional mentor who also checks the application and gives us feedback. This test is successful when every requirement is met. |
| Performance tests | Yes | In order to test the performance of our application, two type of tests will be done: load and soak tests. This will give us a good overview of the performance, but also the scalability and reliability of our application. Our performance tests are successful when the results do not exceed our predefined limits. (TO DO specifieke limits) |
| Security tests | Yes | Security is one of the most important things of an application. It ensures the safety of the users when using the application. Therefore, our team will test the different security aspects of the application. Access security is implemented by Role management. Our team tests this by creating various accounts with different roles. They then test if the functionalities correspond with the associated role. To make sure the application is protected against NoSQL injections and XSS (Cross-Site Scripting), our team tests if the input fields have a limited field length and can verify if the input is a script or tag. |
| Cross-browser and cross-system tests | Yes | Cross-browser and –system tests are done manually by our team members. The tests are executed by using our application on different browsers and systems. The results are successful when our application is supported on every required browser and operating system. |
| Usability tests | Yes | The application gets covered by multiple age groups: youngsters, adolescents, adults and elderly. Each one of them will receive a pre-defined checklist of specific actions that they must perform on the application. After succeeding (or failing) a task, feedback will be asked to see what went well and what can be improved. This way we will get an overall coverage report of each age group. The results of this test get defined by the success rate of all the tasks. A success is achieved when 85% percentage of tasks are performed successfully in each group. Failing this test means we must optimize the *UI*, so it becomes more intuitive. |
| *All the tests with their coverage and criteria* | | | |

# Tools

Provide a brief overview of the software (with version!), Hardware or services used to perform the tests. For software, state the name, version, and distributor, and use the citation below to refer to a website. For hardware (for example specific test servers) you write down the specifications. Where relevant, you should also describe how the test results of the software / hardware / services were captured and converted into the test results in this document. Make sure that no new software is introduced in the later text and that everything is central here

| Name | Version | Supplier | Description | Ref |
| --- | --- | --- | --- | --- |
| IntelliJ IDEA | 2020.1.1 | JetBrains | Back-end IDE for Java and SpringBoot |  |
| WebStorm | 2020.1.1 | JetBrains | Front-end IDE for VueJS, Nuxt and Vuetify. |  |
| JUnit | 4.12 | JUnit | Testing framework |  |
| Spring | 5.2 | VMWare | Framework for developing Java applications. |  |
| Jest? |  |  |  |  |
| Selenium |  |  |  |  |
| NodeJsScan | 3.7 | OpenSecurity | Node SAST-tool to detect security issues inside our application. |  |
| *The tools used to perform the tests* | | | | |

# Literature

|  |  |
| --- | --- |
| [1] | Jan, A. (2015-04-12). The title of this work. Collected from <http://xxxxxxxxx>. |
| [2] | Peter, S. (2012). Title of Peter’s work. *Journal of Infinitesimal Results 46*(2), 123-134. |