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

Part 2: Coding & Testing | Individual Submission

Module Name: ModuleProduceFeedbackForStudent, ModulePDF2b64s

Name & ID: Lin, Lee, s25844

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**

|  |
| --- |
| ModuleProduceFeedbackForStudent.py |
| Inserting image...import configs  import pydantic  import openpyxl  import json  import time  import ModuleLLMQuery  class Comment(pydantic.BaseModel):  detailed\_comment\_on\_student\_performance: str  custom\_error: str = pydantic.Field(..., description="explanation for any fatal error you want to raise. unless a fatal error is what you want to raise, leave this field empty")  def history\_to\_json(p2e):  # Read history excel file  path\_to\_history = p2e  wb\_obj = openpyxl.load\_workbook(path\_to\_history)  sheet\_obj = wb\_obj.active  # Find all the names of the columns  keys = []  right\_most\_col = 0  col = 1  while True:  val = sheet\_obj.cell(row=1, column=col).value  if val:  keys.append(val)  else:  break  right\_most\_col += 1  col += 1  # Find all the values under each column, generating a 'dicts'  # 'dicts' could look like the following example  # dicts = [{"col1":11, "col2":21}, {"col1":12, "col2":22}]  dicts = []  row = 2  while True:  dicts.append({})  all\_empty = True  for col in range(1, right\_most\_col+1):  value = sheet\_obj.cell(row=row, column=col).value  if value:  all\_empty = False  dicts[-1][keys[col-1]] = sheet\_obj.cell(row=row, column=col).value  if all\_empty:  break  row += 1  # Return with json's dumps, making it AI-readable  return json.dumps(dicts, indent=4)  def ProduceFeedbackForStudent(p2e=configs.path\_to\_excel\_of\_testing\_history):  """  Args:  1. p2e: of <class 'str'>  Return:  of <class 'str'>, a summary of a student's performance  Process:  use the information stored in an excel file containing the history of tests that the student has taken to ask an AI to give comment  """  print("Fetching history")  history\_in\_json = history\_to\_json(p2e)  print("Done fetching, now asking AI to give comments")  before = time.time()  comment = ModuleLLMQuery.LLMQuery(  [  {"role": "system", "content": "You are a responsible and experienced teacher who is giving comments on a student's recent performance on exam papers done for practice, and are here to provide a detailed summary of the student's strengths and areas for improvements."},  {"role": "user", "content": "Provided is the recent performance of the student on practice exam papers, in the format of json:"+history\_in\_json}  ],  response\_format=Comment,  model="gpt-5-mini"  )  print(f"AI responded, took {time.time()-before}s")  # Allowing the AI to determine edge cases  if comment.custom\_error:  raise RuntimeError(f"The AI raised an error: {comment.custom\_error}")  print("Comment Produced!")  return comment.detailed\_comment\_on\_student\_performance  if \_\_name\_\_ == "\_\_main\_\_":  with open("test\_folder/ModuleProduceFeedbackForStudent/comment1.txt", 'w') as f:  f.write(ProduceFeedbackForStudent(p2e="test\_folder/ModuleProduceFeedbackForStudent/excel\_of\_testing\_history1.xlsx"))  with open("test\_folder/ModuleProduceFeedbackForStudent/comment2.txt", 'w') as f:  f.write(ProduceFeedbackForStudent(p2e="test\_folder/ModuleProduceFeedbackForStudent/excel\_of\_testing\_history2.xlsx")) |

|  |
| --- |
| ModulePDF2b64s.py |
| import configs  import pymupdf  import base64  def PDF2b64s(pdf\_path):  """  Args:  1. pdf\_path (<class 'str'>)  Return:  an instance of <class 'list'>, each element being a <class 'str'>, which is a base 64 image converted from a page in the pdf  Process:  Convert the pdf into base 64 images, and the base 64 images should be stored in a bunch of <class 'str'>  """  doc = pymupdf.open(pdf\_path)  b64\_imgs = []    for page\_num in range(len(doc)):  page = doc.load\_page(page\_num)  # set to better resolution for better OCR by AI  mat = pymupdf.Matrix(1.5, 1.5) # scaling to make it clearer  pix = page.get\_pixmap(matrix=mat)  img\_data = pix.tobytes(configs.img\_extension)  b64\_imgs.append(base64.b64encode(img\_data).decode('utf-8'))    doc.close()  return b64\_imgs  # For testing  if \_\_name\_\_ == "\_\_main\_\_":  print("Converting")  b64\_imgs = PDF2b64s("test\_folder/ModulePDF2b64s/original1.pdf")  print("Done converting")  with open("test\_folder/ModulePDF2b64s/last\_page2.png", "wb") as f:  f.write(base64.b64decode(b64\_imgs[-1]))  print("Last page saved")  with open("test\_folder/ModulePDF2b64s/first\_page2.png", "wb") as f:  f.write(base64.b64decode(b64\_imgs[0]))  print("First page saved\n")    print("Converting")  b64\_imgs = PDF2b64s("test\_folder/ModulePDF2b64s/original2.pdf")  print("Done converting")  with open("test\_folder/ModulePDF2b64s/last\_page2.png", "wb") as f:  f.write(base64.b64decode(b64\_imgs[-1]))  print("Last page saved")  with open("test\_folder/ModulePDF2b64s/first\_page2.png", "wb") as f:  f.write(base64.b64decode(b64\_imgs[0]))  print("First page saved\n")  print(b64\_imgs[-1]) |

**Test Plan**

Test Plan for ModuleProduceFeedbackForStudent

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Content | Test Data(input) | Purpose | Notes for Clarification | Expected Outcome | Actual Outcome | Test passed |
| Run ProduceFeedbackForStudent with excel\_of\_testing\_history1.xlsx | ‘./test\_folder/ModuleProduceFeedbackForStudent/ excel\_of\_testing\_ history1.xlsx’ | To check if the function works as expected | Because history1 and history2 have opposite strengths and weaknesses(i.e. the strengths in history1 and weaknesses in history2, vice versa), comment1 and comment2 should be the opposite as well | ‘comment1.txt’ and ‘comment2.txt’ are largely the opposite of each other | ‘comment1.txt’ and ‘comment2.txt’ are indeed the opposite | Yes |
| Run ProduceFeedbackForStudent, with excel\_of\_testing\_history2.xlsx | ‘./test\_folder/ModuleProduceFeedbackForStudent/ excel\_of\_testing\_history2.xlsx’ | To check if the function works as expected | Because history1 and history2 have opposite strengths and weaknesses(i.e. the strengths in history1 and weaknesses in history2, vice versa), comment1 and comment2 should be the opposite as well | ‘comment1.txt’ and ‘comment2.txt’ are basically the opposite of each other | ‘comment1.txt’ and ‘comment2.txt’ are indeed the opposite | Yes |
| Test the entire project by inputting the question paper, marking scheme and grading threshold table from test\_folder/data | ‘./test\_folder/data/’ | To check if the function can work with the other modules correctly, as well as if it copies the information from marking\_results correctly | The student’s second try answered more questions | Information appearing correctly at the bottom of the grading result GUI. | Information appeared correctly in the grading result section of GUI. | Yes |

**Steps to reproduce**

For the first test:

1. Run ModuleProduceFeedbackForStudent.py
2. Open test\_folder/ModuleProduceFeedbackForStudent/comment1.txt
3. Open test\_folder/ModuleProduceFeedbackForStudent/comment2.txt
4. Compare the two comments, see if they’re the opposite

For the second test:

1. Run ModuleMainLoop.py
2. Open the URL as instructed by ModuleMainLoop
3. Upload the question paper from ‘test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_qp\_first\_try.pdf’
4. Upload the marking scheme from ‘test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_ms.pdf
5. Upload the threshold table from ‘test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_tt.pdf’
6. Click on the orange button saying ‘Start Processing’
7. Wait for 5 to 10 minutes for the processing to complete
8. Briefly read the grading results
9. Re-upload the question paper from ‘test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_qp\_first\_try.pdf’ to replace ‘test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_qp\_first\_try.pdf’
10. Scroll to the bottom of the grading result and check if the comment is there

Test Plan for ModulePDF2b64s

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Content | Test Data(input) | Purpose | Notes for Clarification | Expected Outcome | Actual Outcome | Test passed |
| Run PDF2b64s with original1.pdf as input and output the first and last pages of the pdf to png files | ‘./test\_folder/ModulePDF2b64s/original1.pdf’ | Check if the function works as expected | Doing this test will first convert the pdf file to base 64 images stored in strings, and then to png files | First page of ‘original1.pdf’ stored in './test\_folder/ModulePDF2b64s/first\_page1.png’,  Last page of ‘original1.pdf’ stored in './test\_folder/ModulePDF2b64s/last\_page1.png’ | Same as expected | Yes |
| Run PDF2b64s with original2.pdf as input and output the first and last pages of the pdf to png files | ‘./test\_folder/ModulePDF2b64s/original2.pdf’ | Check if the function works as expected | Doing this test will first convert the pdf file to base 64 images stored in strings, and then to png files | First page of ‘original1.pdf’ stored in './test\_folder/ModulePDF2b64s/first\_page2.png’  Last page of ‘original2.pdf’ stored in './test\_folder/ModulePDF2b64s/last\_page2.png’ | Same as expected | Yes |
| Run PDF2b64s with original2.pdf as input and output the base 64 image as string | ‘./test\_folder/ModulePDF2b64s/original2.pdf’ | Check if the function works as expected, which is helpful for revealing problems the above did not reveal | The output of this test is not converted back into png, but the base 64 image decoded in ‘utf-8’ to string | A very, very long string appears | Same as expected | Yes |

Steps to reproduce:

1. Simply type in ‘python ModulePDF2b64s.py’ in cmd or powershell. The output of the first two tests will be saved, and the output of the third test will be directly printed in the command line/powershell.