

Path	LLM/API Integration
Document	Project Proposal for DDS-AI Challenge
Name	David Lau Keat Jin
Title	Technical Specification Fulfillment Application

## **Project Progress: Day 1**

### **Project Overview**

This proposal outlines the development of an application designed to streamline the process of evaluating product suitability against tender document requirements. The application will extract individual requirements from tender documents, match them with product specifications from brochures or datasheets, and generate a comprehensive report highlighting fulfilled and unfulfilled requirements. This tool aims to enhance the efficiency of company representatives or pre-sales personnel, enabling them to focus on offering competitive products and negotiating better prices.

### **Objectives**

1. **Automate Requirement Extraction:** Extract individual requirements from sample tender documents.
2. **Match Specifications:** Compare extracted requirements with product specifications from brochures or datasheets.
3. **Generate Reports:** Populate an Excel file with matched requirements and highlight unfulfilled requirements.
4. **Provide Summary:** Summarize the number of fulfilled and unfulfilled requirements.

### **Benefits**

- **Efficiency:** Saves time and effort for company representatives or pre-sales personnel.
- **Product Suitability:** Helps gauge product suitability and identify potential product changes.
- **Competitive Edge:** Allows personnel to focus on offering more competitive products and negotiating better prices.
- **Core Functions:** Supports distributorship companies in inventory management, warehousing, order fulfillment, logistical expertise, and market education.

### **Application Features**

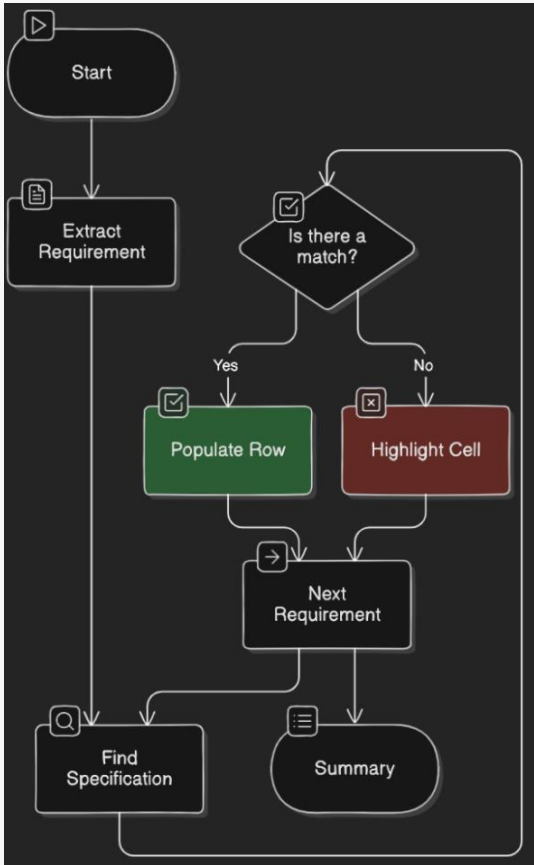
1. **Requirement Extraction:**
  - Use Natural Language Processing (NLP) to extract individual requirements from tender document.
  - Search the required specification from company's repository containing product brochures/datasheets.
2. **Specification Matching:**
  - Parse product brochures or datasheets to extract specifications.
  - Compare extracted requirements with product specifications.
  - Populate corresponding rows in an Excel file if there is a match.
  - Highlight cells as "unable to fulfill" if no match is found.
3. **Report Generation:**
  - Generate an Excel report with matched and unfulfilled requirements.
  - Provide a summary of the number of fulfilled and unfulfilled requirements.
4. **User Interface:**
  - Develop a user-friendly interface for uploading technical requirements

Path	LLM/API Integration
Document	Project Proposal for DDS-AI Challenge
Name	David Lau Keat Jin

### Project Progress: Day 2

### Workflow Diagram

The following diagram illustrates the process flow for the application:



### Example Use Case: IT Product Acquisition

1. **Populate Repository:** The application extracts company product and services information from brochure and datasheet and store it in the repository.”
2. **Upload Technical Requirement:** The user uploads an excel file with a list of requirement in the form of technical specification.
3. **Extract Specifications:** The application extracts specifications from the brochure.
4. **Match Requirements:** The application compares the extracted requirements with the specifications.
5. **Generate Document:** The application generates a compliance sheet showing which requirements are fulfilled and which are not.
6. **Summary:** The application provides a summary of the number of fulfilled and unfulfilled requirements.

### Conclusion

This application will significantly enhance the efficiency of evaluating product suitability against tender document requirements. By automating the extraction and matching process, it will save time and effort, allowing personnel to focus on offering competitive products and negotiating better prices. The tool is particularly beneficial for distributorship companies, supporting their core functions and improving overall productivity.

Path	LLM/API Integration
Document	Project Proposal for DDS-AI Challenge
Name	David Lau Keat Jin

**Project Progress: Day 3**

**Date:** 30 September, 2024

**Github URL:** <https://github.com/renaissance2005/DDS-AI-Challenge-Project/>

---

**Actions Implemented**

- 1. Ollama Installation**  
Installed Ollama on my local machine to facilitate local inference of language models, enabling efficient document analysis without relying on cloud services.
  - 2. Downloaded and Tested Models**  
Successfully downloaded and tested the **bakllava** and **llama3.1:8b** models locally using Ollama, ensuring the system could run language models on-premises.
  - 3. Redis-Server Installation**  
Installed **Redis-Server** to act as the document store, enabling persistent storage of extracted PDF content (text, tables, images) for future retrieval and analysis.
  - 4. Integrated Chroma and Redis-Server**  
Integrated **Chroma** as the vector database for similarity search, combined with Redis-Server for document storage. This setup ensures efficient data retrieval based on content embedding.
  - 5. Execution and Testing of Code**  
Ran the complete code locally to test the functionality of reading and processing PDF content, successfully storing the extracted data based on its format (text, table, or image) without encountering errors.
- 

**Challenges**

Initially attempted to use **Groq Cloud Services** to run **llama3-70b-8192** and **llama-3.2-11b-vision-preview** models but encountered rate limits during the loading of a single PDF document, which significantly hindered progress.

---

**Solution**

To mitigate the rate-limit issue with Groq Cloud Services, I switched to the **Ollama** platform, utilizing local inference models. This transition allowed the models to run smoothly on my local machine, providing better control over model execution without reliance on external cloud resources.

---

**Current Status & Technology Stack**

The project is progressing well, and the code can now be executed locally under normal conditions without errors. The system can process PDFs, extracting and categorizing their

---

Path	LLM/API Integration
Document	Project Proposal for DDS-AI Challenge
Name	David Lau Keat Jin

**Project Progress: Day 3**

content into text, tables, and images, which are then stored for future retrieval. The technology stack is as shown in the following diagram:



---

**Plans for Adjustment**

1. **Model Improvement:** I plan to switch to the **llava-llama3** model, which is expected to offer higher accuracy compared to the bakllava model currently being tested.
2. **Documentation:** A technical specification document outlining the tender requirements from a technical perspective needs to be included in the system to ensure compliance with project goals.

Path	LLM/API Integration
Document	Project Proposal for DDS-AI Challenge
Name	David Lau Keat Jin

## **Project Progress: Day 4**

Minimum technical specification prepared based on the website:

[The Best Laptops For Working From Home 2024 - Forbes Vetted](https://www.forbes.com/sites/forbes-personal-shopper/article/best-laptops-for-working-from-home/)

(<https://www.forbes.com/sites/forbes-personal-shopper/article/best-laptops-for-working-from-home/>)

### **1. API Integration Refinement**

This is not applicable as the application is fully using local databases and models.

### **2. Prompt Engineering**

Prompt for the model is enhanced to produce only specification related output without any image. Error handling is implemented for the purpose of quality testing where the column of 'Expected Response' should be available for calculation of BLEU score.

### **3. Data Handling and Preprocessing**

The Jupyter file (.ipynb) is used for testing to show the output. For error checking for the uploaded excel file, it is implemented with the initial main.py file which is using streamlit for Graphical User Interface.

### **4. Functionality Testing**

The basic QnA is functioning well.

### **5. Fine tuning**

Not implemented

### **6. Performance and Evaluation**

Incorporate BLEU Score generation. I switched to llava-llama model on Ollama.

### **7. Security and Privacy Consideration**

Local models and databases are used. User input is not necessary except to upload the excel file. This application is mainly for internal use by pre-sales team.

### **8. User Interface Testing**

Initial test is performed with streamlit and the result showed excel file can be uploaded by the user.

Path	LLM/API Integration
Document	Project Proposal for DDS-AI Challenge
Name	David Lau Keat Jin

### **Project Progress: Day 5**

Testing with different models for accuracy. Apart from that, requirements.txt file is generated for the environment. The local machine used to test the code has the following specification:

Requirement	Specification
<b>Hardware</b>	
▪ Processor	➤ 13 <sup>th</sup> Gen Intel(R) Core(TM) i7-13700Hx, 2100 Mhz, 16 Cores, 24 Logical Processors
▪ RAM	➤ 16 GB
▪ System Type	➤ 64 -bit operating system, x64-based processor
▪ Operating System	➤ Windows 11 Pro
<b>Software</b>	
▪ Code Editor	➤ Visual Studio Code 1.93.1
▪ LLM Platform	➤ Ollama Inno Setup version 6.3.3
▪ Data Store	➤ Redis Server 3.2.100 64-bit

\*\* The requirements.txt file is available in the project link  
<https://github.com/renaissance2005/DDS-AI-Challenge-Project/>

Path	LLM/API Integration
Document	Project Proposal for DDS-AI Challenge
Name	David Lau Keat Jin

## **Project Progress: Day 6**

### **1. Prompt Optimization**

The prompt to extract relevant data from product brochure/datasheet has been updated.

### **2. Handling Edge Cases**

Error checking has been included for evaluation. The score for accuracy measurement is changed from BLEU score to LLM evaluation which range from 1, 2 or 3 to enhance decision-making by the user.

### **3. Performance Tuning**

The caching of response is not required for the moment. The brochure and content from datasheet need to be extracted only once and store in a persistent manner in the document store and vector store. Hence, repeated processing can be minimized.

### **4. Error Handling and Debugging**

More error handling is incorporated especially during uploading of the excel file to check the format of the file and to ensure the column named 'Minimum Specification' exist.

### **5. UX Enhancement**

The enhancement include a facility to allow user to upload the file first and not processing the file directly. The user has the option to download the file when it is ready.

Path	LLM/API Integration
Document	Project Proposal for DDS-AI Challenge
Name	David Lau Keat Jin

### Project Progress: Day 7

For the sake of clarity, a video ‘explainer’ is created. The files used so far for this program include the following:

Input	Output	File	Usage
HP-dataset.pdf	None	populatedata.ipynb	This is run on Jupyter Notebook for testing and debugging during development
Sample-TenderDoc.xlsx	Updated-Sample-TenderDoc.xlsx	tsfa.py	The evaluation of the program is done using this code by comparing the ‘Expected Response’ and ‘Generated Response’ columns from an excel file with relevance given a score from 1-3. Take note that the processing and storage of brochure/datasheet (company documents) are not done here as it was implemented by the by populatedata.ipynb. This is run with command:  <i><b>python tsfa.py</b></i>
testdoc.xlsx	Updated-Sample-TenderDoc.xlsx	chktechspec.py process.py	This is the actual program where the user can run by giving an excel file with at least a column named ‘Minimum Specification’. A new file will be generated with the column ‘Generated Response’ which the user can use as reference for specification fulfillment of the existing product brochure/datasheet. This is run with the command:  <i><b>streamlit run chktechspec.py</b></i>


\*The program is run using Python 3.10.14 with streamlit installed. The full requirements is included in the file requirements.txt.




Path	LLM/API Integration
Document	Project Proposal for DDS-AI Challenge
Name	David Lau Keat Jin

**Project Progress: Day 8**

The user interface for the completed program is shown below (after uploading the file):

 Drag and drop file here  
Limit 200MB per file • XLSX

Browse files

 testdoc.xlsx 10.4KB

×

File uploaded successfully and contains the 'Minimum Specification' column.

	No.	Minimum Specification	Generated Response
0	1	CPU: Intel Core i5-1335U	None
1	2	RAM: 16GB DDR4	None
2	3	Graphics: Intel Iris Xe	None
3	4	Storage: 512GB SSD	None
4	5	Display: 15.6-inch	None
5	6	Resolution: 1,920 x 1,080	None
6	7	Refresh rate: 120Hz	None
7	8	Battery: Up to 10 hours	None
8	9	Weight: 3.6 pounds	None
9	10	OS: Comes with Windows	None

File has been saved.

Download Processed Excel File

The user need to upload an excel file with the colum ‘Minimum Specification’. Then after the file is processed, the column ‘Generated Response’ will be populated. The user can then download the processed file for further examination.

This conclude the project challenge.

Further enhancement include:

- i) Use better/larger models;
- ii) Incorporate agents to check the first response; and
- iii) Examine effectiveness with multiple PDF files.