User Manual

Ashkan Shokri and Valentijn Pauwels

November 11, 2021

1 Author Acknowledgement

When using the software, please cite the following paper:

Shokri, A., V. Halupka, and **V.R.N. Pauwels**, A Customized and Automated Assignment Management and Marking System for Evaluating Student Performance in the STEM Disciplines, *Proceedings of the Frontiers in Education 2021, Envisioning Convergence in Engineering Education Conference*, paper number 34672, Lincoln, Nebraska, October 13-16, 2021.

2 Downloading and Installing the Code

The code can be found at: https://github.com/vpauwels/CAAM_Public

To install the software, enter the directory "GAME", and enter the following command:

pip3 install -e .

3 File Structure

When using the software, four directories are needed:

- "GAME": This contains the code. No changes need to be made here, except for when the software needs to be used for the generation of an exam instead of an in-semester assignment. In that case, enter the directory "GAME" in the directory "GAME", and modify the file "texMaker.py". Uncomment line 7 and line 23, and set the variable i_ex to 1 on line 33. Some specific exam variables will need to be changed from line 92 onwards.
- "GAME_database": This contains the questions, wich each question having its own directory.
- "GAME_training": This contains the meta-variables for the assignment. These include the gradebook from moodle, and all the settings for the assignment generation and marking.
- "GAME_output_new": This contains the generated questions, that can be uploaded to moodle.
- "studentResults": This directory is not on github, but will be created when the questions are marked. This directory contains the calculated marks, and the individualized feedback files, which can be uploaded to moodle.

The names of the last four directories can be modified (see Section 4), according to the user's preference.

4 Generating an Assignment

4.1 The Required Files

The first step is to download the gradebook from moodle, since this is needed to generate the individualized questions.

Please be aware that each assignment dropbox has its own gradebook. Thus for each assignment a different gradebook must be used.

Enter this gradebook in the directory "GAME_training/settings". In the same directory you will find the files "SAMPLE.yaml" and "Grades-SAMPLE.csv". The csv file is the gradebook from moodle, and is used for the generation and marking of the sample question. the file "SAMPLE.yaml" contains all the information and settings that are needed to generate and mark the assignments. The information in this file includes:

- "assignmentNum": The number of the assignment, which has to be a numeric value.
- "assignmnetTitle": The title of the assignment. This can be, for example, "Probability", "Hypothesis Testing", ...
- "basePath": The directory where the software is installed.

  Please note that the remainder of the directory and file names in this yaml file need to be listed starting from this directory name.
- "questionDatabaseDirectory": The directory where the questions are stored.
- "questionList": The questions that have to be used in the assignment. For each question, enter a "-" and then the question name on one line.

  Please note that these have to be named AxQy, with x the assignment number (equal to the variable "assignmentNum"), and y the question number. x and y must thus be numeric values.
- "assginmentMasterDirectory": The directory where the assignments will be stored.
- "resultDirectory": The directory where the student answers, downloaded from moodle, will be placed. If this directory does not exist, it will be generated by the software.
- "studentIDListFile": The name of the gradebook on moodle.
- "marksOutputPath": The file where the marks of the students will be stored.
- "assginmentSubsetDirectory": This contains specific information regarding the assignments.
  - "enabled": Leave this set as "True".
  - "suffix": For each assignment, the code will generate two directories. A first with all the data and files, and a second with only the files that need to be uploaded to moodle. This variable will be added to the first directory.
  - "filesToBePassed": This is a list of the files that have to be passed to moodle, that are different for each student. In this example this is only the pdf with the assignment. In some cases this may also be the random numbers that the students need to work with.

  - "staticFiles": This is a list of the files that are the same for each student. In most cases this will be the Graphical User Interfaces, and sometimes a file with further information. These files can be placed in the subdirectory "files" in the directory for each question.

- "feedbackSubsetDirectory": This contains specific information regarding the assignments. The explanation of these variables is the same as for the variable "assginmentSubsetDirectory".

When entering the directory names, **please make sure the names end with a "/" (when working with Linux) or a "\" (when working under MS Windows).**

4.2 Programming a Question

The questions are in the directory "GAME_training". For the sample question, enter the subdirectory "A0Q0". The question needs to be entered in the file "text_maker.py". Three subroutines need to be modified:

- ""input_maker": This subroutine generates the random numbers that are used in the questions. In this case the choise was made to store these variables in the pandas data frame, but this is no obligation. A user may choose to store these values in a vector or matrix, depending on the situation. However, a number of requirements must be met:

  1. The format has to be consistent with the subroutine where the inputs are saved (see the next item).
  2. The format has to be consistent with the subroutine where the question is generated (see the second next item).
  3. The format has to be consistent with the subroutine where the inputs are read in to mark the question (see Section 5).

- "save_inputs": This subroutine saves the random numbers so they can be read in when marking the question (see Section 5). As stated above, the user can use any format of choice, but it must be consistent with the subroutine where the inputs are read in to mark the question. The name of the file with the inputs will be "input_ID_Ax_AxQy.csv". ID is the student ID number, x the assignment number, and y the question number.

- "tex_maker": This subroutine generates the LaTeXstring for the specific question. The user can generate this string by any method of choice, as long as the final string is stored as ['The LaTeXstring.'].

4.3 Preparing a Graphical User Interface

4.3.1 Requirements

The Graphical User Interfaces should be prepared in the subdirectory "result_maker". The user can modify the existing html file, or use a different system to allow the students to save their results. There are only two requirements:

- The format in which the Graphical User Interface saves the result must be consistent with the subroutine that reads in the results to mark the question (see Section 5).

- The name of the file with the results must be "answer_ID_Ax_AxQy.csv". ID is the student ID number, x the assignment number, and y the question number.

4.3.2 Preparing a html Graphical User Interface

In the subroutine "GUI" there two codes can be found that provide a straightforward way to generate a Graphical User Interface. Note that the software needs to be installed before these codes can be used.

- structureMaker.py This code generates a file "structure.yaml" based upon which the html Graphical User Interface can be generates. A number of elements can be added for each row. The first row generates a vertical line. In the example, the second line generates a number of cells into which a number can be entered. The four entries in the subroutine gui.addElement() are:

  1. The name of the variable that will be read in by the marker.
  2. "textbox" for entering a number, "dropDown" for a multiple choice question.
  3. The row and column numbers. The text must be entered in odd column numbers.
  4. The text that will be written in the cells.

- make_gui.py This code uses the generated structure.yaml and generates the html Graphical User Interface. At the end of the code, the user must change the name of this Graphical User Interface, and the assignment and question numbers.

4.4 Preparing the Directories With the Files for the Students

When the questions are programmed, enter the "GAME_training" directory, and enter the command

python3 newGenerator.py settings/SAMPLE.yaml

This will generate the directory with the files for all the students (the pdf with the assignment, the Graphical User Interface, the randon numbers if the students need them, and any other required files). If this directory already exists, it will overwrite any files that are already in the directory.

When compiling user-generated assignments, simply replace "settings/SAMPLE.yaml" by the name of the yaml file with the information about the assignment.

4.5 Uploading the Assignments to Moodle

In the directory "GAME_output_new" there will be two generated directories: "SAMPLE" and "SAMPLE_to_moodle". As stated above, the first contains all the files that are needed to generate and mark the assignments, and the second contains all the files that will be passed to moodle. In each of these directories there will be a subdirectory with the files for each student. Enter the "SAMPLE_to_moodle" directory and compress all the subdirectories into one .zip file.

On moodle, enter the assignment dropbox. Make sure the following settings are chosen:

- Submission types: File submissions is checked.
- Accepted file types: .zip,.pdf,spreadsheet,.html
- Feedback types: All four options are checked.
- Use marking workflow: No. If "Yes" is chosen for this option, the students will not be able to see their files for the assignment.

Click "View all submissions". For "Grading action", choose "Upload multiple feedback files in a zip". Upload the .zip file that you generated. Depending on the amount of students this may take a while (for 400 students this can take a couple of minutes). When the files are uploaded, click "Confirm". After this, the students will be able to download the files from the feedback section in the assignment dropbox.

5 Marking an Assignment

5.1 Downloading the Student Results

When the students have uploaded the csv files with their results, enter the assignment dropbox, click "View all submissions", and choose as "Grading action" "Download all submissions". You will see that a directory "studentResults" has been generated, with a subdirectory "SAMPLE_submission". Again "SAMPLE" can be replaced by the assignment name chosen by the user. Enter the directory "SAMPLE_submission" and move the .zip file with the results here, and unzip this file. You can remove the .zip file, as you will not be needing it anymore.

5.2 Programming a Marker for a Question

Each question is marked using the program "marking.py" in the question subdirectory in the directory "GAME_database". Three subroutines must be modified here:

- "input_loader": This subroutine reads in the random numbers that have been generated in Section 4.2, and that are used to generate the assignment. In this case, these numbers are read in using the pandas datastructure. This is not necessary, as long as the numbers are read in consistent with the way they are saved in Section 4.2.
- "result_loader": This subroutine reads in the results generated by the Graphical User Interfaces. Again, in this case the pandas data frame is used, but the user can use any method, as long as it is consistent with the way the results are saved by the Graphical User Interfaces (Section 4.3). If a file is not found, an empty variable will be returned.
- "marker": This subroutine calculates the correct results, and compares them to the results calculated by the students. The user can code this whatever way suits best, as long as one string is returned. In this subroutine also the final mark is calculated (the variable mark_p). Important is that a perfect answer to the question is a total mark of 1.

One subroutine that may be of interest is "check_str". This subroutine may be used with any answer that should be a numerical value. If the students enter a non-numerical value, this subroutine will return -9999, and the answer will thus automatically be marked as wrong.

5.3 Marking the Assigment

To mark the assignment, again enter the directory "GAME_training", and enter the command

python3 newMarker.py settings/SAMPLE.yaml

This will generate the subdirectory "SAMPLE_submission_feedback_to_moodle" in the directory "studentResults". Again, the name "SAMPLE" can be modified into the assignment name chosen by the user.

5.4 Uploading the Feedback Files to Moodle

This is done in exactly the same way as in Section 4.5. Enter the directory "SAMPLE_submission_feedback_to_moodle" and follow the same steps as in Section 4.5.

6 Contact

For any questions regarding the software, please contact Valentijn.Pauwels@monash.edu