

RUANDER Oktatási Kft.

– **LEAP** –

## **Lambda Exam Application**

APPLICATION FOR EDITING AND SOLVING TEST SHEETS

EXTRACT FROM SOFTWARE DEVELOPER VOCATIONAL THESIS

Thesis Advisor:  
**Ferenc Nagy**

Written by:  
**Ádám Szücs-Szabó**  
Software Developer  
MSc Mechanical Engineer

Hungarian National Qualification  
Register Code of the course:  
54 213 05

Budapest, 2022

## Table of contents

1. Setting the goals .....	2
2. The initial block diagram for illustrating the connection between the users and the sub-systems.....	2
3. Use case model.....	3
3.1 Bulk collection of the explored use cases and actors .....	4
3.2 Use case diagrams .....	5
4. Analysis of the system from object-oriented perspective, creating analysis model.....	8
4.1 Identifying the classes and preparing the initial class diagram .....	8
4.2 General user class .....	9
4.3 Teacher class .....	9
4.4 Administrator class .....	9
4.5 Database class.....	10
4.6 Edited test sheet class .....	10
4.7 Task class.....	10
4.8 Multiple-choice task class .....	10
4.9 Essay task class.....	10
4.10 Individual test sheet class .....	11
4.11 The analysis class diagram with the identified associations .....	11
5. Implementation and the classes .....	12
6. System requirements and running the application.....	12
7. Introducing the use of the LEAP software .....	13
8. Summary and opportunities for further development .....	24
9. References .....	25

# LEAP – Lambda Exam Application

## 1. Setting the goals

I tried to set reasonable goals in terms of difficulty and scope at the beginning of making my thesis and the desktop application. It is typical of the software development process that a software can only be completely finished, when it had been used by several users, and thanks to this, the development team received a lot of feedback. Well, in my case, only my supervisor and I could give this kind of feedback to myself. Thus, it is possible that the software will not be perfect by the end of the thesis process, but the main goal was to create an initial application and documentation (by using the knowledge gained during the course), that could be a good basis for further development.



*Figure 1: Self-designed logo of the Lambda Exam Application*

## 2. The initial block diagram for illustrating the connection between the users and the sub-systems

The block diagram is a visual chart generated from the users' requirements in the early stage of the project. It converts and shows the users' needs in a more compact way, than they were originally described in words at the beginning. However, it still closely reflects the user's requirements, in a form that the user still can easily understand.<sup>1</sup> Furthermore, with its help, we are able to structure our system and divide into practical subsystems, which will be important in the later phases of development. After analyzing the requirements and functions, the following block diagram was created which is shown in Figure 2.

---

<sup>1</sup> Ficsor Lajos - Krizsán Zoltán - Dr. Mileff Péter (2011) Szoftverfejlesztés. Miskolci Egyetem, Általános Informatikai Tanszék. 5.1. fejezet

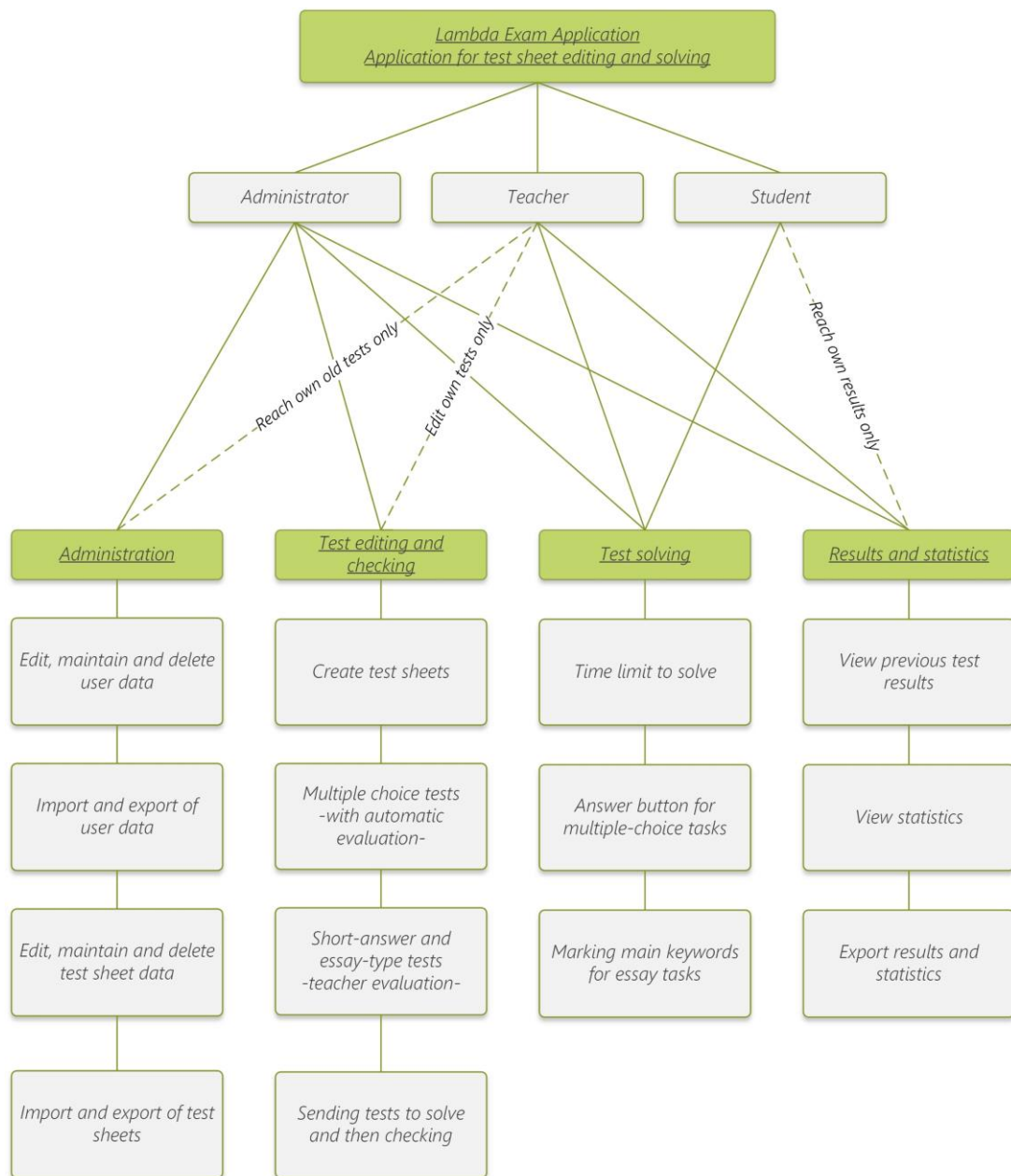


Figure 2: Block diagram illustrates the main-functions of the system

### 3. Use case model

The use case model illustrates the user's view of the system. In this case, the actors symbolize the possible users (and their authorization levels), and the use cases represent the activities of the environment doing on the system.<sup>2</sup> The collection of these diagrams forms the use case model, which will support the creation of class diagrams for later analysis and design models. In addition, this use case model and the block diagram together also form a documentation with simple notations, which will continue to help to specify the (sub)functions, tasks and priorities.

<sup>2</sup> Ficsor Lajos - Krizsán Zoltán - Dr. Mileff Péter (2011) Szoftverfejlesztés. Miskolci Egyetem, Általános Informatikai Tanszék. 8.1. fejezet

### 3.1 Bulk collection of the explored use cases and actors

First, by using the block diagram, a bulk collection of actions and actors will be created, where no connection will be showed between the members. I just write down everything that might be relevant and came to my mind in addition to, what has been written so far. The Figure 3 illustrates the collection of actors and the actions of the system.

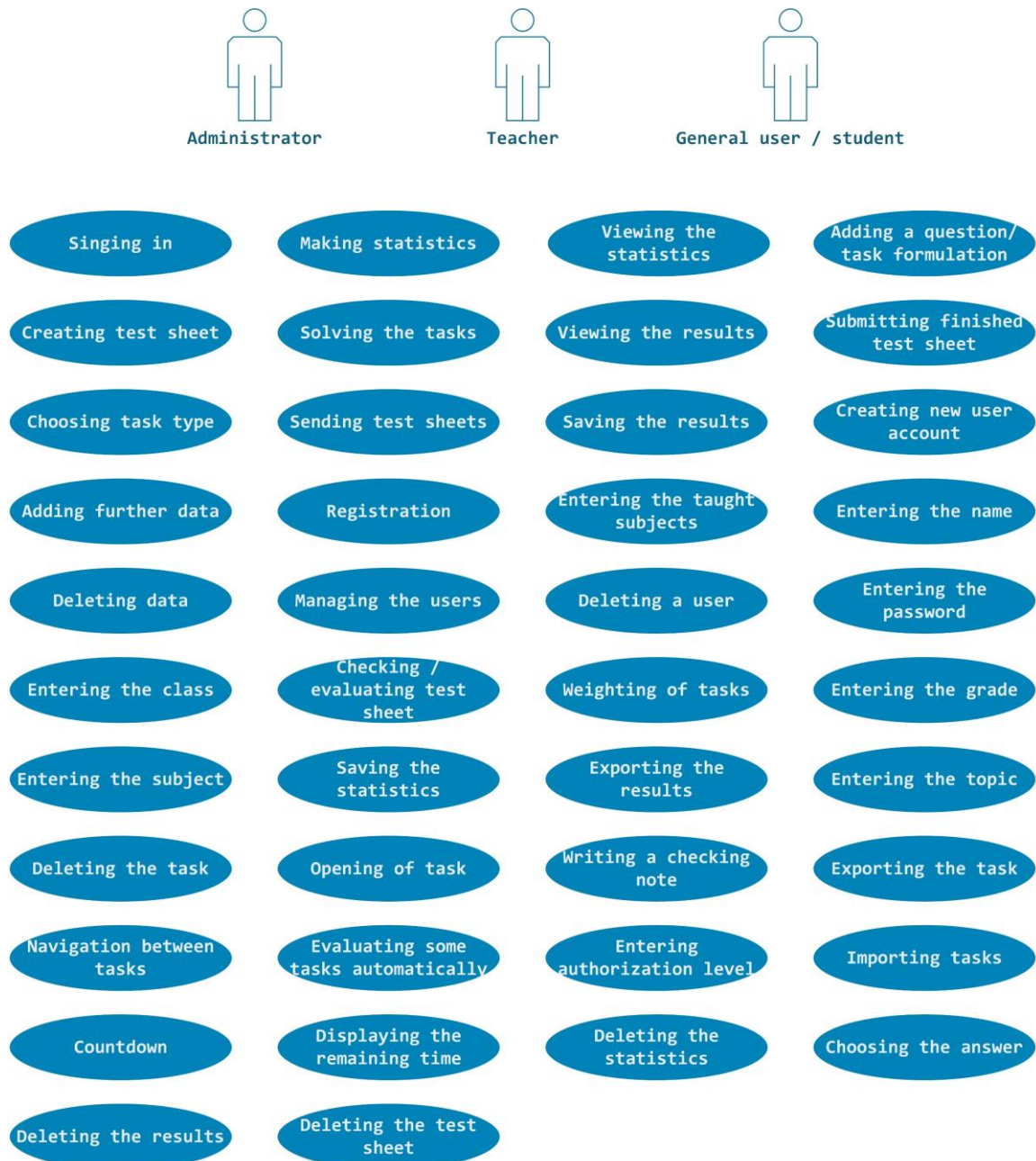


Figure 3: Initial, bulk collection of actions and actors belonging to the system

### 3.2 Use case diagrams

By using the previous elements and connecting them to each other, the following use case diagrams are created.<sup>3</sup> The Figure 4. shows the main diagram of the use case model, where the actors and use cases of the system are shown. However, some use cases have been shown as packages due to their complexity and lack of available space. These packages and the use cases included in them are illustrated in Figure 4, Figure 5, Figure 6, Figure 7 and Figure 8.

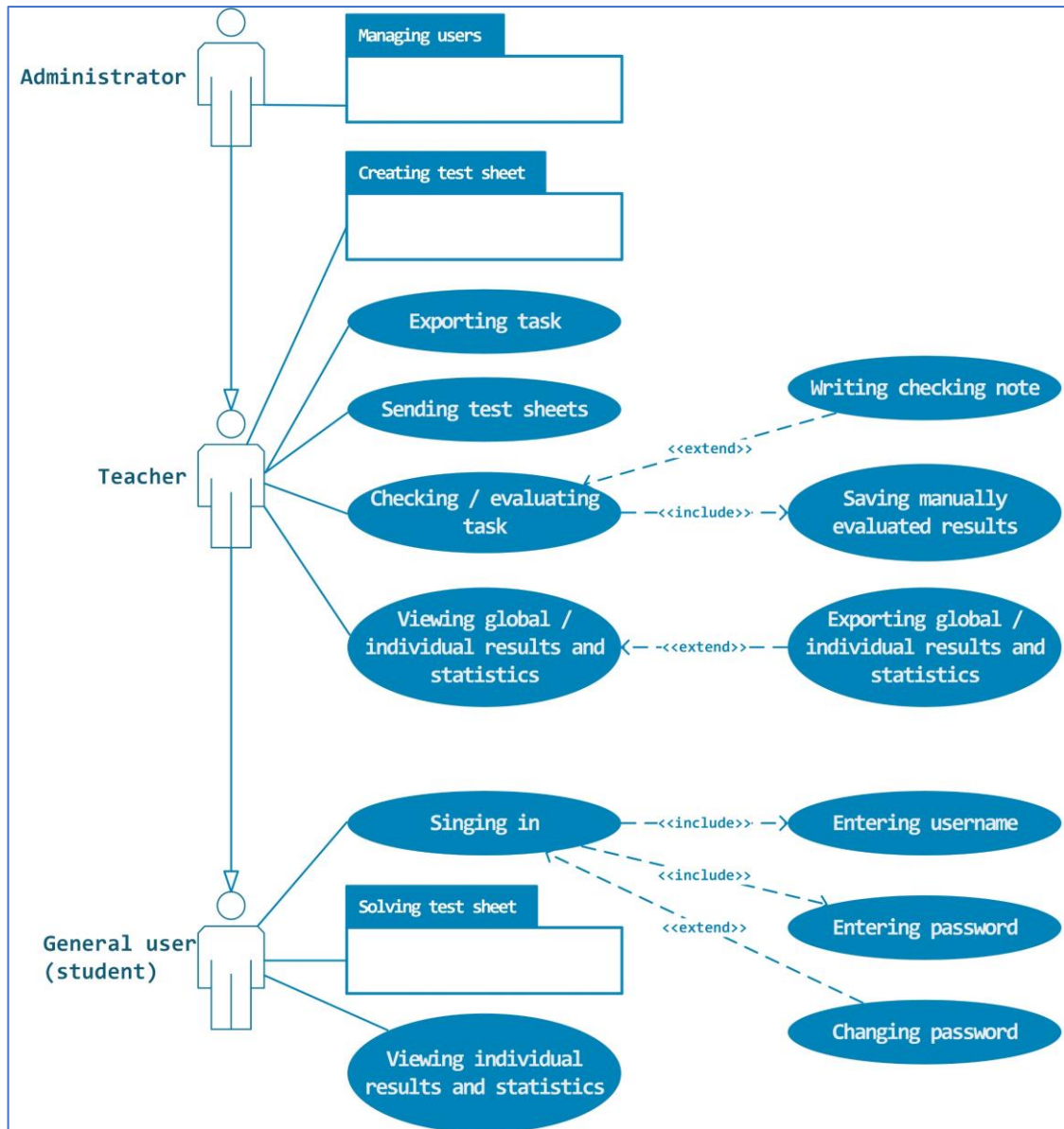


Figure 4: Use case diagram: the main system

<sup>3</sup> Ficsor Lajos - Krizsán Zoltán - Dr. Mileff Péter (2011) Szoftverfejlesztés. Miskolci Egyetem, Általános Informatikai Tanszék. 8.2. fejezet

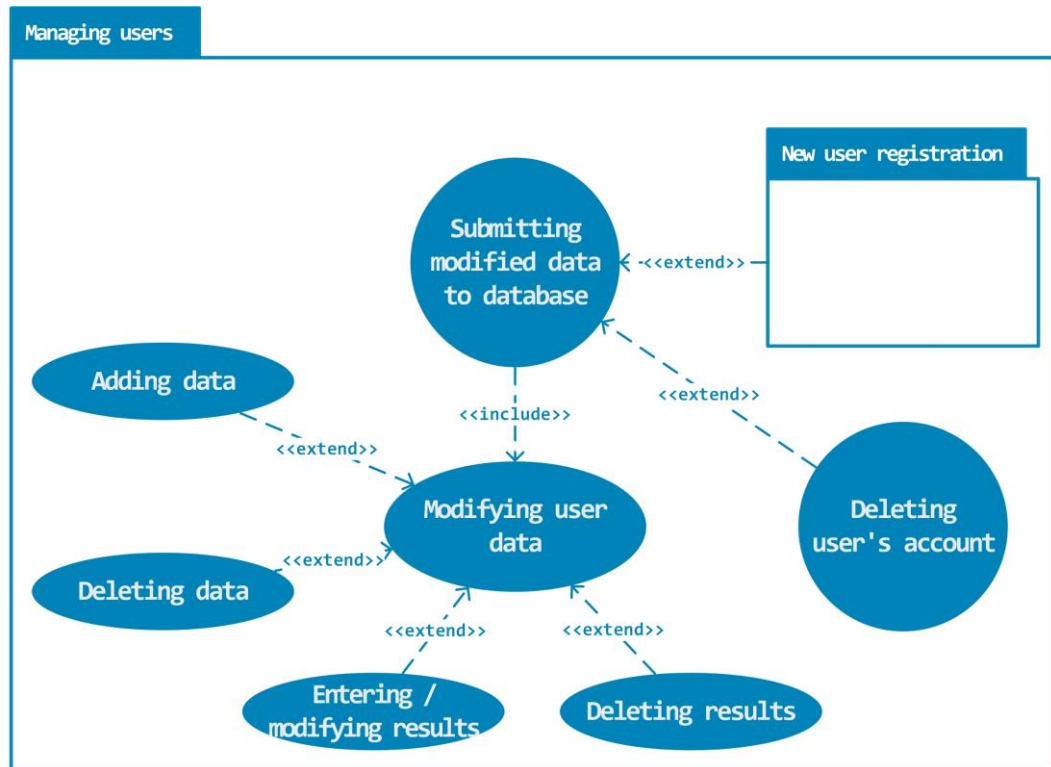


Figure 5: Use case diagram: managing users

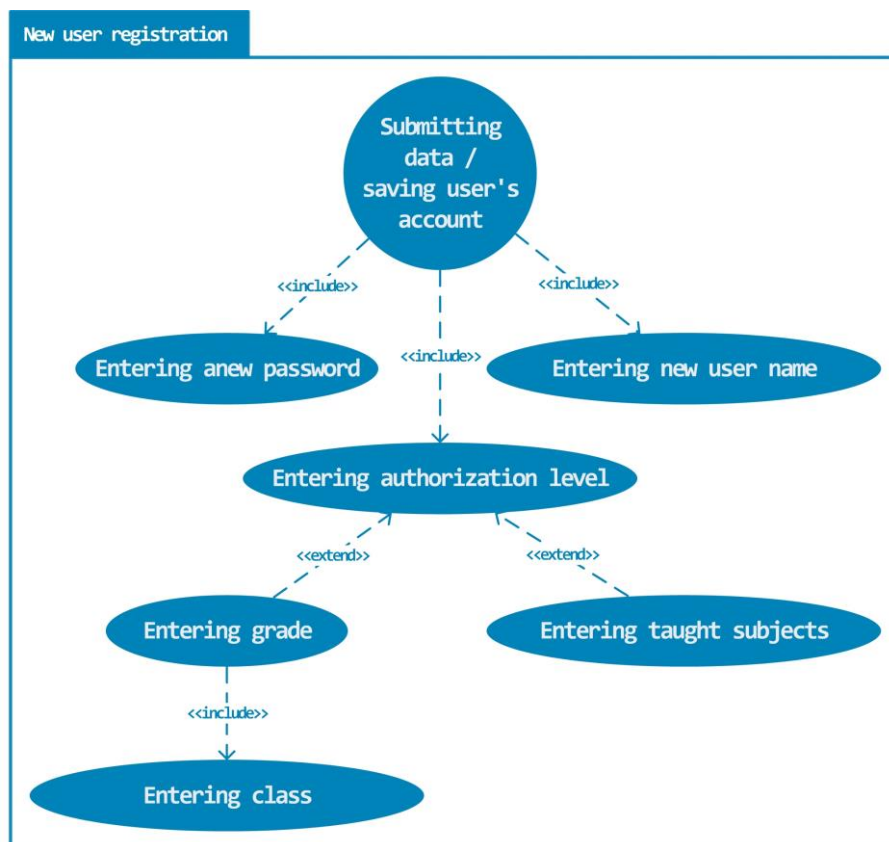


Figure 6: Use case diagram: new user registration



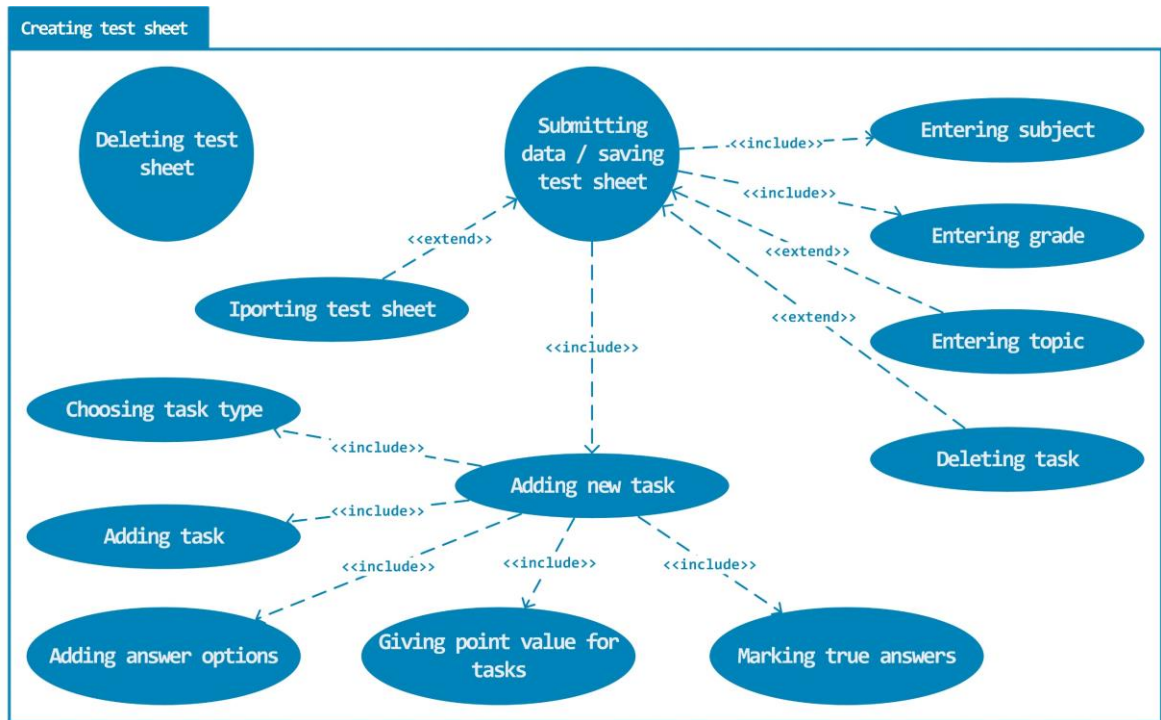


Figure 7: Use case diagram: creating test sheet

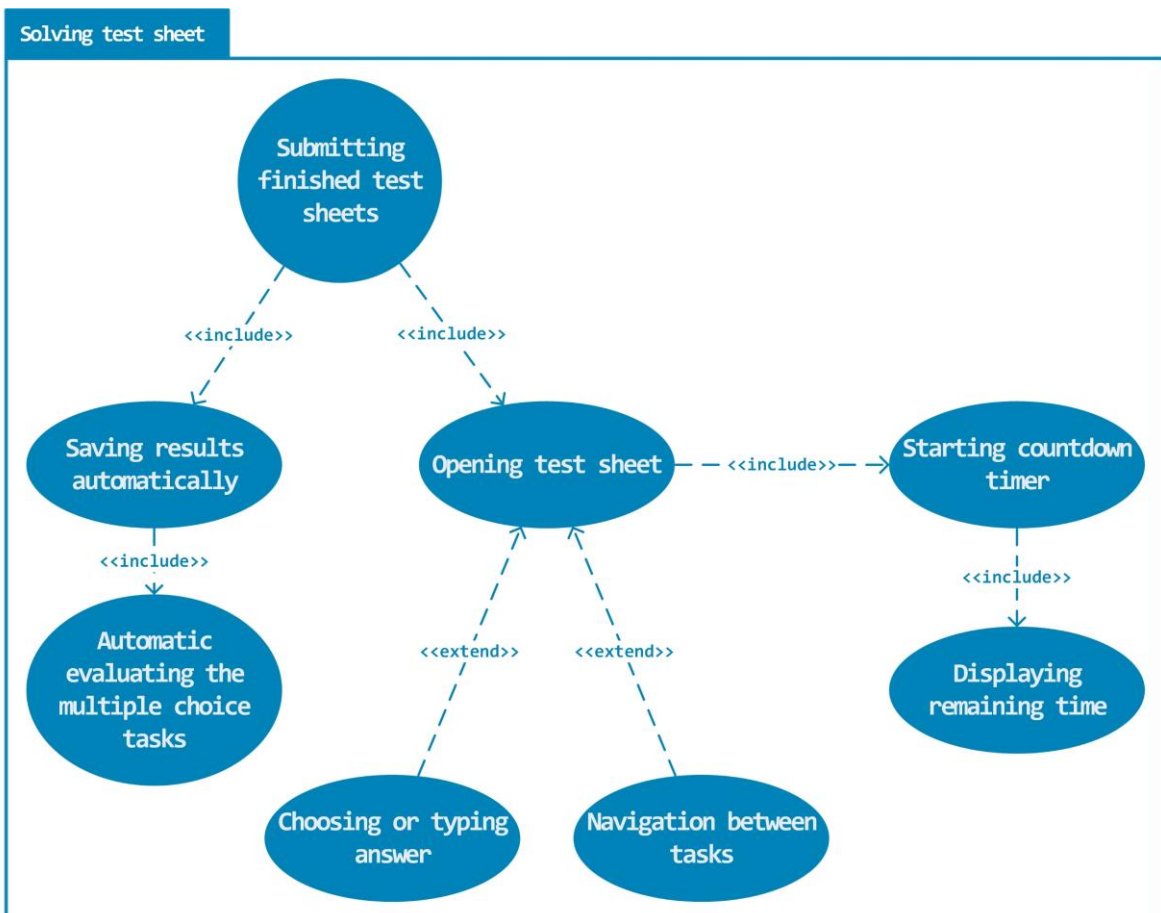


Figure 8: Use case diagram: solving test sheet



## 4. Analysis of the system from object-oriented perspective, creating analysis model

### 4.1 Identifying the classes and preparing the initial class diagram

During the creation of the use case model, I followed the point of collecting the main expressions / acts that provide significant help in terms of the implementation of the main functions. Then I created the use case model by using these words. In the process of identifying classes and creating initial class diagrams I shifted the focus from the acts to the subjects. At this point, it is no longer a purpose to make this document understandable for every user is every detail, but with the help of a developer, it is still understandable to everyone.<sup>4</sup> The main goal is to support the development team by collecting topics, definitions, objects, scenes, events, etc. that define classes, the main pillars of object-oriented programming. Although, as I wrote above, the focus is now on the exploration of subjects, it can be noticed that I only deal with classes that have some kind of operations / methods.

At first, these classes are also illustrated in an initial, summary diagram, where there are no relationships between the classes and they do not have any properties or methods yet.<sup>5</sup> Figure 9. shows the initial class diagram created in this way.

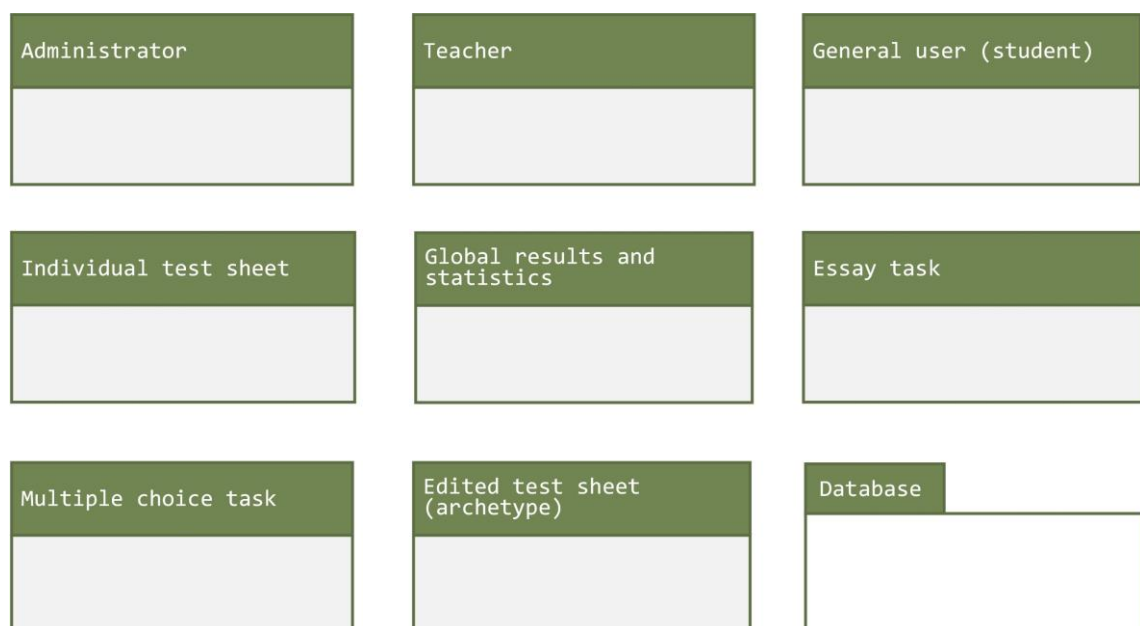


Figure 9: Initial class diagram

<sup>4</sup> Ficsor Lajos - Krizsán Zoltán - Dr. Mileff Péter (2011) Szoftverfejlesztés. Miskolci Egyetem, Általános Informatikai Tanszék. 11.1. fejezet

<sup>5</sup> Ficsor Lajos - Krizsán Zoltán - Dr. Mileff Péter (2011) Szoftverfejlesztés. Miskolci Egyetem, Általános Informatikai Tanszék. 11.1.4. fejezet

## **4.2 General user class**

This class represents the students, who have to solve the test sheets. Its properties are not only the personal, essential data, but also which grade and in which class the student is. Their main task is to solve and submit the received tests sheets. There are two types of tasks that can be answered in different ways. In the case of multiple-choice tasks, you can answer the question by clicking or by marking on the desired answer option. Furthermore, essay type tasks can be solved by answering them in a few or more sentences. Student can navigate between the tasks by scrolling. After answering the questions on the test sheet according to their best knowledge within the given time, they close and submit their own test sheets. (If the student does not do this, the countdown timer will close and submit the test sheet.) In addition to these, the students can view the results of their own test sheets which have been already checked by the teacher. The users can also change their first own password given by the administrator during registration. General user has no permission to manipulate any other data.

## **4.3 Teacher class**

This is a derived class from the general user class. It has the same features, except for one thing. A member of the teacher class does not have grade nor class attribute but it has list of subjects which are taught by her/him. In the application the teacher's first jobs are to prepare the questions, edit the tasks and then assemble a full test sheet. The teacher sends the test sheet for the chosen students to solve it. Then the teacher can check / evaluate the answers given to the essay type tasks. The application automatically evaluates the answers / answer-marks, which given to the multiple-choice type tasks using a truth-table defined in advance by the teacher. When the teacher is done with the evaluation, she/he saves the checked test sheet.

## **4.4 Administrator class**

The administrator class is a descendant of teacher class so it is also a descendant of the general user class. Like the teacher, it also has the attributes of the taught subjects. The special role of administrator is manifested in the user management. The administrator can create new user accounts, during which process she/he specifies the personal data and the authorization level (student, teacher or admin) of the new user. In addition, the administrator can modify or delete this user data. During the modification she/he can add or delete certain properties of the user account or its exam results. Furthermore she/he can delete any user account.

#### **4.5 Database class**

The database class will be the SQL database itself, where all necessary input data and generated data will be stored. At this stage of the development, it can already be said that this class will not be implemented by us using OOP-principles, like the other classes. So even though this SQL database server is a well-made and standalone entity, however we need to mark it as an abstract class in our initial class diagram. All we have to do is create its tables (with the required columns), and implement the connection and the communication with it.

#### **4.6 Edited test sheet class**

The edited test sheet is a kind of an archetype, which is assembled by the teacher. Its primary elements are those tasks which can be the multiple-choice or the essay tasks. The teacher can assemble the tasks in any order and proportion. The edited test sheet also contains the following attributes: what subject and topic is it about, what grade/year is it made for, how many is the totally achievable points derived from the tasks and how much time is available to solve the test sheet? However, this class does not have any attribute about who is the person who will receive or has already received this test sheet assembly.

#### **4.7 Task class**

Task is an elementary unit, whose descendants are the multiple-choice and the essay tasks. The building blocks of the test sheets, which are assembled by the teacher. The properties of the task class include what the question is or the formulation of the task itself, which subject it is from, as well as the point value indicating the difficulty.

#### **4.8 Multiple-choice task class**

This is a child of the task class. Its properties are the formulation of the task, the list of the answer-options (max 10 items), the markers of whether an answer-option is true or not, the subject and the point value. It has an important method, that this type of task can automatically evaluate the answers given by the student.

#### **4.9 Essay task class**

This is a child of the task class. Its properties are the formulation of the task, the list of the keywords (max 10 items), to which keywords the student must refer in the answer. In one hand these keywords help the student by showing her/him about what subtopics should be covered in the essay. On the other hand, these keywords also help the teacher, because he/she can check a task much easier among evaluating the different tasks, when the she/he sees what are the important subtopics in the current essay task.

#### 4.10 Individual test sheet class

The individual test sheet is basically derived from the edited test sheet. Its attributes are a collection of those task types that were assembled by the teacher in the edited test sheet (archetype), supplemented with such properties as: the ID of the student who opened the test sheet, and a point table. The values of this point table come from the evaluation of the essay tasks and the automatically generated points of the multiple-choice tasks. One of the operations of this class is the submission, which closes the test sheet and automatically evaluates the multiple-choice tasks. If the teacher has completed the evaluation of the essay tasks, the total points of the test sheet will be saved.

#### 4.11 The analysis class diagram with the identified associations

Using the elements and properties of the initial class diagram, a new class diagram can be created. This is the analysis class diagram, which shows the relationship between classes, and in some cases also illustrates the directions, the numerosity and the dependency relationships of these connections.<sup>6</sup> The resulting analysis diagram is shown in Figure 10.

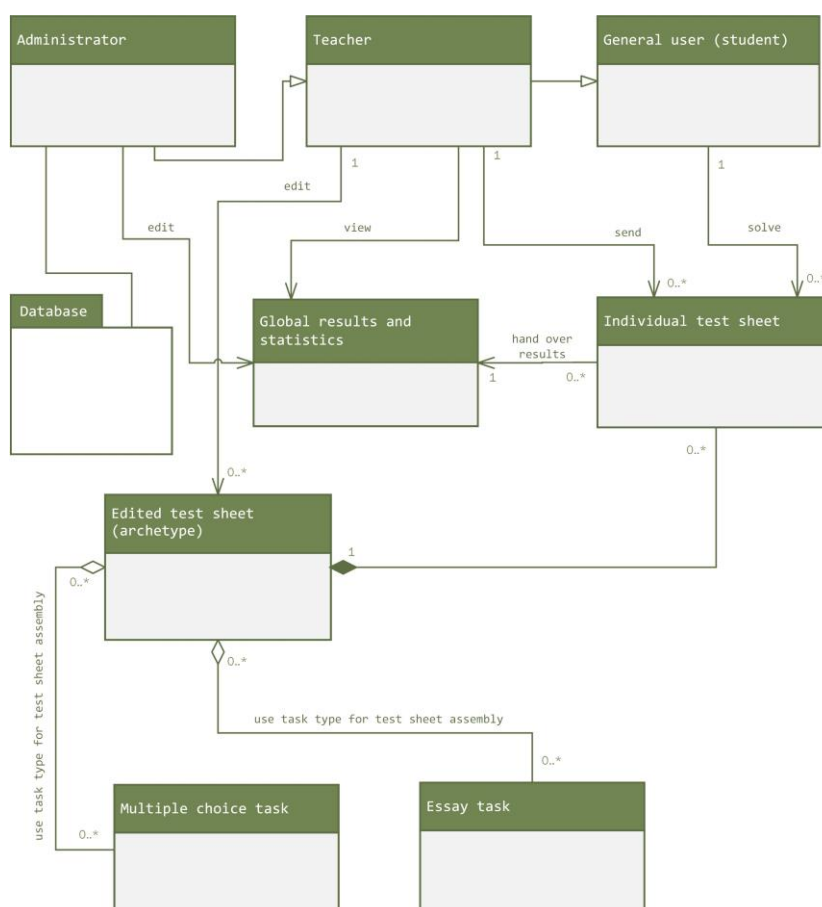


Figure 10: The analysis class diagram with the identified associations

<sup>6</sup> Ficsor Lajos - Krizsán Zoltán - Dr. Mileff Péter (2011) Szoftverfejlesztés. Miskolci Egyetem, Általános Informatikai Tanszék. 11.1.8. fejezet

## 5. Implementation and the classes

After the analyzes were completed and the initial models were set up, the implementation could begin. I used the Microsoft Visual Studio 2019 IDE (Integrated Development Environment) to implement the functions, which IDE provides the opportunity to make source code written in several programming languages, to compile codes, to manage database, to run and debug codes...etc. In summary, it is a rich development environment, that can synthesize the small units and needs of a project into a final solution.

I have chosen the C# language for the implementation, on the one hand, this language formed the main scope of our education, and on the other hand, I was able to implement all the necessary functions and units using C#. The implemented classes and the entire source code can be viewed in the project folder ([www.github.com/szszadam/LEAP-v0\\_3](https://www.github.com/szszadam/LEAP-v0_3)). Descriptions of the model classes are available (as a comment) at the top of the codes in corresponding .cs file.

## 6. System requirements and running the application

Running the LEAP application requires Windows 10 or later versions with at least version 4.8 of the .NET Framework installed. In addition, the Microsoft Visual Studio Community 2019 (or newer) software is also required to run it. Due to the simplicity of the LEAP application, it does not have any particular hardware requirements, rather the system requirements of Visual Studio Community 2019 are more relevant from this aspect. Thus, I would list these requirements as further demand for running LEAP application<sup>7</sup>:

- Processor: min 1,8 GHz
- Memory: min 2 GB
- Storage: 15 GB
- Graphics: DirectX 9 or newer.

---

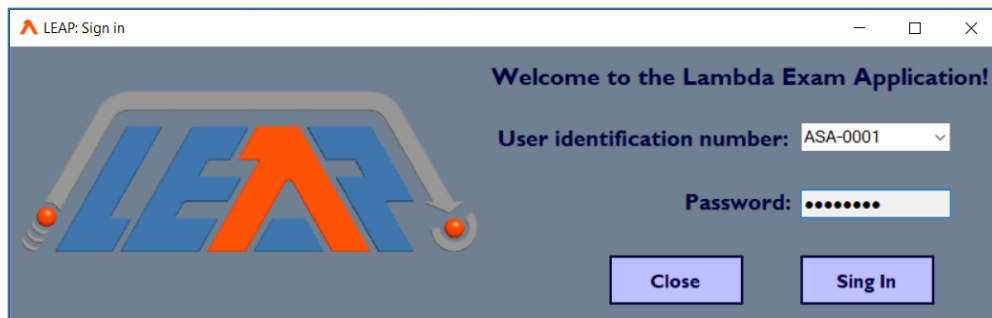
<sup>7</sup> Microsoft – Visual Studio Community 2019 system requirements, [docs.microsoft.com/en-us/visualstudio/releases/2019/system-requirements](https://docs.microsoft.com/en-us/visualstudio/releases/2019/system-requirements)

There is two ways to run the LEAP application:

1. Run the file “LEAP-v0\_3.exe” located in a subfolder of the project
  - Run: `$\LEAP-v0_3\LEAP-v0_3\bin\Debug\ LEAP-v0_3.exe`
2. Run from the Visual Studio Community 2019-ből:
  - Open: `$\LEAP-v0_3\ LEAP-v0_3.sln`
    - Opening project in the Visual Studio
    - Navigation bar
    - „Debug” menu
    - „Start Without Debugging” menu item

## 7. Introducing the use of the LEAP software

**Sign-in window:** Users registered by the administrator can log-in into the application on the Sing-in interface shown on the Figure 11.



*Figure 11: Sign in window*

The authentication process will be successful by entering the correct user identification number (which is automatically generated during the registration process) and the correct password. There is a temporary dropdown list that contains some users identification number, and when the intended ID is chosen, then the password field will be also automatically filled. This behavior is only created to make sign-in procedure easier during the testing and demonstration periods. The initial ‘A’ stands for Administrator, ‘T’ stands for Teacher, and ‘S’ stands for Student. The authentication level is specified by the administrator during the registration.



The first password that is automatically generated during the registration process will be the identification number itself. Currently, the application was made such a way that after the user selects the identification number from the dropdown list, then the password belongs to this id is loaded as well. (For help, there is a chart in the project folder that contains the data of the currently registered users.) If the login is successful, the main window will open. If you had entered wrong ID and/or wrong password, you would receive an error message.

After opening the **main window**, you have the opportunity to access the main functions. The accessibility of the modules depends on the authorization level. Now, the reader will be guided through the functions, based on how a task is prepared, how a test sheet made from it that will be solved in the future, furthermore how the teacher evaluates an individual test sheet and how a user can view her/his own results.

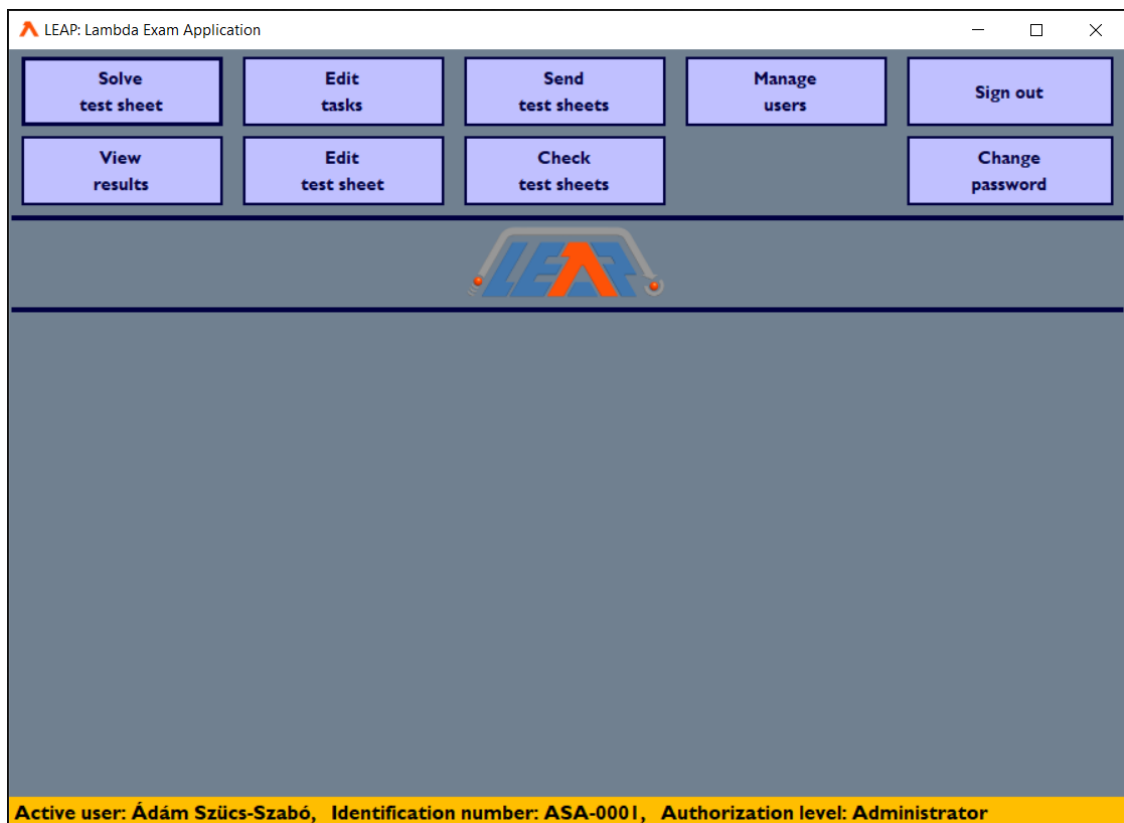


Figure 12: The main window after sign-in

Under the “**Edit tasks**” menu item, you can manage the tasks, from which the test sheets will be assembled by the teacher. Only users with “Administrator” or “Teacher” authorization level can access this menu item. The teacher can only view those tasks, which are matching with her/his taught subject. However, the administrator can view all of the created tasks. Generally, two types of tasks can be created: multiple-choice and essay-type tasks.

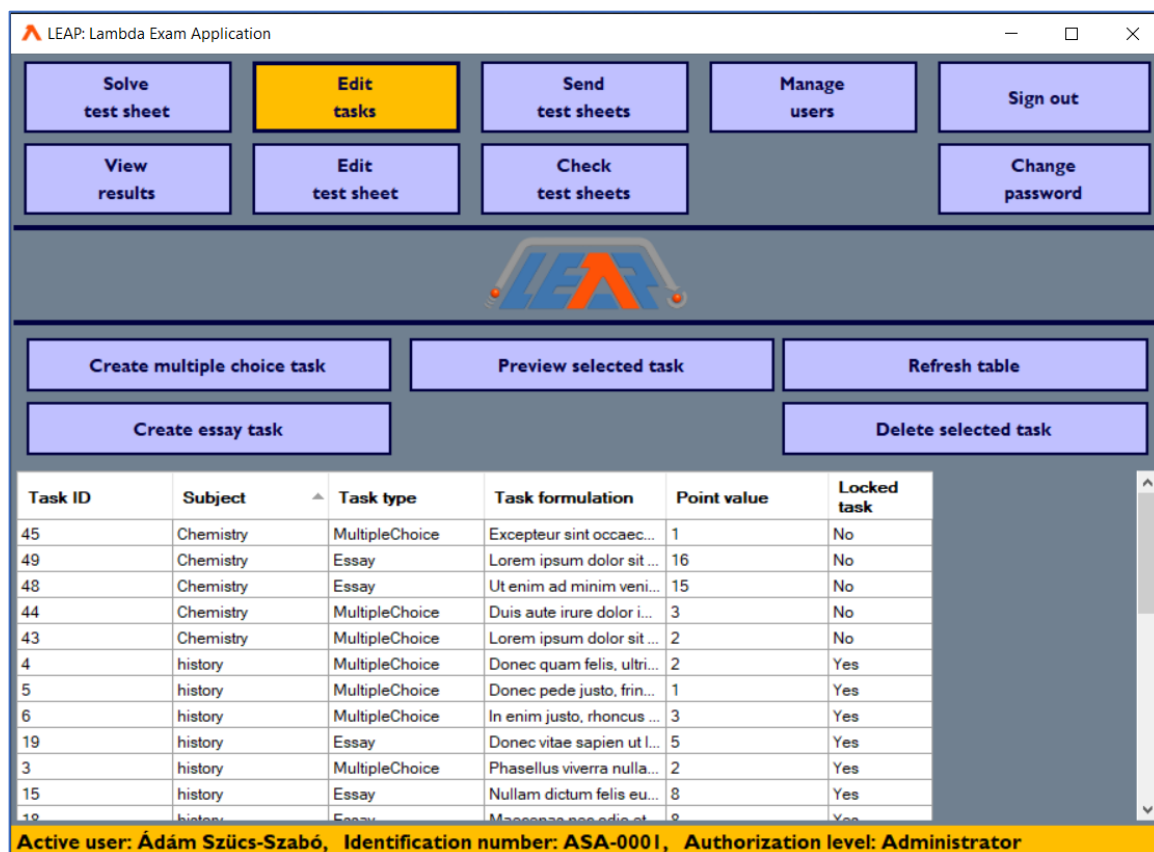


Figure 13: Creating / previewing tasks: interface to choose

When creating a **multiple-choice task**, the following things need to be provided:

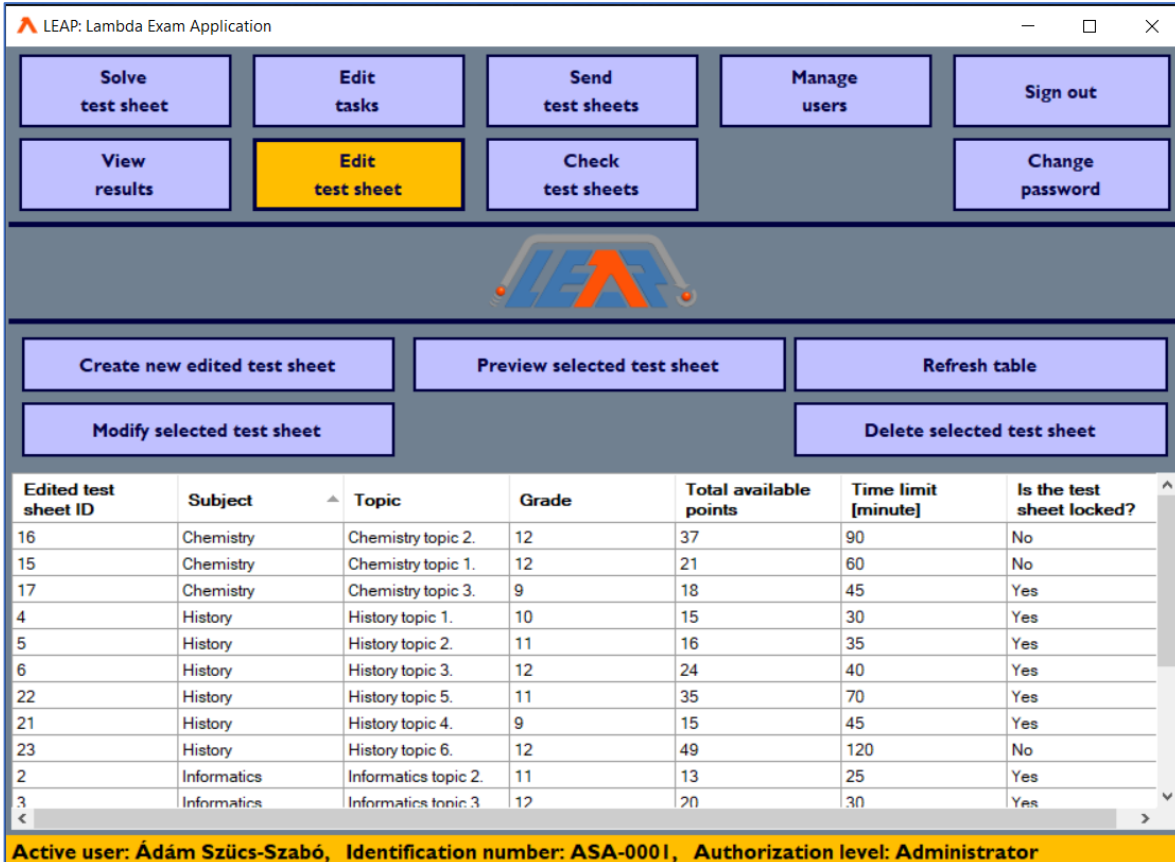
- Subject.
- Formulation of the task.
- Answer options and which of them is correct.
- The amount of the achievable point is automatically generated from the number of the correct answers. The sum of the achievable points is automatically generated from the number of the “true” answers.

The following information need to be provided during the preparation of an **essay task**.

- Subject.
- Formulation of the task.
- What are the displayed keywords that indicate the main topics of the essay, that the student should respond to.
- The teacher enters the achievable points manually.

The tasks which made this way can be viewed by pressing the **“Preview selected task”** button. The task appears in the same form as it will be visible in the assembled test sheet. In preview mode the editor can still see whether an answer option is true or false. Of course, these truth-markers hereinafter will not be shown, during solving the test sheet.

You can compose these tasks into a complete test sheet under the **“Edit test sheet”** menu item. This menu item is only available to users with “Administrator” or “Teacher” authorization level. The teacher can only view those test sheets, which are matching with her/his taught subjects. However, the administrator can view all of the created test sheets. These items called “Edited test sheets”, and in the system these are kind of archetypes, from which the “individual test sheets” will be derived (to be solved later by a student later).



The screenshot shows the LEAP: Lambda Exam Application interface. At the top, there is a navigation bar with buttons: Solve test sheet, Edit tasks, Send test sheets, Manage users, Sign out, View results, Edit test sheet (highlighted in yellow), Check test sheets, and Change password. Below this is a central area with a large blue and orange logo. Under the logo, there are buttons: Create new edited test sheet, Preview selected test sheet, Refresh table, Modify selected test sheet, and Delete selected test sheet. Below these buttons is a table with the following data:

Edited test sheet ID	Subject	Topic	Grade	Total available points	Time limit [minute]	Is the test sheet locked?
16	Chemistry	Chemistry topic 2.	12	37	90	No
15	Chemistry	Chemistry topic 1.	12	21	60	No
17	Chemistry	Chemistry topic 3.	9	18	45	Yes
4	History	History topic 1.	10	15	30	Yes
5	History	History topic 2.	11	16	35	Yes
6	History	History topic 3.	12	24	40	Yes
22	History	History topic 5.	11	35	70	Yes
21	History	History topic 4.	9	15	45	Yes
23	History	History topic 6.	12	49	120	No
2	Informatics	Informatics topic 2.	11	13	25	Yes
3	Informatics	Informatics topic 3.	12	20	30	Yes

At the bottom of the interface, there is a status bar that reads: Active user: Ádám Szűcs-Szabó, Identification number: ASA-0001, Authorization level: Administrator.

Figure 14: Creating Edited Test Sheet surface

By clicking the "**Create a new edited test sheet**" button, an editing interface opens in a new window, where the following input data must be provided.

- Which tasks does the edited test sheet contain?
- Which subject does it belong to?
- Which grade/year is it recommended for?
- How much time does the user have to solve this test sheet?

LEAP: Assembling test sheet

Task ID	Subject	Task type	Task formulation	Point value	Locked task	Add task	Order of tasks
4	history	MultipleChoice	Donec quam felis, ultr...	2	Yes	<input checked="" type="checkbox"/>	
5	history	MultipleChoice	Donec pede justo, frin...	1	Yes	<input checked="" type="checkbox"/>	
6	history	MultipleChoice	In enim justo, rhoncus ...	3	Yes	<input checked="" type="checkbox"/>	
3	history	MultipleChoice	Phasellus viverra nulla...	2	Yes	<input checked="" type="checkbox"/>	
13	history	Essay	Cum sociis natoque p...	8	Yes	<input checked="" type="checkbox"/>	
18	history	Essay	Maecenas nec odio et...	8	Yes	<input checked="" type="checkbox"/>	
19	history	Essay	Donec vitae sapien ut l...	5	Yes	<input checked="" type="checkbox"/>	
16	history	Essay	Curabitur ullamcorper ...	12	Yes	<input type="checkbox"/>	

Subject: **History** Topic: **History topic 12.** Grade: **11** Available time [minute] : **30**

**Refresh preview of test sheet** **Create test sheet**

Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.. Task:  
Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. (1 point(s))

Earned point: **0**

**?** history answer option 11

**?** history answer option 12

**?** history answer option 13

**✓** history answer option 14\*

Figure 15: Surface to assemble, create and preview the edited test sheets

After clicking on the "**Refresh preview of test sheet**" button, you can see below how the test sheet will look like. If you are satisfied with the preview of the task-sequences, then you can create a new "edited test sheet" (archetype) by clicking on the "**Create test sheet**" button.

In the main menu, under the **“Send test sheet”** menu item you can make “Individual test sheets” from the previously prepared “Edited test sheet” by sending this archetype to the selected users. This menu item is only available to users with “Administrator” or “Teacher” authorization level. The teacher can only view those test sheets, which are matching with her/his taught subject. But the administrator can view all of the created test sheets.

**Which test sheet:**

Edited test sheet ID	Subject	Topic	Grade	Total available points	Time limit [minute]	Is the test sheet locked?
16	Chemistry	Chemistry topic 2.	12	37	90	No
15	Chemistry	Chemistry topic 1.	12	21	60	No
17	Chemistry	Chemistry topic 3.	9	18	45	Yes
4	History	History topic 1.	10	15	30	Yes
5	History	History topic 2.	11	16	35	Yes

**Send test sheets**

**For which users:**

User ID	Family name	First name	Authorization level	Taught subjects	Grade	Class	Add user
48	Rozsonits	Oszvald	Student		12	A	<input type="checkbox"/>
42	Kotsis	Kázmér	Student		12	C	<input type="checkbox"/>
38	Kozák	Kálmán	Student		12	B	<input type="checkbox"/>
8	Alfa	Rómeó	Student		12	A	<input type="checkbox"/>
23	Totál	Károly	Student		11	D	<input type="checkbox"/>
47	Milyen	Ferenc	Student		11	B	<input type="checkbox"/>
41	Akashi	Anita	Student		11	A	<input type="checkbox"/>
6	Meggy	Márton	Student		11	B	<input type="checkbox"/>
46	Árpás	Angéla	Student		10	D	<input type="checkbox"/>
25	Kertész	Kamilla	Student		10	A	<input type="checkbox"/>

Figure 16: Sending individual test sheets

You have to choose which “Edited test sheet” should be the basis of the “Individual test sheet” and which student will receive it (checkboxes). You can send the desired “Individual test sheets” to any users for solving, by clicking on the “Send test sheets” button (regardless of the receiver’s authorization level).

Under the **“Solve test sheets”** menu item, the currently logged-in user can browse among those unsolved individual test sheets, which are intended for her/him by the creator. This menu item is available with any authorization level. Thereafter, if a user selects one and clicks on the **“Solve chosen test sheet”** button, a new window will appear containing the tasks of the selected test sheet, that should be solved within the available time. In the upper right corner of the test sheet window, you can check the remaining time for solution. The timer starts to count down from that time-value, that was given by the creator in the editing process.

If you are ready with the tasks, you can submit the test sheet earlier, even before the time is counted down. After the countdown the submission process will happen automatically. (As I wrote earlier, teacher and administrator users can send “individual test sheet” not only for students, but also for other admins and teachers.)

LEAP: Solve test sheet

Submit test sheet

Student and test sheet information:  
Deák Dominika  
Informatics, Informatics topic 4.

Remaining time: 69:47

1. Task:  
Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus. (1 point(s))

? informatics answer option 1 1

? informatics answer option 1 2\*

? informatics answer option 1 3

? informatics answer option 1 4

? informatics answer option 1 5

2. Task:  
Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. (2 point(s))

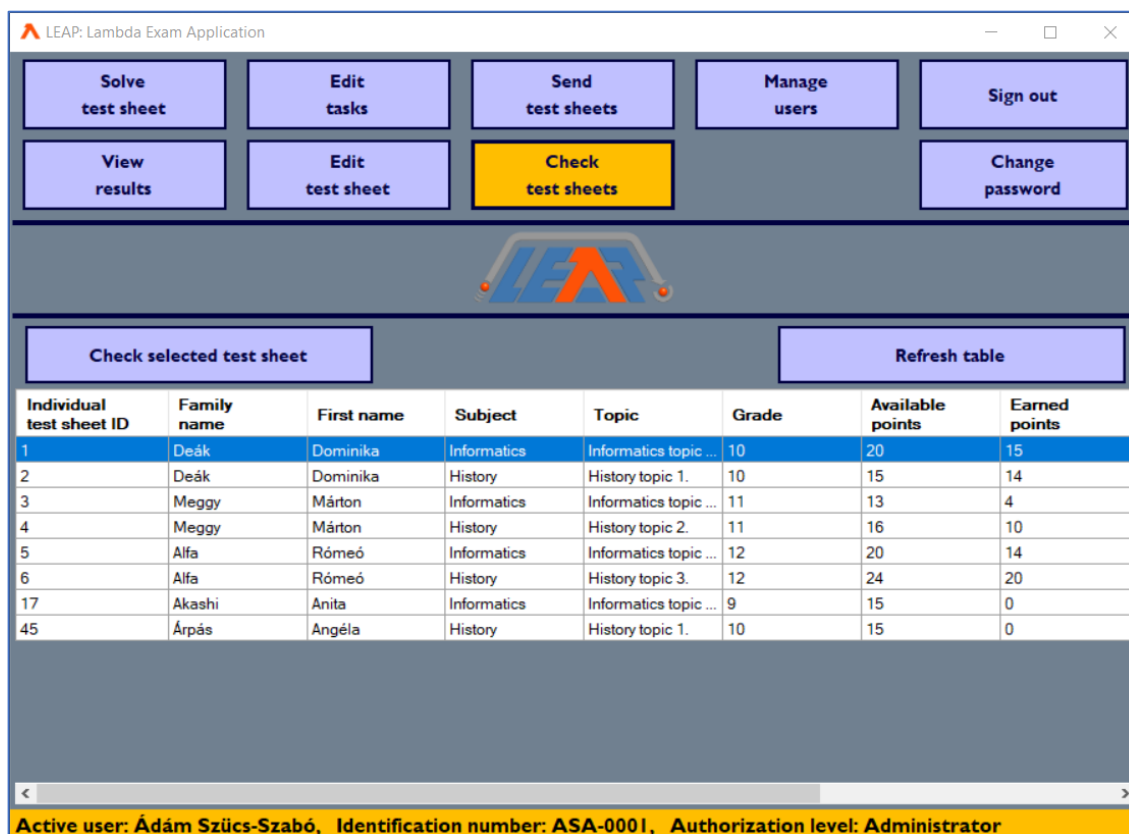
? informatics answer option 1 6

Figure 17.: Solving test sheet interface

A test sheet can consist of two types of tasks: multiple-choice tasks and essay tasks. In the case of multiple-choice tasks, you can click on the question mark (?) on the left side of the answer option to mark your answer as correct. If you are done with the test sheet, you can submit your answers and close the window in two ways. First, by clicking on the “**Submit test sheet**” button on the upper left corner of the test sheet window. Or simply close the window using the usual X button on the very upper right corner of this window. But, before the system passes your answers on and closes the window, it asks you, if you are sure to submit and close. When the time is up, this process will be done automatically without any question.



By clicking on the “**Check test sheets**” button, the teacher can view the submitted test sheets and can give points for the answers to the essay tasks. This menu item is only available to users with “Administrator” or “Teacher” authorization level. The teacher can only view those test sheets, which are matching with her/his taught subject. However, the administrator can view all of the checked test sheets.



LEAP: Lambda Exam Application

Buttons: Solve test sheet, Edit tasks, Send test sheets, Manage users, Sign out, View results, Edit test sheet, **Check test sheets**, Change password

Check selected test sheet Refresh table

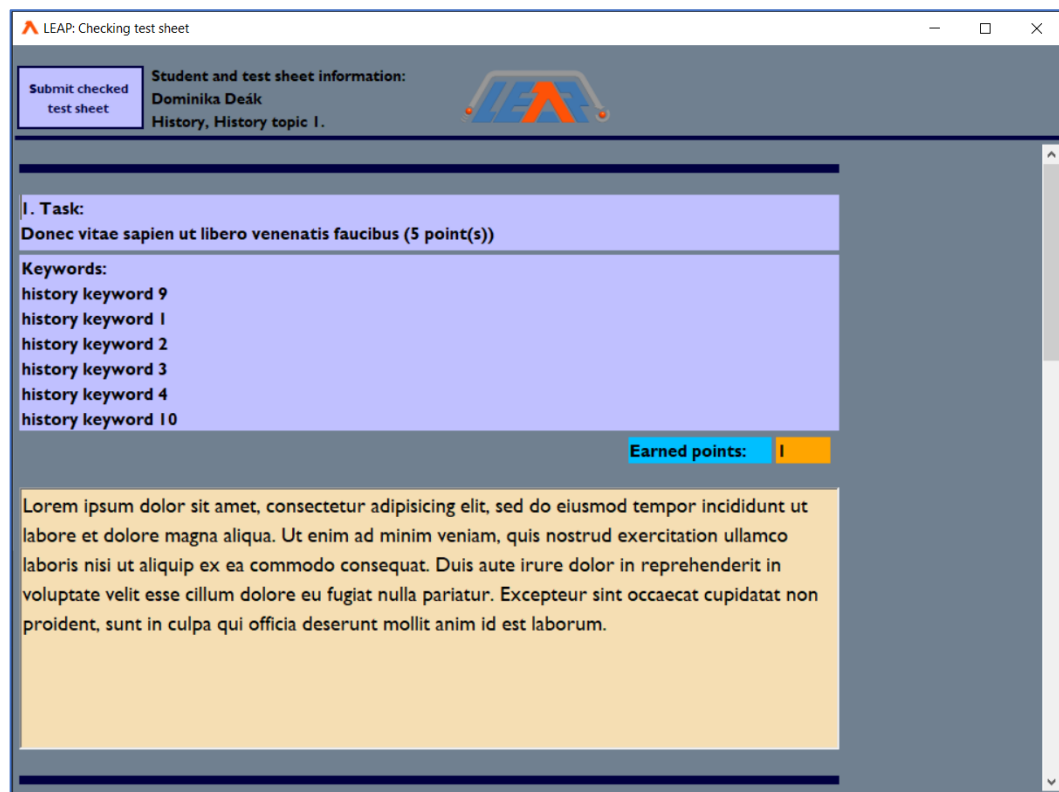
Individual test sheet ID	Family name	First name	Subject	Topic	Grade	Available points	Earned points
1	Deák	Dominika	Informatics	Informatics topic ...	10	20	15
2	Deák	Dominika	History	History topic 1.	10	15	14
3	Meggy	Márton	Informatics	Informatics topic ...	11	13	4
4	Meggy	Márton	History	History topic 2.	11	16	10
5	Alfa	Rómeó	Informatics	Informatics topic ...	12	20	14
6	Alfa	Rómeó	History	History topic 3.	12	24	20
17	Akashi	Anita	Informatics	Informatics topic ...	9	15	0
45	Árpás	Angéla	History	History topic 1.	10	15	0

Active user: Ádám Szűcs-Szabó, Identification number: ASA-0001, Authorization level: Administrator

Figure 18.: Surface to choose, check and evaluate the individual test sheets

As previously can be read, the evaluation of the multiple-choice task is made by an algorithm, thus, the received points cannot be modified by anyone. It works by comparing the student’s answers with the tasks' truth table and then, regarding to the number of good answers, the algorithm gives the appropriate point(s). Currently the algorithm gives one point if one of the student’s mark is correct, but if it is a wrong answer marking, the algorithm takes away one point (the student has to think about clearly, that her/his answer marking whether correct is or not). However, this point-value cannot take on a negative value, so in the worst-case scenario, it cannot be less than 0 point. In the case of essay tasks, the manually entered point value can range from 0 point to that max point value, which specified during the task editing session. Otherwise, an error message will be displayed.

Here, on the test sheet checking/evaluating interface, almost the whole appearance looks the same as on the test sheet solver interface. But the main difference is, that all fields here are read-only, except for the essay tasks' point-value fields.



*Figure 19: Checking / evaluating the chosen individual test sheet*

After you have completed the check and evaluation, then you can finish the procedure by clicking on the **“Submit checked test sheet”** button.

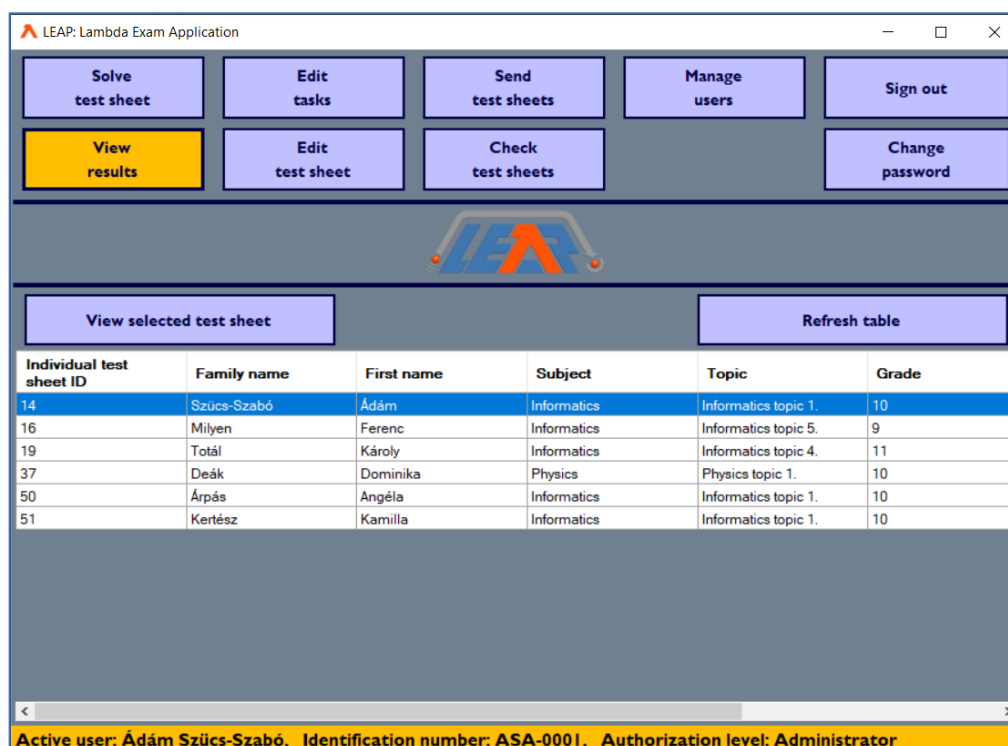


Figure 20: Selection interface for viewing the checked / evaluated test sheets

The own, checked test sheets will be accessible by clicking on the “**View results**” button. In case of multiple-choice tasks, the following things could appear to you:

- what was the student’s / your choice?
- whether this choice was correct or not? (Green check mark ‘✓’ or red x mark ‘✗’.)
- if the choice was wrong, it will show you what the correct answer would have been. (Pale red pointing hand ‘☞’.)

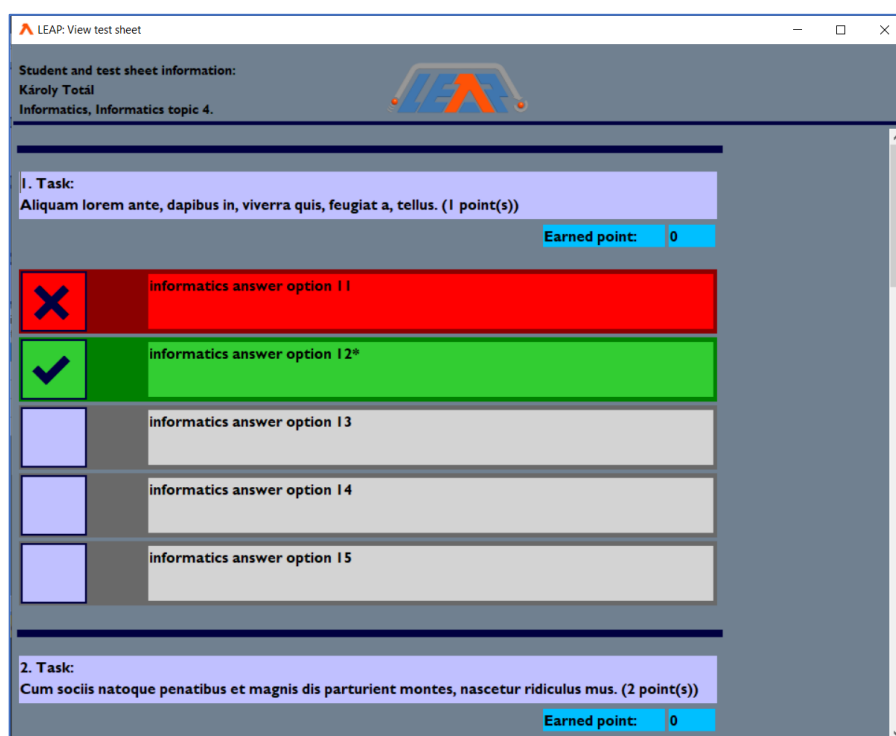
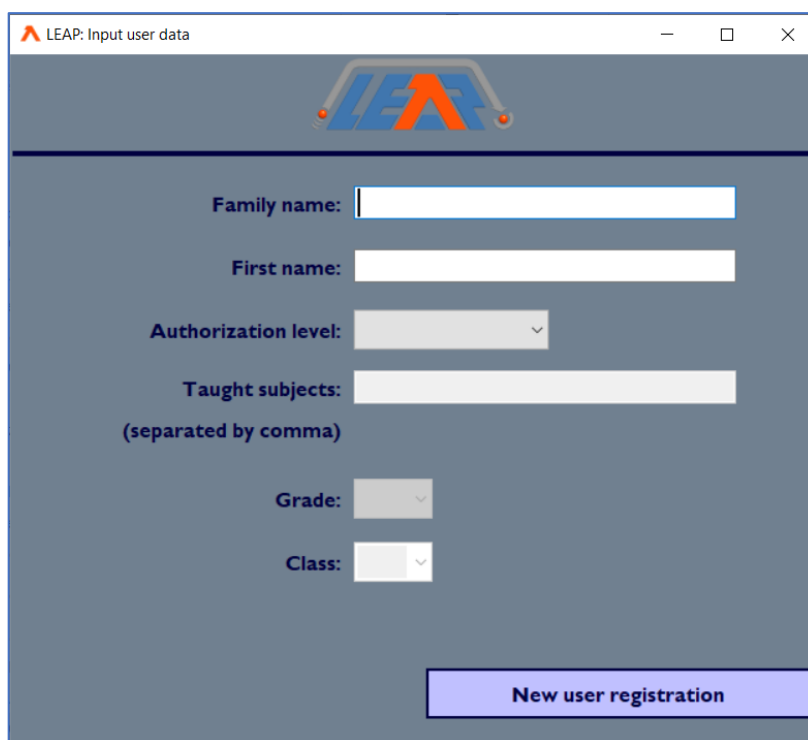


Figure 21: The window for viewing the solved individual test sheet

In case of essay tasks, you also can see your answers as well as how many points did the teacher give to you. This test sheet result viewing interface is very similar to the test sheet solving and checking interfaces, but here all fields and components are read-only. Although, this menu item is available for all users, but the following restrictions apply to the level of access: The student can only view her/his own results. The teacher can only view those test sheet results, that are matching with her/his taught subject. The administrator can view all of the test sheet results.

The image shows a web browser window titled "LEAP: Input user data". The window has a dark blue header with a logo consisting of a stylized 'A' with an orange arrow pointing upwards. Below the header, the form contains several input fields: "Family name:" with a text input, "First name:" with a text input, "Authorization level:" with a dropdown menu, "Taught subjects:" with a text input and a note "(separated by comma)" below it, "Grade:" with a dropdown menu, and "Class:" with a dropdown menu. At the bottom right of the form is a blue button labeled "New user registration".

*Figure 22: Window for registering a new user*

Administrator can view the list of the users by clicking on the **“Manage users”** button. If the administrator clicks on the **“Create new user”** button, a new window will appear and she/he will have the opportunity to register a new user. In addition to giving the name data, it is necessary to decide which authorization level the new user will receive, which choice affects what additional data must be entered. If the new user’s authorization level is **“Administrator”** or **“Teacher”**, then the attribute **“taught subjects”** must be specified. If the authorization level is **“Student”**, then the required attributes are the grade (year) and the class. Filling the visible fields is necessary, otherwise you will receive an error message. The identification number is automatically generated by using the authorization level, the name initials and the SQL ID of the currently registered user. All user’s first password is also their identification number, but it is strongly recommended to change this password after the first sing-in.

The administrator also has the permission to change the personal data of the users or delete a whole user account. If any user's authorization level will be changed in the future, the identification number will stay the same, it will not be changed. And if the administrator wants to delete a user, she/he must accept that all data, individual test sheets and results belonging to this user, will be permanently deleted.

Under the **“Change password”** menu item the user can change her/his password at any time. It is strongly recommended that the users change their first automatically generated password. If the old password is not correct, or the new password and its confirmation do not match, the user will receive an error message.



*Figure 23: Window for changing password*

By clicking on the **“Sign out”** button the user will get back to the sign-in window, where another user can log in.

## **8. Summary and opportunities for further development**

In case of the further development directions, in the early stages of this project, I already formulated several options for the improvement of the given functions and the integration another, additional functions. However, due to the limit of the volume of the thesis and the available time, I could not prepare these further subtasks or functions. But I would like to list a some of them (without the need for completeness):

Data import and export: the editing process and the data-exchange between other systems would be much easier, if there would be opportunity to write into / read from files use for example .csv or .xml extension.

Creating a result-statistics interface, where the users could make statistics using their own or other students' results, regarding to different user preferences. This could be not only an interface where the statistics will appear in texts-form, but these data could appear in well visualized charts.

The visual design of the application definitely needs improvement too. Unique, rounded buttons and further, well-designed controllers will make the LEAP application more nice-looking and more user-friendly.

Nevertheless, comparing the degree of completion of LEAP application to the initial ideas and requirements, I think, that I have succeeded to reach the goals which I formulated at the beginning of the thesis. But I also see that the LEAP is far from a today's modern application, it is still in its very early stage yet. Despite this, I think that I was able to use the knowledge I earned during the education, and I was able to show this knowledge by preparing a complex task. Thus, the application and its documentation created so far, could be a suitable basis to create an even higher-quality application, developed for a wider range of users.

## 9. References

[1]

Ficsor Lajos-Krizsán Zoltán-Dr. Mileff Péter (2011) Szoftverfejlesztés. Miskolci Egyetem, Általános Informatikai Tanszék. 5.1. fejezet

<https://gyires.inf.unideb.hu/KMITT/c02/ch06.html>

Visited: 03-01-2022

[2]

Ficsor Lajos-Krizsán Zoltán-Dr. Mileff Péter (2011) Szoftverfejlesztés. Miskolci Egyetem, Általános Informatikai Tanszék. 8.1. fejezet

<https://gyires.inf.unideb.hu/KMITT/c02/ch08.html>

Visited: 03-01-2022

[3]

Ficsor Lajos-Krizsán Zoltán-Dr. Mileff Péter (2011) Szoftverfejlesztés. Miskolci Egyetem, Általános Informatikai Tanszék. 8.2. fejezet

<https://gyires.inf.unideb.hu/KMITT/c02/ch08s02.html>

Visited: 03-01-2022

[4]

Ficsor Lajos-Krizsán Zoltán-Dr. Mileff Péter (2011) Szoftverfejlesztés. Miskolci Egyetem, Általános Informatikai Tanszék. 11.1. fejezet

<https://gyires.inf.unideb.hu/KMITT/c02/ch11.html#d0e4420>

Visited: 03-01-2022



[5]

Ficsor Lajos-Krizsán Zoltán-Dr. Mileff Péter (2011) Szoftverfejlesztés. Miskolci Egyetem, Általános Informatikai Tanszék. 11.1.4. fejezet

<https://gyires.inf.unideb.hu/KMITT/c02/ch11.html#d0e4420>

Visited: 03-01-2022

[6]

Ficsor Lajos-Krizsán Zoltán-Dr. Mileff Péter (2011) Szoftverfejlesztés. Miskolci Egyetem, Általános Informatikai Tanszék. 11.1.8. fejezet

<https://gyires.inf.unideb.hu/KMITT/c02/ch11.html#d0e4390>

Visited: 03-01-2022

[7]

Microsoft – Visual Studio Community 2019 system requirements

<https://docs.microsoft.com/en-us/visualstudio/releases/2019/system-requirements>

Visited: 03-01-2022