Name: Ding Shutong   Student ID: 1155241390

1. Introduction

This homework aims to develop an AI agent based on LangChain and Google Gemini-2.5-Flash, which parses supermarket invoice images (renamed 1-7.jpg in Google Drive), calculates the total spent amount and original total amount, and only responds to two specified queries while rejecting irrelevant questions. The solution addresses the problem of unselectable text in images via OCR and strictly follows the LangChain tool-use paradigm from the starter code.

2. Methodology

2.1 Core Dependencies

1. **LLM & Framework**: Google Gemini-2.5-Flash (Vertex AI), LangChain (langchain-google-genai, langchain-core)
2. **OCR & Image Processing**: pytesseract + Tesseract OCR engine, pillow, numpy

2.2 System Workflow

The agent adopts a **tool-chain architecture** with 5 core modules, all using JSON-serializable parameters (string/list/dict) to resolve LangChain serialization issues:

1. **load_invoice_paths**: Load valid file paths of 1-7.jpg from the specified Google Drive folder (avoids direct PIL image transmission).

2. **ocr_invoice_by_path**: Preprocess images (grayscale + binarization) and extract text via multilingual OCR (eng+chi_sim), solving unselectable text problem.
3. **extract_invoice_amount**: Use Gemini to extract total_with_discount and total_without_discount from OCR text, outputting structured JSON.
4. **sum_invoice_amounts**: Batch process all invoices, accumulate amounts, and count valid parsing results (skip invalid/inreadable invoices).
5. **classify_query + Agent Orchestration**: Classify queries into total_spent/total_without_discount/irrelevant, execute tool chains in sequence, and generate targeted responses.

2.3 Key Technical Fixes

1. Replaced PIL image object transmission with **file path strings** to eliminate LangChain parameter validation errors.
2. Added image preprocessing (grayscale + binarization) to reduce invoice noise and improve OCR accuracy.

3. Implementation

1. **LLM Initialization**: Follows the starter code exactly, with temperature=0 for deterministic outputs.
2. **Tool Binding**: Uses LangChain's bind_tools to integrate all core tools, maintaining the original paradigm.

3. **Error Handling**: Includes fallback mechanisms for tool call failures and skips invalid invoices without system crashes.
4. **Code Structure**: 4 main sections (environment setup → tool definition → agent orchestration → test execution), consistent with standard AI project norms.