System/Application: Simple Test Paper Marking Tool Based on LLM

Part 2: Coding & Testing | Individual Submission

Module Name: ModuleSaveMarkingResultToExcel, ModuleCreateExcelOfTestingHistory

Name & ID: James, Li, s23236

Due date: 2025/10/27 Week 11, Monday

Teacher: John Barton

Welcome to our project!

This brief intro serves as an aid to read our code and test our entire program and its several modules.

The newest version of our project can always be found at:

<https://github.com/xzxui/CSSillyProject>

Our project is divided into modules in a manner that limits all user input inside the ModuleMainLoop.py, which is the GUI for our program, and this module shall passes these input into other modules when they are called. Therefore, only ModuleMainLoop has instructions for user input, so when grading modules other than ModuleMainLoop, the documentation of the arguments of the functions might be interpreted as the instruction for input.

On the other hand, the output of our program are excel files that the user can directly read and messages shown to the user in the GUI.

To build the environment, a ‘requirements.txt’ is provided. Simply run pip install -r requirements.txt in cmd. Alternatively, use command line and run ./venv/Scripts/activate.bat or use powershell and run ./venv/Scripts/Activate.ps1, and use that command line/powershell window to do all the testing.

To run our project, simply type ‘python ModuleMainLoop.py’ in cmd or powershell, and open the url as instructed.

**Program Code**

|  |
| --- |
| ModuleSaveMarkingResultToExcel.py |
| import configs  import ModuleCreateExcelOfTestingHistory  import openpyxl  def SaveMarkingResultToExcel(save\_path:str, syllabus\_code:str, component\_number:str, marking\_report:list, strengths:str, weaknesses:str, score:int, total\_score\_avail, grade:str, update\_summary=True):  """  Args:  1. save\_path (<class 'str'>): the path of the excel file to save to  2. syllabus\_code (<class 'str'>)  3. component\_number (<class 'str'>)  4. marking\_report (<class 'list'>): the marking report  5. strengths (<class 'str'>): the AI's comment on areas the student did well  6. weaknesses (<class 'str'>): the AI's comment on areas the student did bad  7. score (<class 'int'>)  8. total\_score\_avail (<class 'int'>)  9. grade (<class 'str'>): the grade received, e.g. A, B, C, D, E, U  10. update\_summary (<class 'bool'>): whether to update the excel of testing history  Return: No return  Process:  Save the marking result into an excel file, and call ModuleCreateExcelOfTestingHistory.CreateExcelOfTestingHistory  """  wb = openpyxl.Workbook()  ws = wb.active  # Fill in the first row and some of the second columns  ws["A1"] = "Question Number"  ws["B1"] = "Maximum Mark Possible"  ws["C1"] = "Marks Received"  ws["I1"] = "Syllabus Code"  ws["I2"] = syllabus\_code  ws["J1"] = "Component Number"  ws["J2"] = component\_number  ws["K1"] = "Strengths of the Student"  ws["K2"] = strengths  ws["L1"] = "Weaknesses of the Student"  ws["L2"] = weaknesses  ws["M1"] = "Total Marks Attained"  ws["M2"] = score  ws["N1"] = "Maximum Total Marks Available"  ws["N2"] = total\_score\_avail  ws["O1"] = "Grade Achieved"  ws["O2"] = grade  # Fill in the rest  row = 2  for question\_num, max\_mark, mark in marking\_report:  ws["A"+str(row)] = question\_num  ws["B"+str(row)] = max\_mark  ws["C"+str(row)] = mark  row += 1  # Save  wb.save(save\_path)  print("Marking result saved to " + save\_path, ".")  # Update excel of testing history  print("Updating the excel file of testing history")  ModuleCreateExcelOfTestingHistory.CreateExcelOfTestingHistory()  if \_\_name\_\_ == "\_\_main\_\_":  SaveMarkingResultToExcel(  'test\_folder/ModuleSaveMarkingResultToExcel/marking\_result\_21.xlsx','0917',  '21' ,  [  ['1', 5, 4],  ['2(a)', 10, 4],  ['2(b)(i)', 5, 4],  ['2(b)(ii)', 5, 5]  ] ,  'Good at algebra',  'Not good at geometry',  17,  25,  'B',  update\_summary=False,  )  SaveMarkingResultToExcel(  'test\_folder/ModuleSaveMarkingResultToExcel/marking\_result\_22.xlsx','0917',  '22' ,  [  ['1', 5, 4],  ['2(a)', 15, 4],  ['2(b)', 10, 8],  ['3', 7, 5]  ] ,  'Good at calculating the right answer',  'Not good at showing the work',  37,  21,  'C',  update\_summary=False,  )  SaveMarkingResultToExcel(  'test\_folder/ModuleSaveMarkingResultToExcel/marking\_result\_23.xlsx','0917',  '23' ,  [  ['1', 5, 4],  ['2', 11, 4],  ['3(a)', 5, 4],  ['3(b)', 7, 5]  ] ,  'Good at solving problems quickly',  'Not good at handwritting',  28,  17,  'B',  update\_summary=False,  ) |

|  |
| --- |
| ModuleCreateExcelOfTestingHistory.py |
| import configs  import openpyxl  from pathlib import Path  def CreateExcelOfTestingHistory(marking\_result\_folder\_override=None, output\_override=None):  """  Args:  1. marking\_result\_folder\_override: optional argument of type string, pretty much explains itself  2. output\_override: optional argument of type string, pretty much explains itself  Return:  No return  Process:  Create an excel file about the student's testing history, showing the score and grade achieved in each paper, and the AI's comment on the performance of the student, and save to configs.path\_to\_excel\_of\_testing\_history. Notice that the path of the folder containing all the student's practice paper history is stored in configs.marking\_result\_folder. Every single file ending with 'xlsx' under that folder should be a marking result save.  """  # Create a workbook  wb = openpyxl.Workbook()  ws = wb.active  # Find all the marking results  marking\_result\_folder = marking\_result\_folder\_override or configs.marking\_result\_folder  xlsx\_files = Path(marking\_result\_folder).glob('\*.xlsx')  # Add all the information needed into the excel of testing history  rows = [[]]  # first = [] it is not declared outside because this way when the variable is not created by the iteration, a fatal error occurs in our favor  for file\_path in xlsx\_files:  print(f"Reading: {file\_path}")  new, first = DetermineRowsToAdd(file\_path)  rows.append(new)  rows[0] = first  print(rows)  for row in rows:  ws.append(row)  print(f"Added a total of {len(rows)} rows")  # Save the excel  wb.save(output\_override or configs.path\_to\_excel\_of\_testing\_history)  print(f"Excel of testing history saved to {output\_override or configs.path\_to\_excel\_of\_testing\_history}")  def DetermineRowsToAdd(path\_to\_excel, start\_col=9, end\_col=15):  # Load the marking result  wb\_obj = openpyxl.load\_workbook(str(path\_to\_excel))  sheet\_obj = wb\_obj.active  new\_row = [path\_to\_excel.name]  override\_row = ["Exam Paper Name"] # This is the first row of the final excel file saved by this module  for col in range(start\_col, end\_col+1):  new\_row.append(sheet\_obj.cell(row=2, column=col).value)  override\_row.append(sheet\_obj.cell(row=1, column=col).value)  print(new\_row, override\_row)  return new\_row, override\_row  # For testing  if \_\_name\_\_ == "\_\_main\_\_":  CreateExcelOfTestingHistory(  marking\_result\_folder\_override="test\_folder/ModuleCreateExcelOfTestingHistory/history1/",  output\_override="test\_folder/ModuleCreateExcelOfTestingHistory/excel\_of\_testing\_history1.xlsx"  )  CreateExcelOfTestingHistory(  marking\_result\_folder\_override="test\_folder/ModuleCreateExcelOfTestingHistory/history2/",  output\_override="test\_folder/ModuleCreateExcelOfTestingHistory/excel\_of\_testing\_history2.xlsx"  )  print("Check test\_folder/ModuleCreateExcelOfTestingHistory/excel\_of\_testing\_history.xlsx for the output") |

**Test Plan**

Test Plan for ModuleSaveMarkingResultToExcel.py

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Content | Test Data(input) | Purpose | Notes for Clarification | Expected Outcome | Actual Outcome | Test passed |
| SaveMarkingResultToExcel | SaveMarkingResultToExcel('test\_folder/ModuleSaveMarkingResultToExcel/marking\_result\_21.xlsx','0917', '21' , [ ['1', 5, 4], ['2(a)', 10, 4], ['2(b)(i)', 5, 4], ['2(b)(ii)', 5, 5] ] , 'Good at algebra', 'Not good at geometry', 17, 25, 'B', update\_summary=False) | To check if the function works as expected | This is the first set of inputs | ‘test\_folder/ModuleSaveMarkingResultToExcel/marking\_result\_21.xlsx’ saved | Same! | Yes |
| SaveMarkingResultToExcel | SaveMarkingResultToExcel( 'test\_folder/ModuleSaveMarkingResultToExcel/marking\_result\_22.xlsx','0917', '22' , [ ['1', 5, 4], ['2(a)', 15, 4], ['2(b)', 10, 8], ['3', 7, 5] ] , 'Good at calculating the right answer', 'Not good at showing the work', 37, 21, 'C', update\_summary=False) | To check if the function works as expected | This is the second set of inputs | ‘test\_folder/ModuleSaveMarkingResultToExcel/marking\_result\_22.xlsx’ saved | Same! | Yes |
| SaveMarkingResultToExcel | SaveMarkingResultToExcel( 'test\_folder/ModuleSaveMarkingResultToExcel/marking\_result\_23.xlsx','0917', '23' , [ ['1', 5, 4], ['2', 11, 4], ['3(a)', 5, 4], ['3(b)', 7, 5] ] , 'Good at solving problems quickly', 'Not good at handwritting', 28, 17, 'B', update\_summary=False) | To check if the function works as expected | This is the third set of inputs | ‘test\_folder/ModuleSaveMarkingResultToExcel/marking\_result\_23.xlsx’ saved | Same! | Yes |

**Steps to reproduce**

1. Run ‘python ModuleSaveMarkingResultToExcel.py’, and the three tests are done at the same time
2. Check test\_folder/ModuleSaveMarkingResultToExcel/ for the outputs

Test Plan for ModuleCreateExcelOfTestingHistory

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Content | Test Data(input) | Purpose | Notes for Clarification | Expected Outcome | Actual Outcome | Test passed |
| Test CreateExcelOfTestingHistory | CreateExcelOfTestingHistory( marking\_result\_folder\_override="test\_folder/ModuleCreateExcelOfTestingHistory/history1/", output\_override="test\_folder/ModuleCreateExcelOfTestingHistory/excel\_of\_testing\_history1.xlsx" ) | Check if the function works as expected | history1 has opposite strengths and weaknesses to history2 | Excel file saved to ‘.\test\_folder\ModuleCreateExcelOfTestingHistory\ excel\_of\_testing\_history1.xlsx’, with the strengths and weaknesses being the weaknesses and strengths of ‘. \test\_folder\ModuleCreateExcelOfTestingHistory\ excel\_of\_testing\_history2.xlsx’’ respectively | Same as expected | Yes |
| Test CreateExcelOfTestingHistory | CreateExcelOfTestingHistory( marking\_result\_folder\_override="test\_folder/ModuleCreateExcelOfTestingHistory/history2/", output\_override="test\_folder/ModuleCreateExcelOfTestingHistory/excel\_of\_testing\_history2.xlsx" ) | Check if the function works as expected | history2 has opposite strengths and weaknesses to history1 | Excel file saved to ‘.\test\_folder\ModuleCreateExcelOfTestingHistory\ excel\_of\_testing\_history2.xlsx’, with the strengths and weaknesses being the weaknesses and strengths of ‘. \test\_folder\ModuleCreateExcelOfTestingHistory\ excel\_of\_testing\_history1.xlsx’’ respectively | Same as expected | Yes |

Steps to reproduce:

1. Simply type in ‘python ModuleCreateExcelOfTestingHistory.py’ in cmd or powershell. The output of the two tests will be saved, and could be found at ‘.\test\_folder\ModuleCreateExcelOfTestingHistory\ excel\_of\_testing\_history1.xlsx’ and ‘. \test\_folder\ModuleCreateExcelOfTestingHistory\ excel\_of\_testing\_history2.xlsx’