Wasserstoff Innovation Task Report

# Wasserstoff GenAI Document Research and Theme Identification Chatbot

SHUBH PUNDIR

Date: 22/05/2025

## 1. Introduction

This report details the development of an advanced Generative AI (GenAI) chatbot designed for document research and theme identification as part of the Wasserstoff internship program. The primary objective of this project was to build an intelligent system capable of semantic search, document ingestion, citation-supported answers, and thematic summarization using state-of-the-art AI technologies.

## 2. Project Overview

The core deliverable was a chatbot that enables users to upload documents of various types, query their contents, and receive synthesized, context-aware answers. Unlike traditional keyword-based search, this system leverages vector embeddings and large language models to understand semantic meaning, allowing it to:

* - Ingest and parse documents of any format via OCR and custom parsers  
  - Index content in a vector database for efficient semantic retrieval  
  - Perform contextual search with citation linking  
  - Extract overarching themes and summarize documents intelligently

## 3. Technologies Used

- Programming Language: Python  
- Vector Database: Qdrant — for high-performance vector storage and similarity search  
- Large Language Model (LLM): Gemini 1.5 Flash Pro by Google DeepMind — used for natural language understanding, answer synthesis, and theme extraction  
- AI Orchestration: LangChain — to integrate document embeddings, search, and LLM calls in a seamless pipeline  
- Document Processing:  
 - OCR libraries for text extraction from scanned documents: Pytesseract  
 - Custom parsers to handle multiple document formats (PDF, DOCX, TXT, etc.)  
- Environment: uv python virtual environment

- Databases:  
 -MongoDB: Used for storing metadata in a flexible manner during development

-QdrantDB: A vector DB with semantic searching using cosine similarity equipped with page, paragraph citations to pinpoint the source of our query

## 4. System Architecture

1. Document Upload & Ingestion  
 Users upload documents through a frontend interface or API endpoint. Documents are processed to extract raw text, including OCR for images or scanned files.  
  
2. Parsing & Preprocessing  
 Extracted text is cleaned and segmented based on structure (paragraphs, sections) for better semantic indexing.  
  
3. Embedding Generation  
 Text segments are converted into vector embeddings using language models compatible with the Qdrant vector store.  
  
4. Indexing & Storage  
 Embeddings and metadata are stored in Qdrant, enabling fast semantic similarity queries.  
  
5. Query Interface  
 Users submit natural language queries to the chatbot.  
  
6. Semantic Search & Retrieval  
 The system uses the query to retrieve relevant document segments from Qdrant by vector similarity.  
  
7. Answer Synthesis & Citation  
 Gemini 1.5 Flash Pro synthesizes an answer from the retrieved content, providing contextually relevant citations linking back to original documents.  
  
8. Theme Identification  
 The system analyzes documents to extract main themes and topics, presenting summaries to users.

## 5. Development Process

- Initial Setup:  
 Established the project folder structure based on best practices. Configured Python environments and dependencies.  
  
- Document Upload Module:  
 Developed functionality to handle uploads, support multiple file types, and integrate OCR.  
  
- Document Parser Module:  
 Implemented parsing logic for text extraction and preprocessing.  
  
- Vector Storage Integration:  
 Integrated Qdrant as the primary vector database, implemented embedding generation and indexing workflows.  
  
- LLM Integration:  
 Configured LangChain to manage queries and orchestrate calls to Gemini 1.5 Flash Pro for semantic search, answer generation, and theme extraction.  
  
- Testing & Validation:  
 Conducted thorough tests with various document types and queries to ensure accuracy and robustness.  
  
- Optimization:  
 Improved indexing speed, query response times, and answer quality via prompt engineering and pipeline tuning.

## 6. Challenges and Solutions

- Multi-format Document Handling:  
 Parsing diverse document formats required developing robust preprocessing pipelines and effective OCR integration.  
  
- Citation Accuracy:  
 Ensuring generated answers included accurate citations was challenging; addressed by careful metadata tracking during indexing.  
  
- Model Latency:  
 Balancing response speed with answer quality involved optimizing embedding sizes and LangChain configurations.

## 7. Outcomes and Deliverables

- Fully functional GenAI chatbot capable of ingesting, indexing, and semantically searching documents  
- Accurate, citation-supported answers to user queries  
- Theme extraction and summarization module  
- Modular Python codebase for maintainability and extensibility  
- Documentation covering system usage, deployment, and API interfaces

## 8. Future Work

- Expand support for additional languages and domain-specific documents  
- Improve user interface with richer interaction capabilities  
- Integrate with cloud-based scalable deployments for production use  
- Incorporate feedback loops for continual learning and answer refinement

## 9. Conclusion

This internship project successfully developed a cutting-edge GenAI document research and theme identification chatbot, demonstrating proficiency in AI integration, vector databases, and large language models. The solution meets the Wasserstoff program’s goal of building AI tools that enhance document accessibility and understanding, laying a strong foundation for future AI-powered knowledge systems.

## 10. References

- Qdrant Documentation: https://qdrant.tech/  
- LangChain Framework: https://langchain.com/  
- Google DeepMind Gemini 1.5 Flash Pro (internal reference)  
- OCR Libraries (e.g., Tesseract)  
- Wasserstoff Internship Guidelines and Deliverables