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

Part 2: Coding & Testing | Individual Submission

Module Name: ModuleMainLoop

Name & ID: Joseph, Fu, s23043

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, and open the url as instructed.

**Program Code**

|  |
| --- |
| ModuleMainLoop.py |
| import configs  import gradio as gr  import os  from pathlib import Path  from typing import Tuple, Dict, List  # Import required modules  import ModuleMarkPaper  import ModuleProduceFeedbackForStudent  # Validate uploaded file format  def validate\_file\_format(file\_path: str) -> Tuple[bool, str]:  valid\_extensions = {".pdf"}  file\_ext = Path(file\_path).suffix.lower()  if file\_ext not in valid\_extensions:  return False, f"Unsupported format ({file\_ext}), only {valid\_extensions} are allowed"  return True, ""  # Main processing function (integrated new module calls)  def process\_submission(paper\_file: str, answer\_file: str, threshold\_file: str, progress=gr.Progress()) -> Tuple[str, str, str]:  error\_msg = ""  # Check if files are fully uploaded  if not paper\_file or not answer\_file or not threshold\_file:  error\_msg = "Please upload the exam paper, marking scheme and grading threshold table"  return "", error\_msg  # Validate file formats  for file\_path in [paper\_file, answer\_file, threshold\_file]:  valid, msg = validate\_file\_format(file\_path)  if not valid:  error\_msg = f"File format validation failed: {msg}"  return "", error\_msg  progress(0.05, desc="Starting file processing...")  progress(0.1, desc="Grading in progress...")  try:  # Call marking module  score, max\_score, grade, pros, cons = ModuleMarkPaper.MarkPaper(paper\_file, answer\_file, threshold\_file)  progress(0.8, desc="Generating feedback...")  # Call feedback generation module  comment\_based\_on\_history = ModuleProduceFeedbackForStudent.ProduceFeedbackForStudent()  except Exception as e:  error\_msg = f"Processing failed: {str(e)}"  return "", error\_msg  progress(0.95, desc="Organizing results...")  # Format grading result  result\_str = (  f"Grading completed!\n"  f"The marking results for completed question papers that you've submitted may be found at {configs.marking\_result\_folder}\n6. A summary of all testing records may be found at {configs.path\_to\_excel\_of\_testing\_history}\n"  f"Exam Paper: {os.path.basename(paper\_file)}\n"  f"Reference Answer: {os.path.basename(answer\_file)}\n"  f"Score: {score}/{max\_score}\n"  f"Grade: {grade}\n"  f"Strengths: {pros}\n"  f"Weaknesses: {cons}\n"  f"Feedback based on History: {comment\_based\_on\_history}\n"  )  progress(1.0, desc="Processing completed")  return result\_str, error\_msg  # Create Gradio interface  def create\_gui():  with gr.Blocks(title="Automatic Exam Paper Grading System") as demo:  gr.Markdown("# 📝 Automatic Exam Paper Grading System")  gr.Markdown(  "Please upload exam paper, marking scheme and grading threshold table (supports pdf). The system will automatically grade and generate feedback.")  with gr.Row():  # Left input area  with gr.Column(scale=1):  paper\_input = gr.File(label="Upload Completed Question Paper", file\_types=[".pdf"])  answer\_input = gr.File(label="Upload Marking Scheme", file\_types=[".pdf"])  threshold\_input = gr.File(label="Upload Grading Threshold Table", file\_types=[".pdf"])  process\_btn = gr.Button("Start Processing", variant="primary")  # Right output area  with gr.Column(scale=2):  result\_output = gr.Textbox(label="Grading Result", lines=8)  error\_output = gr.Textbox(label="Error Message", lines=2, interactive=False)  # Bind button event  process\_btn.click(  fn=process\_submission,  inputs=[paper\_input, answer\_input, threshold\_input],  outputs=[result\_output, error\_output]  )  # Instructions  gr.Markdown("""  ### Instructions  1. Upload the student's completed question paper, marking scheme and grading threshold table  2. Click the "Start Processing" button. This should take fewer than 10 minutes.  3. The system will validate file formats, and call the grading and feedback generation modules  4. Results will be displayed in the right area  5. The marking results for completed question papers that you've submitted may be found at """+configs.marking\_result\_folder+"""  6. A summary of all testing records may be found at """+configs.path\_to\_excel\_of\_testing\_history+"""  Note: Grading logic is provided by ModuleMarkPaper, and feedback content is generated by ModuleProduceFeedbackForStudent  """)  return demo  if \_\_name\_\_ == "\_\_main\_\_":  gui = create\_gui()  gui.launch(debug=True) |

**Test Plan**

Test Plan for ModuleProduceFeedbackForStudent

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Content | Test Data(input) | Purpose | Notes for Clarification | Expected Outcome | Actual Outcome | Test passed |
| Run ModuleMainLoop.py and check the localhost server it should creates | None | To check if the function works as expected to create a localhost server |  | Typing in the url shown by the instructions outputted by ModuleMainLoop into the browser and see the webpage being shown correctly | As expected | Yes |
| Run the GUI and click on Start Processing without having all files required uploaded | ‘./test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_qp\_first\_try.pdf’ | To check if the function works as expected when handling such edge case | Normally, all three of the question paper, the marking scheme and the grading threshold table have to be uploaded to start processing, meaning that this test should cause an error | Error msg prompting the user to upload all the files before starting to process | As expected | Yes |
| Run the GUI and click on Start Processing with all files required uploaded | ‘./test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_qp\_first\_try.pdf’  ‘./test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_ms.pdf’  ‘./test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_tt.pdf’  ‘./test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_qp\_second\_try.pdf’ | To check if the function works as expected when regular inputs are given | The student has did the question paper for two times, and therefore needs to click on processing for two times | Information appearing correctly at the bottom of the grading result GUI, files saved correctly to ‘./history/’ and ‘./summary/’ | As expected | Yes |

**Steps to reproduce**

1. Run ‘python ModuleMainLoop.py’ in cmd/powershell
2. Open the URL given by the program as prompted in a browser
3. Check if all the elements are there, and hence get the result for the first test
4. Upload ‘./test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_qp\_first\_try.pdf’ as completed question paper
5. Click on ‘Start Processing’
6. Check if an error message is outputted, and hence get the result for the second test
7. Upload ‘./test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_ms.pdf’ as marking scheme
8. Upload ‘./test\_folder/data/9709\_12\_2024\_MayJune\_Mathematics\_tt.pdf’ as grading threshold table
9. Click on ‘Start Processing’
10. Wait for 5~10 minutes for the processing to be done and grading result to appear
11. Check the grading result and check the files saved under ./history and ./summary for the result of the third test