Costa et al. *Research and Practice in Technology Enhanced Learning* (2023) 18:30
https://doi.org/10.58459/rptel.2023.18030

Research and Practice in
Technology Enhanced Learning

RESEARCH Free and Open Access

Theoretical and practical assessments over SSH

Joana Cabral Costa *, Tiago Roxo, Carolina Lopes, João B. F. Sequeiros, Tiago M. C. Simões and Pedro R. M. Inácio

*Correspondence:
joana.cabral.costa@ubi.pt
Instituto de Telecomunicações, Universidade da Beira Interior, Rua Marquês de Ávila e Bolama, Covilhã, Castelo Branco, Portugal
Full list of author information is available at the end of the article

Abstract

During the COVID-19 pandemic, universities worldwide were forced to close, causing a shift from presential to remote classes. This situation motivated teachers to find suitable tools to evaluate students remotely, fairly, and accurately. However, currently available systems are either survey or exercise evaluation based, not suitable for competency-based assessments. Faced with this context and limitations of available evaluation systems, we developed `TestsOverSSH`, a system to devise, deliver, and automatically correct assessments performed in a Command Line Interface (CLI) environment. Unique assessments are generated per student when they access the proposed system via Secure SHell (SSH). `TestsOverSSH` is composed of shell scripts that orchestrate a series of tools and services that come pre-installed in Linux distributions. It can be used to construct multiple-choice or direct answer questions while also requiring students to perform tasks in the environment per se, namely computer programming or CLI manipulation-related assignments. We present examples of the question types in this system, explaining question formats and operating guidelines. Since the assessments are directly performed in the system, logs and command history can be easily retrieved while keeping information within student devices uncollected. We performed evaluations using this system in a real context and obtained student feedback through a custom survey and the System Usability Scale (SUS). Survey results and SUS score suggest that `TestsOverSSH` is an intuitive evaluation tool, with eased access and usage, making it applicable for e-learning.

Keywords: Automatic assessment tools, Command line interface (CLI), Competency-based assessments, COVID-19, Learning technologies, Remote assessments

Introduction

In March of 2020, COVID-19, a World Health Organization (WHO) pandemic-declared disease, forced multiple countries to announce a nationwide lockdown to flatten the disease dissemination. This situation led to the closure of universities (Nicola et al., 2020), imposing traditional face-to-face methodologies to be switched to online methods (Daniel, 2020). Virtual meetings applications, such as Teams or Zoom, were used to support virtual

classrooms, providing an admissible teaching approach. However, evaluating at a distance was still a challenge. Lack of methods to evaluate students remotely while granting academic integrity prompted broad discussions (Huang et al., 2020; Reimers & Schleicher, 2020) about assessment technologies, stimulating the development of suitable tools to evaluate students remotely (Özyurt & Özyurt, 2015). Furthermore, after this pandemic is over, it is expected that these technologies will be applied and incorporated in future assessment methods (Sahu, 2020).

Aside from the necessity of remote evaluation, there is a shortage of tools designed to evaluate Computer Science students, especially their technical skills. The most common approach for knowledge evaluation is based on written tests. Nevertheless, it is essential to prepare students for real-world practices in this field (Ferreira et al., 2018), given the importance of Computer Science graduates problem analysis skills for companies (Dukhanov et al., 2014). For this reason, it is necessary to define an evaluation that uses a theoretical approach while also requiring competency-based assessments (Poth et al., 2020). An essential skill for Computer Science students is Command Line Interface (CLI) usage, which gives access to powerful functionalities such as operating and file system control, program compilation and execution, among others. Given this context, we focused on the following research questions:

- Is it possible to create a system to assess the practical component of Computer Science students, in particular CLI-based knowledge, in a viable and automatic manner?
- Can we create an assessment tool of CLI-based knowledge without requiring the installation of specific or custom-made software?
- Is it possible to create a remote assessment tool for CLI-based exercises that students find usable and practical?

Based on the aforementioned pandemic context, and the shortage of evaluation tools that also evaluate competency-based assessments, in particular CLI skills, we propose a system capable of achieving such goals, admissible for remote or local evaluations. The system will herein be referred to as `TestsOverSSH`, since it enables theoretical and practical assessments over a Secure SHell (SSH) connection. It can be set up in a Linux system, using only open-source software, which eases its installation and use for students. `TestsOverSSH` is prepared to assess students CLI-based knowledge through tasks and exercises, focusing on multiple-choice and direct answers to automatically correct questions while also allowing variability between questions. The usability of our system was assessed through student feedback in two different student groups using a custom survey and the System Usability Scale (SUS) (Brooke, 1996). Our custom survey showed that no significant difficulties were encountered while assessing, performing, and submitting their exam in `TestsOverSSH`, and a score of 70.99 on SUS was obtained, which

indicates that users perceive this tool as having good usability, with the need for minor improvements to the design (Brooke, 2013).

The remaining of this paper is structured as follows: the "Related work" section exhibits a set of tools that are related to the proposed system; the "Proposed system" section describes the proposed system, discussing its main characteristics, implementation, monitoring functionality, and operating guidelines; the "Questions composition" section briefly discusses the overall question format used in the system, providing an illustrative example; the "Application in real context" section summarizes the survey and SUS responses based on the system application in a real context; main conclusions and future work are presented in the "Conclusion" section.

Related work

Assessment tools

Online assessment tools allow for a constant and dynamic evaluation of students' knowledge. There are several learning management systems that promote an engaging learning environment, with personalized courses and constant connection to students. Guides to use said systems, such as Canvas LMS (John, 2021) or Blackboard Learn (Patterson, 2013) are useful for teachers to prepare classes using remote assessment tools. Another tool is EduZinc (Becerra-Alonso et al., 2020), which enables the creation of individualized learning assessments and automatic grading, applicable to any course. It supports advanced material creation for those with limited or no coding experience and provides competency-based feedback. Students can receive customized exercises, while instructors can assess exercise solutions and daily reports after each class. This tool can also be used as a summative assessment since it registers student performance throughout the course.

Effective learning can be achieved through the stimulation of student creativity. Support for Creativity in a Learning Environment (SCALE) (Richardson & Mishra, 2018) is a framework that focuses on this area and helps teachers and administrators identify ways of teaching to support student creativity. This framework is divided into three categories: the Physical Environment, which evaluates the furniture and spaces where students learn; the Learning Climate, which considers the classroom relationships and how it stimulates communication; and the Learner Engagement, which evaluates how the performed tasks stimulate creativity. Another topic linked to effective learning is the use of gamification in teaching approaches. A tool that brings gamification to the classroom is VRLearner (Sun et al., 2019), which uses Virtual Reality (VR) to teach and evaluate students. It only requires a smartphone and VR cardboard, making it easy to use, and participants only need to gaze to navigate and interact in this VR environment. This tool has 15 pop-up quiz

questions, multiple-choice ones, randomly selected according to the level of competency of the learner.

For additional information on the topic of blended learning tools, Kumar et al. (2021) review the existing tools for e-learning, providing a comparative analysis between them.

Assessment tools for computer science

One of the key aspects of Computer Science relates to code quality evaluation, with several assessment tools being reported in the literature. CodeGrade (CodeGrade, 2021) is a grading environment for coding assignments with automated grading, able to be integrated into other learning management systems such as Moodle, Canvas, and Blackboard. It provides instant and continuous feedback on code submissions, contributing to student code quality improvement. Another reported tool is DomJudge (Eldering et al., 2021), which is an online judge that allows programming problem submission and automatically evaluates them regarding, specifically, syntactic rightness, semantic correctness, and efficiency. For each programming problem, the platform provides details that help the student implement a solution, providing feedback in real-time, while also describing different test cases, with inputs and respective expected outputs, value ranges, and maximum run time (Restrepo-Calle et al., 2019). Suzuki et al. (2019) proposed a software system that allows teachers to create exams for the data structure course unit by generating image-based questions. This tool is mainly based on the topic of binary trees and can generate sample exams automatically from a randomly generated numerical data set, creating questions and respective options, and verifying them.

Aside from code assessment, there are also other topics that relate to Computer Science students evaluation. An important one is data request and manipulation from a database using Structured Query Language (SQL). In this area, the SQL Tester (Kleerekoper & Schofield, 2018) is an online assessment tool comprised of test cases with questions from different categories on this topic. The involvement of students with this tool leads to over 90% of them agreeing to actively engage in practice tests to get good marks. Another tool related to SQL assessment is SQLValidator (Obionwu et al., 2021), a web-based interactive tool. Usable during classes to enable a more dynamic teaching method and to administer exams and practice tests, this tool provides constant feedback to the students, translating into an effective way to improve students learning experience. Another topic related to Computer Science is Cybersecurity. One of the various forms of integrative knowledge assessment is through Capture the Flag (CTF) games, used to perform team-based competitions containing a spectrum of cybersecurity challenges. In this topic, Chicone et al. (2018) proposed using the Facebook CTF platform as a learning tool for undergraduate and graduate cybersecurity students. Challenges regarding vulnerabilities of devices can be integrated into this platform, and students earn points for exploiting them.

Proposed system

The high-level architecture of the proposed system is illustrated in Figure 1. It consists of a CLI-based system where students can submit their answers, and it automatically corrects them. The capacity of this platform for evaluating student CLI skills functions as a competency-based assessment, preparing students for real-world practices and supporting their lifelong development (Poth et al., 2020).

To access this system and take the exam, each student needs to connect to it via SSH, a secure and reliable cryptographic network protocol. Each student has a unique test environment, inside the system, which assures that he/she can only perform his/her test and not one of another student. Additionally, to access the test environment, the student must insert a uniquely generated code provided by the instructor moments before the assessment. This notion, associated with the SSH connection, gives us the best possible assurance that authenticity is a security property present in this system. During the exam taking, the system stores their commands, allowing fraudulent activity tracking and, at the exam submission, it also stores the students answers and evaluates them.

Throughout this paper, we will refer to the person managing the system as the administrator. This role can be taken by the teacher or other designated person.

Main characteristics

The developed tool comprises a Linux environment and several shell scripts that orchestrate its functioning, available as free and open-source software [1]. It is suitable to evaluate Computer Science students, with emphasis on its practical component, in particular CLI-related knowledge, via direct answer and multiple-choice questions. These

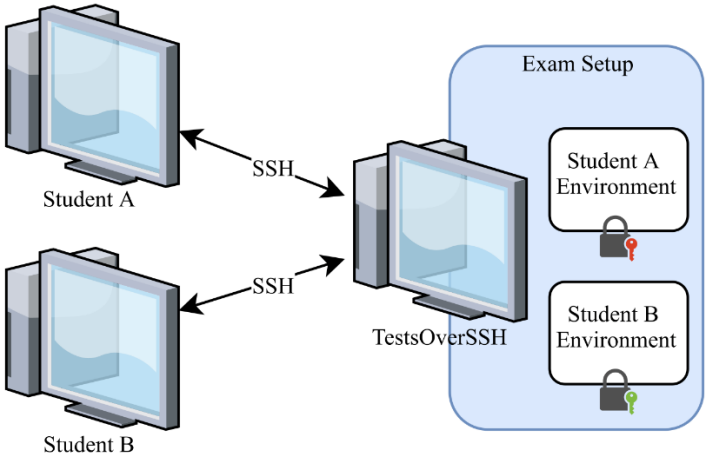

Fig. 1 High-level system architecture. Students connect via SSH to TestsOverSSH, each accessing their own environment, where they can perform their exam. Access is achieved using a student's unique code, depicted by a padlock

types of questions allow for a wide variability of generated tests and are automatically corrected. Furthermore, the installation of specific/custom-made software on the students' computer is not required to use the proposed system.

The distinctive characteristics and main contributions of the system to the field of learning technologies are the following:

- It constitutes a system to conduct both theoretical and practical assignments in the Computer Science area;
- It can be set up in a Linux system, using only open-source software;
- System access is done via a trusted SSH client, not requiring the installation of any custom-made program;
- Since the questions or tasks can be specified using bash commands, the system provides the basis for both simple and complex assessment scenarios;
- It allows for automatic correction of multiple-choice and direct answer questions;
- Since all the evaluation-related tasks are performed in the `TestsOverSSH` operating system, no information within student devices can be collected, ensuring data privacy compliance.

The main drawbacks of the system are the following:

- It is a specialized system that might need some knowledge in shell scripting to achieve its full potential;
- Question and task preparation can be an onerous chore for the administrator.

Implementation details

`TestsOverSSH` directories content is organized based on its functionalities. This section will explore directories and scripts with an essential role in the system functioning.

Under the `Admin` directory, there are several script files and one Comma-Separated Values (CSV) file. The CSV file is responsible for holding student names and respective institution numbers until the administrator manually changes them. All the other scripts of the directory mentioned above relate to manipulating the database, which means the administrator does not need any Data Manipulation Language (DML) knowledge to interact with the database. SQLite3 is the Structured Query Language (SQL) database engine used in this system, and the database is created by running `create-database`. To insert students and reset their test, during assessments, the administrator can take advantage of `insert-student` and `reset-student` scripts, respectively. There are also some information displaying scripts, which exhibit student scores and unique codes.

Assessment necessary information, such as maximum score, start time, and duration, can be viewed and changed in the `config` file, under the `Misc` directory. For text styling (specifically, bold and underline), the American National Standards Institute (ANSI)

escape sequences are provided in file `format`, under variable names, such as `BOLD` and `UNDERLINE`.

One of the most important scripts is the student environment setup, named `run-me`. This script will produce another one that the students will interact with when they connect to the system. It begins by prompting the student with the unique code given by the administrator. If the student inserted code matches the one in the database, he/she can enter the system. Otherwise, the SSH session is automatically closed. To log in again, the student must start another SSH connection. If the student successfully enters the system, a Unix system user and the respective directory will be created, whose name is the student institution number.

The instructions to back up the user environment and delete the user are added to a script, providing an automatic backup and cleanup. From the student viewpoint, the assessment preparation begins with the creation of each question or task individually. This process is possible due to previously prepared questions provided to `TestsOverSSH` by the administrator. For the questions to have the correct display, the system uses the question files content, stored in `Staging` directory, and presentation templates, stored in `Templates` directory. The student environment setup finishes by preparing introduction and submission scripts: the first presents the instructions to take the test, and the latter allows the assessment to be submitted into the system. Figure 2 illustrates a schematic view of the main steps described previously.

All the steps done by the student environment setup script are registered into a log file. This feature allows the administrator to monitor students' login, login failures, test submissions, among other aspects. While creating the scripts, permissions are changed, enabling a more straightforward student test-taking task. All the scripts accessed by

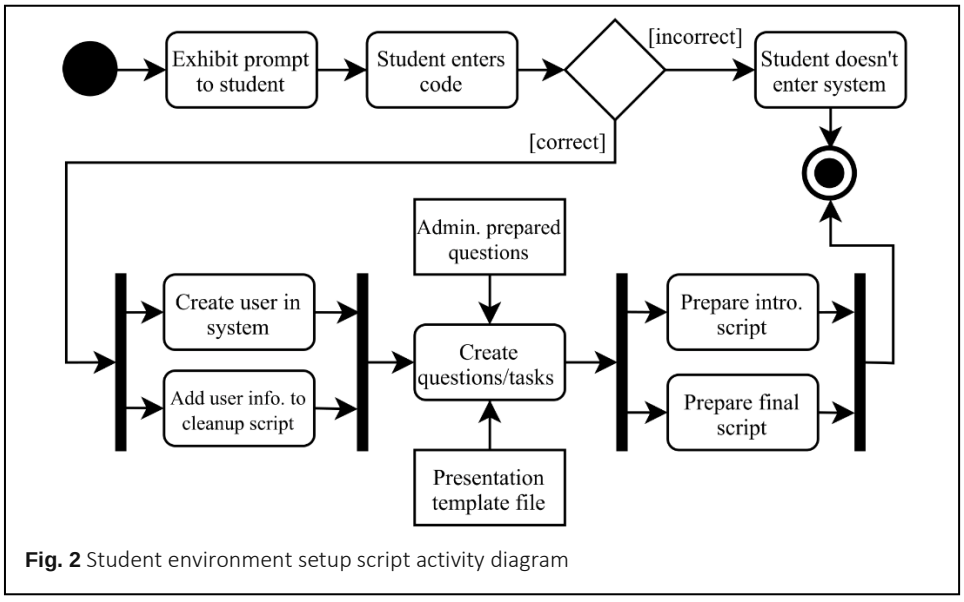

Fig. 2 Student environment setup script activity diagram

students are encrypted and converted into executable files, unreadable by humans in reasonable time.

Questions and tasks are examples of the previously mentioned executable files that students can access. When a student executes one of them, a question is presented, allowing them to answer it. Questions answered have their name changed to end with `.DONE`, which can be revisited and re-answered.

The environment setup script is the main script responsible for providing the correct environment variables and preparing other scripts. This script also creates a Unix system user with the default name and password `guest`, which will be used by the students to establish the SSH connection. Additionally, it changes the `guest` user profile to automatically execute the `run-me` script when a SSH connection is established.

This tool provides the administrator with a monitoring script that continuously reads the log file and displays all the events occurring in the `TestsOverSSH` system; this feature is further explained in the next subsection. Backup and cleanup tasks are eased due to the `cleanup` script. This script executes the `final-clean-up` script, modified each time a user is created in the Unix system to automatically clean the user and its generated files. Moreover, the `cleanup` script does the removal of the user `guest` and log file.

The proposed system has two additional scripts, `create-ap` and `destroy-ap`, whose use is applicable in conditions where an Internet connection is not necessary. The first one allows access point creation, which disables Internet connection, backs up the system network profiles and creates an access point. The latter removes the access point previously created by eliminating all the scripts used to create the access point and restoring all the previously saved network profiles.

Monitoring

The `monitor` script allows the administrator to monitor student entries, exhibiting their Internet Protocol (IP) address on connection. If any connection problem occurs, this script displays a warning message, with the students identified through their unique codes. If the student successfully connects to the system, an informative message is displayed, indicating that a test was generated for the connected student. **Listing *1*** displays the CLI output of both unsuccessful and successful connections.

Listing 1 CLI output of warning and informative message display on unsuccessful and successful connections, respectively

```
 * [WARN] 10Dec2020-12:18 : (169.254.174.105 45142 22)
      User NOT able to connect with code E87AAFD6.
 * [INFO] 10Dec2020-12:19 : (169.254.174.105 45158 22)
      Generated test and user for s12589.
```

Monitoring student submissions is also possible through the `monitor` script execution. When a student submits its evaluation, the system exhibits their institutional number and name, the number of answered questions, the penalty criteria, and their final score. When all the students submit their assessments, the administrator can run the `cleanup` script, leaving the system ready for new assessments. If so, an informative message will be displayed in CLI. Submission and system cleaning output examples are exhibited in **Listing *2***.

Listing 2 CLI output of submission and system cleaning operations

```
 * [INFO] 10Dec2020-13:20 : Submission from s12589 (Pedro)!
 * [INFO] 10Dec2020-13:20 : #Questions: 3; Score: 6; Penalty: 0.
 ---
 * [INFO] 10Dec2020-13:50 : Cleaning and wrapping up.
```

When students access the system, all the commands used by them are stored in logs and available for current and future monitoring. The information stored is used to identify students having connection issues but is also important in fraudulent activity detection. Below is information stored by the logs to aid in this aspect:

- IP address when connecting to `TestsOverSSH`: allows the tracking of people performing an exam from the same location;
- Entrance and exit entries: helps identify connection issues;
- Commands used by students: allows the tracking of steps done by the student to answer questions, which can be used to identify fraudulent activity (e.g., a student knows the answer without executing the necessary steps for it);
- Exit attempts: allows the tracking of commands used by the student to "exit" its environment to corrupt the system or spy on other students environments (i.e., copy the answers of other students);
- Timestamps at the start and end of the exam: helps understand if the student did an exam under an estimated time, which may indicate suspicious activity.

The information presented above in logs requires human intervention and cross-examination with the student exam scores to assess the possibility of fraudulent activity.

Administrator procedures

Initially, the administrator should change the CSV file containing student information to the current year class. Then, and before any assessment, a database should be created to reset its contents, avoiding data sharing from different assessments.

Before a test, the main configuration file should be changed by adjusting the maximum score, initiation hour, and duration time to the current assessment. Afterwards, the main

script, responsible for creating the student interface, should be run. If necessary, the administrator can use the access point functionality by executing the respective script.

While the assessment is taking place, the administrator can monitor the `TestsOverSSH` submissions and connections. After finishing the submissions, if the administrator created an access point, it should be shut down by running the corresponding script. Finally, it should make a backup and remove all the students environments.

Student perspective

Firstly, students must execute `ssh guest@`*`system-ip`* and introduce the user password, provided by the administrator. By default, this password is `guest`. Then, the student can submit his/her unique code, given by the administrator, and the assessment preparation begins.

The student may begin to answer the presented questions by running the correspondent script. For example, the students can visualize the first question by executing `./01-MC.sh`. This example exhibits a multiple-choice question, translated by the `MC` presence in the script name. If the student has submitted all his answers, he can submit the test by running the script designed for that effect.

Questions composition

In this system, there are only two types of questions: multiple-choice and direct answer. Both kinds have the question text divided into three different sections: pre-question, question, and post-question. Neither of the sections is character limited. The question file can contain a set of commands used to prepare the question. **Listing *3*** exhibits an example of a set of commands used; said set is in the variable `COMMANDS`.

Listing 3 Portion of a question script exhibiting variables necessary for its preparation

```
COMMANDS=(
  'OPTION=$((RANDOM%4))'
  ...
  'echo -e "  % My name is Luka\n" >> projeto.tex'
  ...
  'case "$OPTION" in
    0) echo -e "  # I live on the second floor\n" >> projeto.tex
        echo -e "  % I live upstairs from you\n" >> projeto.tex
        echo -e "    Yes I think you ve seen me before\n" >> projeto.tex
    ;;
    ...
    esac'
    ...)
...
SHUFFLE=0 # Can options be randomly shuffled?
...
PENALTY=-1 # Sets the option of having a penalty
```

The system expects students to insert text as a response to the direct answer questions. The administrator can provide a label that shows up right before the answering location. One example of the use of the said label is displayed in **Listing *4*** with the variable `LABELS_COMMANDS`. This label aims to aid the students answering, indicating that the expected answer is a single word.

Listing 4 Excerpt of a direct answer question configuration file

```
...
# COMMANDS for preparing the labels to queries
LABELS_COMMANDS=( 'echo -n "The used cypher mode was (one word): "' )

# COMMANDS for CLEANING UP THE AUXILIARY FILES
CLEAN_COMMANDS=( 'rm solution.txt' )

# How many words should be matched in total for the answer to be correct.
NR_OF_MATCHES=1

CASE_SENSITIVE=0 # Case sensitive

CHECK_WORDS=1    # Check each word
```

The teacher can also state if the response is case sensitive, if the system must verify each word individually, and the number of words that need to be similar for the answer to be considered correct. The variables `CASE_SENSITIVE`, `CHECK_WORDS`, and `NR_OF_MATCHES`, in **Listing *4***, are responsible for each of the previously mentioned features, respectively. Furthermore, the question script can also contain a variable to define the admissible correct answers and another one indicating the question-related files the system must eliminate; this last feature is presented on the variable `CLEAN_COMMANDS`, in the **Listing *4***.

For multiple-choice questions, the student must select one or more options to answer the question. There is the possibility of choosing which option has a penalty associated and if the system can shuffle the options. The penalty option is associated with the variable `PENALTY` and shuffling linked to the variable `SHUFFLE`. In the example presented in **Listing *3***, the multiple-choice answer did not have its options shuffled, and there was no penalizing option.

Some considerations apply to all questions and tasks. The question/task file names should follow a specific terminology, where the first two characters are dedicated to the question number, and the third and fourth characters identify the type of question. The categories available are `DA` and `MC`, which stand for direct answer and multiple-choice, respectively. The administrator can style the text with bold and underline using the variables `BOLD` and `UNDERLINE`, respectively. It is necessary to establish where the text should return to its regular style, achieved with the `NONE` variable usage. **Listing *5*** displays the underline

feature; the expected output, in the CLI, should be: “Which of the following hash related statements is not true?”.

Listing 5 Underline usage example in question text

```
QUESTION_TEXT_COMMANDS=(
  'echo -n "Which of the following hash related statements
  is $UNDERLINEnot$NONE true?"'
)
```

A task is a question that needs an additional effort from the student to respond to it correctly. That said, a task still needs to be evaluated through a direct answer or multiple-choice question to prove its accomplishment. When the exercise is a task, there is always an additional directory containing all the necessary files to finish it successfully. In this type of question, the students need to execute a set of commands and analyze their results to answer correctly, thus evaluating their technical ability.

To better understand question-related concepts, we refer to a task example, whose script is partially presented in **Listing *3***. This task requires students to open a file, named “project.tex”, analyze it, and predict its output, assuming the LaTeX file is compiled to Portable Document Format (PDF). Due to the way the administrator built the task, the “project.tex” can differ from assessment to assessment. In this example, the system generates a random value between zero and three and, following the administrator-defined conditions (`case`), the file has a different content based on the obtained value. This process is illustrated in **Listing *3*** with the bash `case` statement and pattern `0)`; the remaining patterns are not displayed in the code snippet for simplicity purposes. Additionally, it is possible to observe a common segment of all tasks, and variable portions based on the generated number.

`TestsOverSSH` assesses students CLI-based knowledge in a viable and automatic way by focusing on multiple-choice and direct questions. This approach also enables the creation of multiple tests with a wide range of variability. Consider the case where each assessment is composed of 20 questions, whose content differs from student to student. Since each question or task offered four different possibilities, the production of $4^{questions+tasks}$ various assessments is achievable. For example, in this case, a 20 question/task test could have 4^{20} distinct assessments, which is not a manually feasible chore.

Application in real context

`TestsOverSSH` was used to evaluate students from the Computer Science graduation course, at the institution the authors are affiliated with. We divide our evaluation into two parts: 1) feedback from 120 students from a custom usability survey after an exam on

`TestsOverSSH`; 2) feedback, using the SUS, from 80 students that used `TestsOverSSH` throughout a course unit to perform exercises in their practical classes.

Usability survey

To retrieve user feedback from students, we first designed a survey. Students were asked about the system access easiness, informative texts clarity, system navigation easiness, system responsiveness, questions and tasks clarity, answer submission easiness, and assessment submission easiness and system exit. Each aspect was evaluated based on a scale from 1 to 7, with 1 being the most negative evaluation and 7 the most positive one. The scale choice was based on the human mind judgment capability (Miller, 1956). Additionally, students were given the means to express their suggestions and difficulties found while using the system.

Figure 3 exhibits the survey responses received. From its analysis, we can conclude that the overall score for this system is around 6, which proves that users have positive feedback on the system.

The categories more negatively rated were access, navigation, and responsiveness. The exam taking, which preceded this survey, was divided into groups of around 20 people. One of those groups had an unusual access problem with consecutive disconnections. The situation observed in this group was an outlier and not system-related since all the other groups were able to connect to it with relative ease. The students of the said group might have a more negative perception of the system, which would justify the low negative score

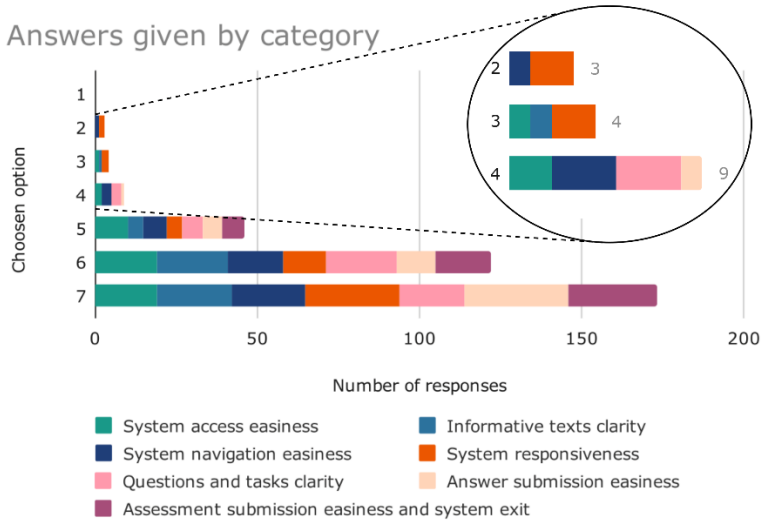

Fig. 3 Cumulative bar graph summarizing the survey responses. Each evaluated category is depicted by a distinct color. Options 2, 3, and 4 are zoomed in for better visualization. Best viewed in color

incidence. To further corroborate this idea, the more negatively rated categories are linked to recurrent system connection issues.

To evaluate the system efficiency, we registered problems encountered when answering questions and submitting the exam, for all the students. The majority of the students (96.70%) reported no problems encountered while taking the assessment. The small percentage of problems encountered did not prevent the students from continuing and finishing his/her evaluation. One example of these problems was environment variable loss following system disconnection. This variable needed to be set to a specific value, related to linker use to perform a specific task, and was not reset to said value if the student was required to reconnect to the system. The problem was temporarily solved, via manual variable setting, during the assessment performance, and the current `TestsOverSSH` implementation permanently solved this issue. Another reported problem was a simultaneous submission error, which occurs when two or more students submit a response at precisely the same time. In this case, the system does not know what operation to perform first and displays a submission error to the student. This situation was solved by sequential submission, in coordination with the administrator. One possible solution for this problem is the transaction mechanism implementation, which we will include in future work.

This system provides a non-typical exam-taking context, which could hinder student performance. However, given their feedback, through this survey, it is possible to conclude that questions were clearly presented and no significant difficulties were encountered while assessing, performing, and submitting their exam.

System Usability Scale

To obtain the feedback of students regarding the usability of `TestsOverSSH`, we also used SUS, which is a short survey that does not require many resources (or time) to distribute and whose template exists and is widely used in a variety of areas. We obtained feedback from 80 students that used `TestsOverSSH` throughout a course unit to perform exercises in their practical classes. This group of students is more familiar with the system and can provide a different perspective from the group of students that only used `TestsOverSSH` on a one-time occasion, for exam taking (Usability Survey, from the previous subsection). The distribution of answers, grouped by intervals, is displayed in Figure 4.

We obtained a score of 70.99 in SUS, which indicates that users perceive this tool as having good usability, with the need for minor improvements to the design (Brooke, 2013). From the analysis of Figure 4, we can observe that the majority (48.72%) of answers lead to a SUS score in the 60-80 interval, with a significant portion (33.33%) in the 80-100 interval. This suggests that most students found the system usable and with good design, with few people finding it difficult to operate. This could be linked to the requirement of

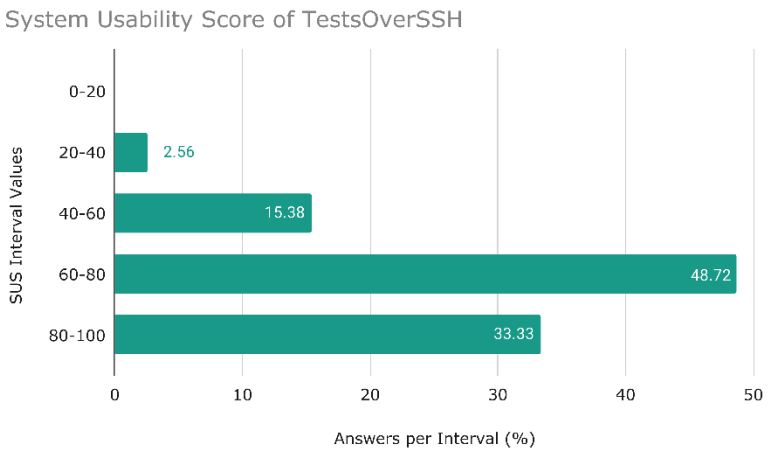

Fig. 4 System Usability Scale (SUS) evaluation of `TestOverSSH`. SUS values of students answers are grouped per intervals to provide a better overview of the system usability evaluation

using CLI, to answer questions and perform tasks, which may represent a challenge to less skilled students, thus hindering their perception of the system usability.

Conclusion

`TestsOverSSH` was developed under the COVID-19 context due to the lack of efficient virtual evaluation methods to assess theoretical knowledge and technical skills, in particular CLI-based knowledge, of Computer Science students. For versatility purposes, the proposed system allows remote, via the Internet, or local evaluation, via access point creation. This system is executed over a Unix-like operating system, accessible via SSH connection, which can log the commands introduced by each student. These two concepts make the system secure and capable of detecting fraud. Additionally, the students do not need to install any software besides the trusted SSH client, ensuring that this system does not trespass on student privacy. User feedback, via custom usability survey and SUS, showed that this system is intuitive and usable for e-learning.

As future work, we suggest developing a Web platform to ease the question and task implementation, detaching the administrator from the steps necessary to execute the system. This improvement would simplify the onerous manual work associated with task and question preparation. Furthermore, we intend to tackle the difficulties presented in the survey, such as connection issues and simultaneous access submission. Finally, we intend to release `TestsOverSSH` as an Open Virtualization Format (OVF), allowing instant execution of the system.

Abbreviations

ANSI: American National Standards Institute; CTF: Capture the Flag; CLI: Command Line Interface; COVID-19: Coronavirus disease of 2019; CSV: Comma-Separated Values; DML: Data Manipulation Language; ID: Identifier; IP: Internet Protocol; OVF: Open Virtualization Format; PDF: Portable Document Format; PIN: Personal Identification Number; SCALE: Support for Creativity in a Learning Environment; SQL: Structured Query Language; SSH: Secure SHell; SUS: System Usability Scale; VR: Virtual Reality; WHO: World Health Organization.

Endnote

[1] Code available at https://github.com/in4cio/TestsOverSSH

Acknowledgements

The authors would like to thank Universidade da Beira Interior, for allowing and providing means for conducting experiments in the classroom, and all the students that accepted to use this platform.

Authors' contributions

Each named author has substantially contributed to conducting the underlying research. All authors read and approved the final manuscript.

Authors' information

T. R.: tiago.roxo@ubi.pt; C. L.: carolina.lopes@ubi.pt; J. B. F. S.: jbfs@ubi.pt; T. M. C. S.: tsimoes@di.ubi.pt; P. R. M. I.: inacio@di.ubi.pt.

Funding

This work was performed under the scope of Project SECURIoTESIGN with funding from FCT/COMPETE/FEDER with reference number POCI-01-0145-FEDER-030657. This work is funded by Portuguese FCT/MCTES through national funds and, when applicable, co-funded by EU funds under the project UIDB/50008/2020 and research grant BIL/Nº11/2019-B00701 and FCT doctoral grants SFRH/BD/133838/2017, 2020.09847.BD, and 2021.04905.BD. It is also supported by project CENTRO-01-0145-FEDER-000019 - C4 - Competence Center in Cloud Computing cofinanced by the European Regional Development Fund (ERDF) through the Programa Operacional Regional do Centro (Centro 2020), in the scope of the Sistema de Apoio à Investigação Científica e Tecnológica - Programas Integrados de IC&DT.

Availability of data and materials

Code for the solution is available at https://github.com/in4cio/TestsOverSSH. The data for the Application in Real Context is not available as it concerns students' personal identifiable information.

Declarations

Competing interests

The authors declare that they have no competing interests.

Author details

Instituto de Telecomunicações, Universidade da Beira Interior, Rua Marquês de Ávila e Bolama, Covilhã, Castelo Branco, Portugal.

Received: 23 November 2021 Accepted: 23 November 2022
Published: 28 February 2023 (Online First: 2 January 2023)

References

Becerra-Alonso, D., Lopez-Cobo, I., Gómez-Rey, P., Fernández-Navarro, F., & Barbera, E. (2020). EduZinc: A tool for the creation and assessment of student learning activities in complex open, online, and flexible learning environments. *Distance Education*, *41*(1), 86–105. https://doi.org/10.1080/01587919.2020.1724769

Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability Evaluation in Industry*, *189*(194), 4–7.

Brooke, J. (2013). SUS: A retrospective. *Journal of Usability Studies*, *8*(2), 29–40.

Chicone, R., Burton, T. M., & Huston, J. A. (2018). Using Facebook's open source Capture the Flag Platform as a hands-on learning and assessment tool for cybersecurity education. *International Journal of Conceptual Structures and Smart Applications*, *6*(1), 18–32.

CodeGrade. (2021). *https://www.codegrade.com/—The flexible learning platform and autograder for coding education*. https://www.codegrade.com/

Daniel, S. J. (2020). Education and the COVID-19 pandemic. *PROSPECTS*, *49*(1), 91–96. https://doi.org/10.1007/s11125-020-09464-3

Dukhanov, A. V., Krzhizhanovskaya, V. V., Bilyatdinova, A., Boukhanovsky, A. V., & Sloot, P. M. (2014). Double-Degree Master's Program in Computational Science: Experiences of ITMO University and University of Amsterdam. *Procedia Computer Science*, *29*, 1433–1445.

Eldering, J., Gerritsen, N., Johnson, K., Kinkhorst, T., & Werth, T. (2021). *DOMjudge—Programming Contest Jury System*. https://www.domjudge.org/

Ferreira, D. J., Ambrósio, A. P. L., & Melo, T. F. (2018). Application of real-world problems in computer science education: Teachers' beliefs, motivational orientations and practices. *International Journal of Information and Communication Technology Education*, *14*(3), 15–28.

Huang, R. H., Liu, D. J., Tlili, A., Yang, J. F., & Wang, H. H. (2020). Handbook on facilitating flexible learning during educational disruption: The Chinese experience in maintaining undisrupted learning in COVID-19 outbreak. *Beijing: Smart Learning Institute of Beijing Normal University*, *46*.

John, R. (2021). *Canvas LMS Course Design: Create and deliver interactive online courses on the Canvas learning management system*. Packt Publishing Ltd.

Kleerekoper, A., & Schofield, A. (2018). SQL tester: An online SQL assessment tool and its impact. *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, 87–92.

Kumar, A., Krishnamurthi, R., Bhatia, S., Kaushik, K., Ahuja, N. J., Nayyar, A., & Masud, M. (2021). Blended learning tools and practices: A comprehensive analysis. *IEEE Access*, *9*, 85151–85197.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *63*(2), 81.

Nicola, M., Alsafi, Z., Sohrabi, C., Kerwan, A., Al-Jabir, A., Iosifidis, C., Agha, M., & Agha, R. (2020). The socio-economic implications of the coronavirus pandemic (COVID-19): A review. *International Journal of Surgery (London, England)*, *78*, 185–193. https://doi.org/10.1016/j.ijsu.2020.04.018

Obionwu, V., Broneske, D., Hawlitschek, A., Köppen, V., & Saake, G. (2021). SQLValidator–An online student playground to learn SQL. *Datenbank-Spektrum*, *21*(2), 73–81.

Özyurt, Ö., & Özyurt, H. (2015). Learning style based individualized adaptive e-learning environments: Content analysis of the articles published from 2005 to 2014. *Computers in Human Behavior*, *52*, 349–358. https://doi.org/10.1016/j.chb.2015.06.020

Patterson, T. (2013). *Blackboard learn administration*. Packt Publishing Ltd.

Poth, C. N., Searle, M., Aquilina, A. M., Ge, J., & Elder, A. (2020). Assessing competency-based evaluation course impacts: A mixed methods case study. *Evaluation and Program Planning*, *79*, 101789. https://doi.org/10.1016/j.evalprogplan.2020.101789

Reimers, F. M., & Schleicher, A. (2020). *A framework to guide an education response to the COVID-19 Pandemic of 2020*. OECD Publishing. https://doi.org/10.1787/6ae21003-en

Restrepo-Calle, F., Ramírez Echeverry, J. J., & González, F. A. (2019). Continuous assessment in a computer programming course supported by a software tool. *Computer Applications in Engineering Education*, *27*(1), 80–89. https://doi.org/10.1002/cae.22058

Richardson, C., & Mishra, P. (2018). Learning environments that support student creativity: Developing the SCALE. *Thinking Skills and Creativity*, *27*, 45–54. https://doi.org/10.1016/j.tsc.2017.11.004

Sahu, P. (2020). Closure of universities due to coronavirus disease 2019 (COVID-19): Impact on education and mental health of students and academic staff. *Cureus*, *12*(4), e7541. https://doi.org/10.7759/cureus.7541

Sun, B., Chikwem, U., & Nyingifa, D. (2019). VRlearner: A virtual reality based assessment tool in higher education. In *Proceedings of 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)* (pp. 1640–1645). The Institute of Electrical and Electronics Engineers.

Suzuki, M. T., Yaginuma, Y., & Kodama, H. (2019). A Command-Line based Exam Generation System for Computer Science Education. In *Proceedings of the 28th ICDE World Conference on Online Learning (WCOL2019)*, 905. International Council for Open and Distance Education.

Publisher's Note