UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Data Science Curriculum

Karel Paan

Development of an Automated Assessment Tool "Silmused" in the Databases course

Master's Thesis (15 ECTS)

Supervisor: Piret Luik, PhD

Tartu 2025

Development of an Automated Assessment Tool "Silmused" in the Databases course

Abstract:

This thesis presents the development and evaluation of Silmused, an automated assessment tool designed for the Databases LTAT.03.004 course at the University of Tartu. The tool addresses the increasing workload of teaching staff and the need for timely feedback for students by automating the evaluation of database assignments. Built as a Python library, Silmused integrates with the Lahendus platform to provide instant and consistent feedback to enhance the assessment process.

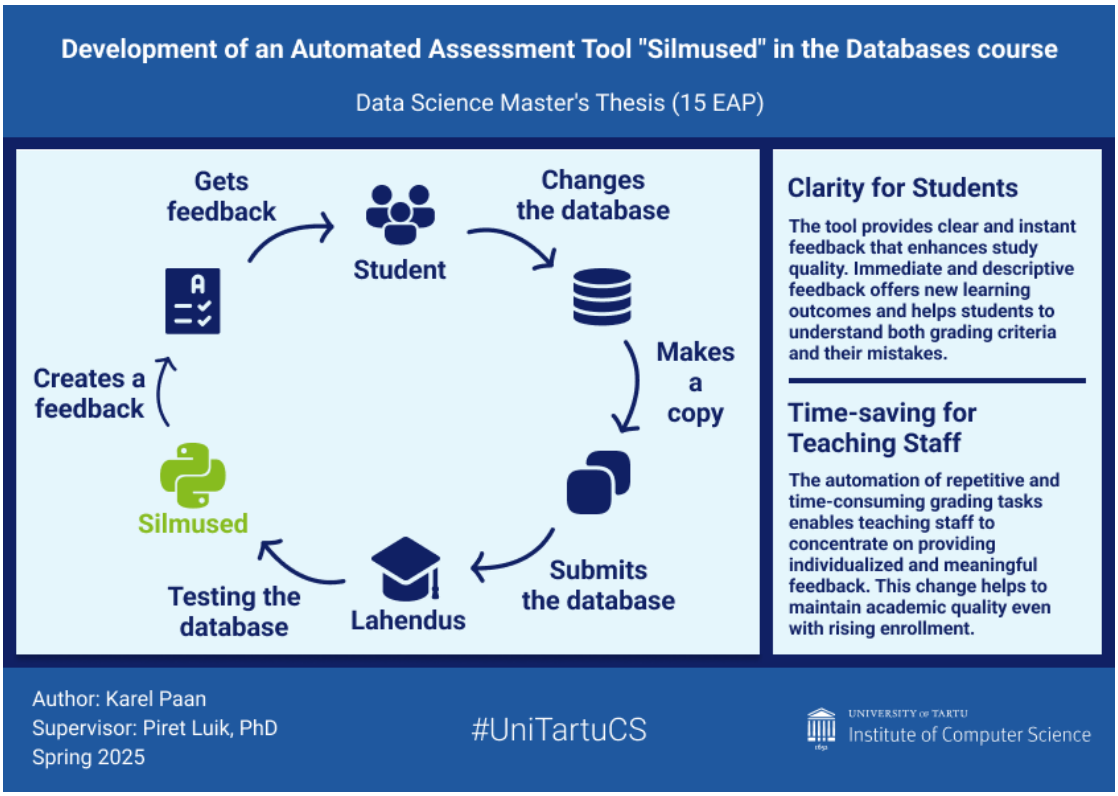
The thesis covers the evaluation of Silmused through questionnaires administered to both teaching staff and students. Results indicate significant time savings for teaching staff, with grading time reduced from 40–100 minutes per task to approximately 20 minutes per task. Students reported high satisfaction with the clarity and immediacy of feedback, though some noted challenges with complex error messages and minor syntax variations.

Key contributions include the tool's modular design for compatibility and its alignment with formative learning principles. While Silmused has proven effective in enhancing efficiency in teaching and learning outcomes, future improvements could focus on refining feedback clarity, accommodating syntax and flexibility.

Keywords:
automated assessment, database, automated tests, programming

CERCS: P170 Computer science, numerical analysis, systems, control; S270 Pedagogy and didactics

Visual Abstract:



Automaatse hindamisvahendi “Silmused” arendamine Andmebaaside kursuse jaoks

Lühikokkuvõte:

Käesolev magistritöö tutvustab automatiseeritud hindamisvahendi Silmused arendust ja hindamist, mis on loodud Tartu Ülikooli Andmebaaside LTAT.03.004 kursuse jaoks. Töö eesmärk on vähendada õppejõudude töökoormust ja pakkuda tudengitele õigeaegset tagasisidet, automatiseerides andmebaasiülesannete hindamise protsessi. Silmused on välja töötatud Python’i teegina ja integreeritud Lahendus platvormiga, pakkudes kohest ja ühtlast tagasisidet, mis lihtsustab hindamisprotsessi.

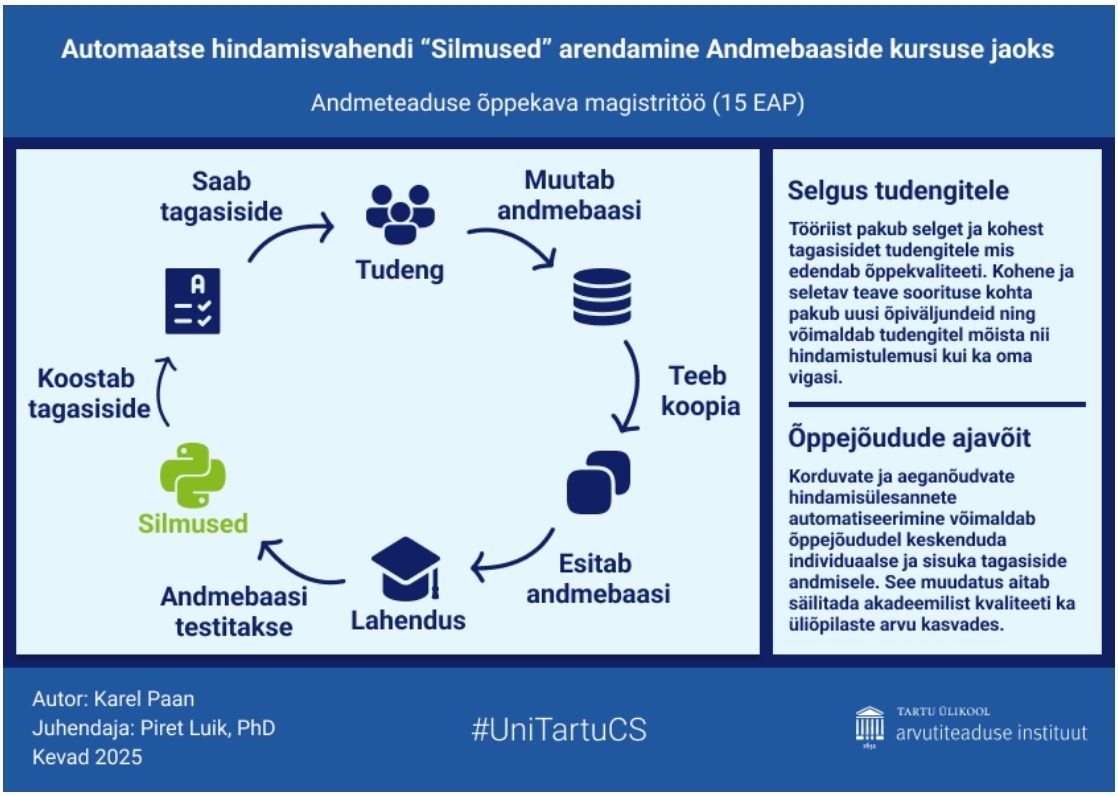
Magistritöö käsitleb Silmused hindamist, tuginedes õppejõudude ja tudengite seas läbi viidud küsimustike tulemustele. Tulemused näitavad õppejõudude jaoks märkimisväärset ajavõitu ja hindamise aja vähenemist 40–100 minutilt ülesande kohta ligikaudu 20 minutini. Tudengid väljendasid suurt rahulolu tagasiside selguse ja kiirusega, kuigi mõned märkisid probleeme keeruliste veateadete ja väikeste süntaksivigadega.

Olulisteks panusteks on tööriista modulaarne ülesehitus ühilduvuse tagamiseks ning selle vastavus kujundava õppimise põhimõtetele. Kuigi Silmused on osutunud tõhusaks õppimistulemuste parandamisel, võiksid tulevased ja täiustused keskenduda tagasiside selguse edendamisele ja süntaksi paindlikkuse suurendamisele.

Võtmesõnad:
automaatne hindamine, andmebaas, automaatsed testid, programmeerimine

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria); S270 Pedagoogika ja didaktika

Visuaalne kokkuvõte:



Contents

Acronyms

AA Automated Assessment 9–14, 17, 18, 21, 28, 30, 31, 36

AAT Automated Assessment Tool 9, 12–15, 17, 21, 25, 27, 28, 30, 31, 33, 34, 36

DB database 11, 14, 15, 17, 18, 21–23, 25, 27, 29–31, 33, 34, 36, 37

PSQL PostgreSQL 15, 17, 19, 22–24, 37

SQL structured query language 14, 15, 17, 18, 28–30, 33

TS teaching staff 9–15, 17–21, 23–30, 33, 36, 37

Introduction

Feedback is an essential part of the learning process. Some need it more, some less, but when gathering new knowledge and skills, one often wants feedback to determine whether one has mastered what one has learned so far, rather than getting the feedback much later. A survey conducted by Hunt and Karm [1] found that the teaching staff (TS) in the University of Tartu are in a loss for time, and this diminishes the feedback given to students. Thus, solutions must be found to reduce the administrative workload of the TS.

The following work focuses on the University of Tartu's course Databases LTAT.03.004, in which the author has been actively involved since 2022. Over the past three years, the course has consistently had more than 300 registered students each year. According to Muuli et al. [2], the number of new information technology students is ever growing. This, in return, increases the workload of the TS. Which prevents the TS from giving timely and meaningful feedback [3].

Prior to this work, a semi Automated Assessment Tool (AAT) by Kakk [4] was already in place in said course to assist with evaluating student submissions. However, it required manual execution by the TS at the end of each assignment period, making it time-consuming and giving no immediate feedback to the students. These limitations highlighted the need for a more automated solution. For this purpose, the AAT Silmused was created by Kakk and Paan.

The purpose of this thesis is to introduce the development process and evaluate the improved approach for the Automated Assessment (AA) of student homework and practice tasks in the Databases LTAT.03.004 course. For collecting evaluations from students and TS, two research questions are proposed:

1. How do TS evaluate the process of providing feedback using the developed AAT?
2. How do students evaluate the clarity, and accuracy of the feedback and assessments provided by the developed AAT?

The master's thesis is organized into four main chapters. The first chapter defines AA, outlines its history and development both at the University of Tartu and globally, and discusses its benefits to the TS and to students. The second chapter details the development process of the AAT created for this thesis, addressing some of the challenges encountered during its creation. The third chapter presents a study involving both the TS and students, using a combination of closed and open-ended questions to gather feedback on the tool.

1 Automated Assessment

In large-enrollment courses, TS often finds it difficult to provide timely and personalized feedback due to a huge number of students [3]. In the course Databases LTAT.03.004 at the University of Tartu, the due date for assignments is one day before the next lab, where TS must provide general feedback on assignments, which is often impractical due to the limited time available for thorough review. Old-fashioned methods, tho they are thorough, are time-consuming and increase the workload of the TS per student. Thus, solutions need to be introduced to remove administrative responsibilities from the TS.

From the student's perspective, receiving timely feedback is highly beneficial, as it allows them to address learning difficulties early, before progressing to new topics. Otherwise, errors may continue in the study process, preventing students from understanding the mistakes made during their learning [5]. Hanmore et al. [6] also suggest that there is a significant drop in quality of study as the time for feedback is delayed. Hence, timely feedback is a core part in a healthy study process.

1.1 What is Automated Assessment?

AA involves using computer-based systems to evaluate student work with little to no human involvement [7]. Such systems utilize predefined rules and algorithms to assess various tasks, including multiple-choice questions, short-answer responses, code submissions, and even open-ended essays [8]. Nowadays, large language models have also paved a new way into the field [9]. The primary goal of AA is to enhance efficiency, consistency, and scalability in the evaluation process, especially in large-scale educational environments where manual grading is not feasible. While rule-based systems use programmed logic to compare student answers with correct solutions, more advanced methods—like intelligent tutoring systems or Artificial Intelligence-based graders can analyze student input in greater depth, providing feedback similar to that of human evaluators [10]. For instance, English et al. [7] discuss the use of AA in computer science courses, highlighting its role in providing immediate feedback and reducing grading workload.

1.2 Formative Learning with Automated Assessment

AA is essential in supporting formative learning, a teaching approach that prioritizes continuous, real-time feedback to help students improve their understanding and develop their skills. Unlike traditional summative assessments, which focus only on grading, formative learning is designed to assist students on their learning journey through iterative feedback and opportunities for self-correction [1, 3, 11]. A AA system promotes formative learning by offering immediate, personalized feedback to students on their

submissions. This feedback assists students in recognizing and addressing misunderstandings before they become ingrained, which promotes a deeper understanding of the material, for instance, in coding tasks, a AA system can identify syntax errors, logical mistakes, or inefficient solutions, guiding students towards better coding practices [5].

AA further enhances formative learning by permitting multiple submission attempts, enabling students to continuously refine their solutions based on the feedback received. This is in line with research indicating that students who receive frequent, high-quality feedback exhibit improved learning outcomes [12, 13]. Formative learning through AA not only boosts student engagement but also fosters a growth mindset, encouraging students to see mistakes as opportunities for learning rather than failures [5, 12]. This approach is especially effective in technical fields like database (DB)s, where hands-on practice and iterative problem-solving are essential for achieving mastery [14].

1.3 Demand for Automated Assessment

Based on Muuli et al. [11], automation is essential for instructors who teach courses with many students to decrease the high workload. Information technology is one of the most popular specialties at universities and the trend keeps on growing. According to the authors, when writing their article, more than 300 information technology students were accepted to the University of Tartu. There is also a big demand from other specialties for information technology courses. As the number of students taking information technology courses is only getting larger, the TS needs tools to maintain the load. The work needed per student (assessment, grading, feedback) is often repetitive and thus can be automated. With a large number of students, it is hard to track everyone's progression. This is another benefit of automation. Furthermore, automation reduces the risk of human error when grading. Automation also helps in creating new assignments as the assessment is programmatic, similar software tools can be used to create more test cases by using an automatic item generator, "Process of using item models to generate test items with the aid of computer technology.".

1.4 Automated Assessment Design

In the article by Muulit et al. [2] AAs must have a well-thought-out grading system, as even a small inconsistency can affect a student's learning outcome. On the other hand, AAs force the TS to justify every grading decision made in an assessment. Another danger of AAs is that the students can get lazy, this can be avoided by e.g., limiting the number of submissions or hiding some test cases before the final assessment. Feedback can be given in two phases, during compilation or execution, and after, when the output is compared to the desired outcome. In the article, Muuli et al. interviewed the University

of Tartu Institute of Computer Science's TS on the question of AA and feedback and concluded that the state of mind of the staff was more towards using automation for non-difficult exercises and novice courses, and also for massive open online courses where the number of learners is even greater. A point was made that automation can only be used to test fact-based understanding, but not the taught process itself. A possible drawback of assessment is that the students might put their effort into the feedback rather than the topic in general. It was found that instant feedback is of the utmost importance, particularly with homework. Furthermore, it is adamant that upon an error, the feedback should point to the origin of the error and should also give instructions on how to solve it. It was stated that the worst outcome of the assessment is when it runs into an internal error, and thus, students get no feedback.

Tasks need to be tailored in such a way as to keep the students from finding out the solutions by hand [15]. The TS should be able to add or edit feedback at each stage of assessment and provide general comments at the end, general feedback should be saved for reuse in future assessments [3]. Kleerekoper et al. [14] solved this issue by making a hidden set of tables that were used for testing but hidden to students, the final test looked at both set of tables, ones public to students and ones hidden, thus assuring that no answers were hardcoded.

In higher education, maintaining the integrity of assessment processes is essential, universities must ensure that assessments accurately measure student learning, providing a fair and consistent evaluation of their skills and knowledge [16]. Automated assessment tools offer a solution by providing scalable, objective, and efficient evaluation methods, ensuring that feedback is both consistent and timely [11]. Nevertheless, universities need to satisfy both internal and external stakeholders, such as the public and future employer,s and uphold academic standards, ensuring course quality, and graduating based on students' achievements [17].

1.5 Benefit of Automated Assessment for Teaching Staff

AATs increase the productivity of TS by providing ease and efficiency in making objective assessments in cases like: multiple-choice questions, fill-in-the-blank assessments, and coding exercises. Repetitive tasks such as these take a large amount of time, so getting rid of them allows TS to spend more time giving instructions to students on more difficult topics [15]. Furthermore, grading consistency is improved across large classes,s and organizational workload is reduced. Kleerekoper et al. [14] found that a lot of the TS's time was spent on grading instead of helping. AATs can give aggregated information about the student's results and trends, which can help TS to plan lessons accordingly [8]. Although the initial effort is significant, AATs provides lasting reductions in workload, enabling the TS to concentrate on student support and research that keeps the curriculum current [15]. Vittorini et al. [18] found that AATs help cohort students, which helps

with tailoring support to a group of students. Additionally, AA keeps feedback clear and consistent, helping the TS to respond fairly to all students and making it easier to moderate and review the results [3].

1.6 Benefit of Automated Assessment for Students

In a study, Kristensen et al. [12] found that students that got feedback from both: TS and AATs presented more accurate solutions and were far more faster than students that only had feedback from TS. This means that TS can not be replaced but rather tools for individual learning should be created to enhance the learning process. In another study conducted by Chirwa [13] they found that students who had more submissions also had better results in the end, nevertheless, when the answers could be obtained only by resubmitting, students opted for that rather than revising their knowledge. This means that the answers that AATs give, should clearly note what is wrong with the submission rather than give exact hints on how to solve a problem. In a study conducted by Queirós et al. [19] they had two groups of students, one that used PETCHA, "a programming exercises teaching assistant" and another that got feedback from TS, they found that the first group was able to solve more exercises in a class but the TS aided group had better and more feedback.

One of the biggest perks of having a AAT is giving live feedback to students so that they would not have to wait in line to ask the TS. As the assessment is done on time, it gives way to students with different capabilities, if tasks in class are done in the traditional way, everyone would move in the same pace. Also, where some students might struggle with one problem and already have the skills for another and vice versa, it is more efficient to not go through these topics in a row. Another benefit is the removal of human bias in assessment as everyone is graded by the same criteria. Furthermore, some students might be dealing with social anxiety or other similar complications, in which case a AAT fits their needs perfectly.

When implementing AATs, one should keep in mind not to remove the human aspect entirely; these tools should rather be used for assessments that have empiric/fixed criteria. Also, after the final assessment, a TS should go over the result and give personalized feedback on points that a student could not cover.

1.7 Prior Works in Automated Assessment

AA in higher education has evolved significantly throughout the past several decades. The earliest notable development was in the 1960s with Ellis Batten Page's creation of Project Essay Grade, a pioneering system that utilized computational linguistics to evaluate student essays [20]. Building on the work of Page's Project Essay Grade, the field of AA in higher education has seen significant advancements over the years. A key

development occurred in the 1970s with the creation of Computer-Assisted Instruction systems. Among these, the PLATO system, created at the University of Illinois, was particularly notable. It utilized the TUTOR programming language to deliver engaging lessons and assessments, paving the way for future systems [21]. During the 1980s and 1990s, there was significant improvement in the creation of advanced Intelligent Tutoring Systems, including examples like Auto-Tutor and Cognitive Tutors. These systems utilized artificial intelligence methods to simulate student understanding and used personalized feedback [22]. In the present time, the emergence of large language models has created new and exciting opportunities for automated evaluation. These models are capable of providing detailed feedback and assessing complicated student answers, giving way to better scalability and adaptability in AA [23].

The evolution of AA in DB education has coincided with advancements in educational technology, meeting the growing demand for efficient and scalable evaluation methods. In the early 2010s, Kovacic et al. [15] introduced an automated grading system integrated into the Moodle learning management platform, which was designed to assess students' skills in spreadsheets and DBs. This system provided immediate feedback and significantly reduced the grading workload for the TS by automatically evaluating student inputs without requiring manual intervention [15]. Building on these foundations, Kleerekoper et al. [14] developed the structured query language (SQL) Tester tool, which was implemented at Manchester Metropolitan University starting in the 2017/18 academic year. This tool was designed to automatically assess SQL programming assignments, providing timely feedback to students and reducing the grading burden on TS, over five years of use, SQL Tester proved effective in enhancing student engagement and was adapted for various educational contexts [14]. Vittorini et al. [18] made significant advancements by developing an artificial intelligence based system designed for both formative and summative assessments in data science courses, this system offered automated feedback throughout the coursework and showed a strong correlation with human grading, which means that the system is both reliable and effective in evaluating complex student responses. These developments show a broader trend in higher education to utilize AATs to enhance both the efficiency and effectiveness of teaching and learning in courses related to DBs.

1.8 Prior Works in Database Automated Assessment in the University of Tartu

The first attempt in the University of Tartu to create a AA solution for databases was in 2015 by Marten Siiber, which was a Java program using the TestNG framework [24]. The second was by Robert Sepp in 2018, the main difference between the first was that it used no framework for assessment [25]. Both solutions were intended for SQL Anywhere, and they both targeted different DBs that were used in the Database's course [25, 24]. In

2019, Õunamaa created a AAT for the course Introduction to Databases MTAT.03.105 which also used Java and was intended to use Moodle for student submissions, but this part was never fully developed [26].

According to Kakk the work done by Simmer was outdated and rarely used, but it was the basis for his work [27]. Kakk's solution was to use SQL scripts rather than a code framework for assessing a SQL Anywhere DB, which improved the time it took for assessment from 10-40 minutes to 2-5 minutes per student [27]. In 2023 there was a similar attempt by Kakk, that used PostgreSQL (PSQL) instead of SQL Anywhere [4].

The author of this thesis has been a TS for the Databases course from the Spring semester of 2022, back then SQL Anywhere was used, and he taught one class (approximately 20-24 students in a class) per semester. The next year, he had two classes which meant an increase in the workload for assessing students' homework. In that year, PSQL was first introduced alongside Kakk's solution [4]. As the assessment time doubled, there came an idea to use Python to automate the whole process altogether. The way that homework was assessed in the Spring of 2023 is as follows: the assessor first needed to make a copy of the DB and then restore a student's backup file, after which they needed to run the test scripts [4]. The result of running the script was an SQL table with the list of assessments and how much points was gained from each, after gaining the result a member of the TS then had to submit the sesutls on Moodle to a student. All of this could be done after the deadline. However, this was time-consuming as for the TS as assessment would take 40-100 minutes per task and the students did not get any instant feedback.

1.9 Database Assessment

In the author's opinion, when writing tests for code, it usually consists of checking a block of logic that should have a desired outcome, in most cases a function and sometimes we might want to test how many of these blocks work together. There can be any number of variables passed in, and each combination should have one deterministic output. This is the most straightforward way to describe the process of testing our code, but more often than not we also need to worry about how data is manipulated by our code. The process remains similar, after a block of code is executed, we check the outcome in our DB. This, in the author's opinion, is the way most back-end developers would think when writing tests. When testing DBs directly, we also need to check the values of our data before and after a DB function, trigger, etc. is executed. This before-and-after testing is the difference between back-end tests and tests needed for a DB AAT.

DB types can be categorized into three big categories: structural testing, functional testing, and non-functional testing [28]. Structural testing focuses on a DB's essential components that are implemented accordingly, such as schemas, tables, relationships, constraints, etc.

[29]. Functional testing focuses on checking if transactions and operations perform the way they are intended to [30]. Non-functional testing consists of checking everything outside of primary functionalities, such as performance, security, etc [29]. As non-functional tests are not required for the DB course, they will be discarded in the thesis. Both functional and structural testing are useful, but their performance can depend on the use case and the characteristics of an error [31].

2 Implementation

This chapter outlines the development of a AAT for DB assignments, created to tackle the dual challenges of alleviating grading workloads for TS and providing prompt, constructive feedback to students. The initiative began in 2023 with a basic script-based approach that leveraged Bash and Python for automating grading. However, it quickly became apparent that a more scalable, secure, and user-friendly system was necessary. This realization prompted the transition to the Lahendus platform, which facilitated containerized execution and seamless integration with Moodle, thereby establishing a consistent and robust environment for assessments. This evolution prompted the package called Silmused, a Python package with an object-oriented solution that allowed for flexible, modular test definitions and clear feedback generation. By exploring the architectural decisions, technical advancements, and pedagogical considerations that influenced this system, this chapter illustrates how an initially simple tool transformed into a comprehensive, scalable assessment solution that not only automates the evaluation of DB tasks but also enriches the learning experience for students.

2.1 First Attempt

The first attempt was made in 2023 with the goal to decrease the assessment load of the TS, it was born out of the author's personal need to grade two classes instead of one as in the year prior, such a need was also clearly stated in Section 1.3. The code can be found in Appendix 1, it is inspired by how SQLAlchemy, a Python Object Relational Mapper which uses simple abstractions to decrease the load of writing preparing SQL. All the test cases were taken from [4] and converted from PSQL into Python The solution used a Bash script (file "convert.sh") to restore PSQL backups and convert them into dump files, to standardize them and replace COPY commands with INSERT so that they could be processed in a similar fashion. Further parts were implemented with Python.

The initial attempt aimed to introduce automation into the DB assessment process, in line with the theoretical framework of AA outlined in Section 1.1. This initiative aimed to provide consistent feedback while reducing the workload for TS, as mentioned in Section 1.3. However, the focus of this approach was mainly on functional DB assessment, described in Section 1.9, emphasizing task validation rather than improving formative learning for students.

The "main.py" script was essential to the testing workflow as it combined everything together, the script performed critical DB operations: It started by creating a temporary PSQL DB for each student's submission with the psycopg2 library, assigning a unique name to ensure that test runs remain isolated from one another; After establishing a connection to the temporary DB, the script populated the DB schema and data by executing a student's dump file; Substitutes schema names and ownership directives

were given to align with the configured environment. This ensured compatibility across different setups, which is important for maintaining assessment reliability, as highlighted in Section 1.4. Further more it provides an interactive command-line interface that allows to choose a specific practical session or homework assignment for evaluation. Upon selection, the program dynamically loads the relevant test cases and runs the corresponding test functions. The test results are then recorded in a structured spreadsheet file, which includes feedback for each test, correctness labels and point allocated.

All the test cases are organized within a dedicated package (folder) called "tests," which serves as a modular repository for evaluating practical sessions and homework assignments. Each submodule within this package—such as "praks3.py" contains a set of test definitions from "checks.py" tailored to the corresponding assignment. These test definitions are structured as Python functions that return lists of test case dictionaries. Each dictionary describes a specific DB (validation) task, such as checking for the existence of a table, verifying a column's default value, or validating the configuration of a SQL function. The "checks.py" file also contains the logic to run said configurations in a DB session created in "main.py", each run returns an array of feedback elements mentioned under "main.py".

For the TS this meant that, they needed to download students submissions from Moodle and put them into the corresponding practice session or homework folder. Then, from the command line, when executing "main.py," they chose said task and as a result, they got back a spreadsheet that contained all the checks that were made, how many points they got/lost, and feedback when something was wrong/missing. The spreadsheet used a structured feedback format, which is essential for formative learning purposes, mentioned in Section 1.2. This way of assessing tasks eliminated the need to manually recreate a student's DB and run the assessment file. Furthermore, this made updating and adding test much simpler as test cases took fewer lines of code and old parts of them could be reused respectfully. This modular design supported scalability, enabling the TS to extend or modify test cases as needed, consistent with the design principles of AA discussed in Section 1.4.

Despite these advantages, the initial solution had several limitations. Most significantly, it did not offer instant feedback to students, which is a critical element of formative learning as mentioned in Section 1.2. Instead, it required the TS to download all the submitted backup files, put them in a dedicated folder and run Python script. Moreover, there were some complications: the solution only ran on ARM chips; the initial setup needed a ".env" file to be configured. All of the weaknesses considered, it was not ideal for the TS nor the students'.

2.2 About "Lahendus" Platform

Lahendus is a platform based on Easy, both are managed by the Institute of Computer Science at the University of Tartu [32, 33]. Lahendus allows students to submit their Python or PSQL solutions and gives instant feedback. It is integrated with Moodle such that Lahendus'results are directly sent there. Lahendus is implanted so that each assessment starts a Docer container which runs a set of evaluations in Python. Originally, it was used for assessing students' tasks in Python-related courses.

Logic for assesment in Lahendus is added as Python package, for Python type tasks, "python-grader" package is used. The package was initially created by Puulmann in his bachelor thesis and later has been forked by Papli [34, 35]. This package allows users to specify test cases in a structured and reusable format, similar to unit testing in software development. A simple example to test a "Hello World!" task with python-grader is shown in Figure 1.

```python
from grader import *
from grader.utils import *


@test
@expose_ast
@set_description("Hello world!")
def test1(m, AST):
    assert not ast_contains_name(AST, 'input'), 'No input() required for this exercise.'

    out = m.stdout.read().strip()

    assert not ("Hello" in out), 'Expected to find' + quote_text_block('Hello') + ' but found: ' + quote_text_block(out)
    ...

```

Figure 1. Lahendus' Python Test Example [32]

The transition to Lahendus addressed several limitations observed in the earlier assessment attempt which relied on a Bash script and a local PSQL setup, both of which posed platform compatibility issues and introduced significant setup overhead for TS. By contrast, Lahendus encapsulates all dependencies within Docker containers, removing the need for any local installation and guaranteeing cross-platform compatibility. From a security perspective, containerization provides important safeguards. Each student's submission is processed in a clean, ephemeral environment that is destroyed after execution, ensuring that no malicious or broken code can affect nor bypass the system. This isolation also guarantees that every submission is graded under identical conditions, thereby upholding fairness and consistency in assessment.

The whole process works as follows: a Docker file is run and it includes a test framework as a PyPI package. The container runs "evaluate.sh" which goes to the file with a student's submission "student-submission" and runs the assessment script that is responsible for loading the task data and running assessments from a dedicated file. The runner container

should give a response in a specified JSON format, shown in Figure 2 to Lahendus portal.

```
{
  "message_type": "OK_V3",
  "message":
  {
    "result_type": "OK_V3",
    "producer": "tiivad v1.2.3",
    "finished_at": "2023-08-26T21:56:00Z",
    "points": 100,
    "pre_evaluate_error": null,
    "tests": [
      {
        "title": "<test title>",
        "status": "PASS",
        "exception_message": null,
        "user_inputs": ["<input 1>", "<input 2>"],
        "created_files": [
          {
            "name": "<filename>",
            "content": "1\\n2\\n3"
          }
        ],
        "actual_output": "1\\n4\\n9",
        "converted_submission": "<modified submission code>",
        "checks": [
          {
            "title": "<check title or asserted condition>",
            "feedback": "",
            "status": "PASS"
          }
        ]
      }
    ]
  }
}
```

Figure 2. Lahendus Assessment Response [36]

Lahendus offers a clear and organized overview of each student's submission history. For every task, the platform tracks the number of submission attempts, the scores achieved, and the timestamps for each attempt. This feature allows both students and TS to easily monitor progress, identify patterns of improvement, and recognize potential difficulties early in the learning process. Additionally, by permitting multiple submission attempts with immediate feedback, Lahendus promotes iterative learning, encouraging students to refine their solutions based on the feedback received. An example of the submission overview available to both students and instructors can be seen in Figure 3.

Figure 3. Lahendus' Submission Overview Example [32]

To extend Lahendus for DB-related assessment, building upon the first attempt, mentioned in Section 2.1, a testing module designed specifically for DB assessment, called Silmused, was made by Kakk and Paan that encapsulates the definitions of DB tests in a Python package.

2.3 Final Solution

There were some subtle hints from a few members of the Didactics of Informatics Working Group of the University of Tartu that there is a need for a AAT in the DB course. As there already was a platform for hosting AATs as mentioned in Section 2.2 and an initial prototype for assessing DB tasks in Python, mentioned in Section 2.1, the decision was to combine them. This way a seamless solution could be offered to the TS and the students could receive instant feedback. As the assessment logic in Lahendus needs to be a PyPI package a package called Silmused was introduced by Kakk and Paan, the code for this is shown in Appendix 3.

2.3.1 Methodology

This subsection discusses the development and implementation of Silmused. The author's original contribution to the platform is shown in Appendix 2. Lahendus platform is utilized to directly address the theoretical requirements for scalable, consistent, and robust AA as described in Section 1.2. This approach not only improved the consistency of feedback but also provided real-time feedback for students, which is a crucial aspect of formative learning mentioned in Section 1.4. Additionally, the implementation of Docker

containers resolved security and compatibility issues, ensuring fairness by assessing each submission in the same standardized environment.

One goal of the new solution was also to make the code more readable. For this, the author opted for an object-oriented programming approach rather than a functional approach, which was described in Section 2.1. This allows the test cases to share logic through abstractions and follow similar patterns, defined in "TestDefinition.py". Which in turn makes implementing the tests easier, as all the cases present a uniform frame. Such an approach is inspired by how unit tests in software development work in general, namely Laravel's object–relational mapping - Eloquent, a structured way of describing a DB in PHP was taken as a reference point. The initial rewrite to an object-oriented approach was done by the author shown in Appendix 2. Further improvements and implementing the logic as a package were done together with Kakk. The package structure blueprint was taken from "python-grader" [35]. Also, an improved feedback system with translations was added by Kakk.

2.3.2 Structure

The author tackled several technical challenges during implementation. DB connection management was enhanced by using an abstraction layer that streamlines credential management and connection establishment. File validation was also introduced to ensure that only valid PSQL dump files are processed, preventing errors that may arise from malformed input files. Test execution follows a standardized pattern, which includes pre-execution setup, query execution, result validation, and response generation. This consistent approach simplifies the maintenance and extension of the framework. As a result, new test types can adhere to the established pattern while incorporating their specific validation logic. Error handling was given careful consideration, with the author implementing transaction rollback mechanisms to preserve DB integrity, even when tests fail due to execution errors. This approach ensures that subsequent tests remain unaffected by previous failures, thereby maintaining the reliability of the assessment process.

The implementation utilizes Python's psycopg2 library for interacting with PSQL, taking advantage of its robust cursor-based execution model. System processes are managed using the subprocess module, enabling the framework to work with PSQL command-line tools for tasks such as DB creation and restoration, which is later used by the Docker container. Test results are organized in a consistent format, provided by Lahendus requirements, providing clear information about outcomes, points awarded, and specific feedback messages. This structured approach allows for flexible presentation of results, ranging from simple console output to integration with educational platforms. Using this framework, the author has created a solid foundation for a consistent and objective assessment of DB implementations in educational settings. This approach addresses

the dual needs of reducing instructor workload while delivering valuable feedback to students.

2.3.3 Usage of Silmused

After Silmused was introduced in 2024, the way students submit their assignments is as follows: they open up the Lahendus platform, select an assignment, upload a SQL dump file of their DB, receive assessment and feedback. The feedback is given in an orderly fashion as shown in Figure 4. The assessment is propagated to Moodle and thus relieves the TS from having to change it manually. Tests are well organized under each assignment and accessible to the TS.

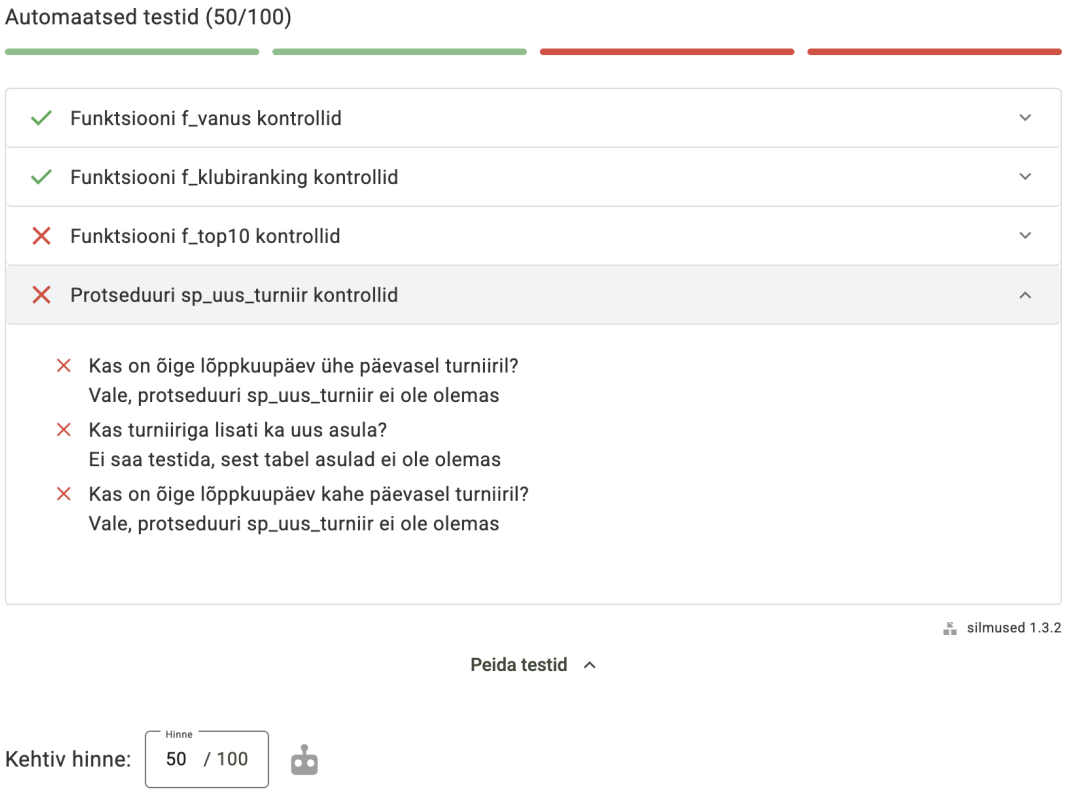


Figure 4. Example of Silmused Feedback in Lahendus [32]

2.3.4 Comparison with the First attempt

The original solution relied on a Bash script and a locally configured PSQL instance to automate the assessment of DB tasks. While functional, this approach presented challenges: TS had to manually download student submissions, configure their own

PSQL environment, and contend with compatibility issues, especially since it only operated on ARM architecture.

In contrast, the updated solution features a robust, object-oriented design encapsulated in a Python package called Silmused. It utilizes the Lahendus platform, which employs Docker for secure, cross-platform execution, ensuring that each student's submission is evaluated in a clean, isolated environment. This transition eliminated the need for local PSQL setup and manual file management.

Additionally, the test definitions were reorganized using a class-based structure, which enhanced clarity and extensibility. The feedback system was significantly improved with multilingual support, developed in collaboration with Kakk. The new solution also provides instant feedback to students, offers seamless integration with Moodle, and enhances security through containerization. Overall, it has made the assessment process more scalable, maintainable, and user-friendly for both students and TS.

3 Evaluation

This chapter presents an evaluation of the AAT Silmused, used in the University of Tartu's DB course, guided by two research questions: (1) How do TS evaluate the process of providing feedback using the developed AAT? (2) How do students evaluate the clarity and accuracy of the feedback and assessments provided by the developed AAT? The evaluation focuses on two primary stakeholders: TS and students. Feedback from TS assesses the tool's impact on grading efficiency, feedback quality, and workload reduction. Student feedback examines the clarity, accuracy, and usefulness of automated feedback. This chapter offers an analysis of the AAT's effectiveness, highlighting its strengths and areas for improvement.

3.1 Teaching Staff Questionnaire

To evaluate the effectiveness of the AAT Silmused on the TS, a questionnaire was conducted for those actively involved in delivering the DB course at the University of Tartu. The purpose of the questionnaire was to assess whether the AAT met the expectations and needs of the TS, particularly regarding workload reduction and the ability to provide timely and consistent feedback. These factors are directly related to the thesis's research question: "How do TS evaluate the process of providing feedback using the developed AAT?"

3.1.1 Methodology

The questionnaire shown in Appendix 4 was distributed to members of the TS in the year 2025 who had firsthand experience utilizing the AAT to evaluate student submissions. The form included both closed and open-ended questions. Key areas of focus included time savings, ease of use, clarity, and usefulness of the feedback from the AAT, and overall satisfaction with the system as a whole. Participation was voluntary and anonymous to promote honest and constructive results. Descriptive statistics were used, and in the open-ended questions, the most frequently mentioned thoughts were highlighted, as well as the thoughts that, in the author's opinion, were most important.

3.1.2 General Feedback

On a scale from one to five, the TS were asked to rate satisfaction, trustworthiness, and understandability of Silmused, as shown in Figure 5. The satisfaction average was 4.4, indicating a high level of user satisfaction. The trustworthiness average was 4.2, suggesting that TS generally found the system reliable and credible. The understandability average was 4.3, reflecting that the TS found the system's interface and instructions to be clear and easy to comprehend. Results demonstrate that Silmused achieved a strong

positive response across all three metrics from the TS. Satisfaction and understandability were also measured by Kakk in 2023 when the old system was in use, he also questioned 12 members of the TS. The average feedback for satisfaction was 3.4 and 4.4 for understandability [4]. This shows that the satisfaction has risen by a whole point, whereas understandability has dropped 0.1 points, which could be explained by the TS having to use a whole new process for grading, whereas Kakk's solution was similar to the old one.

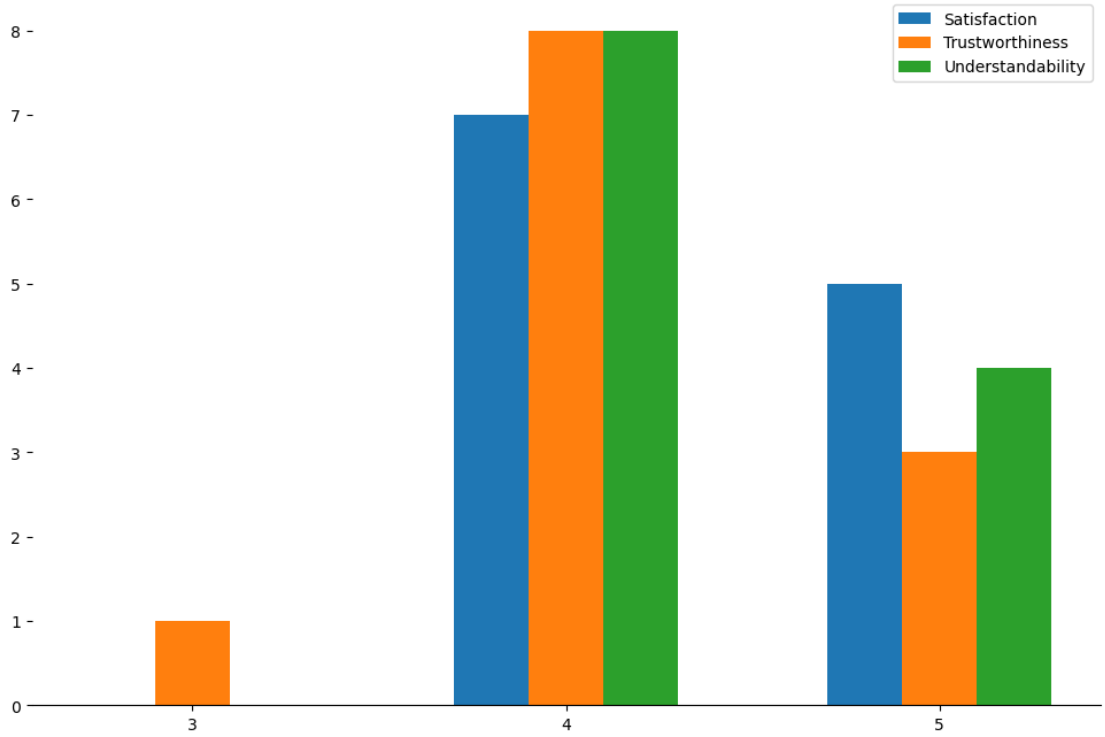


Figure 5. General feedback from the TS

On average, the TS spent 20.4 minutes total on grading homework and on average a total of 17.8 minutes on grading Lab Work. Total time spent per member of the TS is shown in Figure 6. Each bar in the figure illustrates the total time each member of the TS dedicates to grading a single homework or lab work assignment. The distribution of grading time varies among TS members, some allocate considerably more time to homework than to lab work, whereas others exhibit a more balanced approach. Notably, grading homework typically demands more time for the majority of members, likely due to the complexity and detailed evaluation involved. In contrast, lab work usually requires less time, indicating a more straightforward assessment process. In 2020, Kakk measured that the time it took for a TS member per student assignment was 2-5 minutes

[27]. As there are approximately 20 students per class, this means that grading took 40 - 200 minutes per assignment. From the results, we can see a positive reduction of time spent per member of the TS.

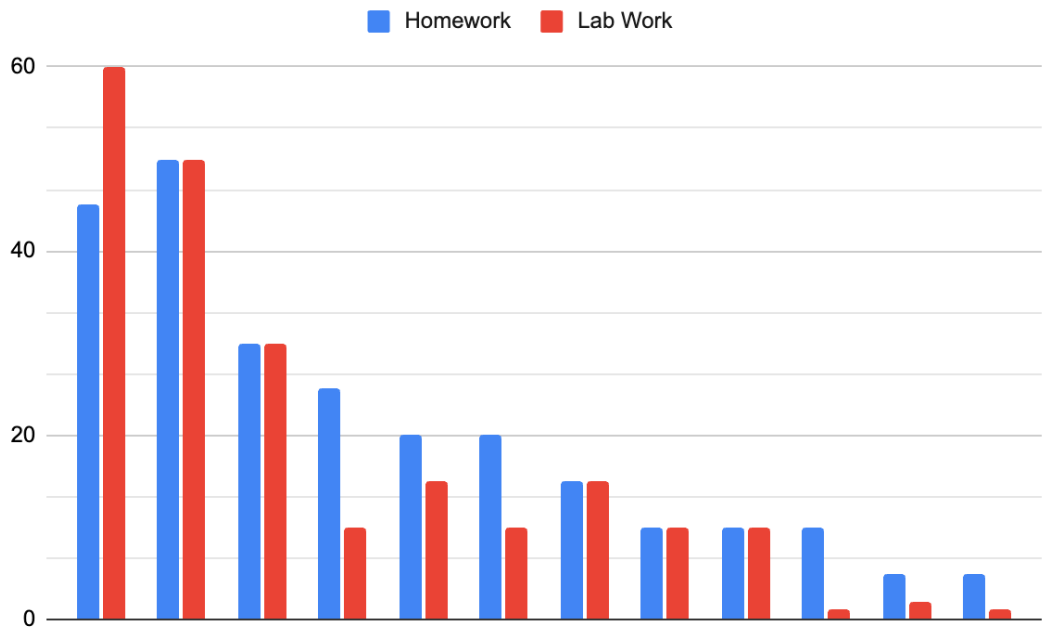


Figure 6. Average time for grading per member of the TS

Feedback from TS members revealed a predominantly positive response to the AAT. Numerous members of the TS highlighted a significant decrease in time dedicated to grading, largely owing to the tool's capacity to automatically restore student DBs, execute test cases, and provide structured feedback. Many TS members observed that the transition from manual assessment to automated workflows enabled them to concentrate more on pedagogical elements, such as individualized consultations and the creation of new exercises.

A common observation was that, although the initial setup and familiarization with the system demanded considerable effort, the long-term benefits far outweighed this investment. The capability to reuse and enhance test cases through Python-based test definitions was noted as a significant advancement over prior solutions. Additionally, the integration with the Lahendus platform and its connection to Moodle facilitated a streamlined grading experience, effectively eliminating redundant administrative tasks.

Some members of the TS identified technical limitations in earlier versions, including container compatibility challenges and issues with clarity in feedback during edge cases

(for instance, when a SQL dump was malformed or had an unexpected encoding). These concerns were subsequently addressed in later iterations of the tool, featuring the introduction of detailed error messages and improved input handling capabilities.

3.1.3 Suggestions from the Teaching Staff to Improve Silmused

The TS were asked to provide feedback on Silmused, with the aim of identifying its strengths, limitations, and potential areas for improvement. Specifically, they were encouraged to share their experiences regarding the need to correct assessment results generated by the tool manually, highlight any recurring issues they had encountered, and suggest enhancements that could make the tool more beneficial for both them and students. This provided valuable insights into the practical challenges associated with AA and revealed several critical areas where the tool could be improved.

One notable issue is the inflexibility of Silmused in recognizing alternative correct answers; for instance, situations where student submissions included additional columns in database tables, which were not explicitly required but did not affect the correctness of the solution, were mentioned. The inflexible design of the AAT led to these submissions being incorrectly marked as incorrect.

Another recurring challenge with Silmused is its tendency to penalize students for minor errors like syntax or formatting mistakes. For instance, a student may lose significant points simply because a column name contains an underscore instead of a space, even if the logic of the query is correct. This strict scoring approach not only discourages students but also misrepresents their actual understanding of the subject. To enhance the educational value of the assessment, the system should adopt a more nuanced evaluation method that can identify the nature of such errors and provide targeted feedback, or not penalize them for minor errors.

The feedback generated by Silmused can be unclear, especially when assessing functions or procedures. Students have reported receiving vague or misleading error messages, such as indications of unexpected errors during automatic assessment, which do not accurately reflect the nature of the problem in their solutions. Such feedback leaves students confused and unable to effectively correct their mistakes. For example, in a specific assignment, a student was penalized for sorting by the wrong column, even though the final result was correct. These ambiguous messages make the learning process more difficult, prompting a need for a feedback mechanism that can clearly identify errors and suggest possible solutions. One proposed solution was to include the links from Problem Solver, which is a tool inside the course used to help students.

Finally, the process of maintaining and updating the automated assessment system is demanding, changes in the underlying data structures or the introduction of new content even more so. For exampel dates changes each year and need to be adjusted, also new

test cases are added each year. Developing a more adaptive assessment system, which can dynamically adjust to changes without extensive manual intervention, would greatly improve the problem.

3.1.4 Suggestions from the Teaching Staff to Improve Lahendus

As Silmused ran in the background of Lahendus, several suggestions came to light on how the whole process of automation, which uses both Silmused and Lahendus, could be further improved. One proposal was to enhance the feedback customization features, allowing instructors to add personalized notes directly within the Lahendus interface. Another recommendation was to introduce analytics dashboards for TS, enabling the tracking of class-wide performance metrics over time. These enhancements would further support the objective of providing individualized, data-informed feedback.

In the initial practical sessions, students submit SQL code, which facilitates the TS's ability to review and discuss various solutions in real-time. This interactive approach promotes immediate demonstration and clarification. However, as the course progresses, especially during later practical sessions and homework assignments, where students provide their solutions as backup files, accessing specific segments of students' code becomes increasingly complex. The existing process requires TS to go through a student's backup scripts when errors occur, which is inefficient. To address these challenges, the grading interface was recommended to include direct access to specific portions of each student's code. For instance, the instructor should be able to select a particular sub-task within an assignment and view the corresponding solutions for each student directly. This capability would significantly enhance the efficiency of code review, reducing the need for downloading DBs or locating solutions manually.

Furthermore, the implementation of an advanced solution comparison feature would be beneficial. This feature would allow the instructor to select a specific sub-task and automatically generate a side-by-side comparison of all student solutions for that task. Such a comparative display would support a more efficient and objective evaluation process, enabling instructors to identify differences in solution strategies and assess their correctness quickly. Additionally, the inclusion of a performance indicator next to each student's name, such as a percentage reflecting the proportion of tasks solved correctly, would aid in distinguishing between students who made minor errors and those whose solutions were predominantly incorrect. Currently, both categories are represented similarly (e.g., indicated in yellow), which dilutes meaningful distinctions.

Moreover, a group-level report describing all incorrect test cases for each student within the group would prove advantageous. This enhancement would mitigate the need for instructors to individually examine each student's submission to uncover common errors. An analytical overview of solution patterns would also provide significant insights. This

could encompass the frequency of specific solution strategies, the diversity of approaches employed, and the level of similarity among the solutions. Such an analysis would deepen understanding of student performance and their problem-solving methodologies. Finally, if submission logs could be managed through this interface, it would further streamline the grading process.

3.1.5 Reflection of the First Research Question

The AAT has significantly alleviated the grading burden on the TS, achieving a high level of satisfaction among TS. This corresponds to the principles of AA as discussed by Muuli et al. [11], who emphasized the importance of automating repetitive, time-consuming tasks in educational environments. Many TS members expressed that the AAT has decreased the time allocated to grading, particularly for assignments needing repetitive assessments. Prior studies have demonstrated that AA not only reduces the workload for TS but also improves feedback consistency and quality [12, 13]. For example, the grading time, which previously ranged from 40 to 100 minutes per task using manual methods, has now been reduced to approximately 20.4 minutes for homework and 17.8 minutes for lab assignments. Such improvements are consistent with the findings of Kakk [4], where a semi AAT significantly minimized time spent on repetitive grading tasks. This reduction in time, which aligns with Section 1.3, which emphasizes the necessity to save time, is linked to the automation of tasks such as the restoration of student DBs, execution of test cases, and structured feedback.

The AAT has also received praiseworthy ratings for understandability and trustworthiness, which supports the findings of Kleerekoper et al. [14], who highlighted the importance of feedback clarity and consistency in automated systems. These scores indicate that the feedback delivered by the system is largely clear and comprehensible, enabling TS members to rapidly assess the accuracy of student solutions. However, a few TS members noted instances where the feedback lacked clarity, particularly in relation to complex DB tasks or SQL errors. Such issues reflect the challenges identified by Queirós et al. [19], who noted that automated feedback systems often struggle with accurately diagnosing complex or context-sensitive issues. Challenges arose when automated feedback was unable to fully capture the detailed aspects of SQL syntax or errors related to table structure, occasionally necessitating manual intervention by TS members for clarification. Such instances show the limitations of AA systems when addressing complex or context-sensitive issues, as discussed in Section 1.4.

The feedback from TS members clearly demonstrates that the AAT has fulfilled its primary objective of reducing the workload for TS, which importance was mentioned in Sections 1.1, 1.3 and 1.5 while maintaining or even enhancing the quality of feedback provided to students. The system has effectively automated routine grading tasks, thereby allowing TS members to engage in more meaningful educational interactions. This

aligns with the foundational objectives of AA, which prioritize efficiency and consistent feedback delivery [7, 18]. The system has automated routine grading tasks, such as the restoration of student DBs, execution of test cases, and structured feedback. These outcomes also support the findings of Pack et al. [9], who demonstrated that automated assessment tools can significantly enhance the grading process without compromising the quality of feedback.

3.2 Student Questionare

This section provides an evaluation of the AAT Silmused, as perceived by students enrolled in the DB course at the University of Tartu. The aim of the student questionnaire was to gather feedback on how students assessed the clarity, accuracy, and usefulness of the feedback generated by the AAT. This evaluation seeks to address the second research question of this thesis: How do students appraise the clarity and accuracy of the feedback offered by the developed AAT?

3.2.1 Methodology

Students were asked about their experience with the AAT Silmused that is used in the Lahendus platform. The questionnaire was about the first three tasks they had to use the AAT to be assessed. The questionnaire asked both closed and open-ended questions from the students. Quantitative feedback was given as a rating from 1 to 5 on how clear the feedback from the AAT was and how clear the assessment made by the AAT was. Qualitative feedback asked the students to comment the same aspects plus there was a free form for submitting general thoughts in a quantitative manner, the form itself was not mandatory. In 2024 course, ratings for Lab 3, Lab 4 and Homework 3 were asked and 143 out of 336 students answered to the questionnaire shown in Appendix 5, the questionnaire also contained additional questions unrelated to the AAT, which are not included in the appendix due to copyright protection. In 2025 course, the Lab 3 and Lab 4 ratings were asked together and 209 out of 340 students answered to the questionnaire shown in Appendix 6, the questionnaire also contained additional questions unrelated to the AAT, which are not included in the appendix due to copyright protection. Descriptive statistics were employed, and in the open-ended questions, the most frequently mentioned responses were highlighted, along with the thoughts that the author considered most significant.

3.2.2 General Feedback

Students were asked to rate the assessment and feedback quality given by the AAT in 2024 and 2025 courses, aggregated results are shown in Tables 1 and 2.

In 2024, the results demonstrated a generally positive reception. For Lab 3, most students rated the feedback clarity favorably, with 60.1% assigning the highest rating (5) and 26.6% a rating of 4. Only a small proportion 4.9% gave the lowest rating (1). The assessment accuracy was also highly rated, with 66.4% of students awarding it a 5 and 21.7% a 4, while only 3.5% rated it as 1. For Lab 4, feedback clarity improved, with 69.2% of students assigning a rating of 5 and 19.6% a rating of 4. Assessment clarity was similarly well-received, with 67.8% giving it a 5 and 18.9% a 4. Homework 3 demonstrated the highest satisfaction levels, with 74.8% of students rating the feedback clarity as 5 and 74.13% rating the assessment clarity at the highest level. Negative ratings remained minimal across all tasks.

In 2025, the combined evaluation of Lab 3 and Lab 4 reflected continued positive reception. Feedback clarity was rated 5 by 69.9% of students and 4 by 16.8%. Assessment clarity was also well-received, with 67.5% of students assigning a rating of 5 and 20.1% a rating of 4. For Homework 3, feedback clarity reached a high level, with 71.3% of students awarding it a 5, while assessment clarity was rated 5 by 70.3% of respondents. Negative feedback was rare in both years.

Table 1. Feedback and Assessment Results for Tasks in 2024

Task	Type	5	4	3	2	1
Lab 3	Feedback	86 (60.1%)	38 (26.6%)	10 (7.0%)	2 (1.4%)	7 (4.9%)
Lab 3	Assessment	95 (66.4%)	31 (21.7%)	11 (7.7%)	1 (0.7%)	5 (3.5%)
Lab 4	Feedback	99 (69.2%)	28 (19.6%)	9 (6.3%)	1 (0.7%)	6 (4.2%)
Lab 4	Assessment	97 (67.8%)	27 (18.9%)	13 (9.1%)	0 (0%)	6 (4.2%)
HW 3	Feedback	107 (74.8%)	25 (17.5%)	8 (5.6%)	0 (0%)	3 (2.1%)
HW 3	Assessment	106 (74.13%)	25 (17.5%)	7 (4.9%)	2 (1.4%)	3 (2.1%)

Table 2. Feedback and Assessment Results for Tasks in 2025

Task	Type	5	4	3	2	1
Lab 3 and 4	Feedback	146 (69.9%)	35 (16.8%)	22 (10.5%)	6 (2.9%)	0 (0%)
Lab 3 and 4	Assessment	141 (67.5%)	42 (20.1%)	19 (9.1%)	2 (1.0%)	5 (2.4%)
HW 3	Feedback	149 (71.3%)	37 (17.7%)	20 (9.6%)	1 (0.5%)	2 (1.0%)
HW 3	Assessment	147 (70.3%)	40 (19.1%)	20 (9.6%)	0 (0%)	2 (1.0%)

General feedback was mostly positive, with many students wanting a similar solution in other courses. Most said that the feedback is helpful in correcting their mistakes. Some students pointed out that the tests should go a bit furthen even if everything is correct, the tests should show how to improve their solution even more (e.g. efficiency). One recommendation was to add a button "send to TS" to make it simpler to ask questions about an assessment. Some students complained that in some cases Lahendus gives an error: "psycopg2.OperationalError: connection to server at "localhost"(127.0.0.1), port 5433 failed: FATAL: the database system is starting up".

The feedback shows that the AAT effectively supported student learning by providing timely and clear feedback. The rise in positive ratings from 2024 to 2025 indicates that the improvements made to the AAT were well-received, increasing its usefulness and user satisfaction.

3.2.3 Feedback of Assessment and Points Given

One general point was made by students who usually get 100% that they do not get any feedback other than everything is correct. Some students had difficulty with long errors, which occur when SQL can not evaluated and system error is returned. Some students pointed out that the AAT only tells what is wrong or missing, but does not give further explanation why. Some students had an issue that the feedback does not clearly show which subtask has an error, but it checks the DB as a whole. Some students had an issue that when a table name of parameter name is not exactly correct, the assessment shows it as wrong e.g. using "õ" instead of "o". Some students mentioned that they forgot how to create a backup file. Some students complained about points not propagating to Moodle.

3.2.4 Reflection of the Second Research Question

The evaluation of the AAT Silmused from the student perspective directly addresses the research question: How do students assess the clarity and accuracy of the feedback provided by the developed AAT? An analysis of student feedback from 2024 and 2025 indicates a generally positive perception of the tool, while also highlighting specific areas that require improvement. Students generally appreciated the clarity, accuracy, and immediacy of the feedback provided by Silmused, which aligns with the principles of formative learning as emphasized by Taipalus and Perälä [5].

Quantitative feedback consistently indicated that a majority of students found the feedback delivered by the AAT to be clear and accurate. This aligns with previous studies showing that automated feedback can enhance student understanding and performance when it is timely and clearly presented [7, 18]. However, qualitative responses uncovered several recurring concerns. Some students noted difficulties in comprehending the

feedback when error messages were excessively lengthy or lacked sufficient clarity. For example, during 2025, students reported challenges in interpreting error messages due to their length or complexity, thereby complicating the identification of the specific problem. Similarly, in 2024, students expressed confusion when the feedback merely stated that a DB was incorrect without detailing the problematic section or the specific test that triggered the issue. Such issues are common for AATs, as highlighted by Pack et al. [9].

The feedback also revealed that students who attained perfect scores (100%) often did not engage with the feedback, as they had no errors to address. This lack of engagement suggests that the AAT primarily serves those encountering mistakes, an issue identified by Queirós et al. [19], who emphasized that even high-performing students benefit from feedback that offers optimization suggestions.

Another common issue noted was the occurrence of technical errors, which disrupted the assessment process for certain students. While such errors were relatively infrequent, they significantly affected the experiences of the impacted students, resulting in confusion and frustration. This is consistent with the findings presented by Kleerekoper et al. [14], who indicated that maintaining technical stability is crucial for maintaining trust in AATs.

Furthermore, some students expressed uncertainty when minor syntax variations, such as differing column names, led to assessment errors. For instance, a student in 2025 reported confusion when the tool indicated that a column named "tegevuskoht" was absent, despite having renamed it to "asukoht". This problem is consistent with the findings of Kleerekoper et al. [14], who emphasized the importance of designing AATs with enough flexibility to accommodate for minor syntax variations without penalizing students for acceptable alternative solutions. The findings also reflect the findings of Vittorini et al. [18], which suggests that overly strict systems can hinder student understanding and discourage creative problem-solving.

Despite these challenges, many students valued the immediacy of the feedback, which facilitated rapid identification and correction of mistakes. This aligns with the principles of formative learning, where timely feedback is essential for effective learning. This corresponds to Hanmore et al. [6] who stated that timely feedback is essential for effective learning. Students also appreciated that the AAT clearly delineated which aspects of their solutions were correct or incorrect, thereby supporting a better understanding of the material.

In conclusion, while the AAT Silmused effectively provided clear and accurate feedback for the majority of students, its utility could be further improved by addressing the identified issues. Potential enhancements could include simplifying error messages, offering clearer explanations for detected errors, ensuring robustness against technical failures,

and providing optimization suggestions for accurate solutions. Such improvements would render the tool more accessible and beneficial to a wider range of students.

Conclusion

Development of Silmused involved continuous improvement influenced by insights from the first attempt. The initial implementation faced scalability issues and lacked real-time feedback, prompting the need for a new solution that works on top of the Lahendus platform. This shift from a basic script-based approach to a robust, containerized solution improved assessment quality, provided real-time feedback, and saved time for the TS. Silmused's modular, object-oriented design enhanced efficiency in writing and maintaining tests and aligned with formative learning principles by providing immediate, clear, and consistent feedback to students.

This thesis outlined the design, implementation, and evaluation of the AAT Silmused, which was developed for the University of Tartu's DB course. The primary aim of the tool was to streamline the assessment process, thereby reducing the workload for TS and providing students with immediate, accurate feedback on their DB assignments. Silmused was seamlessly integrated into the Lahendus platform, facilitating straightforward submission and prompt feedback for students. The evaluation of the tool involved questionnaires directed at both TS and students, focusing on the clarity, accuracy, and usefulness of the provided feedback.

The results indicated that the AAT effectively achieved its objectives. TS reported a significant decrease in grading time, allowing them to dedicate more effort to pedagogical activities. Student feedback suggested that the AAT delivered clear and timely responses, which were essential to their learning process. Additionally, the tool's positive impact was underscored by a notable increase in student satisfaction between 2024 and 2025, reflecting continuous improvements within the system.

Silmused offers a practical solution for courses that rely on repetitive and standardized assessment tasks, particularly in technical fields such as DB management. Its automated feedback system facilitates formative learning, allowing students to promptly identify and rectify mistakes. The integration of Silmused into the Lahendus platform ensures scalability, making it well-suited for large courses with hundreds of students. Additionally, the tool's modular structure allows for easy adaptation to other similar subjects where AA of DBs is applicable.

Despite its effectiveness, Silmused has notable limitations. Firstly, the clarity of feedback can suffer when error messages are overly complex, making it challenging for students to comprehend the issues at hand. Secondly, the tool lacks flexibility in recognizing valid variations in student solutions, such as minor syntax differences in column names. Additionally, occasional technical errors, such as "psycopg2.OperationalError," have been reported, which can disrupt the assessment process. Furthermore, while the tool primarily aids students who make mistakes, those who achieve perfect scores receive little to no feedback for further enhancement. Finally, there are methodological limitations as-

sociated with the student questionnaire evaluation process. As not all students participate in answering the questionnaire, the results may be biased. Typically, the most diligent students are the ones who engage with the tool, leading to a potential positive bias in the overall feedback. As a result, the evaluation of the student questionnaire may appear overly optimistic and may not accurately reflect the experiences of all students.

Further improvements to Silmused have been made by Kakk, namely improving the feedback given and implementing feedback translations; adding the possibility to assess single queries rather than the whole DB. A PSQL log analyzer tool has also been produced by Kangro, which shows how many queries a student made and which were successful [37]. Future work could combine these two to give a more descriptive feedback about a student's progress to the TS. Furthermore as setting up and maintaining a DB has proven to be a challange to students, resulting in much time spent on fixing environment issues, the solution proposed earlier could be one that is hosted in a server, removing said difficulties and making it more easier to access a students progress as everything they have done can be monitored.

To further enhance the effectiveness of Silmused, several improvements are recommended. First, the feedback system could be improved by providing more detailed explanations for common errors and offering suggestions for optimizing correct solutions. This would benefit not only struggling students but also high-performing individuals seeking further enrichment. Additionally, simplifying error messages for better clarity would help users understand and correct their mistakes more efficiently.

The tool should also exhibit greater flexibility in recognizing minor syntax variations to accommodate diverse user inputs. On the technical side, enhancing stability by addressing known errors, such as connection issues, is crucial for ensuring consistent performance. Expanding the tool to include a comprehensive analysis of student performance trends would offer valuable insights for both TS and students, fostering a more effective learning environment. Additionally, Silmused could be enhanced by gathering more information from students and instructors through user interviews or other means. The feedback system could be enhanced with more detailed explanations for common errors and suggestions for optimizing correct solutions, making it beneficial even for high-performing students. Error messages should be simplified for better clarity, and the tool should be made more flexible in recognizing minor syntax variations. Technical stability could be improved by addressing known errors, such as connection issues, and ensuring consistent performance. Finally, the tool could be expanded to provide a detailed analysis of student performance trends, offering valuable insights for both TS and students.

References

[1] Pihel Hunt and Mari Karm. “Feedback is like a tail wind”: pre-service teachers’ and teacher educators’ perceptions of feedback. In *Estonian Journal of Education 10(1)*, pages 143–170, 2022.

[2] Muuli et al. Automation of assessment and feedback in it teaching from the teaching staff perspective. In *IEEE Frontiers in Education Conference, 2021*, pages 1–9, 2021.

[3] Trevor Barker. An automated individual feedback and marking system: An empirical study. In *Electronic Journal of e-Learning; 2011, Vol. 9 Issue 1*, pages 1–14, 2011.

[4] Martti Kakk. Automaatkontrollide arendus õppetöö tagasisidestamiseks aines andmebaasid, 2023.

[5] Toni Taipalus and Piia Perälä. “What to Expect and What to Focus on in SQL Query Teaching. In *SIGCSE ’19: Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 198–203, 2019.

[6] Hanmore et al. Is time really of the essence? timeliness of narrative feedback in ophthalmology cbme assessments. In *Medical Teacher Volume 46, 2024 - Issue 5*, pages 705–710, 2024.

[7] English et al. Experiences of using automated assessment in computer science courses. In *Journal of Information Technology Education: Innovations in Practice, Volume 14*, pages 237–254, 2015.

[8] Paiva et al. Automated assessment in computer science education: A state-of-the-art review. In *ACM Transactions on Computing Education (TOCE), Volume 22, Issue 3, Article No.: 34*, pages 1–40, 2022.

[9] Pack et al. Large language models and automated essay scoring of english language learner writing: Insights into validity and reliability. In *Computers and Education: Artificial Intelligence, Volume 6*, pages 1–9, 2024.

[10] Mark D. Shermis and Jill Burstein. Handbook of automated essay evaluation: Current applications and new directions (1st ed.)., 2013.

[11] Muuli et al. Automation of it faculty work from the teaching staff perspective. In *INTED2021 Proceedings*, pages 2839–2848, 2021.

[12] Kristiansen et al. Feedback on student programming assignments: Teaching assistants vs automated assessment tool. In *ACM Proceedings of the 22nd Koli Calling International Conference on Computing Education Research (Koli 2023)*, 2023.

[13] Lawrence Cloepass Chirwa. A case study on the impact of automated assessment in engineering mathematics. In *Engineering Education, Volume 3, Issue 1 (2008)*, pages 13–20, 2008.

[14] Kleerekoper et al. Automated assessment for databases units. In *CEP '24: Proceedings of the 8th Conference on Computing Education Practice*, pages 17–20, 2024.

[15] Kovačić et al. Automatic grading of spreadsheet and database skills. In *Journal of Information Technology Education Volume 11*, pages 53–70, 2012.

[16] Bretag et al. 'teach us how to do it properly!'. an australian academic integrity student survey. studies in higher education., 2013.

[17] Gigi Foster. Grading standards in higher education: Trends, context, and prognosis. In *Handbook of academic integrity*, pages 307–324, 2016.

[18] Vittorini et al. An ai-based system for formative and summative assessment in data science courses. In *International Journal of Artificial Intelligence in Education*, pages 159–185, 2020.

[19] Queirós et al. Petcha: a programming exercises teaching assistant. In *ITiCSE '12: Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, pages 192–197, 2012.

[20] Ellis Batten Page. The imminence of... grading essays by computer. In *The Phi Delta Kappan, Vol. 47, No. 5*, pages 238–243, 1966.

[21] Smith et al. Educational uses of the plato computer system. In *American Association for the Advancement of Science Vol. 192, No. 4237*, pages 344–352, 1976.

[22] Gobert et al. Intelligent tutoring systems: a history and an example of an its for science. In *International Encyclopedia of Education*, pages 460–470, 2022.

[23] Sembey et al. Intelligent tutoring systems: a history and an example of an its for science. In *Journal of Systems and Software, Volume 211*, 2024.

[24] Marten Siiber. Aine „andmebaasid“ automaattestimise vahend, 2015.

[25] Robert Sepp. Automaattestide loomine andmebaasile edu, 2018.

[26] Mikk Õunmaa. Automaattestide loomine sessioonõppe ainele „sissejuhatus andmebaasidesse“, 2019.

[27] Martti Kakk. Automaattestid andmebaaside ainele, 2020.

[28] GeeksforGeeks. Database testing – software testing. `https://www.geeksforgeeks.org/software-testing-database-testing` (9 Dec. 2024).

[29] Katalon. Database testing: A full guide. https://katalon.com/resources-center/blog/database-testing (9 Dec. 2024).

[30] Javatpoint. Database testing. https://www.javatpoint.com/database-testing (9 Dec. 2024).

[31] Natalia Juristo and Sira Vegas. Functional testing, structural testing, and code In *Lecture Notes in Computer Science volume 2765*, pages 208–232.

[32] Lahendus. https://lahendus.ut.ee (12 Dec. 2024).

[33] Github. Easy. https://github.com/kspar/easy (21 Apr. 2025).

[34] Karl-Aksel Puulmann. Python module for automatic testing of programming assignments, 2014.

[35] Github. Python grader. https://github.com/kspar/python-grader (21 Apr. 2025).

[36] YouTrack. Lahendus api communication protocol. https://easy.myjetbrains.com/youtrack/issue/EZ-1509/Design-executor-core-communication-protocol (30 Apr. 2025).

[37] Külli Kangro. Veebirakendus päringuvigade analüüsimiseks postgresql-i logide abil kursusel „andmebaasid“, 2024.

Appendix

1. Source Code of the First Attempt

https://github.com/kare22/database_course_automatic_tests

2. Source Code of Original Contribution

https://github.com/kare22/silmused/tree/PAAN_ORIGINAL

3. Source Code

https://github.com/kare22/silmused

4. Teaching Staff Questionnaire

2025 Automaatkontrolli tagasiside

Tere!

Antud küsitlus on mõeldud Andmebaasid (LTAT.03.004) õppejõududele ja õppeassistentidele, et hinnata uue automaatkontrolli (mis töötab Lahendus keskkonnas) toimimist ja kasulikkust.

Saadud vastuseid kasutatakse Karel Paan magistritöös ning automaatkontrolli edasises arenduses.

NB! Vastamisel palun lähtuda 3. praktikumist ja 3. kodutööst alates ehk need kontrollid, mis kontrollivad kogu andmebaasi.

* Indicates required question

Nimi

NB! Kasutatakse ainult selleks, et näha, kes on vastanud. Antud vastuseid ja tagasisidet ei seostata neid andnud isikuga.

Your answer

Kui rahul oled **automaatkontrolliga**? *

	1	2	3	4	5	
Vajab suurt parandamist	○	○	○	○	○	Igati rahul

Keskmiselt mitu minutit kulutad **kodutööde** hindamisele? (ühe rühma kõikide kodutööde peale kokku) *

Välja arvatud logi failide kontrollimine.

Choose

Keskmiselt mitu minutit kulutad **praktikumi** ülesannete hindamisele? (ühe rühma kõikide praktikumi ülesannete peale kokku) *

Välja arvatud logi failide kontrollimine.

Choose

Kui **usaldadusväärne** on automaatkontrolli poolt antud hinnang? *

	1	2	3	4	5	
Punkte arvestatakse ilma asjata maha	○	○	○	○	○	Hindab ausalt

Kas oled pidanud automaatkontrolli poolt antud hinnangut korrigeerima? *

- ○ JAH
- ○ Ei

Kui JAH, too palun näide...

Your answer

Kui **arusaadav** on automaatkontrolli poolt antud tagasiside? *

	1	2	3	4	5	
Jääb segaseks, miks punkte on maha arvestatud	○	○	○	○	○	Saan alati aru, mille eest on punkte maha arvestatud

Kas oled pidanud tudengi(te)le automaatkontrolli poolt antavat tagasiside täpsustama? *

- ◯ JAH
- ◯ Ei

Kui JAH, too palun näide...

Your answer

Kui tihti kontrollid automaatkontrolli poolt antud tagasiside/hinnangut? *

- ☐ Iga kord
- ☐ Vaatan ainult neid töid, millel pole maksimum punktid
- ☐ Vaatan, kui tudengil on probleem
- ☐ Vaatan pisteleliselt
- ☐ Usaldan automaatkontrolli täielikult
- ☐ Other:

Milliseid probleeme veel on automaatkontrolliga esinenud?

Your answer

Kuidas saaks automaatkontroll olla kasuks Sinu töö veelgi paremaks tegemisel?

Your answer

Kuidas saaks automaatkontroll tudengitele kasulikum olla?

Your answer

Muu

Kõik mõtted, tagasiside ja **ettepanekud** on oodatud!

Your answer

6. Student Questionnaire 2024

Erinevate vahendite kasutamine kursusel "Andmebaasid"

Hea õppija!

Oleme kursust sellisel kujul õpetanud kolmandat aasta ja pidevalt arendame seda edasi püüdes kiirendada tagasisidet ja aidata materjalidega. Teadmaks, kui kasulikud on meie loodud materjalid (kodulugemised, loenguvideod, rühmatöö juhend) ja vahendid (murelahendajad, automaatkontroll lahenduses) on ning mida peaksime muutma, et need veel kasulikumad oleks, on teie tagasiside meile väga väärtuslik. Siis saame juba teie hinnangute alusel järgmiste selle kursuse teemade juures seda kõike arvesse võtta. Samuti huvitab meid, kuivõrd on õppimist ja vahendite kasutamist muutnud AI.

Vastamine on anonüümne kuigi küsime ka veidi taustaandmeid ja boonuspunkti jaoks nime, siis analüüsi tulemused esitatakse vaid üldistatuna ning Teie nimi koheselt kustutatakse kui boonus on hindetabelisse lisatud. Teie vastused on meile väga olulised, et edaspidi kursusel kasutatavaid vahendeid paremaks muuta. Küsitluse täitmine võtab aega umbes 10 minutit.

Ette tänades.

Piret Luik, Martti Kakk ja Karel Paan

* Indicates required question

Automaatkontrollide tagasiside

1. Kui selgesti oled aru saanud Sinule antud 3. praktikumi tagasisidest, mis on automaatkontrolliga kontrollitud? *

	1	2	3	4	5	
Ei saa üldse aru	○	○	○	○	○	Saan väga hästi aru

2. Mis on Sinu jaoks jäänud 3. praktikumi tagasisidest puudu või tekitanud segadust?

Your answer

3. Kui selgesti saad aru Sulle antud 3. praktikumi tagasisides saadud/kaotatud punktidest? *

	1	2	3	4	5	
Ei saa üldse aru	○	○	○	○	○	Saan väga hästi aru

4. Mis on Sulle tekitanud segadust 3. praktikumi punktide saamises?

Your answer

5. Kui selgesti oled aru saanud Sinule antud 4. praktikumi tagasisidest, mis on automaatkontrolliga kontrollitud? *

	1	2	3	4	5	
Ei saa üldse aru	○	○	○	○	○	Saan väga hästi aru

6. Mis on Sinu jaoks jäänud 4. praktikumi tagasisidest puudu või tekitanud segadust?

Your answer

7. Kui selgesti saad aru Sulle antud 4. praktikumi tagasisides saadud/kaotatud punktidest? *

	1	2	3	4	5	
Ei saa üldse aru	○	○	○	○	○	Saan väga hästi aru

8. Mis on Sulle tekitanud segadust 4. praktikumi punktide saamises?

Your answer

9. Kui selgesti oled aru saanud Sinule antud 3. kodutöö tagasisidest? *

	1	2	3	4	5	
Ei saa üldse aru	○	○	○	○	○	Saan väga hästi aru

10. Mis on Sinu jaoks jäänud 3. kodutöö tagasisidest puudu või tekitanud segadust?

Your answer

11. Kui selgesti saad aru Sulle antud 3. kodutöö tagasisides saadud/kaotatud punktidest? *

	1	2	3	4	5	
Ei saa üldse aru	○	○	○	○	○	Saan väga hästi aru

12. Mis on Sulle tekitanud segadust 3. kodutöö punktide saamises?

Your answer

13. Kas soovid midagi automaatkontrollide kohta lisada?

Your answer

7. Student Questionnaire 2025

Erinevate vahendite kasutamine kursusel "Andmebaasid"

Hea õppija!

Oleme kursust pidevalt edasi arendanud püüdes kiirendada tagasisidet ja aidata materjalidega. Teadmaks, kui kasulikud on meie loodud materjalid (kodulugemised, loenguvideod, rühmatöö juhend) ja vahendid (murelahendajad, automaatkontroll lahenduses) on ning mida peaksime muutma, et need veel kasulikumad oleks, on teie tagasiside meile väga väärtuslik. Siis saame juba teie hinnangute alusel järgmiste selle kursuse teemade juures seda kõike arvesse võtta. Samuti huvitab meid, kuivõrd on õppimist ja vahendite kasutamist muutnud AI.

Vastamine on anonüümne kuigi küsime ka veidi taustaandmeid ja boonuspunkti jaoks nime, siis analüüsi tulemused esitatakse vaid üldistatuna ning Teie nimi koheselt kustutatakse kui boonus on hindetabelisse lisatud. Teie vastused on meile väga olulised, et edaspidi kursusel kasutatavaid vahendeid paremaks muuta. Küsitluse täitmine võtab aega umbes 10 minutit.

Ette tänades.

Piret Luik, Martti Kakk ja Karel Paan

Next Clear form

Automaatkontrollide tagasiside

1. Kui selgesti oled aru saanud Sinule antud 3. ja 4. praktikumide tagasisidest, mis on automaatkontrolliga Lahendus keskkonnas kontrollitud? *

	1	2	3	4	5	
Ei saa üldse aru	○	○	○	○	○	Saan väga hästi aru

2. Mis on Sinu jaoks jäänud 3. või 4. praktikumi tagasisidest puudu või tekitanud segadust?

Your answer

3. Kui selgesti saad aru Sulle antud 3. või 4. praktikumi tagasisides saadud/kaotatud punktidest? *

	1	2	3	4	5	
Ei saa üldse aru	○	○	○	○	○	Saan väga hästi aru

9. Kui selgesti oled aru saanud Sinule antud päringuülesannete (kodutööd 1 ja 2 ning kodutöö 3 alamülesanded 7-8) tagasisidest? *

	1	2	3	4	5	
Ei saa üldse aru	○	○	○	○	○	Saan väga hästi aru

10. Mis on Sinu jaoks jäänud päringuülesannete tagasisidest puudu või tekitanud segadust?

Your answer

11. Kui selgesti saad aru Sulle antud päringulausete (kodutööd 1 ja 2 ning kodutöö 3 alamülesanded 7-8) tagasisides saadud/kaotatud punktidest? *

	1	2	3	4	5	
Ei saa üldse aru	○	○	○	○	○	Saan väga hästi aru

12. Mis on Sulle tekitanud segadust päringulausete (kodutööd 1 ja 2 ning kodutöö 3 alamülesanded 7-8) punktide saamises?

Your answer

9. Kui selgesti oled aru saanud Sinule antud 3. kodutöö 1-6 alamülesande tagasisidest? *

	1	2	3	4	5	
Ei saa üldse aru	○	○	○	○	○	Saan väga hästi aru

10. Mis on Sinu jaoks jäänud 3. kodutöö 1-6 alamülesande tagasisidest puudu või tekitanud segadust?

Your answer

11. Kui selgesti saad aru Sulle antud 3. kodutöö 1-6 alamülesande tagasisides saadud/kaotatud punktidest? *

	1	2	3	4	5	
Ei saa üldse aru	○	○	○	○	○	Saan väga hästi aru

12. Mis on Sulle tekitanud segadust 3. kodutöö 1-6 alamülesande punktide saamises?

Your answer

13. Kas soovid midagi automaatkontrollide kohta lisada?

Your answer

Back Next Clear form

7. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Karel Paan**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Development of an Automated Assessment Tool "Silmused" in the Databases course,

   supervised by Piret Luik.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Karel Paan
15/05/2025