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

Part 2: Coding & Testing | Individual Submission

Module Name: ModuleProduceFeedbackForStudent, ModulePDF2b64s

Name & ID: Lin, Lee, s23043

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

Teacher: John Barton

**Program Code**

|  |
| --- |
| *ModuleProduceFeedbackForStudent.py* |
| **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/original.pdf")**  **print("Done converting")**  **with open("test\_folder/ModulePDF2b64s/last\_page.png", "wb") as f:**  **f.write(base64.b64decode(b64\_imgs[-1]))**  **print("Last page saved")**  **with open("test\_folder/ModulePDF2b64s/first\_page.png", "wb") as f:**  **f.write(base64.b64decode(b64\_imgs[0]))**  **print("First page saved")** |

**Test Plan**

*The*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |