DeepSeek Email Classification & OCR - Test Steps

1. Environment Setup

- Ensure the Python environment is set up with required dependencies.
- Activate the virtual environment (if applicable):
- source venv/bin/activate (Linux/Mac)
- venv\Scripts\activate (Windows)
- Install dependencies if not already installed:
- pip install -r requirements.txt

2. Running the API

- Navigate to the API script location:
- cd scripts
- Start the FastAPI server using Uvicorn:
- uvicorn api:app --reload
- Verify that the server is running at http://127.0.0.1:8000/docs.

3. Preparing Test Data

- Collect sample .eml and .msg files representing different request types.
- Ensure that some test files contain attachments (PDFs, images) for OCR testing.

4. Uploading Emails for Classification via API

- Use Postman or CURL to send a request to the API endpoint:
- curl -X 'POST' \
- 'http://127.0.0.1:8000/classify-email' \
- -H 'accept: application/json' \
- -H 'Content-Type: multipart/form-data' \
- -F 'file=@sample_email.eml'
- Verify that the response includes a classified request type and extracted details.

5. Testing OCR Extraction via API

- Submit emails with PDF or image attachments.
- Confirm extracted text from images/PDFs is included in the response.

6. Validating API Responses

- Check classification accuracy against expected request types.
- Ensure extracted details (amount, date, deal name) are correctly identified.
- Log results in a CSV file for analysis.

7. Testing api.py End-to-End

- Run the API and upload test emails.
- Check logs and ensure proper processing of .eml and .msg files.
- Verify OCR extraction and classification outputs.
- Test error handling for unsupported file types and incorrect formats.

8. Performance & Error Handling Tests

- Test handling of unsupported file formats.
- Assess response time for various email sizes.
- Verify API stability with multiple concurrent requests.

9. Logging & Exporting Results

- Collect API responses and store them in test_results.csv.
- Review and analyze the accuracy of classification and extraction.

10. Model Fine-Tuning Validation

- Run the fine-tune script using:
- python finetune.py
- Re-test classification accuracy after fine-tuning.
- Ensure the newly trained model is used in model.py.

11. Final Review & Documentation

- Verify all functionalities work as expected.
- Update documentation with any additional findings or improvements needed.