Assessment Report

---

1. Problem Solving Process

The development of this project followed an iterative refinement approach, evolving through three distinct phases:

Phase 1: Basic Implementation

Initially implemented a simple Python script that collected user input directly and saved it to JSON without validation. The saved information was passed to the LLM after each input. This approach proved problematic as it:

- Lacked input validation and sanitization, I could input anything and it would take it as valid
- Had no structured data extraction capabilities
- Produced inconsistent and unreliable data collection

Phase 2: Structured AI Integration

Transitioned to using the Instructor library with Gemma3:4b model for structured information extraction. This phase introduced:

- Pydantic models for data validation
- JSON-mode extraction for reliable parsing
- Initial conversational capabilities
- However, the model exhibited significant hallucination issues without proper context

Phase 3: Context-Aware System

The final implementation addressed hallucination through comprehensive context management:

- Added conversation history to maintain context
- Enhanced prompting strategies with detailed system messages
- For the optional pathway generation task, upgraded to Qwen3:4b MoE model due to its superior reasoning capabilities through "thinking" mechanisms

2. Overall Framework

The project employs a modern microservices architecture with the following components:

Architecture

- **FastAPI** serves as the REST API server with comprehensive endpoint management
- **Instructor + Pydantic** handles structured information extraction with type safety
- **Weaviate Vector Database** provides conversation storage and semantic similarity search
- **Docker containerization** ensures consistent deployment across environments

Data Flow

1. User initiates chat session through React frontend
2. FastAPI creates unique session with UUID
3. User messages processed through Instructor for information extraction
4. Structured data validated using Pydantic models
5. Conversations stored in Weaviate for similarity search
6. Real-time progress tracking and completion status updates

3. Tools and Systems Selection

Language Model Selection

- **Gemma3:4b**: Initially chosen for lightweight deployment and reasonable performance
- **Qwen3:4b**: Upgraded for optional tasks due to superior reasoning through chain-of-thought processing
- **Nomic-embed-text**: Selected for vector embeddings due to efficiency and quality

Database Technology

Weaviate:

- Native vector search capabilities for conversation similarity
- Seamless integration with Ollama embedding models

Development Framework Rationale

- **Instructor Library**: Ensures reliable structured extraction from LLMs with type safety
- **Pydantic**: Provides robust data validation and serialization
- **FastAPI**: Offers automatic API documentation, type hints, and async support

- **Docker Compose**: Simplifies multi-service orchestration and deployment

4. Evaluation Results

Information Collection Accuracy

For the course equivalency data parsing:

- It successfully extracts the university approximately 95% of the time. Sometimes it fails to extract the information.
- For the course it again has a high success rate but sometimes LLM hallucination plays a role and it fails to extract simple course names.
- After all data is collected the data is successfully stored as JSON.

For the Pathway Gen Task:

- The university extraction here has also seen a really high success rate, same as course equivalency.
- The program extraction has seen the most differences. Sometimes it extracts the program perfectly such as “MS in Data Science and AI”, but sometimes it only extracts “MS”. Though this decreased after switching the model to Qwen, it was noticed once after that too.
- For the courses taken, it experiences the highest number of failed extraction attempts. Sometimes it extracts all the courses mentioned, but sometimes it fails to. If specifically mentioned by the user that these courses are taken, a high success rate is seen in extraction. Same goes for areas of interest.

5. GenAI Reflection

GenAI was used for frontend for designing the website. Especially the live tracking of information collected.

Apart from that, the instructor python library was suggested by Gemini, which helped me get structured responses through the ollama models. Since my task required me to get structured responses from the LLM.

Apart from the above, GenAI was used for error resolution that came during development and prompt refining.

6. Conclusion

In the end, a highly accurate LLM parser was made. It is able to identify full university names as well as abbreviations such as SFSU/SJSU and extract the information. Minimal amount of hallucination is observed during this, decreasing the temperature has helped reduce this, yet it has not been completely removed. The hardest part of this whole process was to develop intelligent prompts so that the LLM knows what to expect and what to return.