electronics

Article

An Open-Source System for Generating and Computer Grading Traditional Non-Coding Assignments

Pablo Lapeña-Mañero *, Carmen García-Casuso, José Miguel Montenegro-Cooper, Robert W. King and Edwin M. Behrens

Department of Civil Engineering, Universidad Católica de la Santísima Concepción, Concepción 4090541, Chile; mcgarciacs@gmail.com (C.G.-C.); jmontenegro@ucsc.cl (J.M.M.-C.); rking@ucsc.cl (R.W.K.); ebehrens@ucsc.cl (E.M.B.)

* Correspondence: plapenamanero@gmail.com

Citation: Lapeña-Mañero, P.; García-Casuso, C.; Montenegro-Cooper, J.M.; King, R.W.; Behrens, E.M. An Open-Source System for Generating and Computer Grading Traditional Non-Coding Assignments. *Electronics* **2022**, *11*, 917. https://doi.org/10.3390/electronics11060917

Academic Editor: Manuel Palomo-Duarte

Received: 28 January 2022
Accepted: 10 March 2022
Published: 16 March 2022

Abstract: One of the most time-consuming activities in higher education is reviewing and grading student evaluations. Rapid and effective feedback of evaluations, along with an appropriate assessment strategy, can significantly improve students' performance. Furthermore, academic dishonesty is a major issue in higher education that has been aggravated by the limitations derived from the COVID-19 pandemic. One of the possible ways to mitigate this issue is to give different evaluations to each student, with the negative cost of increasing reviewing time. In this work, an open-source system developed in Python to automatically create and correct evaluations is presented. Using Jupyter Notebook as the graphical user interface, the system allows the creation of individual student question sheets, with the same structure and different parameter values, to send them to students, grade them, and send the final score back to the students. The proposed system requires little programming knowledge for the instructors to use it. The system was applied in Civil Engineering and Geological Engineering programs at the Universidad Católica de la Santísima Concepción, drastically reducing grading time while improving students' performance.

Keywords: computer grading; automatic grading; assessment; engineering education; open source

1. Introduction

The assessment of what students have learned during the semester for a particular course is one of the main responsibilities of instructors in higher education. Although the main objective of exams, assignments, and projects is to assess students' progress in the course, an appropriate design of evaluation strategies throughout the course can help students to obtain a functional and meaningful learning experience. Designing a course with shorter and more frequent evaluations improves students' study progress in the course [1,2]. Furthermore, proper and rapid feedback of the evaluations also helps students in their progress in the course [2–4]. Although project-based learning is gaining popularity in engineering education [5–7] and the evaluation of some courses currently relies solely on group projects [8], individual assessments are still a common means of evaluation in undergraduate programs. The possibility of increasing the number of evaluations done in a course is heavily conditioned by the evaluation reviewing time.

Another aspect to consider when designing an evaluation strategy is the possibility of students committing academic dishonesty during the evaluation, commonly known as cheating. Academic dishonesty is frequent and it has become a major problem in higher education in general and in engineering education in particular [9]. There are numerous studies in the literature on why students cheat [9–12] and how they do it depending on the typology of the examination activity [12–14]. Passow et al. [12] reported that students admitting this type of behavior at least once during their college education doubled from 1963 to 1993, from 26% to 52%. More recently, in 2006, Carpenter et al. [9] reported that this number had risen to 80% among US students. Although harsher institutional policies and rules can reduce the incidence of academic dishonesty [15], the existence of rigid honor codes is reported to have little effect on the students' perception of cheating [16].

It must be noted that all data summarized above were obtained from face-to-face learning and in-class exams. In the last two years, the majority of higher education institutions around the world were forced to implement online distance teaching and assessment as a consequence of the restrictions derived from the COVID-19 pandemic. Although there is not much data available concerning the changes that this has produced in assessment, a review study by Montenegro-Rueda et al. [17] concludes that, due to the restrictions of online evaluations, assessment in higher education has become more qualitative compared to when evaluations were carried out in presential written format. The authors also state that a total redesign of the exams is needed to replace a face-to-face assessment with a virtual one. Furthermore, based on both the perception of teachers and students, some studies indicate that cheating during an online evaluation is easier [14]. A possible way to prevent cheating in an online evaluation is to have different test instructions, questions, and results for each student [18]. On the downside, preparing and grading an individual evaluation for each student also results in a substantial increase in reviewing time.

A common way to reduce reviewing time is the usage of computed-graded evaluation tools. Unfortunately, the available tools in commonly used Learning Management Systems (LMS), such as Moodle [19], Canvas [20], or Blackboard [21], only have built-in tools to create multiple choice evaluations or activities that allow students to submit written assignments for the instructor to grade manually. There are a number of systems to computer-grade complex assignments for programming-related assignments in a variety of programming languages [22–27]. Similarly, there are systems available for the creation of online assignments, such as Webassign [28] (proprietary) and WeBWork [29] (open-source). These alternatives are unfortunately limited to basic math, and their usage limits the complexity of the problems that can be proposed. In addition, both applications are actually online tools, and the students are required to be connected to the internet during the whole length of the test. This can be a problem for students with poor internet connections.

An open-source project allows access to its source code and its free redistribution under certain conditions. Among the different licenses available under the open-source movement, some are more permissive regarding the creation of derivate products and others are more restrictive. However, source code modification is allowed for people different from the original project owners in all cases. This software distribution paradigm introduces several advantages to software development. From the end user point of view, the allowance of creating derivate products makes it possible to introduce modifications to the code to adapt it to particular needs, without the intervention of the original development team. Moreover, being able to inspect the code allows users to help in the development of the original project and ensure that there are no malicious codes. Finally, code inspection is a valuable source in computer programming learning.

Among the different programming languages used in open-source projects, Python has gained growing interest in the last decade, both in industry and academia. It is widely used by researchers in different engineering fields, such as machine learning [30], electronics [31,32], and coastal engineering [33], to name a few. It has become one the most popular programming languages to teach introductory computer science courses, above all, in non-computer-related programs. Moreover, Python is free to use, open-source, and cross-platform, making it a perfect candidate for small open-source projects such as the one described in this article.

This document describes the implementation of a system to create and computer-grade assignments for Civil Engineering education based on the Python programming language [34] and with Jupyter Notebooks [35] as a user interface. All students in the course are assigned a different set of instructions, and thus, the correct solution is also different for each student. Instructions and grades are quickly and automatically sent to students using an Office 365 account.

The rest of this document is structured as follows. In Section 2, an overview of the system is presented, with special emphasis on the structure of the developed module and the user interface. Section 3 describes the suggested workflow of the system and typical usage instructions. Two different experiences using the system in Civil and Geological Engineering education are described in Section 4. Finally, the main conclusions of the present study are presented.

2. System Overview

The main purpose of this developed open-source system is to generate and computer-grade personalized assignments for students using the same template document with random parameter values. This approach allows students to be graded by checking whether their answers to each question are within the established tolerance from the correct solution, considering that the answers of every student are different and depend on the randomly generated parameter values of the evaluation.

There are several tools already available to generate and computer-grade assignments, but their usage in engineering education is limited. To solve the problems typically set out in engineering evaluations, it is usually necessary to use specialized toolboxes or advanced calculation techniques currently unavailable in those common systems. Furthermore, in some engineering problems, a change in the value of a given initial parameter can change the way the problem is solved, and thus, it is required to implement advanced logic to address all possible scenarios. With these two limitations in mind, the present work describes a newly created, publicly available, open-source system based on the Python programming language [34] that addresses the issues found in current tools to generate and computer-grade traditional engineering assignments.

In addition, civil engineering problems are usually complex to describe and often need schematics to provide data. Using the systems currently available also limits the complexity of both the instructions and the questions. Lastly, the system is designed to provide an almost indistinguishable experience to the students compared to traditional exams and assignments.

The system is designed to work in all major platforms (Mac, Windows, and Linux) and requires little computing power to run correctly. It can work on any system capable of running Jupyter Notebooks. However, it is important to consider that the data are stored without any kind of encryption, and thus caution needs to be taken on the access to the computer where they are stored.

2.1. Assignment Package

The open-source system is totally programmed in Python using packages commonly used in data science and engineering, such as Pandas [36,37] and NumPy [38], and uses Jupyter Notebooks [35] as a graphical user interface. The system is built around the class Assignment, which contains several Pandas DataFrames to store the data generated along the assignment creation process, student question sheet (henceforth, the term “sheet” will be used when referring to the student question sheet or questionnaire) sending, grading, and results sending. All DataFrames are stored on disk in a .xlsx workbook for convenience and interoperability with other systems. Lists with instance variables and the main methods of the class Assignment are shown in Tables 1 and 2, respectively. The open-source system can be used with little experience in Python programming but can be further customizable by a user with programming capabilities. Some of the methods contained in the class Assignment provide interactivity using ipywidgets [39] and ipysheet [40] with Jupyter Notebooks, and thus can only be used in a Jupyter Notebook-compatible environment (Table 2). An example of the outcome of this type of method is shown in Figure 1.

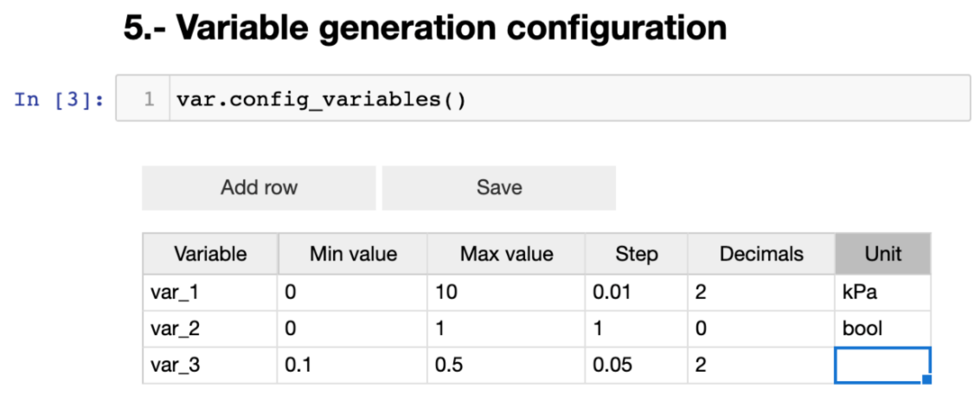
Table 1. Instance variable name list in the class Assignment.

Name	Description
config	Stores the main configuration of the assignment, such as professor, course name, assignment name, etc.
student_list	Stores a standardized list with student name, ID, and email address.
var_config	Stores the data needed to create random variables.
variables	Stores the values of the generated variables.
solutions	Stores the correct answers for the questions in the assignment.
answers	Stores students' answers to the questions in the assignment.
grading_config	Stores the tolerance on correction and the points assigned to each assignment question.
grades	Stores the point per question and the final grade for each student.

Table 2. Main methods of the class Assignment.

Name	Description
configure() [1]	Allows the user to set the assignment's basic parameters and stores them in the config.
load_students() [2]	Asks for student list, formats it, and stores it in student_list instance variable.
config_variables() [1]	Allows the user to set the ranges and steps of the randomly generated variables and stores them in var_config.
generate_variables()	Generates random variables with the configuration stored in var_config and stores the result in variables instance variable.
generate_solutions(solver)	Accepts a function that loads the necessary variables from variables, computes solutions, and stores them in solutions.
load_answers() [2]	Asks for answers collected in a form in Comma Separated Values (CSV) format and stores them in answers.
config_grading() [1]	Allows the user to set the score assigned to every question and the accepted tolerance for the student's answers to be considered correct.
grade()	Grades assignments and stores partial and total grades in grades.

[1] Methods with Jupyter Notebook interactivity. [2] Methods using PyQt5 for dialog creation.



5.- Variable generation configuration

```
In [3]: var.config_variables()
```

Variable	Min value	Max value	Step	Decimals	Unit
var_1	0	10	0.01	2	kPa
var_2	0	1	1	0	bool
var_3	0.1	0.5	0.05	2	

Figure 1. Example of interactivity using Jupyter Notebook, ipywidgets, and ipysheets.

Along with the main Python file containing the class Assignment definition, there are three companion modules to handle different actions. Figure 2 shows the structure of the assignments package. The module generate_pdf.py handles the creation and encryption of the individual pdf files containing the instructions and questions for every student from a single pdf file containing the generated sheet for every student. Furthermore, the gui.py module uses the package PyQt5 [41] to create dialog windows when other parts of the system request a file from the user. Finally, office_365_mail.py is used to send emails to students using the package O365 [42]. Office 365 was chosen as it is the institutional email provider of our university. However, all email functionalities have been encapsulated in a separate module to facilitate the adaptation of the system for other email service providers. All dependencies can be easily installed using common Python package installers, such as pip [43] or conda [44], and are free and open-source.

```
assignments
├── __init__.py
├── assignment.py
├── generate_pdf.py
├── gui.py
└── office_365_mail.py
```

Figure 2. Assignments package folder structure.

2.2. Structure

The user interface is based on Jupyter Notebooks. The code necessary to execute all the steps has been divided into seven notebooks to make the system more usable. It should be noted that most of the logic used in the process is contained in the assignments package described above. Figure 3 shows the structure of the main folder of the system.

```
assignment_generator
├── 00_system_configuration.ipynb
├── 01_variable_generation.ipynb
├── 02_solution_generator.ipynb
├── 03_pdf_generator.ipynb
├── 04_send_emails.ipynb
├── 05_grader.ipynb
├── 06_grade_sender.ipynb
├── LICENSE.txt
├── assignments
│   ├── __init__.py
│   ├── assignment.py
│   ├── generate_pdf.py
│   ├── gui.py
│   └── office_365_mail.py
├── complements
│   └── __init__.py
├── gen
├── resources
│   ├── assignments_conda_environment.yaml
│   └── templates
│       ├── email_template.html
│       └── grade_email_template.html
└── sheets
```

Figure 3. System main folder structure and contents.

Along with the seven notebooks and the folder with the assignments package described in Section 2.1, in the root directory of the system, there are four sub-directories used to store different components that are needed or are produced in the process. The folder complements contains an __init__.py file in order to act as a package. All the additional Python modules needed for solving the problem should be placed in this folder. This design keeps the structure of the system simple and easy to maintain. The folder gen is used to store all generated data within the system and outside of it. The sheets folder is used to store the generated individual pdf files with the instructions and questions for each student prior to being sent. Lastly, in the folder resources, configuration files are stored. In the sub-folder templates, HyperText Markup Language (HTML) templates for the email messages are stored.

3. System Usage

Before using the system, some configurations need to be made to the computer where it will be running. Using a Python virtual environment is highly recommended. The file assignments_conda_environment.yaml (contained in resources folder) can be used to create a new environment using conda [44] directly, both using the terminal version and the graphical user interface Anaconda-Navigator [45]. The file contains a list with all the dependencies of the project and the version used in development. All the packages are included in PyPi [43] and can also be installed using pip on another virtual environment system or directly on the main Python installation.

Once the Python environment is configured, an Azure [46] app must be created to handle email sending. The O365 package [42] is used to connect to Office 365 services.

An updated tutorial on how to configure the Azure app necessary to use the package is available on the readme in the O365 GitHub repository.

The system workflow is divided into seven Jupyter Notebooks, numbered from 0 to 6. The suggested workflow in an assignment is shown in Figure 4, and details on the usage of the notebooks are included in the notebooks themselves. In the figure, the procedures that must be performed using external applications are outlined in orange.

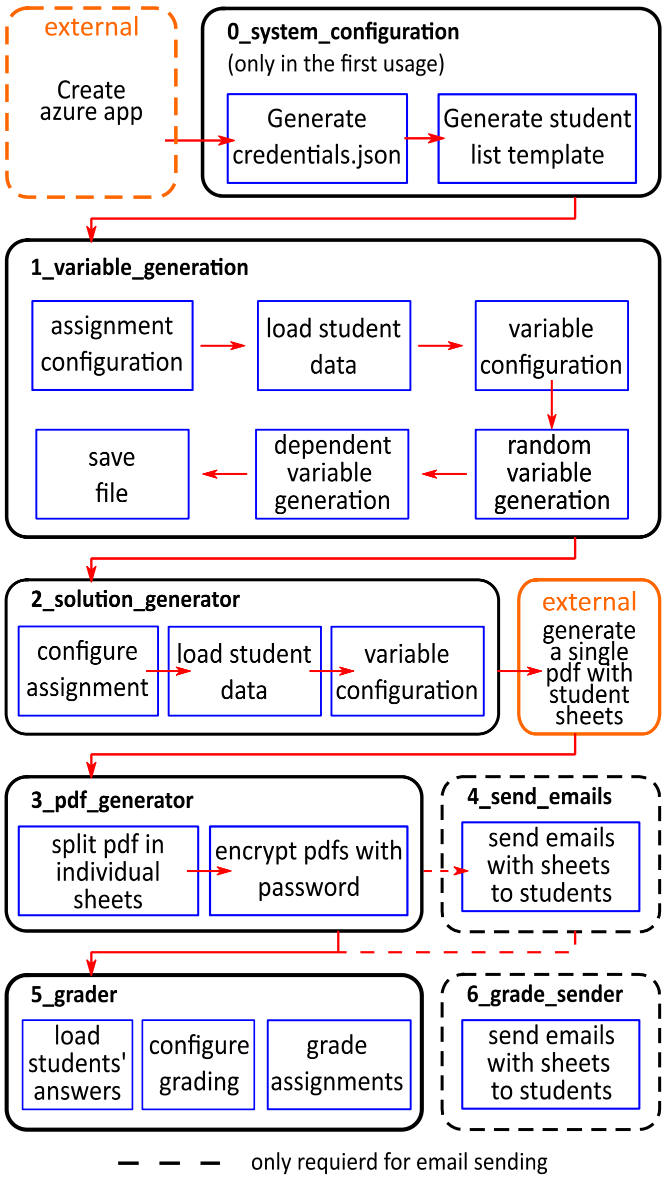

external
Create azure app
0_system_configuration
(only in the first usage)
Generate credentials.json
Generate student list template
1_variable_generation
assignment configuration
load student data
variable configuration
save file
dependent variable generation
random variable generation
2_solution_generator
configure assignment
load student data
variable configuration
external
generate a single pdf with student sheets
3_pdf_generator
split pdf in individual sheets
encrypt pdfs with password
4_send_emails
send emails with sheets to students
5_grader
load students' answers
configure grading
grade assignments
6_grade_sender
send emails with sheets to students
only requierd for email sending

Figure 4. Suggested workflow.

The first external action is to create the Azure app. Once the app is created, notebook 0_system_configuration manages the creation of the Office 365 configuration file (credentials.json) and stores it in the resources folder. This file needs to be created only once as long as the configuration of the Azure app remains constant. To reuse the file in a different set of notebooks, simply copy the file to the resources folder.

Notebook 01_variable_creation handles the importing of the student list and creates a set of random values of the problem variables for each student in the list. Notebook 02_solution_generator is used to compute the correct solution for every generated variable set. In the notebook, a function called solver() must be defined. This function's task is to compute the solution for a single student. This design approach was preferred to a vectorized function to compute the solution of every variable set in order to make coding the function easier.

With the student list imported and formatted, and variable values available, creating a pdf document containing the instructions and questions sheets for every student is necessary. This is typically achieved using word-processing software and the mail merge utility included in most of them. The information to populate the sheets is contained in the variable sheet of the Excel workbook data.xlsx placed in the gen folder. Once the pdf file with all sheets is generated, the notebook 3_pdf_generator is used to create the individual sheet for every student and store them in the sheets folder. The system allows the encryption of the sheets with a password. The sheet generation process is schematized in Figure 5. Notebook 04_send_emails is used to send the pdf files generated in the previous notebook to the corresponding student.

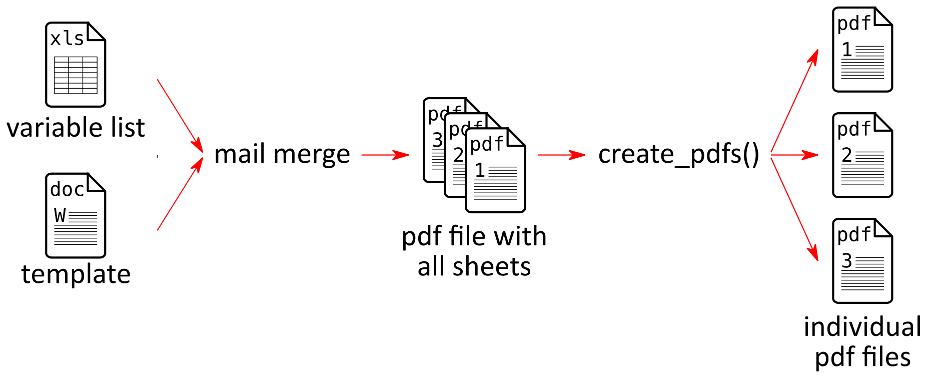


Figure 5. Individual sheet creation scheme.

Answers are typically collected using Google Forms or similar. The system accepts the answers in Comma Separated Values (CSV) format and stores them in data.xlsx.

The notebook 05_grader is used to automatically calculate the score and the final grade obtained for each student. Grading is done by comparing the answer provided by the student with the correct solution computed previously. The system assigns a full score if the answers are within the tolerance indicated for each question. Finally, notebook 06_grade_sender is used to send an email to students with the answers registered in the system and the correct solution for each question, along with their scores and final grade. Typically, a document describing how to solve the problem is uploaded to the course's Learning Management System (LMS) page to improve feedback.

4. Experiences Using the System

Several versions of the system developed in different programming languages have been used in Civil Engineering and Geological Engineering programs at the Universidad Católica de la Santísima Concepción (UCSC) in Chile since 2015. In the past two years, with the changes in higher education caused by the COVID-19 pandemic, the need for a comprehensive system for assignment and test generation with the capability to automatically generate, send, and grade evaluations has arisen. This section describes two approaches or case studies taken in different areas for both programs, along with the benefits for both students and instructors.

4.1. Solid Mechanics

The first contact with the evaluation system described in this work for both programs is in the first course on the mechanical behavior of materials, called "Strength of Materials" and "Solid Mechanics" in the Civil Engineering and Geological Engineering programs, respectively. Both courses have similar content and learning objectives and the instructors for the courses belong to the same faculty group. Thus, essentially the same methodology has been applied in both courses. Within the curriculum of both programs, Strength of

Materials is programmed in the fifth semester of Civil Engineering and Solid Mechanics in the fourth semester for Geological Engineering.

The methodology described in this work has been used in both courses since 2015. The initial adoption of the system was motivated by the increasing number of students enrolled in the courses, and the need to create individual assignments with different parameter values and hence different answers, to prevent students from cheating during exams. Initially, the methodology was used for classroom-based exams. In this scenario, each exam sheet was assigned with an ID number that identified the set of parameter values and correct answers and was handed out to students in the class randomly. Along with instructions and questions, each exam sheet included a table for the students to write down their answers. These answers were then introduced in the system manually by teaching assistants and graded automatically using a previous version of the system, developed in Visual Basic for Applications (VBA) with Microsoft Excel as the user interface. The system used fundamentally worked in the same way as the system described in this work. The adoption of this assessment methodology drastically reduced the copying of answers between students, as all answers are different. It also reduced the time spent by instructors and teaching assistants to review the exams and assign grades. Furthermore, from the instructors' point of view, the methodology makes it necessary for the students to better understand the fundamental concepts learned by reducing the possibility of ambiguous answers.

The usage of the methodology also allowed the redesign of the assessment strategy, allowing the creation of a second step in which students can try to solve the problem again at home to improve their grades. Currently, evaluations are conducted in two stages. In the first stage, students solve their exams independently with a strict time limit, usually around 120 min. This stage can be done in class or remotely, especially in the last four semesters with the limitations derived from the COVID-19 pandemic, but in both cases, the students are not allowed to receive any external assistance to answer their questions. When the test is done remotely, students upload their answers using an online form, usually Google Forms. In the second stage, students are allowed to take their exams home and try to solve them again. In this stage, students are encouraged to work together, as the main objective is to improve the students' abilities and help them fill any gap they could have in their initial knowledge. This stage usually spans several days, and students upload their answers directly to an online form.

The final grade of the first part is calculated by adding the score obtained for each question, which is then converted to a scale of 1 to 7. To make it attractive for students to develop the second stage, only if the answer to a question is correct, the final score of that question is calculated using Equation (1).

$$Score_{stage_1+stage\_2} = 0.85 \cdot Score_{satage_1} + 0.15 \cdot Score_{stage_2} \tag{1}$$

Similarly, the final grade is calculated by adding the stage 1 and stage 2 scores of each question, which is then converted to a scale of 1 to 7. If the answer to the question in stage 2 is incorrect, the score assigned in stage 1 is maintained. With this strategy, the final grade is usually 0.1 to 0.4 higher than the grade obtained in stage 1.

The platform to upload answers is open for the whole length of the assignments, and students are allowed to modify their answers within that time frame, but the system only keeps the last answer provided. After the time limit is reached, the platform stops accepting answers and the instructors can download students' answers in CSV format, which can be read directly with the developed system.

Finally, the assessment system and the reduction in reviewing time associated with it make it possible to increase the number of evaluations during the semester and the realization of a recuperative exam for all students. This type of recuperative exam is usually offered only to students that could not take the exam for medical reasons, but the usage of the system allows it to be offered to all students, as increasing the number of students taking the exam does not translate into additional reviewing time.

4.2. Soil Mechanics

The system is also used in Soil Mechanics courses in the same programs discussed above. The objective of these courses is for the students to learn the basics of saturated soil mechanics. The learning outcomes of both courses are eminently applied and designed to help the students understand the behavior of soils. The knowledge gained in these courses can be applied directly in professional practice and is the basis of subsequent courses on geotechnical works and foundations. In Civil Engineering, only the Soil Mechanics course belongs to the minimal curriculum in this area, and it is delivered in the seventh semester. On the other hand, in Geological Engineering, two courses related to the area are mandatory and are offered in the fifth (Soil Mechanics) and seventh (Geotechnics) semesters. As previously described for the Strength of Materials and Solid Mechanics courses, the high number of students enrolled in the courses required an intensive use of time to review and grade evaluations and assignments, and thus, the number of student evaluations was limited.

The methodology described in this work was first introduced in the courses for the last test of the second semester of 2019, when the teaching mode in the university changed to an online modality. This first implementation was considered successful, and all evaluations have used the methodology since the first semester of 2020. The system has allowed the assessment strategy to be redesigned. In this case, the number of term exams has been raised from two to four. Additionally, prior to each exam, an assignment is sent to the students to help them prepare for the exam. These assignments are implemented as a formative evaluation, and thus, their main objective is not to provide instructors with information about students' performance but rather to help students have real feedback on their knowledge. The system allows the assignments to be reviewed rapidly and provides feedback to students within minutes to help them prepare for their exams. The assignments have a developing time of around four to seven days, and students are encouraged to work together in order to solve their individually generated problems. A week after each assignment is turned in, the same learning objectives are evaluated using traditional exams. These exams are meant to be solved individually and have a duration between 60 and 120 min. To obtain the course's final grade, the average grade of the assignments and the average grade of the exams are weighted, and the final grade (excluding laboratory assessments) is obtained using Equation (2).

$$Grade_{course} = 0.08 \cdot Grade_{asignments} + 0.92 \cdot Grade_{exams} \tag{2}$$

The answers for all the activities are collected using Google Forms. Students can submit an unlimited number of answers; however, only the last is considered for grading. A CSV file with the answers submitted by the students is downloaded and fed to the system to obtain students' grades.

The impact of the methodology on students' performance has been evaluated based on the evolution of the percentage passing and the mean grade of students in the last eight semesters in the course Soil Mechanics in the Civil Engineering program, as this course has the highest student enrollment. Figure 6 shows the pass and fail percentage for the Soil Mechanics course in the last eight semesters, and Figure 7 shows the average grade in the same periods. It should be noted that grades in the Chilean system range from one to seven, and the passing grade is usually established at four.

The percentage of passing students has increased from around 20% before the change in the methodology to around 50% on average for the last three semesters, with no significant variation in the difficulty of the proposed exams and with minor changes in teaching methodology. Regarding average grades, while the mean grade of the students that pass the course has not changed noticeably, the general average grade has increased significantly in the last three semesters.

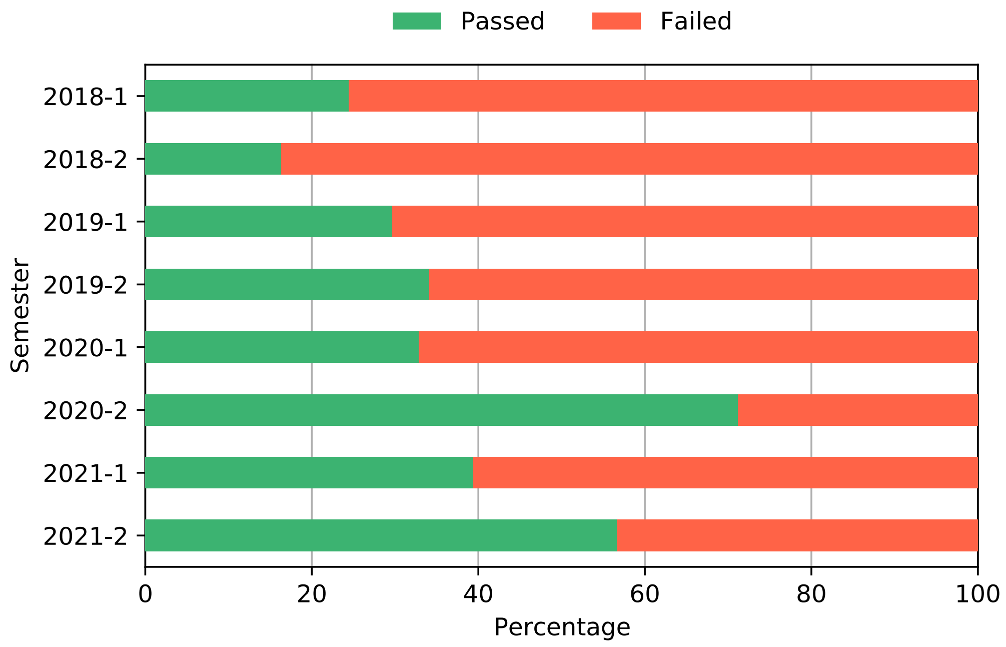


Figure 6. Pass and fail percentage per semester in Soil Mechanics for the last eight semesters.

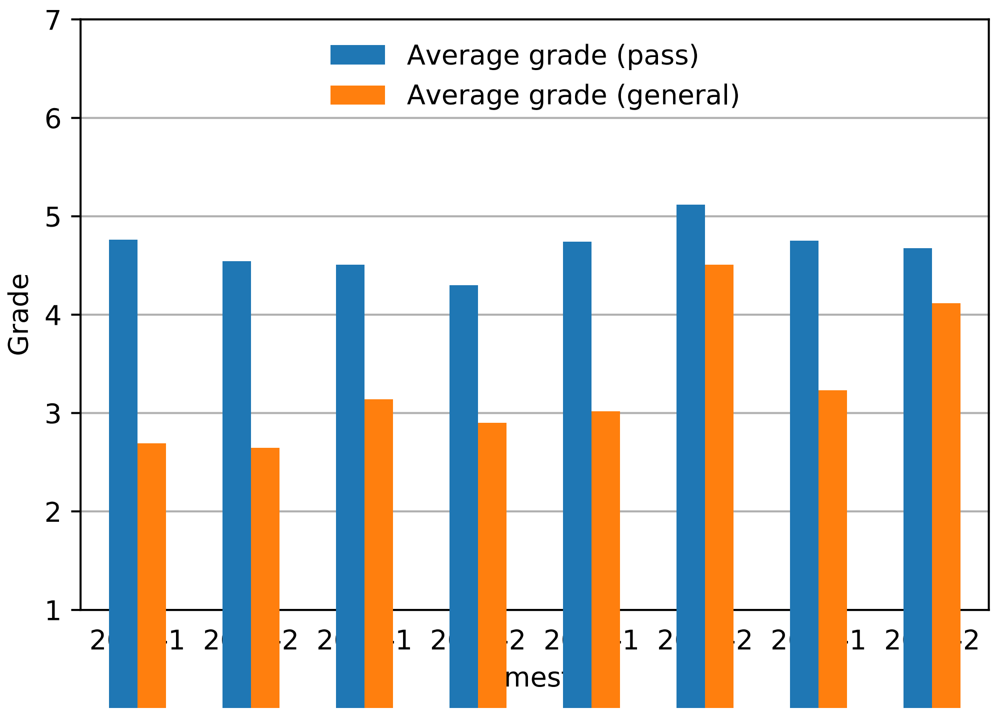


Figure 7. Average grades in Soil Mechanics for the last eight semesters.

To assess the statistical significance of the assessment strategy change, two datasets have been created from the original data, one for the grades with the traditional assessment approach (Old) and another with the grades obtained by students after the implementation of the new assessment (New). In the first dataset (labeled Old), all the grades of the semesters in which the proposed system and the new assessment strategy were not applied are included. There are a total of 278 records for this period, which spans five semesters, from the first semester of 2018 to the first semester of 2020. On the other hand, in the second dataset (labeled New), there are fewer records (190) as it only spans the last three semesters, from the second semester of 2020 to the second semester of 2021. It bears mentioning that the grade data have been anonymized before data analysis, and thus each record corresponds with the final grade of a student in the corresponding semester. Basic descriptive statistics for both datasets are shown in Table 3, while Figure 8 shows boxplots of the two datasets.

Table 3. Descriptive statistics of grades with Old and New assessment strategies.

Assessment Strategy	Count (N)	Mean ($\bar{x}$)	Variance (s^2)	Q1 (25%)	Median (50%)	Q3 (75%)
Old	278	2.9	1.569	1.9	2.6	4.0
New	190	3.9	1.567	2.9	4.1	4.9

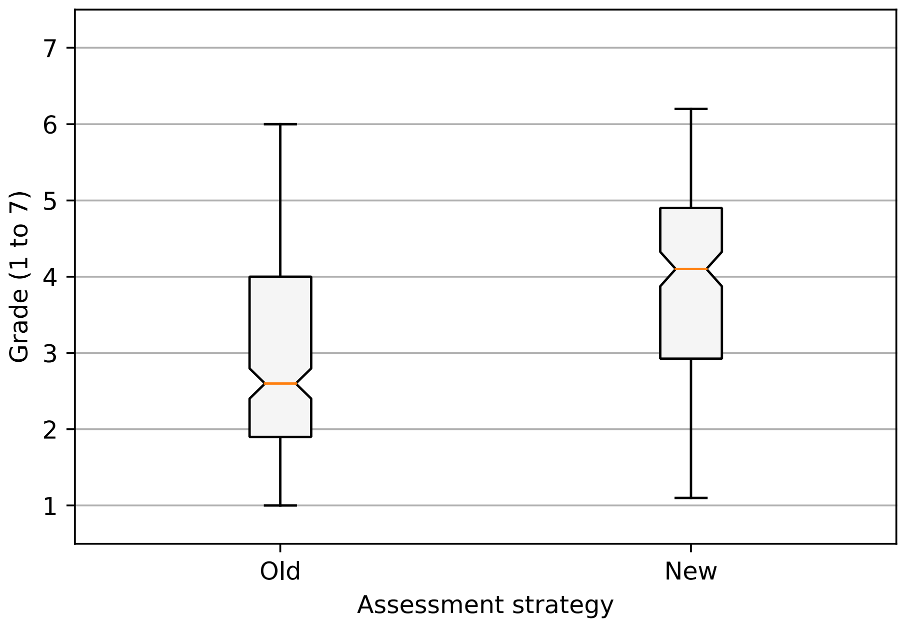


Figure 8. Boxplots of grades with Old and New assessment strategies.

Both boxplots shown in Figure 8 indicate that the data distribution is not Gaussian. To corroborate this, Shapiro–Wilks's normality tests were applied to both datasets, obtaining values of p of 2.361×10^{-8} and 4.403×10^{-5} for the Old and New datasets, respectively. The Mann–Whitney U rank test is an alternative to the t-tests when sample distribution is not normal, and variances can be considered equal, as is the case in the studied data. A Mann–Whitney U test was conducted on the available data, consisting of a total of 468 records grouped based on the assessment strategy. The obtained mean ranks and sum of ranks for both datasets are shown in Table 4, while the Mann–Whitney U test statistics are summarized in Table 5.

Table 4. Mann–Whitney U test ranks.

Assessment Strategy	N	Mean Rank	Sum of Ranks
Old	278	193.48	53,788.50
New	190	294.51	55,957.50

Table 5. Mann–Whitney U test statistics summary.

	Grade [1]
Mann–Whitney U	15,007.500
Wilcoxon W	53,788.500
Z	−7.939
Asymptotic Significance (2-tailed) p-value	1.0235×10^{-5}

[1] Grouping variable: Assessment strategy.

The values of the mean rank and the U and W statistics depend on the number of records in the datasets analyzed; thus, to determine the statistical independence of large datasets ($N \geq 30$), the values of the Z and p statistics must be analyzed, whose values do not depend on the amount of data analyzed. In the case of the Z statistic, it is considered that the null hypothesis can be rejected and, therefore, that the samples come from different populations if the absolute value of that statistic is greater than 2.58 (for a 99% confidence interval). The analyzed data yield a Z value of -7.939, so it can be concluded that the samples are statistically independent. On the other hand, the value of the p parameter obtained is less than 0.001 (p-value = 1.0235×10^{-5}); therefore, based on the value of this statistic, it can be assured with a confidence level of 99% that the two samples can be considered to belong to two non-identical populations.

5. Conclusions

This document describes an open-source system developed to generate and computer-grade assignments in engineering education. The creation of individual assignments reduces the possibility of academic dishonesty during the evaluations while also drastically reducing the grading time.

The system has been used successfully in several engineering courses at the Universidad Católica de la Santísima Concepción, Chile, both for presential evaluations and online distance modality. The system allows the evaluation strategy to be redesigned, significantly improving the number of students that succeed in the courses.

Currently, the main limitation of the system is that only problems with an exact and unique solution can be graded. This can be a problem in advanced design courses, where multiple solutions can be achieved depending on the technical decisions made during the design process. However, the system is ready to be used in most science and engineering courses with the current approach. The system is still under development, and more features will be added and made public as they are considered production-safe. Development is currently focused on an improved variable creation process and adapting the system to work as a web application instead of using Jupyter Notebooks.

The implementation of the system in two courses in the Civil Engineering and Geological Engineering programs in the Universidad Católica de la Santísima Concepción has made it possible to redesign the assessment strategy of the courses, with a significant impact on students' performance.

The described software source code is available at a public repository under MIT license.

Author Contributions: Conceptualization, P.L.-M. and E.M.B.; formal analysis, P.L.-M.; investigation, P.L.-M. and C.G.-C.; methodology, P.L.-M. and C.G.-C.; project administration, P.L.-M.; software, P.L.-M., C.G.-C. and E.M.B.; supervision, J.M.M.-C. and R.W.K.; validation, C.G.-C., J.M.M.-C. and R.W.K.; visualization, P.L.-M.; writing—original draft, P.L.-M.; writing—review and editing, J.M.M.-C., R.W.K. and E.M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code of the system described in this article is available at https://github.com/plapenamanero/assignment_generator (accessed on 27 January 2022).

Acknowledgments: The authors want to thank our colleagues Frank Sanhueza and Nelson Maureira in the Civil Engineering Department of the Universidad Católica de la Santísima Concepción, who tested some previous versions of the system. The authors are also grateful for the patience and understanding of their students both in Civil Engineering and Geological Engineering during the process of developing this project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nonis, S.A.; Hudson, G.I.; Logan, L.B.; Ford, C.W. Influence of Perceived Control over Time on College Students' Stress and Stress-Related Outcomes. *Res. High. Educ.* **1998**, *39*, 587–605. [CrossRef]
2. Jansen, E.P.W.A. The Influence of the Curriculum Organization on Study Progress in Higher Education. *High. Educ.* **2004**, *47*, 411–435. [CrossRef]
3. Evans, C. Making Sense of Assessment Feedback in Higher Education. *Rev. Educ. Res.* **2013**, *83*, 70–120. [CrossRef]
4. Pereira, D.; Flores, M.A.; Simão, A.M.V.; Barros, A. Effectiveness and Relevance of Feedback in Higher Education: A Study of Undergraduate Students. *Stud. Educ. Eval.* **2016**, *49*, 7–14. [CrossRef]
5. Fini, E.H.; Awadallah, F.; Parast, M.M.; Abu-Lebdeh, T. The Impact of Project-Based Learning on Improving Student Learning Outcomes of Sustainability Concepts in Transportation Engineering Courses. *Eur. J. Eng. Educ.* **2018**, *43*, 473–488. [CrossRef]
6. Guo, P.; Saab, N.; Post, L.S.; Admiraal, W. A Review of Project-Based Learning in Higher Education: Student Outcomes and Measures. *Int. J. Educ. Res.* **2020**, *102*, 101586. [CrossRef]
7. Megayanti, T.; Busono, T.; Maknun, J. Project-Based Learning Efficacy in Vocational Education: Literature Review. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *830*, 042075. [CrossRef]
8. Cifrian, E.; Andrés, A.; Galán, B.; Viguri, J.R. Integration of Different Assessment Approaches: Application to a Project-Based Learning Engineering Course. *Educ. Chem. Eng.* **2020**, *31*, 62–75. [CrossRef]
9. Carpenter, D.D.; Harding, T.S.; Finelli, C.J.; Montgomery, S.M.; Passow, H.J. Engineering Students' Perceptions of and Attitudes Towards Cheating. *J. Eng. Educ.* **2006**, *95*, 181–194. [CrossRef]
10. Drake, C.A. Why Students Cheat. *J. High. Educ.* **1941**, *12*, 418–420. [CrossRef]
11. Simkin, M.G.; McLeod, A. Why Do College Students Cheat? *J. Bus. Ethics* **2010**, *94*, 441–453. [CrossRef]
12. Passow, H.J.; Mayhew, M.J.; Finelli, C.J.; Harding, T.S.; Carpenter, D.D. Factors Influencing Engineering Students' Decisions to Cheat by Type of Assessment. *Res. High. Educ.* **2006**, *47*, 643–684. [CrossRef]
13. Garavalia, L.; Olson, E.; Russell, E.; Christensen, L. How do students cheat? In *Psychology of Academic Cheating*; Anderman, E.M., Murdock, T.B., Eds.; Academic Press: Burlington, NJ, USA, 2007; pp. 33–55. ISBN 978-0-12-372541-7.
14. Chirumamilla, A.; Sindre, G.; Nguyen-Duc, A. Cheating in E-Exams and Paper Exams: The Perceptions of Engineering Students and Teachers in Norway. *Assess. Eval. High. Educ.* **2020**, *45*, 940–957. [CrossRef]
15. Hall, T.L.; Kuh, G.D. Honor Among Students: Academic Integrity and Honor Codes at State-Assisted Universities. *NASPA J.* **1998**, *36*, 2–18. [CrossRef]
16. Tatum, H.E.; Schwartz, B.M.; Hageman, M.C.; Koretke, S.L. College Students' Perceptions of and Responses to Academic Dishonesty: An Investigation of Type of Honor Code, Institution Size, and Student–Faculty Ratio. *Ethics Behav.* **2018**, *28*, 302–315. [CrossRef]
17. Montenegro-Rueda, M.; Luque-de la Rosa, A.; Sarasola Sánchez-Serrano, J.L.; Fernández-Cerero, J. Assessment in Higher Education during the COVID-19 Pandemic: A Systematic Review. *Sustainability* **2021**, *13*, 10509. [CrossRef]
18. Böhmer, C.; Feldmann, N.; Ibsen, M. E-exams in engineering education—On-line testing of engineering competencies: Experiences and lessons learned. In Proceedings of the 2018 IEEE Global Engineering Education Conference (EDUCON), Santa Cruz de Tenerife, Spain, 17–20 April 2018; pp. 571–576.
19. Moodle—Open-Source Learning Platform. Available online: https://moodle.org/ (accessed on 24 January 2022).
20. Canvas Overview. Available online: https://www.instructure.com/canvas (accessed on 24 January 2022).
21. Blackboard Learn—An Advanced LMS. Available online: https://www.blackboard.com/teaching-learning/learning-management/blackboard-learn (accessed on 24 January 2022).
22. Von Matt, U. Kassandra: The Automatic Grading System. *ACM Spec. Interest Group Comput. Uses Educ. Outlook* **1994**, *22*, 26–40. [CrossRef]
23. Morris, D.S. Automatic grading of student's programming assignments: An interactive process and suite of programs. In Proceedings of the 33rd Annual Frontiers in Education, FIE 2003, Boulder, CO, USA, 5–8 November 2003; Volume 3, pp. S3F–1.
24. Nabil, R.; Mohamed, N.E.; Mahdy, A.; Nader, K.; Essam, S.; Eliwa, E. EvalSeer: An Intelligent Gamified System for Programming Assignments Assessment. In Proceedings of the 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), Cairo, Egypt, 26–27 May 2021; pp. 235–242.
25. Helmick, M.T. Interface-Based Programming Assignments and Automatic Grading of Java Programs. *SIGCSE Bull.* **2007**, *39*, 63–67. [CrossRef]
26. Ke, H.; Zhang, G.; Yan, H. Automatic Grading System on SQL Programming. In Proceedings of the 2009 International Conference on Scalable Computing and Communications; Eighth International Conference on Embedded Computing, Dalian, China, 25–27 September 2009; pp. 537–540.
27. Jupyter, P.; Blank, D.; Bourgin, D.; Brown, A.; Bussonnier, M.; Frederic, J.; Granger, B.; Griffiths, T.; Hamrick, J.; Kelley, K.; et al. Nbgrader: A Tool for Creating and Grading Assignments in the Jupyter Notebook. *J. Open Source Educ.* **2019**, *2*, 32. [CrossRef]
28. Web Assign. Available online: https://www.webassign.net/ (accessed on 24 January 2022).
29. We BWorK. Available online: https://webwork.maa.org/ (accessed on 24 January 2022).
30. Raschka, S.; Patterson, J.; Nolet, C. Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information* **2020**, *11*, 193. [CrossRef]

31. Durán-Acevedo, C.M.; Carrillo-Gómez, J.K.; Albarracín-Rojas, C.A. Electronic Devices for Stress Detection in Academic Contexts during Confinement Because of the COVID-19 Pandemic. *Electronics* **2021**, *10*, 301. [CrossRef]
32. González, I.; Portalo, J.M.; Calderón, A.J. Configurable IoT Open-Source Hardware and Software I-V Curve Tracer for Photovoltaic Generators. *Sensors* **2021**, *21*, 7650. [CrossRef] [PubMed]
33. Cagigal, L.; Rueda, A.; Ricondo, A.; Pérez, J.; Ripoll, N.; Coco, G.; Méndez, F.J. Climate-Based Emulator of Distant Swell Trains and Local Seas Approaching a Pacific Atoll. *J. Geophys. Res. Oceans* **2021**, *126*, e2020JC016919. [CrossRef]
34. Python Org. Available online: https://www.python.org/ (accessed on 24 January 2022).
35. Project Jupyter. Available online: https://jupyter.org (accessed on 24 January 2022).
36. Reback, J.; Rockmendel, J.B.; McKinney, W.; den Bossche, J.V.; Augspurger, T.; Cloud, P.; Hawkins, S.; Roeschke, M.; Young, G.F.; Sinhrks; et al. Pandas-Dev/Pandas: Pandas 1.4.1. Available online: https://doi.org/10.5281/zenodo.6053272 (accessed on 24 January 2022).
37. McKinney, W. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2010; pp. 56–61.
38. Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array Programming with NumPy. *Nature* **2020**, *585*, 357–362. [CrossRef]
39. Ipywidgets—Jupyter Widgets 7.6.5 Documentation. Available online: https://ipywidgets.readthedocs.io/en/stable/ (accessed on 24 January 2022).
40. Ipysheet 0.4.4 Documentation. Available online: https://ipysheet.readthedocs.io/en/latest/ (accessed on 24 January 2022).
41. Riverbank Computing PyQt. Available online: https://www.riverbankcomputing.com/software/pyqt/ (accessed on 24 January 2022).
42. O365—Microsoft Graph and Office 365 API Made Easy. Available online: https://github.com/O365/python-o365 (accessed on 24 January 2022).
43. PyPI. The Python Package Index. Available online: https://pypi.org/ (accessed on 24 January 2022).
44. Conda—Conda Documentation. Available online: https://docs.conda.io/en/latest/ (accessed on 24 January 2022).
45. Anaconda Navigator—Anaconda Documentation. Available online: https://docs.anaconda.com/anaconda/navigator/ (accessed on 27 January 2022).
46. Microsoft Azure. Available online: https://azure.microsoft.com/ (accessed on 27 January 2022).