Operating Systems



*Midterm Project Documentation*

< **TITLE OF TASK / DOCUMENTATION >**

Authors: **Full name**, Neptun code, E-mail

**Full name**, Neptun code, E-mail

Institute: Óbuda University – John von Neumann Faculty of Informatics

Major: Computer Engineering BSc

Training Mode: Daytime training

Semester: 2022-23-1

1

**Table of contents**

[1 Written summary of task 3](#_Toc26794)

[2 Requirement specification 3](#_Toc26795)

[2.1 use-case 3](#_Toc26796)

[2.2 user-story 4](#_Toc26797)

[2.3 FURPS+ 5](#_Toc26798)

1. Evaluation of implementation options............................................................................................ 7
2. Overview of selected solution .................................................................................................. 8
3. Rendszerterv .................................................................................................................................... 9
   1. Hálózati terv ............................................................................................................................ 9
   2. Szerver1 ................................................................................................................................... 9
   3. Szerver2 ................................................................................................................................. 10
   4. Kliens ..................................................................................................................................... 10
4. Tesztelési terv ................................................................................................................................ 10
   1. Funkcionalitás tesztelése ....................................................................................................... 11
      1. Távoli elérés (SSH) tesztelése ........................................................................................ 11
      2. FTP tesztelése ................................................................................................................ 11
   2. Terheléses tesztek ................................................................................................................. 12
   3. Megbízhatóság ...................................................................................................................... 12
   4. Biztonsági tesztek .................................................................................................................. 12
5. Megvalósítási dokumentáció......................................................................................................... 13
6. Tesztelési jegyzőkönyv .................................................................................................................. 13

2

# 1 Written summary of task

* Write a summary of the choosen task from the perspective of the client. Do no just copy the description from the task sheet. What does the client want, what do you understand of it, what does it need.
* This is needed because the client and the developers can often understand tasks differently, or cannot communicate clearly to the other side.
* Think about elements that were not mentioned by the client directly, but are indirectly contained within the task description.

# 2 Requirement specification

In this section, you must prepare a functional and objective overview of the requirements of the system to be developed. There are some recommended format options for this below. These contain exact requirements that can easily be validated (whether they were included) in the finished system.

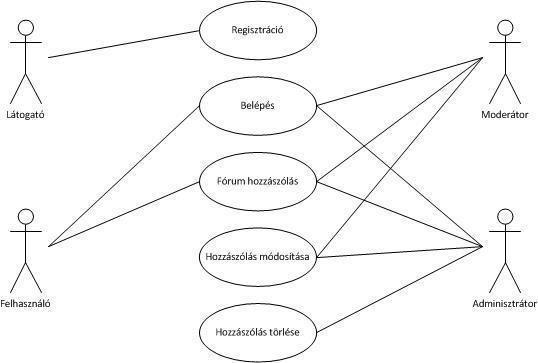
## 2.1 Use-case

1. Take the available user levels
2. Take the available functionality

1. Evaluate which user levels can access which functionalities. This can be done in a matrix, and/or use-case diagram. A text summary is also a nice addition.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | registration | Login | Forum commenting | Edit  comments | Delete comments |
| Guest | x |  |  |  |  |
| User |  | x | x |  |  |
| Moderator |  | x | x | x |  |
| Administrator |  | x | x | x | x |

The use-case diagram is another view of the same data you can see in the matrix above.



3

## 2.2 User-story

Optionally, or instead of the use-case option you can also describe the behavior of your system with a user story. (You can do either use-case or user-story, it is not mandatory to do both )

In this case the elements of the system are not highlighted as data. Instead, you give a detailed explanation of how a given functionality works from the point of view of a user.

For example you can go into details of the registration process of regular users:

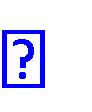
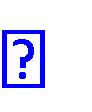
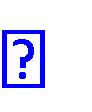
* After clicking on the Registration link, the system redirects me to a page where I can create a user account, which will allow me to participate in the forum.
* On the top of the page, there are warning texts and an option for accepting cookies. There are also the terms of service, which contains the requirements of the registration. I can click the square to declare that I have read the terms of service and agree with them.
* After clicking on the ‘Register’ button, my account was successfully created.
* So on and so on……

Describe some features and what possible interactions users have with them.

## 2.3 FURPS+

This is a very detailed description method. There is no step-by-step tutorial for it, the following pages can help:

<http://en.wikipedia.org/wiki/FURPS> <http://www.ibm.com/developerworks/rational/library/4706.html><http://www.ariasamp.net/brain-dump/guide-to-furps/>



**2.3.1** **Functionality**

**Requirements for the server:**

|  |  |
| --- | --- |
| Component | Requirement |
| FTP service | Anonymus ftp for downloads |
| FTP service | Read-only directory for outside users (clients) |
| FTP service | Write only directory for outside users (for example to upload error logs) |
| FTP service | Only secure connection (SFTP) is allowed from the outside |
| Mailing | Provide mobile (Andorid/iOS) access |

Additional main topics: Requirements for the client (computer), Security requirements, Usability, Trustworthiness, Performance, Maintainability.

This was a short example, if you choose this method please look up the sites listed above, or translate the Hungarian document.

1. **Evaluation of implementation options**

The aim is to explore our options in terms of software, hardware, operating system, network etc. for implementing the desired system.

**Required software for office environment:**

|  |  |  |  |
| --- | --- | --- | --- |
| Applications | Platform | Software | Compatibility\* |
| Ms Word, Excel, PowerPoint | Windows | Ms Office 2010 | Compatible |
| Ms Project | Windows | Ms Project 2010 | Compatible |
| Ms Visio | Windows | Ms Visio 2010 | Compatible |
| Ms Word, Excel, PowerPoint | Windows | LibreOffice | Not fully compatible |

Based on the evaluation above, the installation of Microsoft software (Office, Project, Visio) is recommended.

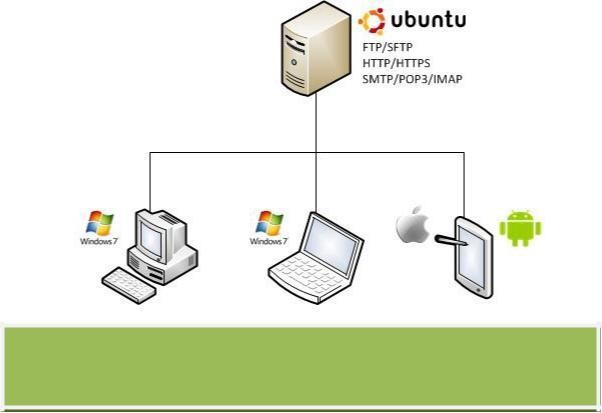
**Operating Systems:**

|  |  |  |
| --- | --- | --- |
| Operating System | Compatibility with MS software | Recommendation |
| Windows XP | Compatible | Limited due to obsolescence |
| Windows 7 | Compatible | Recommended |
| Windows 8 | Compatible | Might be uncomfortable interface Possible, but risky |
| Mac OS | Not fully compatible | Not recommended |
| Linux | not fully compatible | Not recommended |

Based on the evaluation above, clients should be set up with Windows 7.

1. **Overview of selected solution**

You should write a couple sentences about the planned solution, and provide a nice diagram which presents an overview of the system, like the one below.



This is the end of the first milestone for the documentation

.

The deadline for this is about

:

**Week 5**

**.**

1. **System plan**

As indicated by the section name, this is a plan. Therefore, this part shouldn’t consist of your actual implementation (commands and screenshots), but your plans of implementation. This is a plan, in which you think through the necessary steps of implementation. For example, you can list the host IP addresses, domain names, and users ahead of actually implementing them, so that you don’t have to come up with all these values during installation. These values can of course change during implementation, and they should be adjusted accordingly later.

The following sections give you some examples of what needs to be written, but your actual documentation should be in line with your own tasks.

### 5.1 Network plan

Each midterm project tasks requires you to deploy multiple virtual machines and connect them on a network. For this, it is a good practice to come up with a networking plan ahead of time, which includes IP addresses, subnet mask, default gateway, DNS server etc. In addition to the textual summary/table, it is beneficial to also create a diagram visualizing your network configuration.

### 5.2 Server 1

*Partition table:* The disk partitions created during installation, written in text or in table, contains information about partitions (name, filesystem, size..)

*Services:* We can list what services will run on the server. If one of them differs from the basic setting and it is necessary to differentiate it in the plan, then it is worth indicating this. It is possible to describe the exact application with which the service was implemented. Ex: Remote access with command line: Openssh - will work with AD directory users.

*Ports*: for the previously used service, it is possible to plan in advance which proton they should communicate with. For example, SSH uses port 2200 instead of 22.

*Domains*: What domain(s) will be created. For example hellokitty.local. For tasks 3 and 5, more of these are possible. It is worth describing which company or sub-unit it will belong to. The name of the server can also be specified here. E.g., Machine name: Server1.

*Design of users and groups*: At the first milestone, only the groups were defined for the use-case model. In this section, you can define which user will be created in which group. In addition to the burden of castration, usernames can be user1, user2, admin1, etc., but it is not recommended. Here you can determine (even in tabular form) what other properties a given user has. For example, with AD, you can define when and where you can log in, which shared folders you can use. This is optional, as these extra features depend on the task and your own decision.

*Folder structure*: Defining and listing the most important folders. Users' home folders, the location of common folders, or the location of scripts written for the system. The latter also occurs, for example, in optional tasks.

*Monitoring*: Planning what and how to monitor. What are the elements we monitor and where do we collect the information, where can it be viewed.

*Custom*: Eg: RAID type selection | Which print server will be installed, with what settings | Firewall settings, which are necessary for operation (task 5 is optional or ICMP echo reply permissions for tests)

9

### 5.3 Server2

This option comes up for tasks that use multiple servers. Its structure is the same as that of server1. It is true for both that the services that are available on the given server are listed and explained. Eg: Domain names on one server, then it will be explained there, while on the other, LDAP/AD, and the users part there. Duplicate only where it is necessary.

### 5.4 Client

*Machine Name*: Speaks for itself.

*Partition table:* As already known.

*Programs to be installed:* List of programs that will be installed on the client. Which browser will be installed, FTP client, compressor, etc...

*Individual: Ex:* Office installation: which service will be installed (Access, Word, etc.).

**6 Testing plan**

The same is true here as with the system plan. You don't need to do the testing here yet, you just need to provide a framework that describes what WILL be tested. The subject of the testing is whether we meet the requirements that we laid down in the first milestone. Thus, the FURPS used in the requirements can be used to group the tests, or it can also be performed according to the testing of the areas generally examined (functionality, load test, reliability, security tests). But you can even create your own grouping.

Tests should be designed to be reproducible by a third party and give the same results (unambiguous) under the same conditions." In other words, it must be properly described what will be tested, how it will be tested, with whom it will be tested and what it will be tested on. Then you need to define what results you expect from it. What has BEEN done does not need to be specified, that will happen at the third milestone.

### 6.1 Functional testing

It is likely that most of the test cases will be here, so it is worthwhile not to do this in a single table and the huge description before, but to deal with them in a separate section per function. For example:

#### 6.1.1 Testing remote access (SSH)

Console access to the server named "opreserver" from the "elsogep" client will be tested with multiple users. The putty program will be used as SSH client. It is intended to allow only users with administrator privileges to connect.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test** |  | **Username** | **Testcase description**  **Expected** | | **Obtained** |  | **Evaluation** |
| **ID** |  | **/Group** |  | **result** | **result** |  |  |
|  |  |  | Testing SSH |  |  |  |  |
| **F\_SSH\_1** |  | kovacsa/ seller | Connection With seller user | Unsuccessful  connection |  |  |  |
|  |  |  | Group |  |  |  |  |
|  |  |  | Testing SSH |  |  |  |  |
| **F\_SSH\_2** |  | kissp/ administrator | Connection  With administrator  Group | Successful  connection |  |  |  |
| **F\_SSH\_3** |  | ... | ... | ... | ... |  | ... |

#### 6.1.2 FTP testing

Testing FTP connection from "elsogep" to "opreszerver". Testing with directory, anonymous, local and virtual users. Since FTP is not encrypted, it is necessary to exclude the transmission of data of real users.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test** |  | **Username** | **Testcase description**  **Expected** | | **Obtained** |  | **Evaluation** |
| **ID** |  | **/Group** | **result** | | **result** |  |  |
|  |  |  | Testing SSH | |  |  |  |
| **F\_FTP\_1** |  | kovacsa/ seller | Connection Unsuccessful With seller user connection | |  |  |  |
|  |  |  | Group | |  |  |  |
| **F\_FTP\_2** |  | kissp/ administrator | Unsuccessful  Testing  FTP access connection  Administrator group | |  |  |  |
| **F\_FTP\_3** |  | anonymus | Testing Anonymus    FTP connection | Unsuccessful |  |  |  |
| Connection |  |  |  |
| **F\_FTP\_4** |  | virtualbela | Virtual FTP succesful | |  |  |  |

11

These are not the only two functions to be tested. Each task has its own features, it is worth designing a test case for the main one, with at least 3-4 test cases, but preferably more.

### 6.2 Load tests

In reality, this is a legitimate test. It is not required to be TESTED in the semester, but it should be included in the test plan. You can write down what test, software would be used for this, what would be expected - as there is a performance part in the requirements that would need to be met.

### 6.3 Reliability

Also a test required in a real system. For example, high availability can be tested. It is more difficult to test in this exercise: it is worth describing what software would be used to test it.

### 6.4 Security tests

Security test can come up in many forms, also worth writing up in tabular form, but can also be done in text. A few examples:

Vulnerability check with software. E.g.: nessus. Write down what you expect, which server, client to check etc.

Wireshark will be used to test if SSH traffic is encrypted between the two channels.

If the requirements were that SSH needed to have a port number change, then test if it is really not possible to use 22 for this purpose.

etc...

1. **Implementation documentation**

Once the design is complete, you can start building the system. The main steps of this process should be documented. Of course, for this mid-term assignment, it is not necessary to describe every single step - like the "onwards and upwards" of a Windows 7 installation. However, when installing or configuring the services that form the backbone of the system, it is worth describing a command or creating a "print screen" with the appropriate signature. Deciding what is "important" is quite subjective, especially for a first time documentation student, but as a help, think about what would be a step, setting or command that, if you were to do the system again, would be good to have written down and not have to browse again for a solution. After all, one of the purposes of this implementation documentation is precisely that, to aid in understanding and testing when reviewing or reinstalling the system.

At the same time, it can be noted if there is a deviation from what was planned. For example, what was defined in the specification could not be solved due to technical/resource or other problems. In this case, the alternative solution should be described.

1. **Testing logs**

Once the system is ready, testing is necessary, i.e. the test plan in section 6 must be implemented. Here, the system must be tested line by line and the results must be entered in the appropriate line, possibly with comments.

A further solution could be to have pictures to back up the tests. For example, testing a mail server where two users successfully exchange mail.

This concludes the documentation of the **second milestone**.