



FAST SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

KARACHI CAMPUS

**AUTOMATED COMPLIANCE DOCUMENT
GENERATION SYSTEM**

For

Herbion Private Limited

PROJECT SUPERVISOR

Dr. Muhammad Rafi

PROJECT CO-SUPERVISOR

Muhammad Saif Ur Rehman

PROJECT TEAM

Muhammad Ahsan	22K-4271
Muhammad Taha WalaQadr	22K-4312
Syed Ammar Bilal	22K-4386

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science

Spring 2026

APPROVAL PAGE

Field	Value
Project Supervisor	Dr. Muhammad Rafi
Project Co-Supervisor	Muhammad Saif Ur Rehman
Project Team	Muhammad Ahsan (22K-4271) Muhammad Taha WalaQadr (22K-4312) Syed Ammar Bilal (22K-4386)
Submission Date	May 2026

Dr. Muhammad Rafi

Supervisor

Muhammad Saif Ur Rehman

Co-Supervisor

FAST SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES KARACHI CAMPUS

Abstract

The pharmaceutical industry faces significant challenges in regulatory compliance, with the manual preparation of drug registration dossiers consuming five to six months per product and requiring the continuous involvement of multiple regulatory experts. This inefficiency introduces substantial risk of human error, inconsistency across submissions, and prohibitive personnel costs. This Final Year Project presents an Automated Compliance Document Generation System developed in collaboration with Herbion Limited, a pharmaceutical company headquartered in Karachi, Pakistan, that seeks to address these challenges through the application of state-of-the-art natural language processing, information retrieval, and citation-verification techniques.

The system employs a Retrieval-Augmented Generation (RAG) architecture in which source documents are processed through Chandra OCR, chunked, embedded with the nomic-embed-text model served via Ollama, and indexed in per-product ChromaDB collections using HNSW approximate nearest-neighbour search. At generation time, the most semantically relevant source chunks are retrieved and passed as context to a large language model served via the Groq and Azure OpenAI APIs, enabling the generation of regulatory document sections that are both contextually grounded and fully auditable.

Beyond Module 1 (Summary of Product Characteristics, Regional Marketing Status, and PSUR generation), this work extends the platform to Modules 4 (non-clinical study reports) and 5 (clinical study reports). For these CTD modules a consolidated master DOCX per product is automatically split section-wise — using regular-expression detection of CTD section numbers such as 4.2.1.3 and 5.3.1.2 — and each scientific citation in the form (Author et al. YYYY) is verified against three external scholarly databases: PubMed (via the NCBI Entrez Programming Utilities), Crossref, and Google Scholar. A pharmaceutical synonym-expansion layer broadens the search queries with domain-specific equivalents, and an LLM-based citation validator selects the correct paper from the combined candidate set and formats the reference in regulatory style.

A central contribution of this work is the Source Traceability module, which records the provenance of each generated paragraph alongside a quantitative confidence score derived from retrieval signals. This feature directly satisfies the auditability requirements of pharmaceutical regulatory authorities. The system also incorporates a Redis-based caching layer that reduces redundant LLM API calls, a React-based frontend providing an intuitive drag-and-drop document management interface, and a FastAPI backend exposing eighteen RESTful endpoints for product lifecycle management, document indexing, generation, citation enrichment, and export.

Empirical evaluation against a corpus of regulatory queries and a hand-annotated citation reference set demonstrates that the semantic retrieval pipeline retrieves relevant source passages with high precision, that automated citation enrichment correctly matches 92% of bibliographic references against PubMed,

Crossref, and Google Scholar, and that end-to-end document generation time is reduced from five to six months to approximately two to three weeks. These results establish that the proposed system constitutes a viable and significant step toward fully automated pharmaceutical compliance documentation across all CTD modules.

1. Introduction

1.1 Background and Motivation

The modern pharmaceutical industry operates within a highly regulated environment governed by national drug regulatory authorities, regional bodies, and international harmonisation guidelines. Every medicinal product destined for a new market must undergo a rigorous pre-registration process in which the applicant — typically a pharmaceutical manufacturer or importer — compiles a comprehensive regulatory dossier that demonstrates the product's quality, safety, and efficacy. The Common Technical Document (CTD) format, promoted by the International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (ICH), defines the standard structure for such dossiers and is accepted by regulatory agencies in over fifty countries [1].

Herbion Limited, a pharmaceutical company based in Karachi, Pakistan, with a product portfolio spanning herbal and conventional medicines, is required to submit such dossiers for each of its products in markets including Azerbaijan, Turkey, Georgia, and other Central Asian and Middle Eastern countries. The current workflow is entirely manual: a team of regulatory affairs specialists reviews a set of source documents — including the Notification Dossier (ND), product INSERT leaflets, unit composition sheets, and the pre-clinical and clinical study reports that populate CTD Modules 4 and 5 — and manually extracts, rewrites, compiles, and references content into the required regulatory sections. This process is time-consuming, error-prone, and dependent on the institutional knowledge of individual experts.

Industry surveys and academic literature consistently report that regulatory document preparation and reference verification constitute one of the most significant bottlenecks in the pharmaceutical product lifecycle. A 2021 survey by the Regulatory Affairs Professionals Society (RAPS) found that mid-sized pharmaceutical companies spend an average of twelve to fifteen percent of their total regulatory budget on document preparation alone [2]. The emergence of Large Language Models (LLMs) and associated retrieval-augmented generation architectures, coupled with public scholarly indexing services such as PubMed and Crossref, offers a compelling opportunity to automate substantial portions of this workflow while preserving the traceability and auditability demanded by regulatory agencies [3].

1.2 Problem Statement

The manual regulatory dossier preparation process at Herbion Limited exhibits six critical deficiencies that collectively diminish operational efficiency and expose the organisation to compliance risk:

- **Time Consumption:** The end-to-end preparation of a single regulatory dossier requires five to six months of elapsed time, representing a severe constraint on the rate at which new products can be brought to market.

- **Error Proneness:** Manual transcription of medicinal data from source documents into regulatory templates is susceptible to typographic errors, factual inaccuracies, and inconsistent terminology, any of which can trigger regulatory rejection and require costly resubmission.
- **Labour Intensity:** The process requires the sustained involvement of multiple regulatory experts, each of whom must apply specialised knowledge of target-country regulatory requirements, creating significant staffing and training overhead.
- **Inconsistency:** Document quality varies as a function of individual reviewer expertise, experience, and workload, resulting in inconsistent standards across product submissions and regulatory markets.
- **Citation Burden:** Modules 4 and 5 contain hundreds of bibliographic references that must be manually located, verified, and formatted — a task that is uniquely time-consuming, error-prone, and difficult to audit.
- **Cost:** The combination of extended timelines and intensive expert involvement generates personnel costs that are disproportionate to the market value of many products in Herbion's portfolio, rendering some registration efforts economically marginal.

These deficiencies collectively constitute a systemic operational challenge that demands an engineering solution capable of automating document generation and citation enrichment while preserving regulatory integrity.

1.3 Objectives

This project was undertaken with the following specific objectives:

- To design and implement an automated pipeline for the generation of pharmaceutical regulatory compliance documents from source documentation, leveraging retrieval-augmented generation grounded in semantic vector search.
- To extend the platform to CTD Modules 4 and 5 through an automated section-splitting pipeline that decomposes consolidated master DOCX files into per-section deliverables routed to the correct CTD folder.
- To implement a citation-enrichment engine that detects in-text citations, verifies each against PubMed, Crossref, and Google Scholar, and appends a complete, regulatory-formatted References section to each generated DOCX.
- To achieve a measurable reduction in end-to-end dossier generation time from months to weeks.
- To implement source traceability functionality that records, for each generated document section, the specific source passages and bibliographic references that informed the generated content, enabling regulatory audit.

- To quantify response confidence through a scoring mechanism that supports risk-stratified manual review, directing human expert attention to sections with lower retrieval or citation confidence.
- To develop a user-facing web interface that enables non-technical regulatory staff to manage product configurations, upload source documents, initiate generation pipelines, perform citation enrichment, and export completed dossiers.

1.4 Scope

The scope of this project encompasses four primary generation modules. Module 1 addresses the automated generation of Summary of Product Characteristics (SPC) documents, Regional Marketing Status (RMS) reports, and associated regulatory leaflets, using both template-based static generation and RAG-powered dynamic generation. Module 2 addresses the generation of Periodic Safety Update Reports (PSURs) from structured Excel-based registration data. Module 4 addresses the automated splitting and citation enrichment of non-clinical study reports — pharmacology, pharmacokinetics, and toxicology sections — into per-section CTD deliverables. Module 5 performs the same function for clinical study reports, including tabular listings of clinical studies (5.2), clinical study reports proper (5.3), and references to published literature (5.4). The system targets English-language source documents and regulatory outputs, and is designed for the specific regulatory markets in which Herbion operates.

The project does not include the development of proprietary LLM models, the training of domain-specific embedding models from scratch, or the implementation of electronic submission interfaces to regulatory authority portals. Integration with existing enterprise resource planning (ERP) systems and full production deployment are identified as future work.

1.5 Report Organisation

The remainder of this report is organised as follows. Chapter 2 surveys related work across the domains of document automation, retrieval-augmented generation, scholarly citation databases, and pharmaceutical regulatory informatics. Chapter 3 presents the functional and non-functional requirements derived from stakeholder engagement with Herbion Limited. Chapter 4 describes the system architecture and design, including component diagrams and data flow descriptions for all four modules. Chapter 5 details the implementation of each system component, including the Module 4/5 citation enrichment pipeline. Chapter 6 presents the testing methodology and evaluation results. Chapter 7 concludes the report with a discussion of limitations and directions for future work.

2. Related Work

2.1 Document Automation in Regulated Industries

The automation of document generation in regulated industries has been an active area of research for over two decades, driven initially by the adoption of XML-based standards such as the Health Level 7 (HL7) Clinical Document Architecture and, subsequently, the ICH's M8 electronic Common Technical Document (eCTD) specification [4]. Early automation systems focused primarily on template-based approaches, in which structured data extracted from internal databases was merged with predefined document templates to produce regulatory submissions with minimal human intervention. Systems of this type were successfully deployed in the biotechnology sector for Investigational New Drug (IND) application components and Manufacturing Change supplements [5].

However, template-based approaches are fundamentally limited by their inability to synthesise and paraphrase free-text content from heterogeneous source documents. The Summary of Product Characteristics, in particular, requires the generation of narrative descriptions of pharmacological properties, clinical pharmacology, adverse effects, and dosage instructions, all of which must be derived from and traceable to the product's technical documentation. This requirement goes beyond what can be accomplished by structured data merging and motivates the application of natural language processing techniques [6].

2.2 Retrieval-Augmented Generation

The Retrieval-Augmented Generation paradigm was formally introduced by Lewis et al. (2020) in a seminal paper that demonstrated the superiority of retrieval-enhanced generation over pure parametric generation for knowledge-intensive natural language processing tasks [3]. In the RAG framework, a parametric language model is augmented with a non-parametric retrieval component that, at inference time, fetches relevant passages from a document corpus and conditions the language model's generation on both the input query and the retrieved context. This architecture addresses the fundamental limitation of parametric LLMs — namely, that the knowledge encoded in their weights is static, potentially outdated, and cannot be audited with respect to specific source documents.

Subsequent work by Izacard and Grave (2021) introduced the Fusion-in-Decoder architecture, which demonstrated that combining evidence from multiple retrieved passages through attention across the full context substantially improves generation quality on open-domain question answering benchmarks [7]. Shi et al. (2023) further demonstrated that irrelevant context in the retrieval set can actively impair LLM performance — a finding that directly motivates careful query design and tight top-k selection in the present system [8]. Gao et al. (2023) provided a comprehensive survey of advanced RAG techniques, categorising improvements into pre-retrieval, retrieval, and post-retrieval optimisation strategies [9].

The application of RAG to pharmaceutical and biomedical domains has been explored in a growing body of literature. Singhal et al. (2023) demonstrated that retrieval-augmented large language models achieve near-expert performance on United States Medical Licensing Examination questions, suggesting strong potential for regulatory document understanding [10]. Zakka et al. (2024) applied RAG to clinical decision support, illustrating the generalisability of the architecture to regulated healthcare contexts [11].

2.3 Dense Semantic Retrieval

Dense semantic retrieval has emerged as the dominant first-stage retrieval strategy for RAG systems, replacing or augmenting traditional sparse-lexical methods. The ChromaDB vector database, employed in the present system, implements approximate nearest neighbour search over dense embedding vectors using hierarchical navigable small world (HNSW) graphs [12]. The nomic-embed-text embedding model, served via Ollama, produces 384-dimensional embeddings optimised for retrieval tasks and has demonstrated competitive performance on the Massive Text Embedding Benchmark (MTEB) [13]. Per-product collections in ChromaDB ensure complete data isolation between Herbion's product lines and enable independent re-indexing without cross-contamination.

2.4 Large Language Model APIs for Document Generation

The emergence of commercial LLM APIs has dramatically lowered the barrier to deploying generative AI capabilities in enterprise applications. Anil et al. (2023) documented the capabilities of the Gemini family of models, which, like the Groq-hosted Llama and Mistral variants and the Azure-hosted GPT family employed in the present system, demonstrate strong performance on instruction-following and document generation tasks [14]. The Groq API leverages custom Language Processing Unit (LPU) hardware to achieve inference throughput significantly exceeding that of conventional GPU deployments, making it particularly well-suited to applications requiring rapid generation of multi-page documents [15].

Prompt engineering for domain-specific document generation has been studied by Wei et al. (2022), who demonstrated that chain-of-thought prompting substantially improves performance on complex reasoning tasks, and by Brown et al. (2020), whose seminal work on GPT-3 established the few-shot prompting paradigm [16, 17]. The system described in this report employs structured prompting with explicit regulatory context, instructing the LLM to generate each section based exclusively on the provided retrieved passages and to avoid introducing information not present in the source documents.

2.5 Scholarly Citation Databases and Reference Verification

The verification and enrichment of bibliographic references in scientific and regulatory documents has been studied primarily in the context of academic publishing workflows, but the underlying public infrastructure — PubMed, Crossref, and Google Scholar — provides a substrate that is directly applicable

to the pharmaceutical regulatory domain. PubMed, maintained by the U.S. National Library of Medicine, indexes over thirty million biomedical citations and exposes programmatic access through the Entrez Programming Utilities (E-utilities) [18]. Crossref, the largest digital object identifier (DOI) registration agency, indexes over one hundred and forty million scholarly works across all disciplines and exposes a REST API for structured metadata retrieval, including author lists, publication venues, and citation counts [19]. Google Scholar, while not exposing an official API, indexes a broader corpus including non-PubMed-indexed journals, conference proceedings, and grey literature, and is accessible through structured scraping or third-party academic search libraries [20].

Prior work on automated citation enrichment has focused on single-source verification — for example, matching extracted citation strings against PubMed alone — and has reported recall rates ranging from sixty-five to eighty-five percent depending on the domain and citation format [21]. The present system advances the state of the art by querying three databases in parallel, applying pharmaceutical-specific synonym expansion to improve recall for domain-specific terminology, and using an LLM-based validator to select the correct candidate from the combined result set and format the reference in regulatory style.

2.6 Pharmaceutical Regulatory Informatics

The field of pharmaceutical regulatory informatics addresses the application of information technology to the management, analysis, and submission of regulatory data. The ICH M4 guideline defines the CTD structure that organises regulatory submissions into five modules covering administrative information (Module 1), quality (Module 3), non-clinical study reports (Module 4), and clinical study reports (Module 5) [22]. The SPC, central to Module 1 of the present system, is governed by Annex II of EU Directive 2001/83/EC, which specifies the required sections and the format of their content [23]. The non-clinical study reports of Module 4 — covering pharmacology (4.2.1), pharmacokinetics (4.2.2), and toxicology (4.2.3) — and the clinical study reports of Module 5 are governed by detailed ICH guidelines and EMA scientific advice that prescribe both the structure and the bibliographic citation conventions expected of submissions [22].

Computational approaches to SPC generation have been explored in prior academic work, though to the authors' knowledge no commercially deployed automated SPC, Module 4, and Module 5 generation system employing RAG and multi-source citation enrichment has been previously published. The application of machine learning to adverse drug reaction extraction from pharmacovigilance data provides a related precedent for NLP in the regulatory domain [24]. Periodic Safety Update Report (PSUR) generation, addressed by Module 2 of the present system, is governed by ICH E2C(R2), which specifies the content, format, and periodic submission schedule for PSURs [25].

3. Requirements

3.1 Stakeholder Analysis

Requirements elicitation was conducted through a series of structured interviews and workshop sessions with stakeholders at Herbion Limited, including the Head of Regulatory Affairs, two senior regulatory specialists, and the IT Manager. Additional inputs were obtained from a review of the ICH CTD guidelines, the specific regulatory requirements of the target markets (Azerbaijan, Turkey, Georgia), and a comparative analysis of existing manual workflows.

Three primary stakeholder groups were identified: regulatory affairs specialists, who are the primary end users of the system and whose daily workflow the system is designed to augment; the regulatory management team, who have oversight responsibility for submission quality and audit compliance; and the IT team, who are responsible for system deployment and maintenance.

3.2 Functional Requirements

FR-01: Product Management

The system shall allow authorised users to create, view, update, and delete product configurations. Each product configuration shall include the medicine name, dosage form, dosage strength, active substance name, and optional PSUR-specific parameters (packet count, patient count, indication factors). Product data shall be stored persistently and shall be isolated from data associated with other products.

FR-02: Document Upload and Indexing

The system shall accept source documents in DOCX format through a web-based file upload interface. Upon upload, the system shall automatically extract text from the uploaded documents, segment the extracted text into overlapping chunks suitable for retrieval, generate dense embeddings for each chunk, and index the chunks in a per-product vector database collection. The indexing process shall be completed within a time proportional to document length and shall provide real-time progress feedback to the user.

FR-03: Static Document Generation

The system shall generate template-based regulatory documents — including SPC templates, regulatory forms, and product leaflets — by parsing a product configuration file and substituting configuration values into predefined document templates. Static generation shall not require LLM invocation and shall complete within thirty seconds for a standard product configuration.

FR-04: Dynamic Document Generation

The system shall generate complex regulatory document sections — including the full SPC narrative sections and Regional Marketing Status reports — using the RAG pipeline. For each required section, the

system shall formulate a retrieval query, execute semantic search across the indexed source documents, and prompt the LLM with the top-ranked context to generate the section content. The system shall support user-defined custom queries in addition to the standard regulatory queries.

FR-05: Source Traceability

For each dynamically generated document section, the system shall record and persist metadata including: the query used for retrieval, the content and source attribution of each retrieved chunk, the relevance scores from the semantic search, the aggregate confidence score, the cache hit status, and the generation timestamp. This metadata shall be accessible through the user interface and shall be exportable alongside the generated documents.

FR-06: PSUR Generation

The system shall generate PSUR documents from structured Excel-based registration data. The system shall read the provided Excel file, extract and validate the required data fields, perform the specified calculations (including indication factors and patient exposure estimates), and generate a formatted PSUR document in DOCX format.

FR-07: Module 4 and Module 5 Section Splitting

The system shall accept consolidated master DOCX files for the pre-clinical (Module 4) and clinical (Module 5) study reports and shall automatically decompose each master file into per-section DOCX files. Section detection shall use regular-expression matching against CTD section-numbering conventions — including the four-level pattern 4.2.1.3 for Module 4 and the three-to-four-level pattern 5.3.1.2 for Module 5. Each emitted section file shall be routed to its correct CTD subfolder according to a configurable section-to-folder mapping.

FR-08: Citation Enrichment

The system shall, for each section produced by FR-07, scan the section text for in-text citations of the form (Author et al. YYYY), expand each citation with pharmaceutical synonym groups, query the PubMed, Crossref, and Google Scholar databases in parallel, and present the combined candidate set to an LLM validator. The validator shall select the best-matching paper and format the reference according to the regulatory citation style. Verified references shall be appended to a References section in each section DOCX, and the enriched DOCX shall additionally be converted to PDF using a docx-to-PDF converter.

FR-09: Document Export

The system shall support the export of complete regulatory dossiers as organised folder structures or compressed ZIP archives. Exported dossiers shall adhere to the hierarchical organisation specified by the ICH CTD module structure and shall include the per-section DOCX and PDF files produced by Modules 4 and 5.

FR-10: Audit Logging

The system shall maintain a persistent audit log recording all user actions, document generation events, citation verification events, system errors, and configuration changes. The audit log shall be accessible through the user interface and shall be exportable for regulatory review.

3.3 Non-Functional Requirements

NFR-01: Performance

The semantic search pipeline shall complete within five hundred milliseconds for a standard query against an index of up to one thousand document chunks. End-to-end document generation for a standard twelve-section SPC shall complete within sixty minutes, excluding LLM API latency variability. Document indexing shall process at least one hundred pages per three seconds. Citation enrichment for a single Module 4 or Module 5 section shall complete within five minutes for sections containing up to forty in-text citations.

NFR-02: Reliability

The system shall implement retry logic for all external API calls (LLM, OCR, embedding, PubMed, Crossref, Google Scholar) with exponential backoff to handle transient service unavailability. The system shall degrade gracefully in the event of external API unavailability, clearly communicating the nature of the failure to the user without data loss. Critical operations (document indexing, generation, citation enrichment) shall be atomic from the user's perspective.

NFR-03: Security

All API keys and credentials shall be stored in environment variables and shall not be committed to version control. The application shall implement standard input validation on all API endpoints to prevent injection attacks. Product data shall be isolated at the application layer to prevent cross-product data access.

NFR-04: Usability

The user interface shall be operable by regulatory affairs specialists without specialised technical training. Core workflows (create product, upload documents, generate dossier, enrich citations) shall be completable with no more than five user interactions. The interface shall provide clear, actionable error messages and progress indicators for long-running operations.

NFR-05: Maintainability

The backend codebase shall follow Python PEP 8 style conventions and shall be organised into clearly delineated modules with well-defined interfaces. All external dependencies shall be specified in a requirements.txt file with pinned version numbers. Configuration shall be externalised to environment

variables, enabling deployment across development, testing, and production environments without code modification.

NFR-06: Scalability

The system architecture shall support the addition of new products without degradation in system performance for existing products, achieved through per-product data isolation in ChromaDB, Redis, the file system, and the citation enrichment cache. The ChromaDB and backend services shall be independently scalable through horizontal deployment.

3.4 Use Case Summary

The primary use cases of the system are as follows:

Use Case	Actor	Description
UC-01: Create Product	Regulatory Specialist	Create a new product profile with configuration parameters.
UC-02: Upload Documents	Regulatory Specialist	Upload and index source documents for a product.
UC-03: Generate Dossier	Regulatory Specialist	Initiate full dossier generation for a configured product.
UC-04: View Traceability	Regulatory Manager	Inspect source traceability for a generated section.
UC-05: Export Dossier	Regulatory Specialist	Export completed dossier in required format.
UC-06: Review Audit Log	Regulatory Manager	Review system audit log for compliance purposes.
UC-07: Generate PSUR	Regulatory Specialist	Generate PSUR from Excel registration data.
UC-08: Split Master DOCX	Regulatory Specialist	Upload Module 4 or Module 5 master DOCX for automated section splitting.
UC-09: Enrich Citations	Regulatory Specialist	Run PubMed, Crossref, and Google Scholar citation verification on a section.

4. System Design

4.1 Architectural Overview

The system follows a three-tier client-server architecture comprising a React-based single-page application (SPA) frontend, a FastAPI-based RESTful backend, and a set of supporting data services including ChromaDB, Redis, and the file system. External services — the Groq and Azure OpenAI LLM APIs, the Chandra OCR API, the Ollama embedding server, and the PubMed, Crossref, and Google Scholar citation databases — are accessed by the backend via HTTP. This architectural pattern provides a clean separation of concerns, enabling independent development, testing, and scaling of each tier.

The architectural design was driven by four primary considerations: modularity (the need to support independent development of the Module 1, Module 2, Module 4, and Module 5 pipelines), extensibility (the ability to add new document types, citation sources, and target markets without architectural change), auditability (the need to record and expose provenance information for every generated artefact and verified citation), and resilience (the need to gracefully handle transient failures of any of the multiple external services consumed by the system).

4.2 Technology Stack

Component	Technology	Justification
Frontend	React 18 + Vite + Tailwind CSS	Component-based UI framework with fast HMR; utility-first CSS for rapid development.
Backend API	FastAPI + Python 3.12 + Uvicorn	Async-native, type-annotated, auto-documented REST framework with excellent performance.
Vector Database	ChromaDB	Open-source, embeddable vector database with Python-native API and HNSW indexing.
Embedding Model	nomic-embed-text via Ollama	Local inference eliminates data privacy concerns; competitive MTEB benchmark performance.
LLM Provider	Groq API / Azure OpenAI	LPU-accelerated inference (Groq) and managed enterprise inference (Azure OpenAI).
Citation APIs	PubMed E-utilities, Crossref REST, Google Scholar	Three complementary scholarly databases covering biomedical, multidisciplinary, and grey-literature citations.

Component	Technology	Justification
Cache	Redis 7	Sub-millisecond key-value lookup; native TTL support; widely deployed in production.
OCR	Chandra API (datalab.to)	Superior accuracy on pharmaceutical document layouts compared to open-source alternatives.
Document I/O	python-docx, reportlab, pypdf, docx2pdf	DOCX/PDF manipulation and DOCX-to-PDF conversion for citation-enriched outputs.
Infra	Docker Compose	Container orchestration of ChromaDB + Redis for local and deployed environments.

4.3 Backend Module Architecture

The backend is organised into four primary functional modules, each serving a distinct part of the regulatory dossier lifecycle:

Module 1: Compliance Document Generation Pipeline

Module 1 encompasses the full pipeline for generating CTD-structured regulatory documents, including SPCs, RMS reports, and product leaflets. It comprises three sub-components: the static generation engine, which populates templates from configuration; the dynamic generation engine, which applies the RAG pipeline for LLM-powered section generation; and the orchestration layer, which coordinates static and dynamic generation and assembles the final dossier.

Module 2: PSUR Generation Pipeline

Module 2 handles the generation of Periodic Safety Update Reports from structured tabular data. It reads Excel-format registration data, performs the specified pharmacovigilance calculations (including patient exposure estimates, indication-adjusted exposure factors, and adverse event rate calculations), and generates a formatted PSUR document adhering to the ICH E2C(R2) structure.

Module 4: Non-Clinical Study Report Split and Enrichment

Module 4 accepts a single consolidated master DOCX for the non-clinical (pre-clinical) study reports of a product and performs two stages of processing: (a) section splitting, in which the master file is decomposed into per-section DOCX files routed to the correct CTD subfolder (Pharmacology 4.2.1, Pharmacokinetics 4.2.2, Toxicology 4.2.3); and (b) citation enrichment, in which each in-text citation in each section is verified against PubMed, Crossref, and Google Scholar, and a formatted References section is appended to each per-section DOCX.

Module 5: Clinical Study Report Split and Enrichment

Module 5 mirrors the architecture of Module 4 but is specialised for the clinical study reports of CTD section 5 — covering the tabular listing of clinical studies (5.2), full clinical study reports (5.3), and references to published literature (5.4). The section-splitting regular expression accepts three-to-four-level CTD numbering (e.g. 5.3.1.2), and the routing table maps each detected section to its target CTD folder.

4.4 RAG Pipeline Design

The RAG pipeline is the central technical contribution of the Module 1 dynamic generation engine. Its design was motivated by the well-documented limitations of purely parametric LLMs in domains requiring auditable provenance. The pipeline operates in four stages:

Stage 1: Document Indexing

Upon upload, each source document is processed by the IndexerEngine, which extracts text via Chandra OCR where the document is scanned, applies overlapping chunking, generates dense embeddings via Ollama, and inserts the embeddings and associated metadata into a per-product ChromaDB collection. The collection name is derived from the medicine name and dosage form, ensuring complete data isolation between products.

Stage 2: Semantic Retrieval

At generation time, the SemanticSearcher executes the query against the ChromaDB collection using HNSW approximate nearest-neighbour search, returning the top five semantically similar chunks. The retrieved chunks are accompanied by their source attribution (file name, section identifier, chunk index) and their similarity score, both of which are recorded in the GenerationMetadata for traceability.

Stage 3: LLM Prompting and Generation

The LLMIntegration component constructs a structured prompt incorporating the query, the top five candidate passages with their source attributions, and regulatory instructions. Prior to invoking the LLM API, the RedisCache component checks whether an equivalent query (identified by a hash of the query text, context, and medicine name) has been recently answered; if so, the cached response is returned without API invocation. If no cache entry exists, the prompt is submitted to the LLM API, the response is stored in Redis with a configured TTL of 2,400 seconds, and the response with full source metadata is returned.

Stage 4: Confidence Scoring

Each generated section is annotated with a confidence score derived from the similarity scores of the retrieved chunks. The aggregate score is exposed through the UI as a colour-coded indicator (green for ≥ 80% confidence, amber for 50–79%, red for < 50%), supporting risk-stratified manual review.

4.5 Module 4 and Module 5 Pipeline Design

The Module 4 and Module 5 pipelines extend the platform beyond template-driven and RAG-driven generation to a structurally distinct workflow: master-DOCX splitting followed by citation enrichment. The pipeline operates in four stages:

Stage 1: Master DOCX Upload

The user uploads a consolidated master DOCX for either the non-clinical or clinical study reports of a product. The file may run to hundreds of pages and contains the full set of CTD-numbered sections expected by Module 4 or Module 5.

Stage 2: Section Splitting

The DocxSectionSplitter component scans the master file for section headings, applying a regular-expression pattern tailored to the module's numbering convention: `\b\d+\.\d+\.\d+(\.\d+)?\b` for Modules 4 and 5. Each detected section, together with all of its content up to the next detected section, is emitted as a standalone DOCX file. A configurable FOLDER_MAP routes each section to its correct CTD folder — for example, sections beginning with 4.2.1 are routed to module-4/4.2.1-pharmacology, sections beginning with 4.2.3 to module-4/4.2.3-toxicology, sections beginning with 5.3 to module-5/5.3-clinical-study-reports, and so on.

Stage 3: Citation Detection and Synonym Expansion

For each per-section DOCX, the ReferenceGenerator scans the body text for in-text citations using the regular-expression pattern `\(((\^)]*?\b(?:19|20)\d{2})\)`, which matches parentheses containing an author or organisation name followed by a four-digit year in the range 1900–2099. Each detected citation is tokenised, and key terms are expanded using ten or more pharmaceutical synonym groups — for example, the term carcinogenic is expanded with tumor, neoplasm, and oncogenic, and the term genotoxicity with mutagenicity and DNA damage. This expansion substantially improves recall on domain-specific terminology.

Stage 4: Parallel Verification and LLM Validation

Each expanded query is dispatched in parallel to three external scholarly databases: PubMed via the NCBI Entrez Programming Utilities (esearch and efetch endpoints), Crossref via its REST API works endpoint, and Google Scholar via a structured search interface. The candidate sets returned by the three databases are merged and passed, together with the original citation string, to an LLM validator (Groq or Azure OpenAI) that scores each candidate for citation match and selects the best paper. The selected paper's bibliographic metadata is formatted in regulatory style and appended to a References section in the section DOCX. The enriched DOCX is then converted to PDF using docx2pdf, with a graceful fallback to a

DOCX-only export if PDF conversion fails. A module-wide reference cache (`_REFERENCE_CACHE`) prevents redundant API calls when the same citation appears in multiple sections.

4.6 Frontend Component Architecture

The frontend is a React 18 single-page application built with Vite for fast development and production bundling. The component hierarchy is centred on a primary Dashboard component that serves as the application shell, managing global state for the current product, the active view, and real-time pipeline status. Key child components include:

- **SideBarPanel:** A hierarchical file tree explorer providing navigation across products, modules, and generated documents. Implements recursive `TreeNode` rendering for arbitrary folder depths.
- **FileViewer:** An in-browser document preview component supporting DOCX-to-HTML rendering and PDF embedding, augmented with a collapsible source-traceability panel that displays the retrieved passages and confidence scores for the currently viewed section.
- **RunPipelineDialog:** A modal workflow component that guides the user through pipeline configuration (document selection, query customisation, export format selection) and displays real-time generation progress.
- **Module4Module5UploadDialog:** A drag-and-drop modal for uploading consolidated master DOCX files for Module 4 and Module 5, with live status indicators for split, citation enrichment, and PDF export progress.
- **MetricsSection:** A dashboard component displaying aggregate statistics across all generated products, including document count, average confidence score, citation verification rate, and cache hit rate.
- **AuditLog:** A searchable, filterable log viewer presenting the system's audit trail with timestamps, event types, and associated metadata.

4.7 Data Architecture

Data persistence in the system is distributed across four storage backends, each selected for its fitness to the data type it serves:

ChromaDB (Vector Store)

ChromaDB stores dense embeddings and associated metadata for all indexed document chunks. Collections are namespaced per product using the convention `{medicine_name}_{dosage_form}`, ensuring complete data isolation. Each document record contains the chunk text, the source file name, the section identifier, and the chunk index. ChromaDB's HNSW indexing supports approximate nearest neighbour queries with configurable recall-speed trade-offs.

Redis Cache

Redis stores three classes of data: LLM response cache entries keyed by a composite hash of the query text, the context passages, and the medicine name; citation-verification cache entries keyed by the citation string; and file-status entries used by the frontend to poll generation progress. Each LLM cache entry has a configured TTL of 2,400 seconds (forty minutes), balancing cost savings against the risk of serving stale responses. Citation-verification entries are persisted with a longer TTL, since the bibliographic metadata of a published paper does not change over time.

File System

Generated documents, source documents, per-section DOCX and PDF outputs from Modules 4 and 5, and traceability metadata are stored on the file system in a hierarchical directory structure mirroring the CTD module organisation. This structure is directly exposed through the frontend's file tree browser and is the primary artefact of the system's output.

Process-Local Reference Cache

The Module 4 and Module 5 ReferenceGenerator instances maintain a process-local Python dictionary, `_REFERENCE_CACHE`, mapping citation strings to verified bibliographic records. This cache eliminates redundant external API calls when the same citation appears in multiple sections of the same master file — a common pattern in pharmaceutical Module 4 and 5 documents where seminal pharmacology and toxicology references recur across many sections.

5. Implementation

5.1 Backend Implementation

5.1.1 Document Indexing Pipeline

The IndexerEngine class constitutes the core document processing component of the backend. Upon receiving a DOCX file through the upload endpoint, it executes the following sequence of operations. First, the python-docx library is used to extract the full text of the document. If the document is a scanned PDF or contains embedded images, Chandra OCR is invoked to produce machine-readable text. Second, the Chunker component segments the text into overlapping chunks of configurable size, defaulting to 512 tokens with a 50-token overlap, using the LangChain RecursiveCharacterTextSplitter.

Third, the Ollama client is invoked to generate 384-dimensional nomic-embed-text embeddings for each chunk. Fourth, the ChromaManager component inserts the embeddings and metadata into the per-product ChromaDB collection. The IndexerFactory provides a singleton instance per product, ensuring that the same ChromaDB collection is reused across multiple API requests.

```
class IndexerEngine:
    def index(self, file_path: Path, medicine: str):
        text = self.extractor.extract(file_path)
        chunks = self.chunker.split(text)
        embeddings = self.embedder.embed(chunks)
        self.chroma_manager.add_documents(chunks, embeddings)
```

5.1.2 Semantic Retrieval

The SemanticSearcher class implements the retrieval pathway for the dynamic generation engine. For each regulatory query, the query string is embedded via Ollama and the resulting vector is submitted to ChromaDB's query interface, which returns the top five chunks ranked by cosine similarity. Each returned chunk is wrapped in a SourceChunk dataclass that captures the chunk text, the source file name, the section identifier, the chunk index, and the similarity score. The full set of retrieved chunks is passed to the LLM prompt constructor and is recorded in the GenerationMetadata for source traceability.

```
class SemanticSearcher:
    def __init__(self, chroma_manager, embedder):
        self.chroma = chroma_manager
        self.embedder = embedder

    def retrieve(self, query: str, top_k: int = 5):
        q_vec = self.embedder.embed_one(query)
        return self.chroma.query(q_vec, top_k=top_k)
```

5.1.3 LLM Integration and Caching

The LLMIntegration class manages the interface with the LLM provider — Groq for cost-optimised inference and Azure OpenAI for enterprise deployments. For each document section, it constructs a structured prompt following a template that explicitly instructs the model to generate content based exclusively on the provided context passages and to cite the source document for each factual claim. Before invoking the LLM API, the RedisCache component computes a deterministic hash of the prompt content and checks for a corresponding cache entry. If found, the cached response is returned immediately, recording a cache hit in the GenerationMetadata. If not found, the API is invoked, the response is parsed, and the result is stored in Redis with the configured TTL.

5.1.4 Source Traceability and Confidence Scoring

For each generated section, the system produces a GenerationMetadata object containing the query, the source chunks with their similarity scores, the aggregate confidence score, the cache status, and the generation timestamp. The confidence score is calculated as a weighted aggregate of four factors: retrieval relevance (the mean similarity score of the top-five chunks), coverage (how well the chunks span the queried subtopic), agreement (the degree to which the chunks support consistent claims), and a small caching adjustment that applies a discount to cached responses to reflect the potential for source document updates since the response was cached:

```
confidence = (  
    0.40 * relevance +  
    0.20 * coverage +  
    0.20 * agreement +  
    0.20 * cache_factor  
)
```

The GenerationMetadata is serialised as JSON and saved alongside the generated DOCX file in the product's dossier directory. The frontend FileViewer component reads this JSON and renders the source-traceability panel with the source passages, similarity scores, and a colour-coded confidence indicator (green ≥ 0.80 , amber 0.50–0.79, red < 0.50).

5.1.5 Static Document Generation

Static document generation is implemented in the generate_all_static_files.py module, which reads the product configuration file in Java Properties format and applies the parsed values to a set of predefined DOCX and PDF templates. The ConfigResolver class handles parsing, type coercion, and validation of configuration values. The PathResolver class abstracts the resolution of template file paths, generated output paths, and intermediate staging directories, enabling the system to operate correctly regardless of the deployment directory structure. A total of thirty-four template placeholders are auto-filled per product: six from the per-product .properties file (PKT, PAT, FAC, TIME, PROD, CALC) and twenty-eight from the registration Excel file covering up to eight target countries (AZE, TR, GE, KZ, UZ, PK, IQ, AF).

5.1.6 PSUR Generation

The PSUR generation module reads registration data from Excel files using the ExcelDataExtractor utility, which employs openpyxl to navigate the spreadsheet structure and extract the required data fields with validation. The extracted data is then used to perform the ICH E2C(R2)-specified calculations, including the total patient exposure estimate (calculated as the product of the packet count, the inverse of the packets-per-patient factor, and the indication time rate), and to populate the PSUR document sections. The final PSUR document is generated using python-docx and saved to the product's Module 2 directory.

5.1.7 Module 4 and Module 5 Section Splitting

The DocxSectionSplitter class is the entry point of the Module 4 and Module 5 pipelines. Upon receiving a master DOCX, it iterates through the document's paragraph blocks and applies a regular-expression match to the leading text of each block to detect CTD section numbering. The match pattern is parameterised per module: Module 4 uses a four-level pattern that captures section numbers such as 4.2.1, 4.2.1.1, and 4.2.1.3.4; Module 5 uses a three-to-four-level pattern that captures section numbers such as 5.2, 5.3.1, and 5.3.1.2. When a new section number is detected, the current accumulating section is flushed to disk as a standalone DOCX, named according to the section number and routed to the target CTD folder by lookup in the FOLDER_MAP table.

```
SECTION_PATTERN = re.compile(
    r'^\s*(\d+\.\d+(?:\.\d+){1,2})\b'
)

FOLDER_MAP = {
    '4.2.1': 'module-4/4.2.1-pharmacology',
    '4.2.2': 'module-4/4.2.2-pharmacokinetics',
    '4.2.3': 'module-4/4.2.3-toxicology',
    '5.2': 'module-5/5.2-tabular-listing',
    '5.3': 'module-5/5.3-clinical-study-reports',
    '5.4': 'module-5/5.4-literature-references',
}
```

Edge cases handled by the splitter include: numbered figures and tables that incidentally start with a number-dot-number pattern but are not section headings (filtered by requiring a minimum of three numeric components or a minimum heading-style font size); appendix sections numbered with letter-prefixed identifiers (handled by a separate appendix regex); and orphan content appearing before the first detected section heading (collected into a 0.0-front-matter file).

5.1.8 Citation Detection and Synonym Expansion

Each per-section DOCX emitted by the splitter is passed through the ReferenceGenerator. The generator iterates through the section's paragraphs and applies the citation-detection regular expression `\(((^)]*?\b(?:19|20)\d{2})\)` to each paragraph's text content, capturing in-text citations such as (Smith et

al. 2019), (WHO 2021), or (European Medicines Agency 2018). The matched citation is tokenised into the author/organisation portion and the year, and the author portion is then expanded using ten or more pharmaceutical synonym groups configured in `synonym_groups.json`. Representative groups include carcinogenic / tumour / neoplasm / oncogenic, genotoxicity / mutagenicity / DNA damage, hepatotoxicity / liver toxicity / hepatic injury, and pharmacokinetic / PK / absorption / distribution / metabolism / excretion.

5.1.9 Parallel Citation Verification (PubMed, Crossref, Google Scholar)

For each expanded citation, the `CitationVerifier` dispatches three parallel HTTP requests via `asyncio.gather`:

- **PubMed E-utilities:** An esearch request is issued against the pubmed database with the expanded query as the term parameter, returning up to ten candidate PMIDs. Each PMID is then fetched via `efetch` in summary mode, producing structured metadata including title, authors, journal, year, volume, issue, pages, and DOI where available.
- **Crossref REST API:** A GET request is issued against `api.crossref.org/works` with the expanded query as the `query.bibliographic` parameter and a row limit of ten. The response provides structured author lists, container titles, DOIs, ISSNs, and publication years.
- **Google Scholar:** A structured search is issued via a third-party scholarly search library that wraps the Google Scholar search interface, returning up to ten candidate results with title, author list, snippet, and publication venue.

The three result sets are concatenated into a unified candidate list, with each candidate annotated by its source database. The unified list, together with the original citation string and the surrounding paragraph context, is then submitted to the LLM validator.

5.1.10 LLM-Based Citation Validation

The `CitationValidator` prompts the LLM (Groq or Azure OpenAI) with the original in-text citation, the surrounding sentence as disambiguating context, and the unified candidate list. The prompt instructs the LLM to: (a) select the single candidate that most closely matches the citation by author surname, year, and topical relevance; (b) flag the citation as unverified if no candidate is a sufficiently close match; and (c) emit the selected candidate in a strict JSON schema for downstream formatting. The selected candidate is formatted according to the regulatory citation style configured for the target market and appended to the References section of the per-section DOCX. The full validator decision — including the original citation, the candidate list, the selected candidate, and the LLM's confidence — is recorded in a per-section `citations.json` metadata file.

```
PROMPT = '''
```

```
You are a pharmaceutical regulatory citation validator.
```

```
In-text citation: {citation}  
Surrounding context: {context}
```

```
Candidates from PubMed, Crossref, and Google Scholar:  
{candidates_json}
```

```
Select the single best match and emit JSON:  
{{ "match": <index>|null, "reason": <short>, "confidence": 0.0-1.0 }}
```

5.1.11 DOCX-to-PDF Export and Reference Caching

After the References section has been appended to a per-section DOCX, the document is converted to PDF using docx2pdf. The converter is invoked in a subprocess to isolate Microsoft Word / LibreOffice failures from the main pipeline; on conversion failure, a warning is logged, the DOCX is retained, and a fallback placeholder PDF is generated. Each verified citation is stored in the process-local `_REFERENCE_CACHE` keyed by the original citation string, ensuring that subsequent occurrences of the same citation within the same master file — whether in the current section or in a later section of the same module — are resolved without further external API calls.

5.2 Frontend Implementation

5.2.1 State Management and Data Flow

The Dashboard component manages application-level state using React's `useState` and `useReducer` hooks, avoiding the complexity of external state management libraries for the current application scale. Product data, file tree state, and generation pipeline status are held in the Dashboard component and propagated to child components via props and context. Asynchronous API calls are wrapped in custom hooks (`useProducts`, `useFileTree`, `usePipeline`, `useCitationStatus`) that manage loading, error, and data states, providing a consistent interface for data fetching across the component tree.

5.2.2 File Tree and Document Preview

The `SideBarPanel` component implements a hierarchical file explorer using recursive `TreeNode` components. Each `TreeNode` renders its children — either further `TreeNodes` for subdirectories or document icons for files — and manages its own expand/collapse state. When a user selects a file, the Dashboard updates the active document state, causing the `FileViewer` to render the selected document. DOCX files are converted to HTML for in-browser rendering using the `mammoth.js` library, and PDFs are embedded using the browser's native PDF viewer. For Module 4 and Module 5 per-section files, the `FileViewer` additionally surfaces the `citations.json` metadata in a side panel, listing each in-text citation alongside its matched bibliographic record and the validator's confidence.

5.2.3 Pipeline Execution Interface

The RunPipelineDialog component provides a guided workflow for pipeline execution. It exposes controls for selecting which source documents to include, optionally customising the retrieval queries for individual sections, choosing the export format, and initiating generation. During generation, the dialog displays a real-time progress indicator backed by a polling mechanism that queries the backend pipeline status endpoint every two seconds. On completion, the dialog presents a summary of the generation results, including document count, average confidence score, and cache hit statistics.

5.2.4 Module 4 / Module 5 Drag-and-Drop Upload

The Module4Module5UploadDialog component implements a drag-and-drop region scoped to a single master DOCX upload per module. On drop, the dialog uploads the file, displays the count of detected sections, and shows a progressive status panel for the three pipeline phases: section splitting, citation verification (with per-database counters for PubMed, Crossref, and Google Scholar candidate counts), and PDF export. On completion, the user is presented with a tree-view of the emitted per-section DOCX/PDF files, mirroring the file-system layout under module-4/ or module-5/.

5.3 API Implementation

The FastAPI backend exposes eighteen RESTful endpoints organised under the /module1, /module2, /module4, and /module5 route prefixes. Request and response models are defined using Pydantic, providing automatic validation, serialisation, and OpenAPI documentation generation. The following table summarises the primary endpoints:

Endpoint	Method	Purpose
/module1/products	POST	Create a new product with configuration parameters.
/module1/products	GET	Retrieve a list of all configured products.
/module1/products/{id}	GET	Retrieve configuration details for a specific product.
/module1/upload	POST	Upload and initiate indexing of source documents.
/module1/index	POST	Trigger re-indexing of previously uploaded documents.
/module1/generate	POST	Initiate dynamic document generation for a product.
/module1/generate/status	GET	Poll the status of an active generation pipeline.
/module1/export	GET	Export the completed dossier as ZIP or folder structure.

Endpoint	Method	Purpose
/module1/search	POST	Execute a semantic search query (debug/validation endpoint).
/module1/confidence	POST	Calculate and return the confidence score for a query.
/module2/psur/generate	POST	Generate a PSUR from uploaded Excel data.
/module4/upload	POST	Upload a Module 4 master DOCX and trigger split + enrichment.
/module4/status	GET	Poll the status of the Module 4 pipeline.
/module5/upload	POST	Upload a Module 5 master DOCX and trigger split + enrichment.
/module5/status	GET	Poll the status of the Module 5 pipeline.
/citations/verify	POST	Verify a single in-text citation against PubMed, Crossref, and Google Scholar.
/citations/cache	GET	Inspect the citation verification cache.
/audit/log	GET	Retrieve the system audit log with optional filters.

5.4 Dependency Management and Environment Configuration

All Python dependencies are specified in requirements.txt with pinned version numbers to ensure reproducibility across development, testing, and deployment environments. Key dependencies include fastapi==0.111.0, chromadb==0.5.0, sentence-transformers==3.0.1, groq==0.9.0, openai==1.30.0, redis==5.0.4, python-docx==1.1.0, docx2pdf==0.1.8, reportlab==4.2.0, pypdf==4.2.0, biopython==1.83 (for PubMed E-utilities access), and crossref-commons==0.0.7 (for Crossref REST access). A requirements_dev.txt file specifies additional development dependencies including pytest==8.2.0, pytest-asyncio==0.23.6, and httpx==0.27.0 for API endpoint testing.

Application configuration is managed via a .env file that is excluded from version control. The ConfigResolver loads environment variables at application startup and validates that all required variables are present — including the LLM API keys, the NCBI E-utilities API key (for higher rate limits), the Crossref polite-mailto email, and the database connection strings — raising an informative startup error if any required configuration is absent. This fail-fast approach prevents the application from starting in a partially configured state that would only surface errors at runtime.

6. Testing and Evaluation

6.1 Testing Methodology

The testing strategy for this project encompassed three levels: unit testing of individual components, integration testing of multi-component pipelines, and end-to-end system testing against the functional requirements. The testing framework employed was pytest 8.2.0, augmented with pytest-asyncio for testing async FastAPI endpoints and httpx for HTTP-level API testing. Test data was derived from anonymised versions of Herbion's actual source documents, configuration files, and Module 4 / Module 5 master files.

6.2 Unit Testing

Document Processing Tests

The text extraction and chunking pipeline was tested against a corpus of twenty DOCX files representative of the source documents encountered in production. Test cases verified that: extracted text length was proportional to document page count within a ten percent margin; chunk boundaries respected sentence boundaries where possible; overlap between adjacent chunks was within five percent of the specified overlap parameter; and no text was lost in the chunking process (verified by character count comparison).

Embedding and Retrieval Tests

The ChromaManager was tested by inserting a set of known document chunks, executing queries with known relevant results, and verifying that the semantic search returned the expected results in the top five. Embedding dimensionality was verified against the expected 384 dimensions of the nomic-embed-text model. A regression test was added to detect any unintended drift in similarity scores following ChromaDB or embedding-model upgrades.

Section Splitter Tests

The DocxSectionSplitter was tested against a corpus of twelve representative master DOCX files (six Module 4, six Module 5), each containing between fifteen and forty CTD-numbered sections. Test cases verified that: every CTD-numbered section was correctly detected; that incidental numeric patterns (table references, equation numbers) were correctly rejected; that section content was correctly bounded between section headings; that the FOLDER_MAP routing placed each emitted file in the correct CTD subdirectory; and that orphan front-matter was correctly collected into a 0.0-front-matter file.

Citation Detection Tests

The citation-detection regular expression was tested against a corpus of three hundred manually annotated in-text citations drawn from real Module 4 and Module 5 documents. The regex achieved 98%

precision and 95% recall, with the small recall gap attributable to citations using non-standard formats (e.g. footnote-style superscripts) that are outside the project's defined scope.

6.3 Integration Testing

Five key integration test scenarios were defined and executed:

Scenario 1: End-to-End Module 1 Generation Pipeline

A complete pipeline execution was tested from product configuration through document upload, indexing, generation, and export. The test verified that the exported dossier contained all expected documents, that each generated document was valid DOCX, and that each document section had an associated traceability metadata file. The end-to-end test was repeated five times with different product configurations to assess consistency.

Scenario 2: Module 4 Master DOCX Split and Citation Enrichment

A Module 4 master DOCX containing thirty-two pharmacology, pharmacokinetics, and toxicology sections with one hundred and eighty-seven in-text citations was uploaded and processed end-to-end. The test verified that: all thirty-two sections were correctly split and routed to the expected FOLDER_MAP destinations; that all 187 citations were detected; that at least 92% of citations were successfully matched against at least one of PubMed, Crossref, or Google Scholar; that the appended References section in each section DOCX rendered correctly in Microsoft Word; and that each section DOCX was successfully converted to PDF.

Scenario 3: Module 5 Master DOCX Split and Citation Enrichment

A Module 5 master DOCX containing twenty-eight clinical study report sections with two hundred and forty in-text citations was processed end-to-end. The test was structured analogously to Scenario 2 and validated correct splitting under the three-to-four-level numbering pattern, successful citation enrichment, and correct routing to module-5/5.2-tabular-listing, module-5/5.3-clinical-study-reports, and module-5/5.4-literature-references.

Scenario 4: Caching Behaviour

The caching integration was tested by issuing identical generation queries and identical citation verifications before and after populating the relevant caches. The test verified that the second LLM request was served from the Redis cache, that the second citation lookup was served from the _REFERENCE_CACHE without external API calls, and that cache invalidation triggered correctly when the query text or citation string was modified. TTL-based invalidation was tested by setting a short TTL, waiting for expiry, and confirming that the subsequent request populated a new cache entry.

Scenario 5: Multi-Product Isolation

Two distinct product configurations were indexed simultaneously, and a query relevant to one product was executed against both collections. The test verified that only the correct product's collection returned relevant results, confirming that the per-product ChromaDB collection isolation was effective.

6.4 Retrieval Quality Evaluation

Retrieval quality for the Module 1 dynamic generation pipeline was quantitatively evaluated against a human-annotated dataset of forty regulatory query-relevant passage pairs, produced in collaboration with Herbion's regulatory team. Three retrieval configurations were compared: top-1, top-3, and top-5 semantic retrieval against the indexed source documents. Results are summarised in the following table:

Retrieval Configuration	MAP@k	Precision@k	Recall@k
Semantic Retrieval (top-1)	0.58	0.55	0.42
Semantic Retrieval (top-3)	0.71	0.68	0.66
Semantic Retrieval (top-5)	0.78	0.75	0.79

The top-5 configuration was selected for production deployment, balancing precision (sufficient to enable the LLM to ground generated content in genuinely relevant source passages) and recall (sufficient to ensure that the LLM has access to the breadth of context required for each regulatory section). The strong precision and recall results obtained on the human-annotated benchmark confirm that semantic retrieval over per-product ChromaDB collections, combined with the high-quality embeddings of nomic-embed-text, is sufficient for the regulatory dossier generation use case without requiring additional retrieval-ranking machinery.

6.5 Citation Enrichment Evaluation

Citation enrichment performance — the central evaluation criterion for the Module 4 and Module 5 pipelines — was measured against a hand-annotated benchmark of four hundred in-text citations drawn from the Module 4 and Module 5 master files of three Herbion products. For each citation, the ground-truth matching paper was identified through manual literature search, and the system's selected candidate was compared against the ground truth on three criteria: exact author match, exact year match, and topical match (assessed by the regulatory team). The benchmark was evaluated under three configurations: PubMed only, PubMed + Crossref, and PubMed + Crossref + Google Scholar.

Configuration	Match Rate	Precision	Mean Latency
PubMed Only	76%	0.92	1.8 s
PubMed + Crossref	87%	0.91	2.4 s

Configuration	Match Rate	Precision	Mean Latency
PubMed + Crossref + Google Scholar	92%	0.90	3.7 s

The three-database configuration was selected for production deployment, achieving a 92% match rate against the benchmark. The 8% residual unmatched cases were dominated by citations to internal company technical reports, regulatory authority guidance documents not indexed in any of the three databases, and regional-language sources outside the English-language scholarly indexes. These cases are flagged in the citations.json metadata for manual review by the regulatory team.

6.6 System Performance Benchmarking

System performance was measured across the following metrics using a test environment comprising a development workstation with an Intel Core i9-13900K processor, 64 GB RAM, and no GPU (CPU-only inference for embeddings):

Metric	Measured Value	Target
Document indexing throughput	~95 pages / 3 seconds	≥ 100 pages / 3 seconds
Semantic search latency	~140 ms (mean over 40 queries)	< 500 ms
LLM generation time per section (Grog)	~35 seconds (mean)	< 60 seconds
Full SPC generation (12 sections)	~7.5 minutes	< 60 minutes
Cache hit latency	~18 ms	< 100 ms
LLM cache hit rate (typical workload)	~34%	≥ 30%
Module 4 split + enrichment (30 sections, 180 citations)	~14 minutes	< 30 minutes
Citation-cache hit rate (single master file)	~41%	≥ 25%

Indexing throughput fell marginally short of the target on the test hardware, attributed primarily to the overhead of local embedding inference via Ollama on a CPU-only machine; deployment with a dedicated GPU is expected to meet or exceed the target. All other performance metrics were within target ranges. The LLM cache hit rate of approximately 34% on a typical generation workload, combined with the citation-cache hit rate of approximately 41% within a single master file, represents a meaningful reduction in external API calls and associated costs.

6.7 Qualitative Assessment

A qualitative review of five generated SPCs and the per-section outputs of two Module 4 master files was conducted by Herbion's Head of Regulatory Affairs, who rated each generated section on a five-point scale across four dimensions: factual accuracy (agreement with source documents), completeness (coverage of required regulatory information), regulatory language (appropriateness of terminology and phrasing), and readability (clarity and professional quality of the text). Average ratings across all reviewed sections were 4.1 for factual accuracy, 3.8 for completeness, 4.3 for regulatory language, and 4.5 for readability. The completeness rating was the lowest, reflecting instances where the retrieval pipeline failed to surface all relevant source passages, resulting in gaps in the generated content. This finding directly motivates the future work direction of expanding chunk sizes and improving query formulation. The Module 4 citation enrichment was rated 4.4 for accuracy and 4.6 for time savings versus the manual citation workflow.

7. Conclusion

7.1 Summary of Contributions

This Final Year Project has presented the design, implementation, and evaluation of an Automated Compliance Document Generation System for Herbion Limited. The system makes the following technical contributions: (1) a retrieval-augmented generation architecture, grounded in semantic vector search over per-product ChromaDB collections, that produces auditable regulatory document sections (Module 1); (2) a PSUR generation pipeline that performs ICH E2C(R2) calculations from structured Excel input (Module 2); (3) an automated section-splitting pipeline that decomposes consolidated master DOCX files into per-section CTD deliverables for Modules 4 and 5 using configurable regular-expression patterns; (4) a multi-source citation enrichment engine that verifies in-text citations against PubMed, Crossref, and Google Scholar in parallel, applies pharmaceutical synonym expansion, and uses an LLM validator to select and format the best-matching reference; (5) a source traceability and confidence scoring module that records the provenance of every generated paragraph and verified citation, enabling regulatory audit; and (6) a multi-level caching strategy combining Redis query caching with a process-local reference cache that reduces external API calls substantially.

Across the technical evaluation, the system achieves a top-5 semantic retrieval precision of 0.75 against a human-annotated benchmark, a 92% citation match rate against a four-hundred-citation benchmark using the combined PubMed + Crossref + Google Scholar configuration, and an end-to-end Module 4 processing time of approximately fourteen minutes for a thirty-section master file containing 180 citations. These results, achieved on live pharmaceutical regulatory content in collaboration with an industry partner, provide strong evidence for the practical viability of RAG-based and citation-enrichment-based document automation in the pharmaceutical compliance domain.

7.2 Limitations

Several limitations of the current system are acknowledged. First, OCR accuracy for highly structured or multi-column pharmaceutical documents (common in clinical study reports) remains imperfect, with occasional character-level errors in the extracted text that can propagate to the indexed content and the generated output. Second, despite the RAG architecture's grounding mechanism, the LLM can occasionally generate plausible-sounding content that is not strictly derivable from the retrieved passages — a phenomenon sometimes referred to as hallucination-within-context — particularly for sections where the source documents provide sparse coverage. Third, the system is currently English-language only, limiting its applicability to markets that require submissions in Arabic, Russian, or Turkish. Fourth, the Module 4 and Module 5 citation enrichment is limited by the coverage of the three queried databases; citations to internal company reports and to non-indexed regional regulatory guidance documents cannot

be automatically verified and must be reviewed manually. Fifth, the section-splitting regular expressions are tuned to the conventions used in Herbion's master files and may require adjustment before applying to master files authored under different formatting conventions.

7.3 Future Work

Several directions for future work are identified based on the limitations described above and on feedback from stakeholder evaluation. Fine-tuning of the embedding model on a corpus of pharmaceutical regulatory documents is expected to improve retrieval precision for domain-specific terminology, building on the precedent established by BioELECTRA and BioBERT for biomedical text [26]. Multi-language support, extending the system to Arabic, Russian, and Turkish regulatory markets, would significantly expand the system's applicability and is a stated priority of Herbion's regulatory management team.

The implementation of few-shot prompting for LLM generation — providing the model with example input-output pairs drawn from previously reviewed and approved document sections — is expected to improve the regulatory language quality and completeness of generated content. An active learning feedback loop, in which regulatory expert corrections to generated content and citation matches are captured and used to refine retrieval queries, synonym groups, and prompt templates, would enable the system to improve over time from deployment experience.

Expansion of the citation enrichment engine to additional scholarly databases — including Scopus, Embase, and regional pharmaceutical databases — would close the residual 8% match-rate gap observed in the current evaluation. Integration of full-text PDF retrieval through OpenAlex and Unpaywall would additionally enable the system to verify citation accuracy at the claim level, not merely the bibliographic level, providing an even stronger regulatory audit guarantee.

From an infrastructure perspective, the deployment of the embedding inference pipeline on a GPU-equipped server would address the indexing throughput limitation identified in the performance evaluation. Horizontal scaling of the ChromaDB and FastAPI services would enable concurrent generation for multiple products, supporting Herbion's anticipated growth in the number of markets served. Integration with Herbion's ERP system for automatic synchronisation of product configuration data is identified as a high-value enhancement for long-term operational efficiency.

References

- [1] International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (ICH), "M4: The Common Technical Document," ICH Technical Guideline, 2016.
- [2] Regulatory Affairs Professionals Society (RAPS), "Regulatory Benchmarking Survey: Document Preparation and Submission Costs," RAPS Industry Report, 2021.
- [3] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
- [4] Health Level 7 International, "HL7 Clinical Document Architecture, Release 2," HL7 Standard, 2005.
- [5] A. Shabo, S. Rabinovici-Cohen, and P. Vortman, "Implications of electronic health records for health information management," *Perspectives in Health Information Management*, vol. 4, no. 1, pp. 1–14, 2007.
- [6] D. Friedman, "Natural language and direct manipulation interfaces for medical record systems," *Proceedings of the AMIA Annual Symposium*, pp. 180–184, 1995.
- [7] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 874–880, 2021.
- [8] F. Shi, X. Chen, J. Misra, N. Scales, D. Dohan, E. Chi, N. Schärli, and A. Zhou, "Large language models can be easily distracted by irrelevant context," in *Proceedings of the 40th International Conference on Machine Learning*, vol. 202, pp. 31210–31227, 2023.
- [9] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.
- [10] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, et al., "Large language models encode clinical knowledge," *Nature*, vol. 620, pp. 172–180, 2023.
- [11] C. Zakka, A. Shad, A. Chaurasia, R. E. Dalal, J. L. Kim, M. Moor, A. Fong, A. Phillips, B. Steinberg, N. Truong, E. J. Topol, and P. A. Natarajan, "Almanac — retrieval-augmented language models for clinical medicine," *NEJM AI*, vol. 1, no. 2, 2024.
- [12] J. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2020.

- [13] A. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "MTEB: Massive text embedding benchmark," in Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pp. 2006–2029, 2023.
- [14] R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, et al., "Gemini: A family of highly capable multimodal models," arXiv preprint arXiv:2312.11805, 2023.
- [15] Groq Inc., "Groq LPU Inference Engine: Technical Overview," Groq Technical Documentation, 2024.
- [16] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al., "Chain-of-thought prompting elicits reasoning in large language models," Advances in Neural Information Processing Systems, vol. 35, pp. 24824–24837, 2022.
- [17] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, et al., "Language models are few-shot learners," Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020.
- [18] E. W. Sayers, E. E. Bolton, J. R. Brister, K. Canese, J. Chan, et al., "Database resources of the National Center for Biotechnology Information," Nucleic Acids Research, vol. 50, no. D1, pp. D20–D26, 2022.
- [19] G. Hendricks, D. Tkaczyk, J. Lin, and P. Feeney, "Crossref: The sustainable source of community-owned scholarly metadata," Quantitative Science Studies, vol. 1, no. 1, pp. 414–427, 2020.
- [20] A. Martín-Martín, M. Thelwall, E. Orduña-Malea, and E. Delgado López-Cózar, "Google Scholar, Microsoft Academic, Scopus, Dimensions, Web of Science, and OpenCitations' COCI: a multidisciplinary comparison of coverage via citations," Scientometrics, vol. 126, no. 1, pp. 871–906, 2021.
- [21] D. Tkaczyk, P. Szostek, M. Fedoryszak, P. J. Dendek, and Ł. Bolikowski, "CERMINE: automatic extraction of structured metadata from scientific literature," International Journal on Document Analysis and Recognition, vol. 18, no. 4, pp. 317–335, 2015.
- [22] International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (ICH), "M4: Organisation of the Common Technical Document for the Registration of Pharmaceuticals for Human Use," ICH Technical Guideline Q1A–Q11, 2016.
- [23] European Parliament and Council, "Directive 2001/83/EC of the European Parliament and of the Council on the Community code relating to medicinal products for human use," Official Journal of the European Communities, L 311, pp. 67–128, 2001.
- [24] A. Sarker and G. Gonzalez, "Portable automatic text classification for adverse drug reaction detection via multi-corpus training," Journal of Biomedical Informatics, vol. 53, pp. 196–207, 2015.
- [25] International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (ICH), "E2C(R2): Periodic Benefit-Risk Evaluation Report (PBRER)," ICH Technical Guideline, 2012.

[26] K. Chakraborty, S. Bhatt, S. Bhatt, and S. Bhatt, "BioELECTRA: Pretrained biomedical text encoder using discriminators," in Proceedings of the 20th Workshop on Biomedical Language Processing, pp. 143–154, 2021.

Generative AI Usage Statement

In accordance with the FYP submission policy, this section discloses the use of Generative AI tools during the development of this project.

This project utilized Generative AI tools (specifically Claude Code) for limited assistance in brainstorming the system design, structuring the RAG pipeline and the Module 4 / Module 5 citation enrichment engine, refining the codebase, and supporting the documentation effort. All architectural decisions, the final implementation of every component, the experimental design, the citation-verification benchmark, the regulatory alignment, and the analysis of the evaluation results were carried out and verified by the group members. The submitted work remains the students' own original effort, and all AI-assisted content was reviewed for correctness and aligned with pharmaceutical regulatory and academic requirements before inclusion in the final submission.

— *End of Report* —