2. Analysis and specification of requirements

2.2. Guidelines for writing requests

Abstract

A well-structured requirements specification document makes it easier for us to organize, write, verify, and understand a set of requirements for a software solution. However, if the individual requirements are written poorly, a good structure will not guarantee that the requirements will be correctly interpreted, and therefore not correctly realized in design, implementation and testing activities. In this lesson, we deal with the topic of writing quality individual claims and a set of claims with the help of the widely accepted INCOSE guidelines [1]. Guidelines define a set of characteristics that individual requirements and sets of requirements should possess, and a set of rules that should be followed in order to achieve said characteristics.

Introduction

In order for the requirements specification to provide a solid foundation for further activities of the software development process, the requirements document should be clearly structured and easy to understand. A document organized in this way allows us to easily find requirements, understand their context, spot omissions or redundant requirements, eliminate conflicting or poorly written requirements, etc. However, another key factor in the quality of the requirements specification is the quality of individual requirements. The structure of the document itself will not help us much if the individual requirements are not written in such a way that they are clear, consistent, and precise in expression. The authors of the templates for defining the requirements specification (e.g., Volere and IEEE 830-198 SRS) were aware of this, so in addition to the structure of the entire document, they also provided general guidelines for the design of individual requirements. However, much more detailed and numerous guidelines for formulating individual requests can be found in the INCOSE guidelines for writing requests, which this lesson specifically focuses on.

Before we focus on guidelines for writing requests, we will look at the criteria by which we can classify requests. One of these criteria is the level of abstraction at which the request is written. In this sense, we can talk about several types of requests (ordered from higher level of abstraction to lower):

[1]Incose.org

- Statement of need ,
- **Stakeholder (user) requests,**
- **System (software) requirements,**
- System component requirements,
- Subsystem component requirements.

Given the level of complexity of the problem domain and the software solution that we will deal with, it will be sufficient for our purposes to talk about user requirements (i.e., what the user wants to achieve using the software) and software requirements (i.e., what the software must implement). We mentioned these two types of requirements in the last lesson when we talked about the specification of user requirements and the specification of software requirements.

Another important classification of software requirements that we will mention, and also the classification that we will focus on during this lesson, takes into account the purpose of the requirements. In this sense, we distinguish between functional requirements and non-functional requirements. The functional requirements of the software define what the software should do, how it reacts to stimuli from the environment and interacts with the user. Such requirements are sometimes called capabilities or options. Which functional requirements will be found in the requirements specification largely depends on the problem domain that the software deals with, the type of software, the users of the software and their needs and expectations, etc.

On the other hand, non-functional software requirements represent limitations that apply to the entire software. They affect the entire system architecture, not just individual components. Because of this distinct influence on the quality of the software solution, they are often called quality requirements, and can be more critical than functional requirements. Namely, if we fail to identify and thus implement a functional requirement, we can, with reasonable effort, implement it later. However, given that it potentially involves the entire software solution architecture, the subsequent implementation of a non-functional requirement is extremely demanding and sometimes impossible within reasonable deadlines and resources. Non-functional requirements usually deal with the topics of performance, reliability, availability, portability, ease of use, security, privacy, hardware, etc. Although it is important for functional requirements to be precisely expressed, it is even more important for non-functional requirements because it is more difficult for them to be objective determine whether they are fulfilled or not. Therefore, non-functional requirements often contain a quantitative component that will be an indicator of the fulfillment of the requirements (e.g., number of transactions per second, average time to failure, response time, megabytes of memory, number of simultaneous users, etc.). In this way, less space is left for subjective interpretation of success or failure in fulfilling the given requirement.

In the software requirements specification template that we started to fill out in the previous lesson (chapters 1 and 2), functional (chapter 3 of the template) and non-functional requirements (chapter 4) have their place. Unfortunately, as with most activities that fall under the domain of creative work, there is no precisely prescribed series of steps that we should follow in order to have guaranteed correctly formulated requirements. However, guidelines such as the INCOSE requirements writing guidelines offer heuristics that can help us do this. Of course, like any other heuristic, these guidelines should be applied carefully and meaningfully, and not literally without critical judgment.

Guidelines for writing requests

The INCOSE guidelines are often used in practice and cited in the literature for writing requests. They begin by defining 9 characteristics (C1 – C9) that each individual claim should possess in order to be considered correctly written. In addition to these, the guidelines also define 5 characteristics (C10 – C14) that a set of requirements should have in order to be considered correct. Regardless of whether it is the characteristics of a single request or a set of requests, each characteristic is associated with a rule or more rules that help request authors achieve the prescribed characteristics. There are a total of 41 rules (R1-R41), where each rule can contribute to the achievement of one or more characteristics. Given that the original guidelines are in English, some of the rules are specific to the English language and could not be meaningfully applied to the Croatian language. Such rules are omitted (e.g., R3 and R5).

Requirements characteristics

#	Characteristic	Description of the characteristic	Rules
C1	Need	The requirement should define the necessary capability, characteristic, limitation or quality factor. If this requirement were omitted from the set of requirements, its lack would not be possible to compensate for the other requirements from the set.	R30
C2	Suitability	The specific intent and amount of detail provided in the request should be appropriate given the level of abstraction we are at at the time.	R31
C3	Unambiguity	The request should be written in such a way that it can be interpreted in only one way, regardless of who interprets it.	R1-R37
C4	Completeness	The requirement describes the necessary capability, characteristic, limitation or quality factor in a way that provides	R6-R9,

		sufficient information for understanding, i.e. no additional information is needed to understand the requirement.	R18, R24, R25, R33, R34, R35, R39
C5	Singularity	The request should refer to exactly one possibility, characteristic, limitation or quality factor.	R9, R18-R23, R39
C6	Feasibility	The request should be feasible regarding the existing limitations (e.g., budget, deadline, equipment, legislation, security, etc.) and with an acceptable risk.	R26, R33
C7	Verifiability	The request should be formulated in such a way that it can be checked whether the request has been fulfilled or not.	R1-R10, R15, R17, R18, R24, R26, R28, R32-R35
C8	Accuracy	The request should be an accurate representation of the user's need on the basis of which it was written.	R6, R32, R33, R36
C9	Compliance	The application should be written in accordance with an approved template, style or application writing guidelines.	R12, R18, R30, R36, R37-R40

Requirements set characteristics

#	**Characteristic**	**Description of the characteristic**	**Rules**
C10	Completeness	The set of requirements should sufficiently describe the necessary capabilities, characteristics, limitations and quality factors of the software system. No other set of requirements is needed to understand a set of requirements.	R3, R29, R41
C11	Consistency	A set of requirements should contain individual requirements that are unique, do not conflict with each other, do not overlap, and use consistent wording.	R4, R29, R30, R36-R41
C12	Feasibility	The set of requirements should be feasible regarding the existing limitations (e.g., budget, deadline, equipment, legislation, security, etc.) and with an acceptable risk.	R26, R29, R30, R33, R34
C13	Understandability	A set of requirements should be written in such a way that it is clear what is expected.	R4, R18, R36-R41
C14	Verifiability	The set of requirements should be formulated in such a way that it can be checked whether the set of requirements is fulfilled or not.	R3, R4, R36-R41

Rules for defining requirements

Note that you can always check page 3 of document *Short Incose Summary Sheet* (Moodle).

Rule	**Description**
R1: Sentence structure	Define a request as a complete sentence of the form such as *"The system shall enable <functionality> <object> with <constraints>"* .
R2: Active form	Use active when defining requirements.
R3: Subject and verb appropriate to the entity	Sentence makes sense and has clear structure.
R4: Defined terms	Use the terms you previously defined in the requirements specification.
R5: “The” instead of “a”	Use the definite article “the” rather than the indefinite article “a”.
R6 : Units of measure	In case quantitative indicators are used in the request, indicate the correct units of measure.
R7: Unclear terms	Avoid using vague terms, such as: approximately, enough, usually, approximately, something, any, etc.
R8: Escape expressions	Avoid using escape clauses such as: if possible, if necessary, etc.
R9 – Open expressions	Avoid open-ended expressions such as "and so on".
R10 – Redundant words	Avoid using redundant words just to "decorate" the request.
R12 – Correct grammar-	Define the request grammatically correctly (e.g. correct case, correct person,...).
R14 – Punctuation	Use punctuation correctly.
R16 – Negation	Avoid using absolute negations. Claims such as *"The system WILL NOT contain bugs* ." are very difficult or even impossible to verify.
R17 – Symbol "/"	Avoid using the "/" symbol as it can have many interpretations ("and", "or",...).
R18 – One sentence	Write the request as one sentence containing one thought.
R19 – Conjunctions	Avoid using conjunctions when defining requirements. Conjunctions such as *i, pa, te, ni, nor, a, but,* can mean that it is more than one request. The same applies to *"however", "in addition", "on the other hand", "while".*
R20 – Purpose	Avoid using words or expressions that indicate the purpose of the request (e.g., *"because of", "in order to"* , etc.). The purpose, that is, the answer to the question why the request is important, should be specified in a separate attribute *Reasoning,* and not in the request sentence itself.
R21: Brackets	Avoid using parentheses when defining requirements. Information in parentheses is often redundant, or not clear enough, so it should either be removed or discussed with the user and listed in the *Reason attribute* .
R22: Enumeration	If the request refers, for example, to a set of operations or entities, it is necessary to explicitly state the elements of that set and not just state the name of the set. For example instead of "The system will enable the creation of reports," it is necessary to explicitly state which reports are involved. It is very likely that a separate requirement will need to be defined for each element of the set.
R23: Context	If the request refers to an extremely complex structure or behavior, it would be preferable to attach a diagram (e.g. UML diagram) for the sake of clarity.
R24: Pronouns	Avoid using pronouns (that, that, that, he, they...). Rather, repeat the noun you are referring to. Also, avoid using indefinite pronouns such as *all, any, others, any, every, etc.* , because they create the possibility of multiple interpretations of the same request.
R25: Titles	The requirement must not be defined in such a way that it depends on the title of the chapter or subchapter in which it is located in order to be interpreted. The request must be self-explanatory.
R26: Absolute values	Avoid using unattainable absolute values. For example "The system will be online 100% of the time" is not an achievable goal. In addition to quantitative absolute indicators, expressions containing the words *"all", "always", "never"* and the like should be avoided.
R27: Conditions	In case the request is applicable only under certain conditions, these conditions should be explicitly stated.

R29: Classification	Classify the requirements according to the aspects of the problem domain or the solution itself to which the requirements refer (e.g., functional requirements, non-functional requirements, security, privacy, ...).
R30: Express once	Each request should be expressed once and only once. The appearance of duplicate requirements leads to problems in the upcoming activities of the software development process.
R31: No solution	Avoid stating or suggesting solutions when defining requirements. The answer to the question of how to realize the requirements should be given by design and implementation activities, not by the activity of defining requirements.
R33: Range of values	If the request cannot be quantified with a specific value, use a range of values (e.g., "The system will read temperature values with a precision of +/- 0.1°C").
R34: Measurability	If the requirement can be quantified, define it in such a way that it is measurable. Avoid words that vaguely or flatly quantify the request, such as *immediately, quickly, normally, maximally, minimally, optimally, easily* , etc.
R35: Time indefiniteness	Avoid words that vaguely define the time aspect of the request, such as before, after, sometime, at one time, simultaneously, etc. Instead, be explicit in expressing the time aspect.
R36: Consistent terms	Use terms and units of measure consistently throughout the requirements set.
R37: Acronyms	If requirements contain acronyms, use a consistent set of defined acronyms.
R38: Abbreviations	Avoid using non-standard abbreviations as they can lead to different and wrong interpretations.
R39: Style guidelines	Use request writing guidelines to standardize the form and syntax of requests across your team or enterprise.
R40: Grouping	Group related requests based on type, scenario, or other convenient characteristics.
R41: Structure	Stick to a defined structure or template when writing requests for all requests in a set.

Request attributes

In order to be easy to understand, the requirements should be expressed in the form of simple sentence constructions. However, such a form is certainly not sufficient to express all the necessary request data. Instead of requirements becoming long, complex and hard-to-understand sentence constructions, so-called "requirements" are often used to define additional information about requirements. *attributes* . They enable easier understanding, grouping, filtering and sorting of requests. Which specific attributes will be used to structure additional requirements information is up to the authors of the requirements specification. However, although there is no prescribed standard, the benefit of certain attributes is recognized both in the literature and in practice. Some of these attributes are as follows:

Attribute	**Label**
Identifier/Mark	Unique request identifier. Necessary to uniquely identify the requirements in the requirements specification.
Date of request	The date the request was created.
Type of request	The type of requirement with respect to the classification used by the authors (e.g., functional/non-functional).

Priority	Indicator of the importance and/or urgency of the request. It can take on quantitative (e.g., 1-10) or nominal values (e.g., key, mandatory, optional, desirable...).
Source	The name of the document by which the request is prescribed or of the stakeholder who submitted the request.
Status	The status in which the request is (e.g. proposed, accepted, rejected, fulfilled, etc.
Explanation	Reasoning why the request exists, or why it is needed.
Verification method	A verification criterion or a test scenario that will allow determining whether a requirement is met or not.
Risk	The level of risk related to the realization of the request.

A practical example

Software requirements specification for Evaluation Manager

3. SPECIFIC FUNCTIONAL REQUIREMENTS

Identifier	**FZ-1**
Claim	The system will allow data access only to authenticated users.
Explanation	The Evaluation Manager must restrict access to sensitive student data (results and evaluation points, and grade) so that only authorized persons (teachers on the course) can read and enter them.
Verification method	Entering correct user data should result in successful authentication and enable the user to continue working in the system. In case of incorrect user data, authentication should be unsuccessful and work in the system will not be possible.
Priority [1 -5]	1
Source	Teachers of the program engineering course

Identifier	**FZ-2**
Claim	The system will enable the import of data on enrolled students from a file exported from the ISVU system.
Explanation	At the beginning of each academic year, it is necessary to enter a larger number of students (expected about 200) into the Evaluation Manager. Given that the students who enrolled in the course are already registered in the ISVU system, the data entry procedure can be accelerated by importing data from the ISVU system.
Verification method	After the data import, all students who have been exported from the ISVU system should be permanently recorded and visible in the Evaluation Manager.
Priority [1 -5]	3
Source	Teachers of the program engineering course

Identifier	**FZ-3**
Claim	The system will enable manual entry of data on students attending the course.
Explanation	At the beginning of the academic year, students enrolled in the course are imported from the ISVU system. However, for various reasons, a certain number of students are enrolled in the course late. Such students must then be entered manually into the system.
Verification method	A manually entered student should be permanently recorded and visible in the system.
Priority [1 -5]	3
Source	Teachers of the program engineering course

Identifier	**FZ-4**
Claim	The system will enable the display of students enrolled in the course.

Explanation	Teachers must be able to see the students enrolled in the course at all times in order to be able to record their activities.
Verification method	The list of students enrolled in the system should be visible to the teacher.
Priority [1 -5]	1
Source	Teachers of the program engineering course

Identifier	**FZ-5**
Claim	The system will enable the definition of a list of monitoring elements in the course.
Explanation	Given that the way the course is conducted can be changed (e.g. reduce or increase the number of colloquiums and assignments), it is necessary to enable changes in the monitoring elements by which the students' work will be evaluated.
Verification method	Changes resulting from entering a new tracking element, or changing and removing an existing one, are permanently recorded and visible in the system.
Priority [1 -5]	4
Source	Teachers of the program engineering course

Identifier	**FZ-6**
Claim	For each tracking element, the system will enable the definition of conditions that the student needs to fulfill in order to exercise the right to sign.
Explanation	Teachers can decide that there are minimum conditions that a student must fulfill for the monitoring element in order to be able to record that he has "listened" to the course and can take the exam. In case of non-fulfillment of such minimum conditions, the student loses the right to sign and must re-enroll in the course.
Verification method	It is possible to enter minimum conditions for signature in the system, and these conditions should be permanently recorded and visible.
Priority [1 -5]	4
Source	Teachers of the program engineering course

Identifier	**FZ-7**
Claim	For each monitoring element, the system will enable the definition of conditions that the student needs to fulfill in order to be positively evaluated.
Explanation	Teachers may decide that there are minimum requirements that a student must meet for the tracking element in order to be graded positively. In case of non-fulfillment of such minimum conditions, the student will be evaluated negatively.
Verification method	It is possible to enter minimum conditions for a positive evaluation into the system, and these conditions should be permanently recorded and visible.
Priority [1 -5]	4
Source	Teachers of the program engineering course

Identifier	**FZ-8**
Claim	The system will enable the definition of a point scale for evaluating students.

Explanation	Depending on the introduced changes in the performance of the course, it may be necessary to correct the point scale in terms of expanding or reducing the grade for a particular grade.
Verification method	It is possible to enter and correct the point scale in the system. The mentioned changes should be permanently recorded and visible.
Priority [1 -5]	4
Source	Teachers of the program engineering course

Identifier	**FZ-9**
Claim	The system will enable the recording of points achieved during the evaluation for each enrolled student according to the defined monitoring elements.
Explanation	Recording points by monitoring elements is necessary so that we can evaluate students' knowledge in a theoretical and practical sense, and
Verification method	After entering points for some element of continuous monitoring of the student, the points must be permanently recorded and visible.
Priority [1 -5]	1
Source	Teachers of the program engineering course

Identifier	**FZ-10**
Claim	The system will be able to determine for each student whether the conditions for signature have been met or not.
Explanation	A student who did not meet the conditions for signing did not meet the minimum conditions prescribed by the monitoring model. For such a student, it will be necessary to register a ban on signatures, thus preventing him from attending the exam dates of the course.
Verification method	A student for whom the system has determined that he meets the conditions for signing must have recorded at least the minimum points from all monitoring elements defined by the model. If points are not recorded from all monitoring elements defined by the model, or if points are recorded for at least one monitoring element below the minimum, the student must not have the right to sign.
Priority [1 -5]	1
Source	Teachers of the program engineering course

Identifier	**FZ-11**
Claim	At the end of continuous monitoring, the system will propose a grade based on the total number of points achieved for students who have met the signature requirement.
Explanation	In order to be able to enter a grade on the deadline for continuous monitoring at ISVU in accordance with the achieved points on continuous monitoring, it is necessary for each student to know whether he has met the condition for entering the grade and which grade is being worked on.
Verification method	The grade should only be proposed if the student has met the condition for signing, and it must be in accordance with the defined point scale.
Priority [1 -5]	1
Source	Teachers of the program engineering course

Identifier	**FZ-12**

Claim	The system will create a report with a list of students who did not exercise the right to sign.
Explanation	Before opening the deadline for continuous monitoring, the teacher should record the prohibition of signatures in the ISVU system for all students who have not exercised the right to sign.
Verification method	All students who have not exercised their right to sign must be on the list. Also, there must not be a single student on the list who has exercised the right to sign.
Priority [1 -5]	3
Source	Teachers of the program engineering course

Identifier	**FZ-13**
Claim	The system will enable the printing of continuous monitoring results for students who have signed. At the same time, for each student, the number of points of individual monitoring elements, the total number of points, and the proposed evaluation should be visible.
Explanation	After opening the period for continuous monitoring, the teacher should enter the grades in the ISVU system for all students who have registered for the examination period (i.e. who have received a signature). In addition to the proposed grade, the teacher also needs an overview of points by individual elements of monitoring, in order to make a decision more easily in case of marginal situations and answering for a higher grade.
Verification method	The list must include all students who have exercised the right to sign, and accordingly registered for the period for continuous monitoring. The list must not include students who have not exercised the right to sign. All displayed points for monitoring elements should correspond to the factual situation. The proposed grade must be in accordance with the point scale.
Priority [1 -5]	2
Source	Teachers of the program engineering course

3.1. Dynamics of request realization

In the initial version of the software, only requests with the highest priority will be implemented. That includes:

- **FZ-1 -** The system will allow access only to authenticated users.
- **FZ-4 -** The system will enable the display of students enrolled in the course.
- **FZ-9 -** The system will enable the recording of points achieved during the evaluation for each enrolled student according to the defined monitoring elements.
- **FZ-10 -** The system will be able to determine for each student whether the conditions for signing have been met or not.
- **FZ-11 -** At the end of continuous monitoring, the system will propose an evaluation based on the total number of points achieved for students who have met the requirement for signature.
- **FZ-13 -** The system will enable the printing of continuous monitoring results for students who have signed. At the same time, for each student, the number of points of individual monitoring elements, the total number of points, and the proposed evaluation should be visible.

In subsequent versions, the implementation of other requirements is planned:

- **FZ-2 -** The system will enable the import of data on enrolled students from a file exported from the ISVU system.
- **FZ-3 -** The system will enable manual entry of data on students attending the course.
- **FZ-5 -** The system will enable the definition of a list of monitoring elements in the course.
- **FZ-6 -** For each monitoring element, the system will enable the definition of conditions that the student must fulfill in order to exercise the right to sign.

4. NON-FUNCTIONAL REQUIREMENTS

4.1. Software layout

NFZ-1 – The system will interact with the user via a graphical interface.

NFZ-2 – The system will follow a formal/corporate graphical interface style.

4.2. Usability of software

NFZ-3 – The system will offer mechanisms that will reduce the possibility of errors when entering evaluation results by teachers.

4.3. Software performance

NFZ-4 – The system will provide precision for decimal numbers at the level of 2 decimal places.

NFZ-5 – The system will be available 24 hours a day for the duration of continuous monitoring.

NFZ-6 – The system will provide the possibility of simultaneous use by at least 5 users.

4.4 Software Execution and Environment

NFZ-7 – The system should work on computers with Windows 10 or a newer operating system installed.

4.5. Security and privacy

NFZ-8 - The system will only allow teachers to access the results of monitoring students,

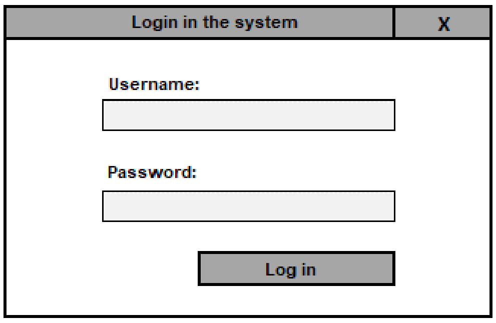
NFZ-9 – The system will use student data in accordance with the provisions of the GDPR.

4.6. The rest

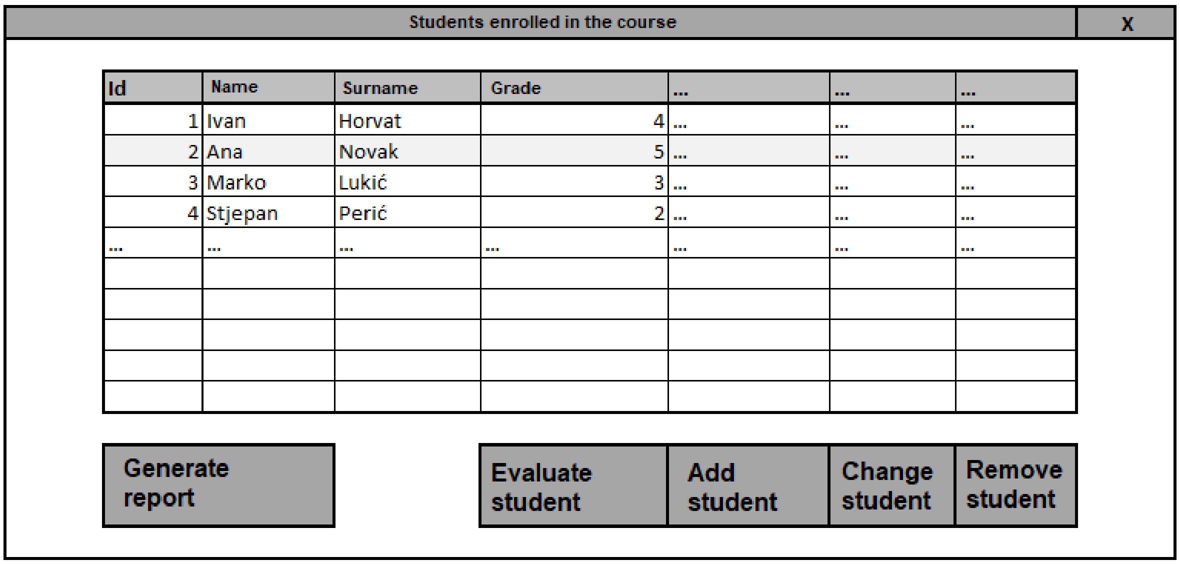
No additional non-functional requirements have been identified.

5. SCREEN SKETCHES

5.1. Sketch of the login screen

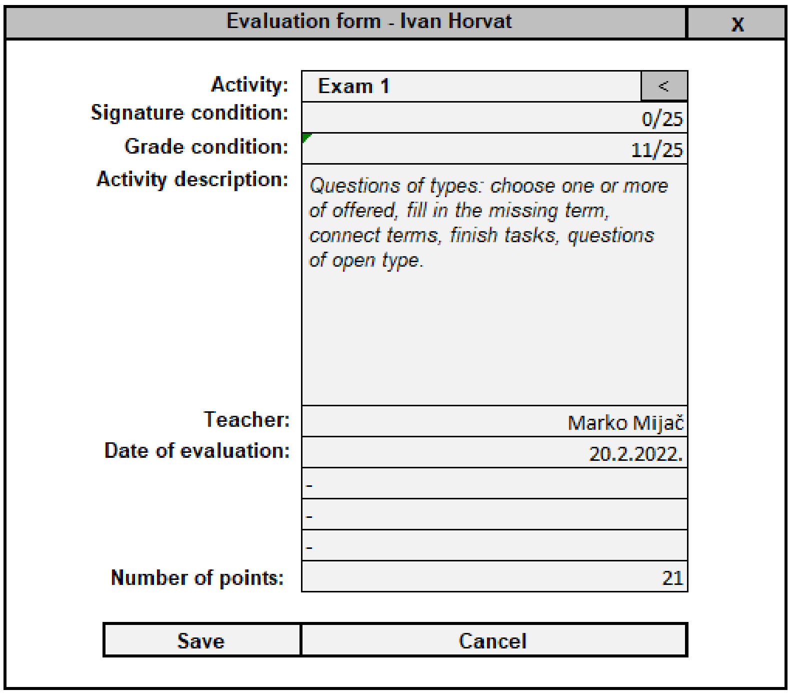


5.2. Screen sketch for displaying students enrolled in the course



Id	Name	Surname	Grade	...	...	...
1	Ivan	Horvat	4	...	...	...
2	Ana	Novak	5	...	...	...
3	Marko	Lukić	3	...	...	...
4	Stjepan	Perić	2	...	...	...
...	...	...	...	...	...	...

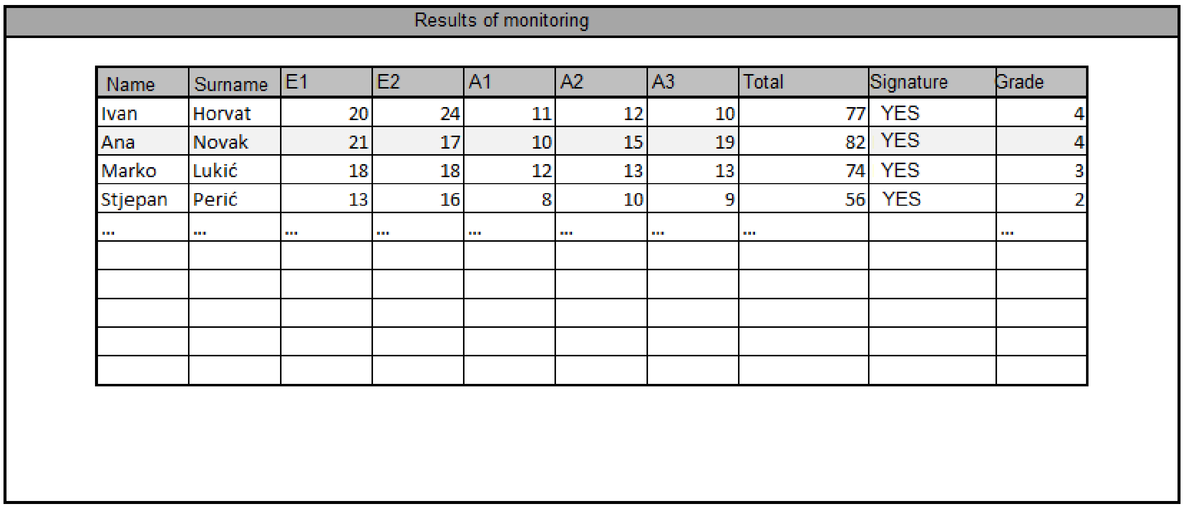
5.3. Screen sketch for conducting student evaluation



Evaluation form - Ivan Horvat

Activity:	Exam 1
Signature condition:	0/25
Grade condition:	11/25
Activity description:	*Questions of types: choose one or more of offered, fill in the missing term, connect terms, finish tasks, questions of open type.*
Teacher:	Marko Mijač
Date of evaluation:	20.2.2022.
	-
	-
	-
Number of points:	21

5.4. Screenshot of continuous monitoring results display



Results of monitoring

Name	Surname	E1	E2	A1	A2	A3	Total	Signature	Grade
Ivan	Horvat	20	24	11	12	10	77	YES	4
Ana	Novak	21	17	10	15	19	82	YES	4
Marko	Lukić	18	18	12	13	13	74	YES	3
Stjepan	Perić	13	16	8	10	9	56	YES	2
...	...	...	...	...	...	...	...		...

Questions and tasks

1. Explain the difference between user requirement and software requirement?
2. Explain the difference between a functional requirement and a non-functional requirement?
3. Why are non-functional requirements often critical and functional requirements are not?
4. State what characteristics individual requirements should possess?
5. State what characteristics the sets of requirements should possess?
6. Why do we use additional attributes to describe the request?
7. List some rules for writing requests?
8. What are screen sketches useful for in the context of requirements specification?