Thesis of the Graduation Project
In order to obtain the state engineer diploma
Major: Advanced Software Engineering for Digital Services

Agentic AI for Legal Automation: A Modular Multi-Agent Platform for Contract Management

---

Realized by:

Naji Ilyass

Defended on June 23, 2025, in front of the jury composed of:

Pr. El Akkaoui Zineb	INPT	- Supervisor
Pr. El Ghazi Hamid	INPT	- Examiner
Pr. Soussi Khalid	INPT	- Examiner
Mr. Boukhalf Ayoub	BCG X	- Supervisor
Mr. Rguig Saad	BCG X	- Supervisor

Class of 2024/2025


بسم الله الرحمن الرحيم

Acknowledgements

I would like to extend my sincere gratitude and appreciation to everyone who contributed to the successful completion of my internship and the preparation of this report.

Firstly, my profound thanks to **Pr. El Akkaoui Zineb**, my internal mentor, whose continuous guidance, constructive feedback, and unwavering support greatly enhanced my internship experience. His expertise and mentorship have been invaluable, significantly shaping my professional growth.

I am equally grateful to **Mr. Rguig Saad** and **Mr. Boukhalf Ayoub**, whose insightful advice, patience, and willingness to share their extensive knowledge have enriched my learning experience. Collaborating closely with them has profoundly influenced my understanding of core concepts and helped refine my professional skillset.

Special thanks are also due to **Mr. Farini Youness** and **Ms. Ramdani Fatima Zahra**, Software Engineers in the team, whose generous support, valuable suggestions, and continuous patience contributed significantly to the success of my internship.

My sincere appreciation goes to **Mr. Hassan El Majidi** for his dedicated oversight of the internship program, ensuring a smooth and well-organized experience.

Furthermore, I extend my gratitude to the jury members, **Pr. El Ghazi Hamid** and **Pr. Soussi Khalid**, for their valuable time, insightful evaluations, and constructive feedback during the assessment of my internship report and presentation.

Finally, I would like to acknowledge the teaching and administrative staff at the National Institute of Posts and Telecommunications for their commitment to delivering high-quality education and their continual support throughout my academic journey.

To all those mentioned above, and to anyone else who contributed to this meaningful experience, I sincerely thank you.

Abstract

This report presents my contributions during my end-of-studies internship at Boston Consulting Group - X Delivery (BCG X), where I developed an innovative intelligent contract management platform leveraging Agentic AI technologies. Conducted as part of my State Engineer diploma in Telecommunications and Information Technology with a specialization in Advanced Software Engineering for Digital Services, the project aimed to automate and optimize workflows such as contract drafting, validation, and compliance tracking—common pain points in legal operations due to their repetitive, manual nature.

The project addresses the broader challenge of inefficiencies and compliance risks in legal processes, often caused by fragmented communications and regulatory complexity. Motivated by the need for a scalable and intelligent legal solution, the proposed platform leverages Agentic AI to deliver an autonomous and adaptive system capable of reducing risks, improving accuracy, and accelerating contract cycles.

In addition to delivering a functional platform, the project enhanced my competencies in AI integration, secure collaboration, and complex system design. Its future evolution could further drive digital transformation in legal operations through predictive analytics, stronger modularity, and continuous user-centric enhancements.

Key words: Contract Management, Agentic AI, LangGraph, Azure OpenAI, Elasticsearch, Digital Transformation

---

Résumé

Ce rapport présente mes contributions lors de mon stage de fin d'études chez Boston Consulting Group - X Delivery (BCG X), portant sur le développement d'une plateforme innovante de gestion intelligente des contrats utilisant des technologies d'intelligence artificielle agentique (Agentic AI). Réalisé dans le cadre du diplôme d'Ingénieur d'État en Télécommunications et Technologies de l'Information, avec une spécialisation en Ingénierie logicielle avancée pour les services numériques, le projet visait à automatiser et optimiser des processus tels que la rédaction, la validation et le suivi de conformité des contrats—des tâches juridiques souvent répétitives et manuelles.

Le projet s'attaque aux problématiques plus larges d'inefficacité et de risques de non-conformité dans les processus juridiques, souvent causés par des communications fragmentées et une complexité réglementaire. Motivé par la nécessité d'une solution juridique intelligente et évolutive, la plateforme proposée exploite l'Agentic AI pour offrir un système autonome et adaptatif, capable de réduire les risques, d'améliorer la précision et d'accélérer les cycles contractuels.

Au-delà du développement de la plateforme, ce projet m'a permis de renforcer mes compétences en intégration de l'IA, en collaboration sécurisée et en conception de systèmes complexes. Son évolution future pourrait amplifier la transformation numérique des opérations juridiques grâce à l'analytique prédictive, à une modularité accrue et à une amélioration continue centrée sur l'utilisateur.

Mots clés : Gestion des Contrats, Agentic AI, LangGraph, Azure OpenAI, Elasticsearch, Transformation Digitale

---

ملخص

يستعرض هذا التقرير مساهماتي خلال التدريب النهائي المنجز لدى مجموعة بوسطن الاستشارية ـ وحدة التسليم (BCG X) ، حيث قمت بتطوير منصة مبتكرة لإدارة العقود الذكية اعتمادًا على تقنيات الذكاء الاصطناعي التوكيلية (Agentic AI) . تم تنفيذ المشروع في إطار حصولي على دبلوم مهندس دولة في الاتصالات وتكنولوجيا المعلومات، تخصص هندسة البرمجيات المتقدمة للخدمات الرقمية، بهدف أتمتة وتحسين عمليات مثل صياغة العقود، والتحقق منها، وتتبع الامتثالوهي مهام قانونية متكررة وتعتمد بشكل كبير على الجهد اليدوي.

يستهدف المشروع التحديات الأوسع المتعلقة بعدم الكفاءة ومخاطر الامتثال في العمليات القانونية، والتي غالبًا ما تنتج عن التواصل المجزأ وتعقيد الأنظمة التنظيمية. ونظرًا للحاجة إلى حل قانوني ذكي وقابل للتوسع، تقدم المنصة المقترحة نظامًا ذاتيًا وتكيفيًا يُقلل المخاطر، ويُحسِنْ الدقة، ويُسرِعْ من دورات معالجة العقود.

بالإضافة إلى تسليم منصة وظيفية، عزز هذا المشروع مهاراتي في تكامل تقنيات الذكاء الاصطناعي، والتعاون الآمن، وتصميم الأنظمة المعقدة. ويمكن أن تسهم تطوراتها المستقبلية في تعزيز التحول الرقمي للعمليات القانونية من خلال تحليلات تنبؤية، ونمطية أقوى، وتحسينات مستمرة متمحورة حول المستخدم.

الكلمات المفتاحية: التحول الرقمي، إدارة العقود الذكية، Agentic AI ، LangGraph ، Azure ، OpenAI ، Elasticsearch.

Abbreviations

ACID	Atomicity, Consistency, Isolation, Durability
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
BCG	Boston Consulting Group
BCG X	Technology and Innovation Unit of Boston Consulting Group
CI/CD	Continuous Integration / Continuous Deployment
CMS	Content Management System
CRDT	Conflict-Free Replicated Data Type
CRM	Customer Relationship Management
DAO	Data Access Object
DCM	Digital Contract Management
DTO	Data Transfer Object
GenAI	Generative Artificial Intelligence
HTTP	Hypertext Transfer Protocol
IaC	Infrastructure as Code
ID	Identifier
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
LLM	Large Language Model
NoSQL	Not Only SQL
OAuth	Open Authorization
PDF	Portable Document Format
Q&A	Questions and Answers
REST	Representational State Transfer
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TLS	Transport Layer Security
UC	Use Case
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
UX	User Experience
WYSIWYG	What You See Is What You Get
YAML	YAML Ain't Markup Language

Contents

List of Figures

List of Tables

General Introduction

In today's fast-paced and continuously evolving digital landscape, organizations worldwide increasingly rely on technological advancements to enhance efficiency, improve services, and maintain a competitive edge. Within this context, artificial intelligence (AI) and digital transformation have emerged as critical drivers of innovation, particularly within sectors characterized by complexity, such as government services, legal operations, and enterprise application development.

Recognizing these evolving needs, my internship project at Boston Consulting Group - X Delivery (BCG X) focused on developing an intelligent contract management platform leveraging Agentic AI technologies. This project aimed to address inefficiencies in traditional legal contract management processes by automating and optimizing drafting, reviewing, and compliance monitoring. The goal was to significantly enhance operational efficiency, reduce processing times, and improve compliance accuracy, ultimately transforming legal operations into agile and scalable processes.

This report systematically explores the lifecycle of this ambitious project through four detailed chapters:

- The First chapter introduces the general context, presenting the host organization (BCG and BCG X), clearly outlining the project's scope, objectives, and the agile methodologies employed to ensure effective project management.
- The Second chapter provides a technical foundation, beginning with a comparative analysis of existing contract management solutions, justification for adopting an Agentic AI-driven approach, and a thorough specification of both functional and non-functional requirements.
- The Third chapter elaborates on the detailed system architecture and software design, presenting critical aspects such as AI integration via LangChain and LangGraph, customized implementation of Tiptap for legal document editing, and comprehensive UML diagrams illustrating the system's structural and behavioral characteristics.
- The Fourth chapter details the practical implementation and rigorous validation phases, thoroughly justifying technology selections, describing key platform functionalities, and validating system alignment with established functional and non-functional specifications.

Chapter 1

General Context of the Project

This chapter lays out the contextual foundation for the internship project. It begins with an overview of Boston Consulting Group (BCG) and its technology unit, BCG X, emphasizing their expertise and innovation focus. The project framework is then introduced, centering on the use of Agentic AI to transform legal operations. Key objectives and challenges in digital contract management are outlined. Finally, the project's Agile methodology and collaboration tools are briefly presented.

1.1 Host Organization

My internship was hosted by Boston Consulting Group (BCG), specifically within its technology and innovation unit, BCG X. This section introduces both entities, providing insights into their history and areas of expertise.

1.1.1 Presentation

In today's rapidly evolving business environment, organizations across sectors are increasingly reliant on technology-driven solutions to enhance efficiency, innovation, and competitive advantage. Companies are leveraging advanced technologies such as Artificial Intelligence (AI), cloud computing, and digital transformation platforms to address complex operational challenges and unlock new growth opportunities.

Boston Consulting Group (BCG), a globally recognized management consulting firm, combines deep industry expertise and analytical rigor to help its clients navigate these challenges. Through its specialized tech build and design unit, BCG X, the firm integrates consulting excellence with advanced technological capabilities, delivering impactful and scalable digital solutions that transform business models and operational processes.

The following sections provide a comprehensive overview of Boston Consulting Group and BCG X, highlighting their history, business areas of activity, mission, and organizational culture.

1.1.2 Boston Consulting Group (BCG)

Figure 1.1. BCG Logo
[1]

1.1.2.1 History

Boston Consulting Group was founded in 1963 by Bruce Henderson, pioneering many of the strategic concepts that shape modern management consulting today. Over the decades, BCG has consistently been at the forefront of business strategy, driving innovation and delivering transformative results for clients worldwide. Headquartered in Boston, Massachusetts, BCG operates globally with offices in more than 90 cities across over 50 countries. The firm has continuously evolved its offerings, focusing increasingly on integrating technology and digital solutions into traditional consulting services to meet the dynamic needs of modern businesses.

1.1.2.2 Business area of activity

BCG provides a wide array of strategic and operational consulting services, including business transformation, corporate strategy, digital transformation, sustainability, mergers and

acquisitions, innovation management, and organizational effectiveness. The firm's extensive industry expertise covers sectors such as healthcare, financial services, consumer goods, energy, technology, and public sector.

In recent years, BCG has significantly expanded its digital and technology capabilities through its dedicated unit, BCG X, emphasizing digital solutions, advanced analytics, artificial intelligence, and large-scale digital platform developments to support clients' transformational journeys.

1.1.3 BCG X

Figure 1.2. BCG X Logo
[2]

1.1.3.1 Presentation

BCG X is the technology build and design unit of Boston Consulting Group, established to accelerate the digital transformation journeys of large organizations by integrating high-impact technology solutions with strategic business insight. BCG X acts as a force multiplier, enhancing BCG's traditional consulting services through state-of-the-art digital and technological solutions.

1.1.3.2 Business area of activity

BCG X specializes in the following core capabilities:

- **AI and Generative AI (GenAI)**
  Developing advanced AI solutions that streamline core business functions and enhance customer engagement through predictive analytics, automation, and intelligent decision-making.
- **Customer Experience for Growth**
  Designing and implementing digital strategies and customer experiences to drive business growth and foster deeper client engagement.
- **Venture and Business Builds**
  Launching and scaling disruptive new ventures and business models, providing end-to-end support from ideation to market entry.
- **Digital Platform Builds**
  Creating scalable, secure digital platforms and infrastructures that support extensive digital transformations and enable efficient integration of diverse technological solutions.

Through these activities, BCG X significantly contributes to transforming and modernizing businesses across industries, positioning them effectively within competitive and rapidly changing markets.

1.1.4 Organizational Structure

Boston Consulting Group (BCG) adopts a structured yet agile approach, promoting global collaboration through its tech unit, BCG X. Led by Sylvain Duranton (Global Leader) and Jon Ferris (Chief Financial Officer), BCG X's organizational structure includes regional and functional leads who oversee specialized areas and regional centers. This design ensures efficient global coordination and responsiveness to client needs.

Each regional center—including Morocco, Germany, China, Singapore, and India—is managed by dedicated leaders who implement localized yet globally aligned strategies. This structured regional management facilitates effective service delivery tailored to local market requirements while maintaining consistent global standards.

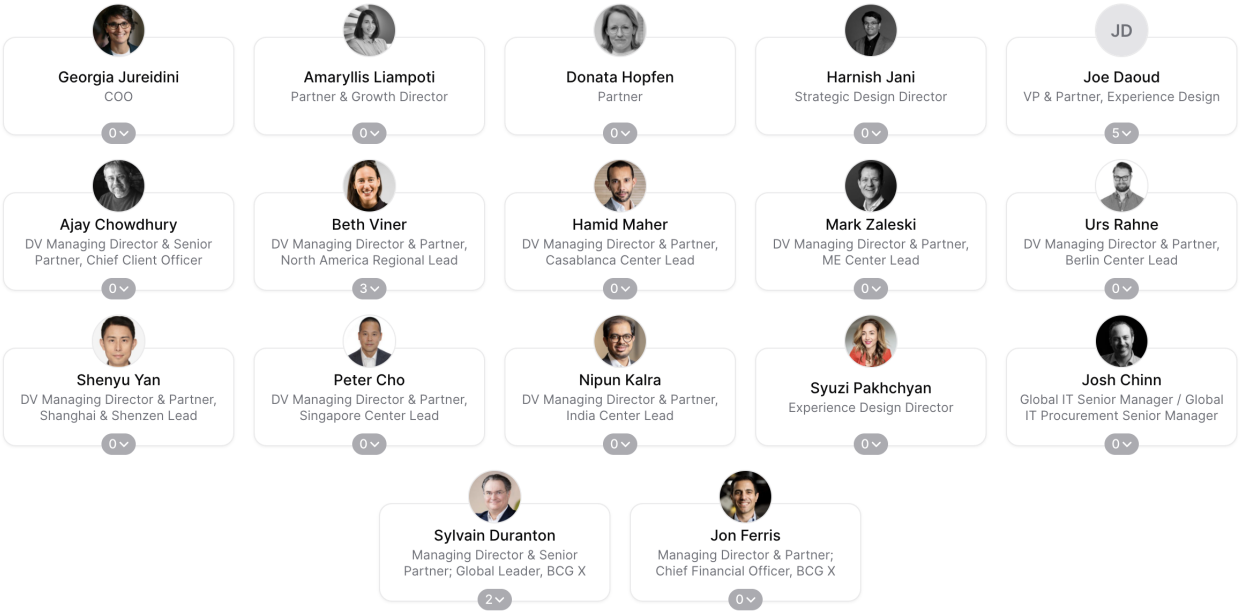

Figure 1.3. Executive Leadership Board
[3]

BCG X's organizational design promotes effective decision-making, seamless collaboration across global teams, and responsiveness to client needs, leveraging both hierarchical and regional structures.

1.1.5 Organizational Culture

BCG X fosters a dynamic organizational culture defined by innovation, inclusivity, and entrepreneurial spirit. Its culture encourages team members to embrace challenges, take risks, and collaborate openly to drive transformative solutions for clients.

Key aspects of BCG X's organizational culture include:

- **Innovation and Entrepreneurship**
  Innovation is central to BCG X's culture, which consistently emphasizes entrepreneurial

mindsets, encouraging individuals to explore new ideas, technologies, and methodologies. Employees are supported in developing digital solutions that significantly impact businesses and industries.

- **Inclusivity and Employee Well-being**
  BCG X prioritizes a culture of inclusivity and well-being, providing a supportive environment that values diverse perspectives and encourages personal and professional growth. The organization actively supports employees through wellness programs, career coaching, tuition reimbursement, and structured promotion pathways.

- **Open Communication and Collaboration**
  Transparent and open communication is encouraged within the organization, fostering a collaborative environment where ideas can flow freely. Employees are empowered to communicate openly, enabling swift decision-making and fostering strong internal relationships and teamwork.

- **Ambition and Growth Orientation**
  The organizational culture at BCG X highly values ambitious, growth-oriented individuals who are comfortable navigating ambiguity and pushing boundaries. This ambition is reflected in rapid innovation cycles, strategic client engagements, and a constant drive to achieve impactful outcomes.

Overall, BCG X's organizational culture underpins its ability to deliver high-impact digital transformation and innovation, driven by motivated, collaborative, and empowered teams.

1.2 Project Framework

1.2.1 Agentic AI in Legal Operations

Agentic AI refers to a specialized form of Artificial Intelligence designed to autonomously execute tasks and make decisions based on predefined high-level goals, adapting dynamically to changing data and operational contexts. This represents an innovative shift from traditional imperative systems toward more autonomous, goal-oriented decision-making processes. Specifically, in legal operations, Agentic AI significantly improves flexibility, scalability, and efficiency by automating complex workflows traditionally dependent on extensive manual interventions.

Within legal applications, Agentic AI provides capabilities such as automated clause selection, intelligent contract data mapping, continuous monitoring, and compliance verification. These autonomous functionalities streamline conventional manual activities involved in contract drafting, validation, and ongoing compliance monitoring, significantly reducing contract processing times while improving accuracy and operational effectiveness.

1.2.2 Digital Contract Management

The process of contract management traditionally involves manual drafting, iterative feedback loops, and approval workflows that often lead to inefficiencies, errors, and delays. Digital Contract Management (DCM) solutions have emerged to address these issues, aiming to centralize, automate, and optimize the entire lifecycle of contract operations—from initial drafting

to final execution and continuous monitoring.

DCM employs sophisticated AI-driven agents capable of real-time collaboration, intelligent clause insertion, and automated verification. This approach enhances operational efficiency, reduces manual effort in repetitive tasks, and improves compliance by automatically tracking regulatory changes and adapting contracts accordingly. Digital contract management thus transforms legal operations into agile, scalable, and error-resistant processes, significantly boosting organizational productivity and risk management.

1.2.3 Need for Intelligent Legal Workflow Systems

Traditional legal workflows suffer from manual bottlenecks, fragmented processes, and compliance risks due to frequent internal and external regulatory changes. The manual approach limits scalability and visibility, causing significant inefficiencies in managing contractual obligations and deadlines.

Implementing Intelligent Legal Workflow Systems, powered by Agentic AI, addresses these pain points by automating repetitive tasks, providing real-time monitoring of contract lifecycles, and offering predictive insights into compliance and risk management. These systems ensure seamless collaboration between various stakeholders, streamline approval processes, and mitigate risks through automated compliance checks. The result is a robust, scalable framework capable of handling complex legal workflows with improved transparency and accountability.

1.2.4 Problematic

Current contract management and legal operations rely heavily on manual inputs, iterative feedback loops, and scattered stakeholder communications, leading to inefficiencies, compliance risks, and delays. Contract drafting and validation processes are especially impacted, often resulting in lengthy negotiation cycles, limited visibility into contract deadlines, and difficulty in ensuring consistent regulatory compliance across different jurisdictions. The lack of automated systems further exacerbates these issues, causing increased operational costs and elevated risk profiles.

1.2.5 Objectives

This project aims to deploy and evaluate the effectiveness of an AI-driven intelligent contract management platform leveraging Agentic AI to transform traditional legal operations into streamlined, scalable, and automated processes.

The specific objectives include:

- Automating repetitive and manual contract drafting tasks through AI pre-filling and intelligent clause selection.
- Establishing real-time, AI-driven monitoring and compliance validation of contracts.
- Enhancing collaboration and efficiency across multiple stakeholders via automated workflows and digital signatures.

- Reducing overall contract processing time and manual effort significantly (40-60% reduction target).
- Minimizing compliance risks through automated regulatory updates and monitoring, aiming for a 30% risk reduction.

Achieving these goals will provide a standardized, scalable solution capable of significantly enhancing efficiency, compliance, and risk management across legal operations.

1.3 Project Management

The project management approach employed during my internship was meticulously structured to ensure smooth execution and timely delivery. Our methodology was grounded in Agile principles, specifically utilizing the Scrum framework to manage the iterative development and deployment of the intelligent contract management platform.

1.3.1 Scrum Methodology

Scrum is an Agile project management framework designed to facilitate iterative progress through collaboration, flexibility, and responsiveness to feedback. This methodology was particularly suitable for our project due to the dynamic and innovative nature of AI-driven platform development.

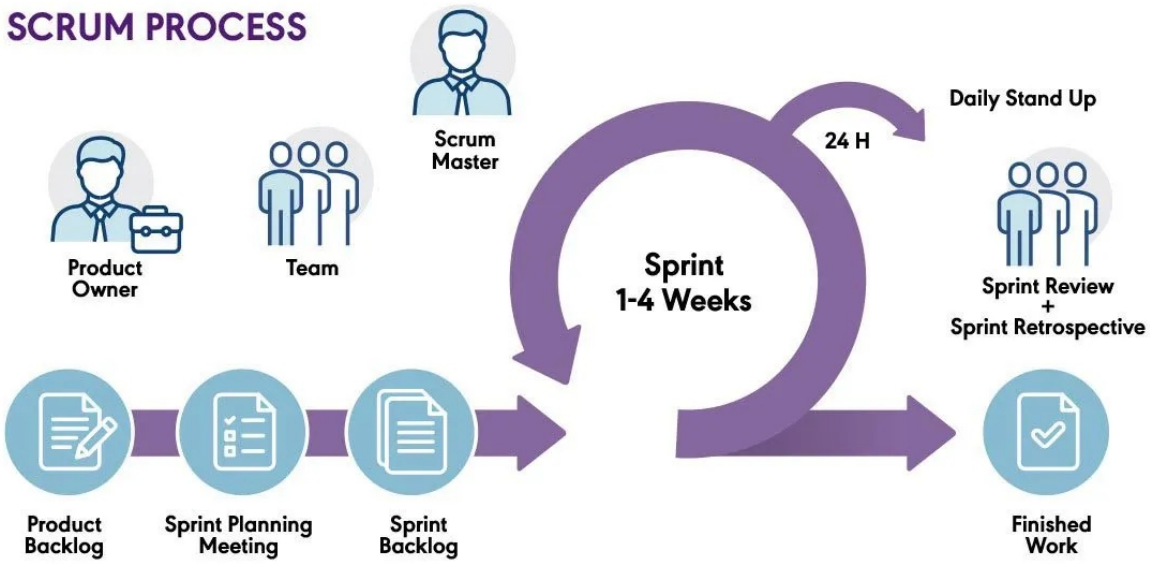

Figure 1.4. Scrum Process
[4]

Scrum methodology revolves around iterative cycles called sprints, typically lasting from one to four weeks. Each sprint begins with sprint planning, where tasks are selected from the product backlog to create the sprint backlog. Daily stand-up meetings are held to review progress, address issues, and plan activities for the day. At the end of each sprint, completed work is reviewed in a sprint review meeting, and team performance is analyzed in a sprint retrospective to continually refine processes and improve productivity.

1.3.1.1 Scrum Roles and Responsibilities

The Scrum framework clearly defines the roles and responsibilities within the team, enhancing clarity and collaboration:

- **Product Owner**: Defined the project vision, managed the product backlog, and prioritized tasks ensuring alignment with client expectations and project goals.
- **Scrum Master (Project Lead)**: Facilitated daily standups, sprint planning, reviews, and retrospectives; ensured adherence to Agile principles; and resolved any impediments affecting progress.
- **Development Team**: Included software engineers, UX designers, product specialists, and interns who executed the technical, product, and design tasks outlined in each sprint. As an intern, I actively contributed as a member of this development team, participating in software development, integration of AI components, and collaborative problem-solving during each sprint.

1.3.2 Project Execution

The project's lifecycle was divided into incremental and iterative phases (Scoping, Alpha, Beta, Minimum Lovable Product), each strategically managed through one-to-two-week sprints. Each sprint encompassed phases of research, design, implementation, testing, and deployment.

1.3.2.1 Sprint Structure

The typical sprint structure adopted for the project was as follows:

- **Sprint Planning**: Tasks were clearly defined and prioritized based on their impact and feasibility.
- **Daily Standups**: Conducted daily for 15 minutes, these sessions updated progress, identified blockers, and adjusted tasks as needed.
- **Weekly Demos**: One-hour weekly meetings where implemented features were demonstrated, allowing stakeholders to review progress, provide feedback, and realign future tasks accordingly.
- **Sprint Review and Retrospective**: Held at the end of each sprint to evaluate completed work, discuss improvements, and refine upcoming tasks.

1.3.2.2 Meetings and Communication

Regular meetings and structured communication were crucial to maintaining project momentum and addressing challenges promptly:

- **Steering Committee (SteerCo)**: Held every two to three weeks, these strategic meetings, involving leadership from BCG and the client, focused on validating use cases, reviewing project approaches, and aligning on vision and roadmap.

- **Weekly Demos**: Facilitated regular feedback loops involving product owners, business analysts, development teams, and client representatives to ensure alignment and adapt quickly to feedback and evolving requirements.
- **Ad-Hoc Sessions**: Dedicated weekly time to address emergent challenges, conduct user testing, validate product iterations, and ensure continuous alignment with legal and regulatory frameworks.

1.3.3 Collaboration Tools

Our team utilized various collaborative and productivity-enhancing tools, significantly streamlining communication, documentation, and project management:

- **ClickUp**: Functioned as the central hub for managing diverse project activities including sprint planning, task assignment, documentation storage, issue tracking, and progress monitoring. It facilitated comprehensive documentation through organized folders for engineering, product, and design documentation, as well as centralized spaces for meeting notes and general project insights.

Figure 1.5. ClickUp Logo
[5]

- **Slack**: Facilitated instant communication and collaboration among team members, enabling quick discussions and efficient problem-solving.

Figure 1.6. Slack Logo
[6]

- **Zoom and Microsoft Teams**: Used extensively for virtual meetings, sprint demos, and daily standups, providing robust audio-visual communication capabilities.

Figure 1.7. Zoom and Microsoft Teams Logos
[7] - [8]

- **Microsoft Outlook**: Managed meetings, reminders, and task scheduling, ensuring structured daily and weekly routines.

Figure 1.8. Microsoft Outlook Logo
[9]

- **Miro**: Supported visual collaboration, particularly useful during ideation, design sessions, and feedback workshops.

Figure 1.9. Miro Logo
[10]

1.3.4 Project Timeline and Milestones

The project followed a clearly structured roadmap divided into four primary phases, each critical for incremental delivery and validation:

- **Scoping & Tech Foundations (Weeks 1-3)**: Defined objectives, system design, and initial architecture.
- **Alpha Build (Weeks 3-9)**: Developed the first operational version, enabling preliminary testing and stakeholder feedback.
- **Beta Build (Weeks 9-15)**: Refined the product based on real-world usage, addressing user feedback and enhancing system reliability.

- **Minimum Lovable Product (Weeks 15-21)**: Delivered an enterprise-grade, production-ready platform.

This structured and iterative management approach, supported by Agile Scrum methodologies and robust tool usage, enabled our team to effectively respond to changing requirements, maintain alignment with project objectives, and deliver a comprehensive, innovative solution tailored to enhancing efficiency and compliance in legal operations.

1.4 Conclusion

This chapter introduced BCG and BCG X, framed the project's objectives around Agentic AI for legal workflows, and described the Agile-based management approach. It laid the groundwork for the technical analysis presented in the next chapter.

Chapter 2

Analysis and Specifications

This chapter presents the technical analysis and specifications underlying the contract management solution. It starts with a benchmark of existing tools and justifies the choice of an Agentic AI-based approach. Then, it reviews the core technologies used, including LLMs, LangChain, and LangGraph. The integration of Langfuse and Tiptap is also introduced. Lastly, the chapter details the system's functional and non-functional requirements.

2.1 Comparative Analysis

This section benchmarks existing contract management and legal automation platforms to highlight their features, strengths, and limitations, providing the rationale for developing our specialized Agentic AI solution.

2.1.1 Benchmarking Existing Solutions

To evaluate the landscape of intelligent contract management platforms, we analyzed three prominent solutions: PandaDoc, Harvey AI, and DocDraft. Each platform was selected specifically for its focus on contract management and legal automation capabilities.

2.1.1.1 PandaDoc

PandaDoc is a comprehensive document automation platform designed to streamline the creation, approval, and management of digital documents, including contracts, proposals, and quotes.

Figure 2.1. PandaDoc Logo
[11]

Here are the key features of PandaDoc:

- **Dynamic Contract Templates**: Create and manage reusable templates to ensure consistency and efficiency in document creation.
- **Integrated eSignatures**: Facilitate secure and legally binding electronic signatures within documents.
- **CRM Integrations**: Seamlessly integrate with platforms like Salesforce and HubSpot to synchronize data and streamline workflows.
- **Contract Repository**: Centralized storage for contracts, enabling easy access, tracking, and management.
- **Workflow Automation**: Automate approval processes and notifications to reduce manual intervention and accelerate deal closures.

2.1.1.2 Harvey AI

Harvey AI is a generative AI platform designed for legal professionals, enabling advanced legal research, document drafting, and compliance analysis with the support of large language models.

Figure 2.2. Harvey AI Logo
[12]

Harvey AI's main functionalities include:

- **AI-Powered Legal Research**: Provides fast, accurate legal research across jurisdictions with references and citations.
- **Document Drafting Automation**: Automates the generation and review of legal documents, reducing time and manual effort.
- **Predictive Analytics**: Leverages historical legal data to forecast case outcomes and support strategic decisions.
- **Secure Document Management**: Ensures safe handling and storage of sensitive legal documents.
- **Customizable AI Models**: Allows tailoring of legal AI agents to specific practice areas, workflows, or jurisdictions.

2.1.1.3 DocDraft

DocDraft is an AI-powered contract drafting solution focused on automating the generation, review, and management of legal documents for legal teams and firms.

Figure 2.3. DocDraft Logo
[13]

DocDraft highlights the following core features:

- **Automated Document Drafting**: Quickly generates contracts and legal documents using intelligent pre-built logic.
- **Clause Library**: Offers a rich repository of standard and customizable clauses for efficient contract assembly.
- **Compliance Checks**: Reviews documents to ensure alignment with relevant laws and regulations.
- **Collaboration Tools**: Enables multiple stakeholders to co-author and comment on documents in real-time.
- **Attorney Matching Service**: Connects users to legal experts for document review and legal consultation.

2.1.1.4 Comparative Analysis

The following table summarizes the key features of the three platforms:

Feature	PandaDoc	Harvey AI	DocDraft
Document Automation	Yes	Yes	Yes
eSignature Integration	Yes	No	No
Legal Research Capabilities	No	Yes	No
Predictive Analytics	No	Yes	No
Compliance Checks	Limited	Advanced	Advanced
CRM Integration	Yes	No	No
Collaboration Tools	Yes	Limited	Yes
Customizable AI Models	No	Yes	No
Clause Library	Limited	No	Yes
Attorney Consultation	No	No	Yes

Table 2.1. Imperative vs. Agentic Approach Comparison

2.1.2 Why We Chose to Build a New Solution

The comparative study clearly revealed that existing contract management platforms were not sufficient to meet our needs. Most tools offer limited automation capabilities and lack the flexibility required for complex legal workflows. Additionally, their inability to support deep integration with enterprise systems, limited orchestration features, and lack of control over agent behavior made them unsuitable for large-scale deployment.

To overcome these limitations, we chose to develop a custom solution based on agentic AI. This approach allows us to build a modular, scalable system capable of automating contract drafting, compliance checks, and collaboration, all within a secure and integrated enterprise environment. The goal was not only to improve current processes, but also to lay a solid foundation for future legal automation use cases.

To achieve this, we evaluated two principal approaches for building intelligent automation solutions: the traditional imperative approach and the more advanced declarative (agentic) approach. The imperative approach relies on explicit instructions and manual intervention, while the declarative approach leverages autonomous AI agents capable of dynamically adapting to tasks and goals.

A detailed comparison of these two methodologies, highlighting the specific advantages of the agentic approach, is presented in Table 2.2. This comparison clearly demonstrates why the agentic (declarative) methodology is more aligned with our requirements and vision, guiding our architectural direction.

Aspect	**Imperative Approach** (Traditional)	**Declarative Approach** (Agentic AI)
Definition	Step-by-step instructions to achieve a goal.	Specifies the goal and constraints, allowing AI agents to figure out the best steps.
Control	Explicitly defines every action and decision logic.	AI autonomously determines actions based on high-level goals.
Flexibility	Rigid and requires manual updates for changes.	Adaptive, dynamically adjusting based on new data and conditions.
Example	Writing explicit loops and conditionals to handle a process.	Using AI agents that decide when and how to execute tasks.
Scalability	Limited by human-defined logic; harder to extend.	Easily scales as AI agents can generalize solutions across tasks.
Error Handling	Requires explicit error handling in the code.	AI can self-correct and learn from past interactions.
Human Involvement	Heavy involvement in coding and debugging logic.	Minimal intervention; humans define objectives, and AI executes.
Efficiency	Time-consuming, as every scenario must be manually handled.	AI optimizes execution and improves efficiency over time.
Use Case Example	A chatbot that follows predefined decision trees.	An autonomous AI assistant that adapts to user queries and context dynamically.

Table 2.2. Imperative vs. Agentic Approach Comparison

2.2 Technical Background

This section provides an overview of key technologies enabling the platform's intelligent functionalities.

2.2.1 Large Language Models (LLMs)

2.2.1.1 Overview

Large Language Models (LLMs) have revolutionized natural language processing through transformer-based architectures, extensive computational resources, and vast datasets, achieving near-human performance in various linguistic tasks.

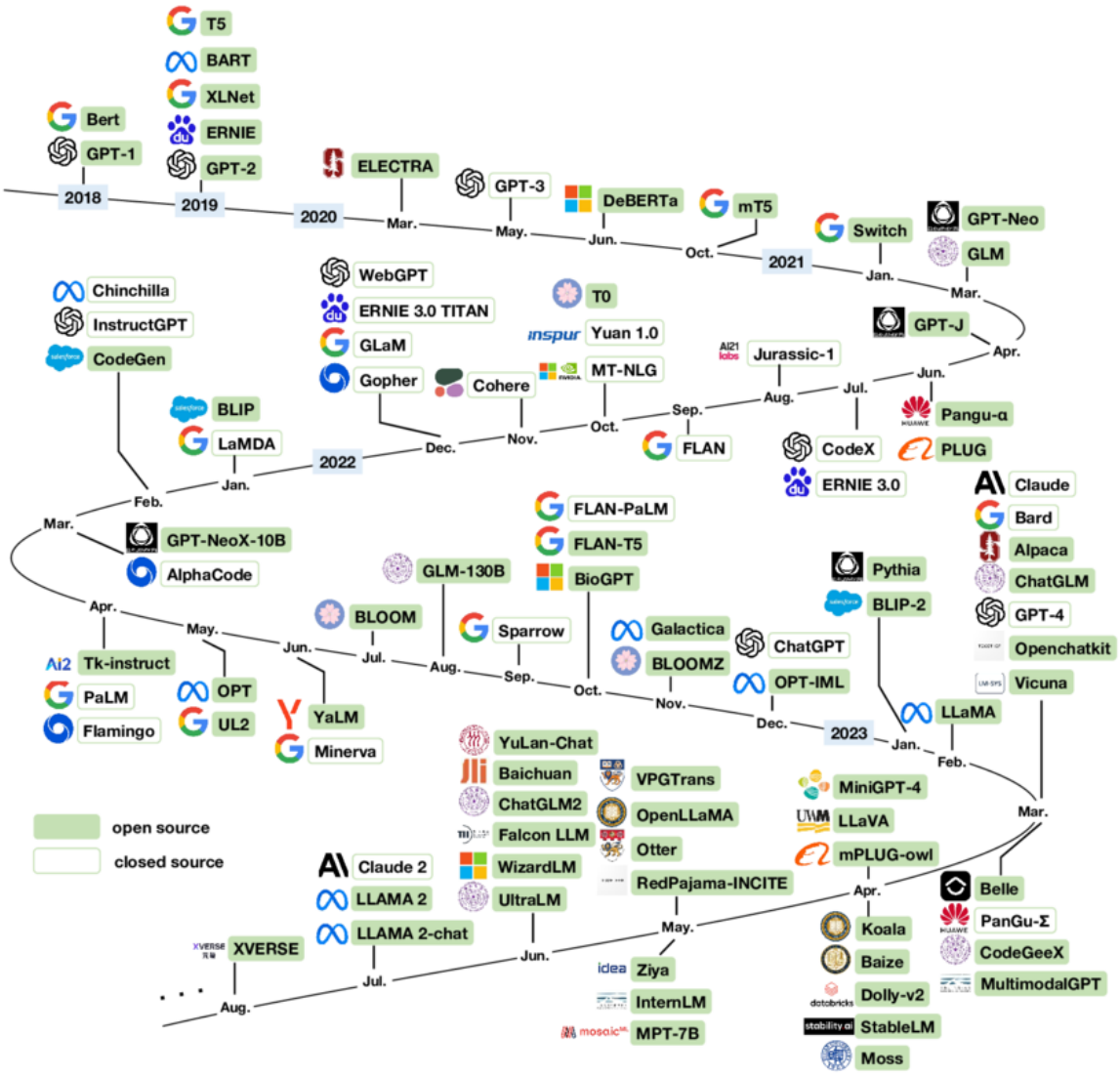

Figure 2.4. Chronological Evolution of Large Language Models (LLMs) [14]

2.2.1.2 Capabilities and Emergent Properties

Large Language Models have significantly evolved, exhibiting capabilities such as reasoning, decision-making, and in-context learning, often emerging spontaneously at scale rather than through explicit programming or training. These emergent properties enable LLMs to effectively generalize across diverse tasks without extensive task-specific data, highlighting their robust adaptability. Techniques such as fine-tuning and prompt engineering further enhance model alignment, ensuring outputs align closely with user intents and expectations.

LLMs' adaptability and versatility have found applications in various sectors including healthcare, finance, legal operations, and customer service, underscoring their transformative potential across industries.

2.2.1.3 Challenges and Limitations

Despite these advancements, LLM deployment is constrained by several critical limitations. Key challenges include:

- **Computational Cost and Efficiency**: The resource-intensive nature of LLM training and inference demands considerable computational power, resulting in high operational costs.
- **Input Size Constraints**: Current LLM architectures typically handle context windows of up to 128k tokens, limiting their ability to directly process large-scale data sources comprising millions of tokens.
- **Inference Speed and Latency**: Longer inputs proportionally increase inference time and computational demands, challenging real-time or large-scale applications.
- **Ethical and Bias Considerations**: Models may inadvertently propagate biases present in training data or generate misleading and potentially harmful outputs, necessitating robust ethical oversight.

2.2.1.4 Future Directions

Addressing these constraints has become an active area of research. Innovations in model compression techniques, such as pruning, quantization, and distillation, aim to enhance efficiency and accessibility of LLMs. Moreover, emerging architectures and retrieval-augmented generation methods offer promising approaches for handling extensive datasets and large input contexts effectively.

Future advancements are expected to mitigate current limitations, further unlocking the full potential of Large Language Models across diverse, complex applications.

2.2.2 Agentic AI

2.2.2.1 Overview

Agentic AI represents a significant evolution in artificial intelligence, transitioning from reactive systems to proactive, autonomous entities capable of setting goals, making decisions, and adapting to dynamic environments. Unlike traditional AI agents that operate within predefined parameters, Agentic AI systems exhibit autonomy, learning, and adaptability, enabling them to handle complex, multi-step tasks with minimal human intervention.

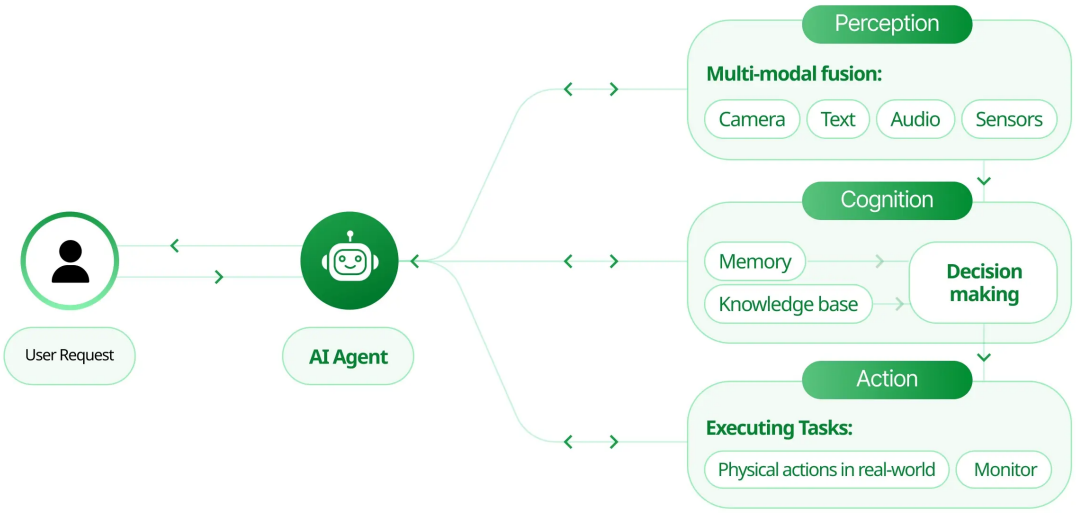

Figure 2.5. Agentic AI System Architecture
[15]

2.2.2.2 Core Characteristics

Agentic AI systems are distinguished by several key features:

- **Autonomy**: They operate independently, making decisions without continuous human oversight.
- **Goal-Oriented Behavior**: These systems can set, pursue, and adjust goals based on environmental feedback.
- **Adaptability**: Agentic AI learns from experiences, refining its strategies to improve performance over time.
- **Complex Decision-Making**: They evaluate multiple options and potential outcomes to make informed decisions.
- **Collaboration**: Agentic AI can coordinate with other agents or systems to achieve shared objectives.

2.2.2.3 Agentic AI vs. Generative AI

While both Agentic AI and Generative AI leverage advanced machine learning techniques, their functionalities differ significantly:

- **Generative AI**: Focuses on creating content (text, images, etc.) based on input data, operating primarily in a reactive manner.
- **Agentic AI**: Emphasizes autonomous decision-making and goal pursuit, enabling proactive interactions with the environment.

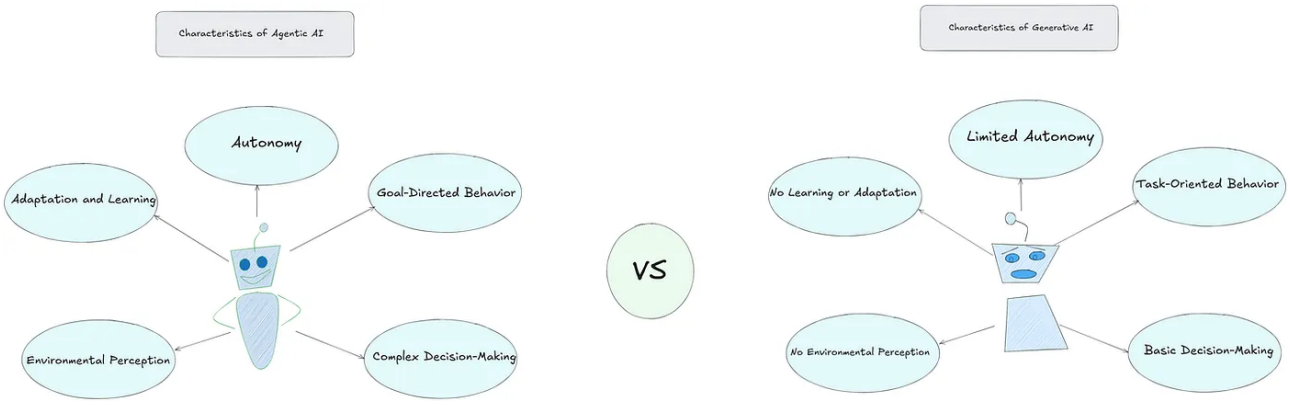

Figure 2.6. Comparison Between Agentic AI and Generative AI
[16]

2.2.2.4 Multi-Agent Systems

Agentic AI often operates within multi-agent systems, where multiple autonomous agents collaborate to solve complex problems. These systems benefit from:

- **Specialization**: Agents can focus on specific tasks, enhancing efficiency.
- **Scalability**: Systems can be expanded by adding more agents to handle increased complexity.
- **Robustness**: Collaboration among agents can compensate for individual agent failures or limitations.

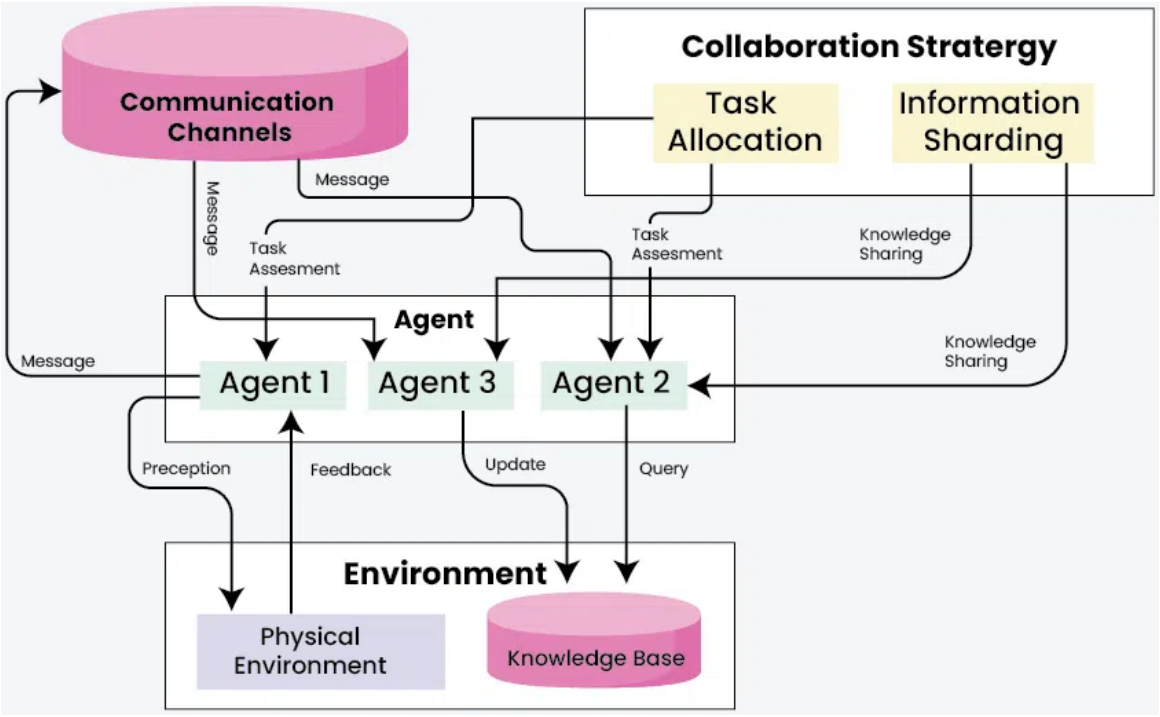

Figure 2.7. Multi-Agent System Collaboration
[17]

2.2.2.5 Challenges and Future Directions

Despite its potential, Agentic AI faces several challenges:

- **Ethical Considerations**: Ensuring decisions align with human values and societal norms.
- **Transparency**: Making decision-making processes understandable to users.
- **Security**: Protecting systems from malicious manipulation or unintended behaviors.
- **Integration**: Seamlessly incorporating Agentic AI into existing infrastructures.

Ongoing research aims to address these challenges, focusing on developing frameworks for ethical AI, enhancing interpretability, and establishing standards for safe deployment.

2.2.3 LangChain and LangGraph

2.2.3.1 Overview

LangChain and LangGraph are complementary frameworks designed to facilitate the development of applications powered by Large Language Models (LLMs). While LangChain provides a modular approach to constructing sequential workflows, LangGraph introduces a graph-based paradigm, enabling the creation of complex, dynamic, and stateful AI systems.

2.2.3.2 LangChain: Modular Workflow Construction

LangChain is an open-source framework that simplifies the integration of LLMs into applications by allowing developers to build chains—sequences of calls to LLMs and other utilities. Its modular design supports the composition of various components such as prompt templates, memory modules, and tool integrations. This structure is particularly effective for tasks that follow a linear progression, such as document summarization, question answering, and conversational agents.

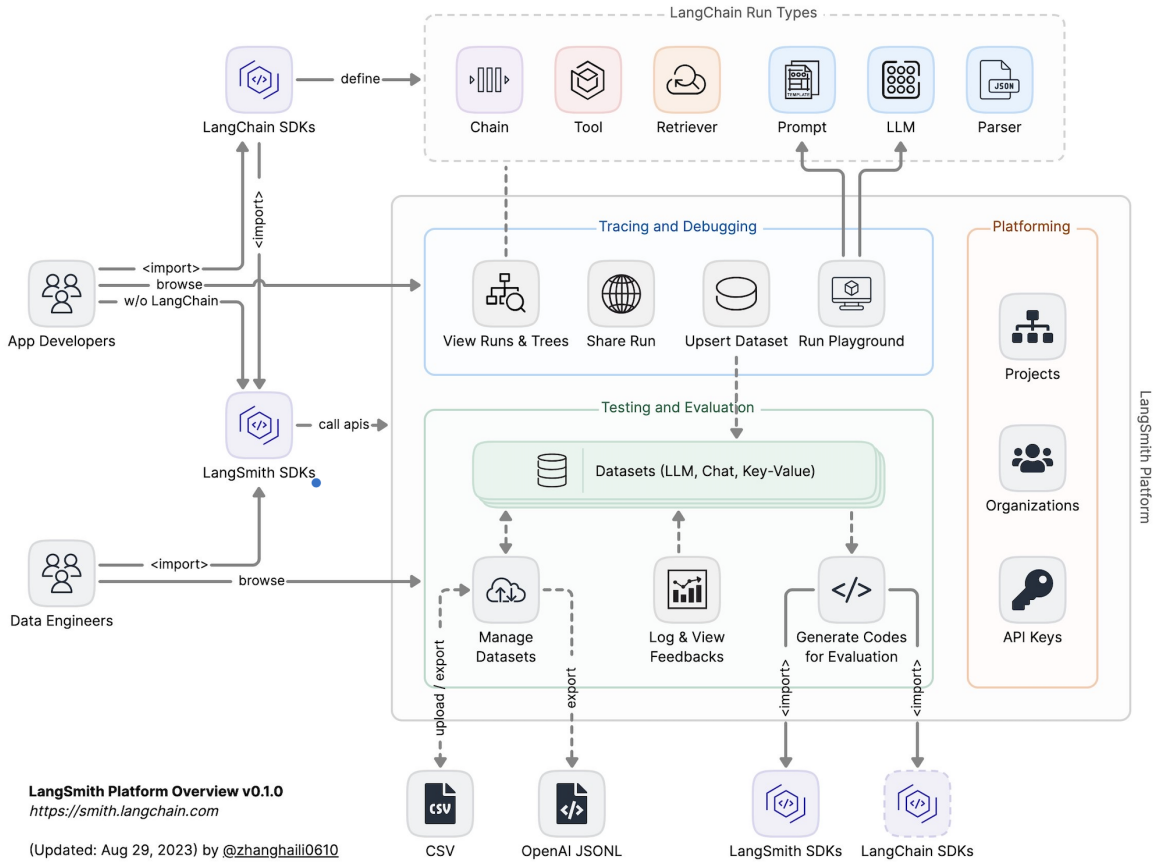

Figure 2.8. LangChain Architecture
[18]

Key features of LangChain include:

- **Prompt Templates**: Facilitate the creation of dynamic prompts for LLMs.
- **Chains**: Enable the linking of multiple components to form a cohesive workflow.
- **Agents**: Allow for decision-making capabilities by selecting appropriate actions based on user input.
- **Memory**: Maintain context across interactions to support stateful conversations.

2.2.3.3 LangGraph: Graph-Based Workflow Orchestration

Building upon the foundations of LangChain, LangGraph introduces a graph-based approach to workflow orchestration, allowing for the modeling of complex, non-linear, and cyclical processes. In LangGraph, workflows are represented as directed graphs comprising nodes (representing operations or agents) and edges (defining the flow between nodes). This structure is particularly advantageous for applications requiring iterative processing, conditional branching, and multi-agent coordination.

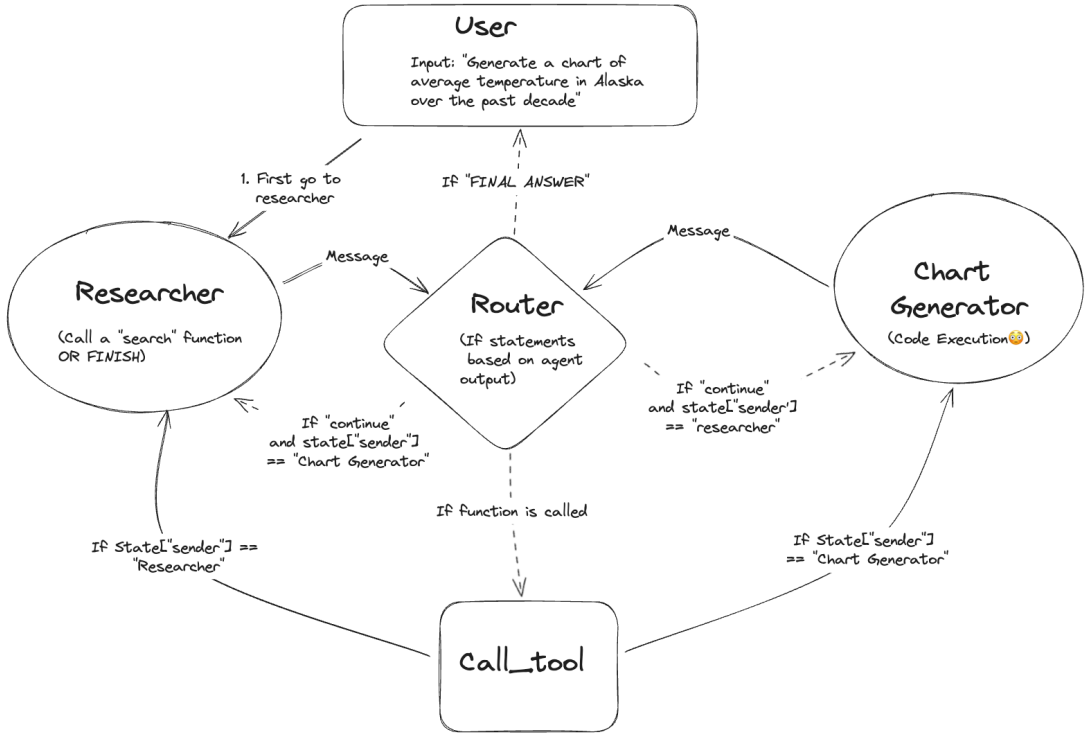

Figure 2.9. LangGraph: Multi-Agent Workflows
[19]

Notable capabilities of LangGraph include:

- **Cyclic Workflows**: Support for loops and iterative processes within workflows.
- **Conditional Routing**: Dynamic decision-making to determine the next node based on runtime conditions.
- **State Management**: Persistent tracking of the system's state throughout the workflow execution.
- **Multi-Agent Collaboration**: Facilitation of interactions between multiple agents, each with specialized roles.

2.2.3.4 Comparative Analysis

LangChain and LangGraph streamline the development of LLM-powered applications but address varying complexity levels. LangChain is optimal for linear workflows with straightforward tasks and limited iterative processes. In contrast, LangGraph's dynamic, graph-based structure efficiently manages more intricate workflows involving conditional logic, iterative loops, and robust multi-agent coordination.

Feature	LangChain	LangGraph
Workflow Structure	Sequential chains	Dynamic graphs
Complexity Handling	Simple/moderate workflows	Moderate/complex workflows
Iterative Processing	Limited	Fully supported (loops)
Conditional Logic	Basic conditions	Advanced conditions
Multi-Agent Coordination	Basic support	Robust collaboration
State Management	Basic retention	Advanced tracking
Use Case Suitability	Straightforward tasks	Complex adaptive tasks
Flexibility	Rigid structure	Highly flexible
Debugging and Monitoring	Good support	Enhanced visibility
Real-time Decision Making	Limited	Extensive support
Ideal Applications	Simple Q&A, summarization	Complex multi-agent apps

Table 2.3. Comparison of LangChain and LangGraph

The choice between these frameworks thus depends primarily on the complexity of the intended application, where LangChain offers simplicity for linear scenarios, while LangGraph is suited for adaptive, complex applications requiring enhanced flexibility and sophisticated state management.

2.2.4 Langfuse and Observability

2.2.4.1 Overview

Langfuse is an open-source observability platform designed specifically for applications powered by Large Language Models (LLMs). It provides developers with tools to trace, debug, and analyze the behavior of LLM applications, ensuring better performance and reliability.

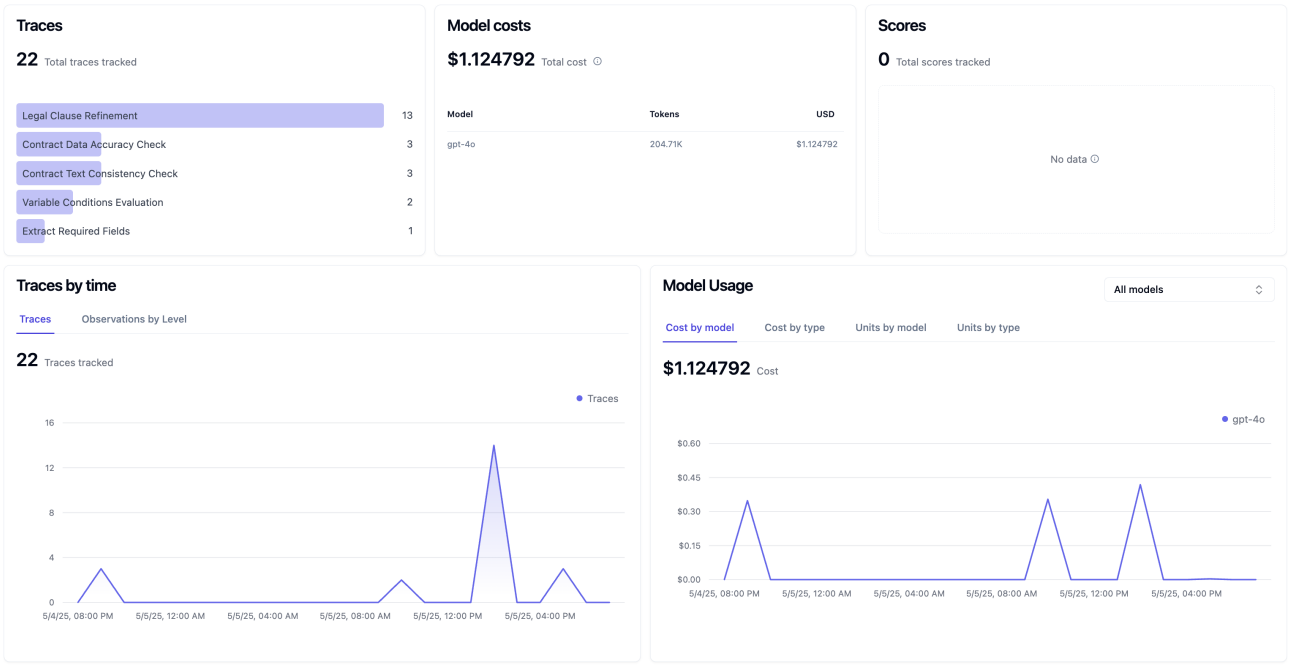

Figure 2.10. Langfuse Dashboard

2.2.4.2 Key Features

- **Comprehensive Tracing**: Langfuse captures detailed traces of LLM operations, including prompts, responses, and intermediate steps, allowing developers to understand the flow of data and identify issues effectively.
- **Session Management**: It groups related interactions into sessions, providing a holistic view of multi-turn conversations or complex workflows.
- **Analytics Dashboard**: Langfuse offers dashboards that display metrics such as latency, token usage, and cost, enabling teams to monitor and optimize application performance.
- **Prompt Management**: The platform includes tools for managing and versioning prompts, facilitating experimentation and iterative development.
- **Integration Support**: Langfuse integrates seamlessly with popular frameworks like LangChain, LlamaIndex, and OpenAI SDKs, as well as supports OpenTelemetry for broader observability needs.

2.2.4.3 Deployment Options

Langfuse can be deployed in various environments:

- **Cloud Hosting**: Utilize Langfuse's managed cloud service for quick setup and scalability.
- **Self-Hosting**: Deploy Langfuse on-premises or in private clouds using Docker, providing full control over data and configurations.

2.2.4.4 Use Cases

- **Debugging Complex Workflows**: By tracing each step of LLM operations, developers can pinpoint and resolve issues in complex applications.
- **Performance Monitoring**: Track metrics to identify bottlenecks and optimize resource usage.
- **Quality Assurance**: Analyze user interactions and feedback to improve response quality and user satisfaction.
- **Compliance and Auditing**: Maintain detailed logs for auditing purposes, ensuring compliance with regulatory standards.

Incorporating Langfuse into the development lifecycle of LLM applications enhances transparency, reliability, and efficiency, making it an invaluable tool for teams aiming to build robust AI solutions.

2.2.5 Tiptap for Rich Text Editing

2.2.5.1 Overview

Tiptap is a headless, open-source rich text editor framework built on ProseMirror, designed for developers seeking full control over their content editing interfaces. Its modular architecture and extensive extension ecosystem make it a powerful tool for building customized, collaborative, and AI-enhanced editing experiences.

Figure 2.11. Tiptap Editor
[20]

2.2.5.2 Architecture and Core Concepts

Tiptap is a headless, framework-agnostic rich text editor built on top of ProseMirror, designed to provide developers with full control over the editor's functionality and appearance. Its architecture is modular and extensible, allowing for the creation of customized editing experiences.

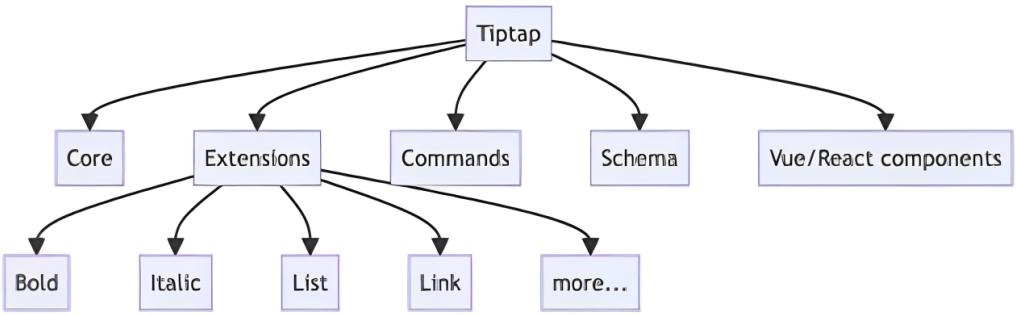

Figure 2.12. Tiptap Architecture

Key Architectural Components:

- **Schema**: Defines the structure of the document, specifying the allowed nodes (e.g., paragraphs, headings) and marks (e.g., bold, italic). This strict schema ensures content consistency and predictability.
- **State**: Represents the current content and selection within the editor. It is immutable and updated through transactions, enabling features like undo/redo and collaborative editing.
- **Transaction**: Encapsulates changes to the state, such as text insertions or formatting. Transactions ensure that state changes are predictable and manageable.
- **Extensions**: Modular units that add functionality to the editor, such as new nodes, marks, commands, or plugins. Tiptap provides a rich set of core and community extensions, and developers can create custom ones as needed.
- **Commands**: Functions that perform actions within the editor, often used to manipulate the document or respond to user input. Commands can be chained for complex operations.

2.2.5.3 Key Features

- **Extension-Based Modularity**: Tiptap offers over 100 extensions, including both open-source and Pro options, allowing developers to tailor the editor's functionality to specific needs.
- **Real-Time Collaboration**: Through integrations like Hocuspocus and CRDTs, Tiptap supports collaborative editing with features like live cursors and offline synchronization.
- **AI Integration**: The Content AI extension enables in-editor AI capabilities such as text suggestions, translations, and content generation, enhancing the writing experience.
- **Framework Agnostic**: Tiptap's design allows it to be integrated into various JavaScript frameworks, providing flexibility in application development.

2.2.5.4 Use Cases

Tiptap is suitable for a wide range of applications, including:

- Building Notion-like editors with custom blocks and interactions.

- Developing collaborative document editing platforms.
- Creating AI-assisted writing tools with real-time suggestions.
- Implementing rich text editors in CMS platforms.

Tiptap stands out as a versatile and developer-friendly rich text editor framework. Its combination of headless architecture, modular extensions, and support for real-time collaboration and AI integration makes it a compelling choice for modern web applications requiring customized content editing solutions.

2.3 Specification of Requirements

Given the agile nature of the project, the actors as well as functional and non-functional requirements are subject to evolve. The elements below reflect the current state of the project.

2.3.1 Business Architecture

The business architecture depicted in Figure 2.13 outlines the overarching structure and planned components of the intelligent contract management platform. The platform integrates three main components supported by robust underlying layers of data management, security, observability, and tool integration capabilities.

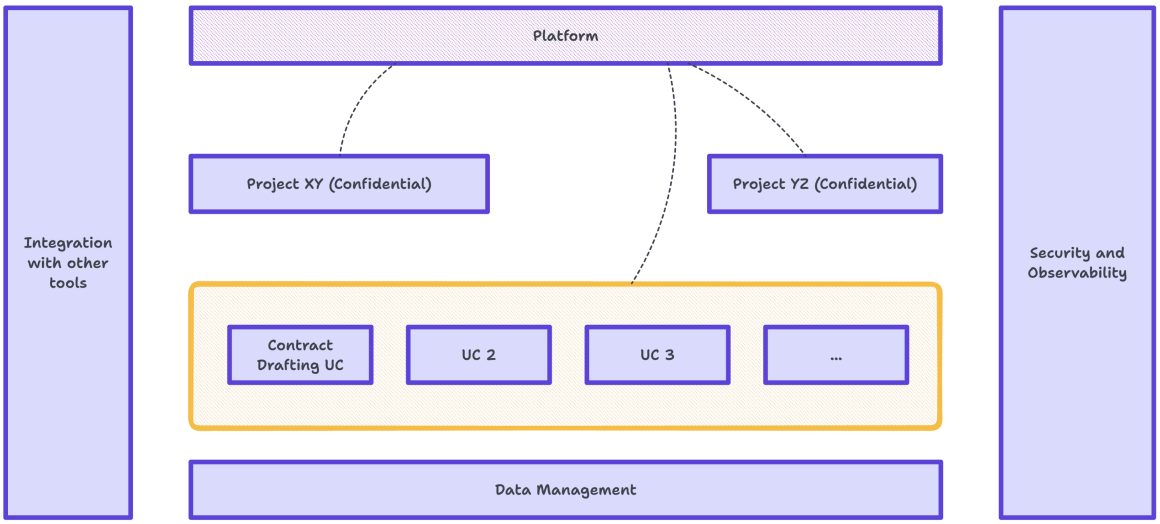

Figure 2.13. Business Architecture Overview

The platform structure is organized into three core areas:

- **Cockpit (Steering and Monitoring)**: This area provides oversight and real-time analytics, enabling stakeholders to monitor system performance, track usage, and manage workflows effectively.
- **Agentic Marketplace**: Designed to host various autonomous AI agents, this component facilitates the scalable deployment and orchestration of intelligent services, supporting diverse, dynamic business needs and agent interactions.

- **Use Case Features**: These represent modular functionalities tailored to specific business processes. The primary use case currently prioritized is the **Contract Drafting Use Case (UC)**, scheduled for completion within a 21-week timeframe. This use case is central to my internship project and focuses on automating and enhancing the contract drafting process through intelligent clause selection, AI-assisted drafting, compliance verification, and collaboration enhancements.

These foundational components are encapsulated within a broader environment of:

- **Data Management**: Ensures consistent data governance, quality, and efficient access to support the platform's intelligent features.
- **Integration with Other Tools**: Provides seamless interoperability with external and internal enterprise applications, extending the platform's capabilities and ensuring a cohesive user experience.
- **Security and Observability**: Delivers comprehensive security measures and observability tools to protect data integrity, ensure compliance, and provide transparent system monitoring and diagnostics.

This structured business architecture aims to offer a scalable, secure, and integrative platform capable of addressing complex legal operations through advanced technological solutions.

2.3.2 Contract Structure and Domain Terminology

In the context of this platform, a **contract** refers to a formal legal document defining the obligations and expectations between two parties. Structurally, each contract is composed of **articles**, which are referred to in our system as **sections**. Each section can contain one or more **clauses**, which are modeled as **subsections** of a section. Both sections and subsections are technically treated as hierarchical `Section` objects within the platform's data model.

A **contract template** serves as a predefined document structure containing static legal content along with dynamic placeholders (variables) that must be completed by the user based on the context or client-specific information.

Contracts are also categorized by **type**, such as:

- **Spot contracts** (short-term agreements)
- **Long-term contracts** (ongoing or renewable arrangements)

Additionally, each contract is associated with an **Incoterm**—an international commercial term standardizing shipping responsibilities. For example, **FOB (Free On Board)** means the seller delivers the goods on board the vessel chosen by the buyer, whereas **CFR (Cost and Freight)** means the seller pays for transport to the destination port, but the risk is transferred once goods are on the ship.

2.3.3 Actor Identification

An actor specifies a role played by a user or any other system interacting with the solution. Actors are always external to the system, interacting by initiating a use case, providing inputs, or receiving outputs. In the context of our platform, we have two types of actors:

Actor	Role
Sales User	Responsible for creating and managing contract documents and negotiations with clients.
Legal User	Oversees contract compliance, manages legal clauses, and reviews contracts.
Administrator	Manages platform users, roles, permissions, and oversees system configurations.

Table 2.4. Actors and Their Roles

2.3.4 Functional Requirements

2.3.4.1 Authentication

The application must enable users to authenticate using a username or email and password.

2.3.4.2 Sales User Functionalities

Contract Generation:

- The sales user must be able to create a new contract.
- The sales user must be able to create a new client.

Contract Editing:

- The Sales user must be able to access the rich-text contract editor.
- The Sales user must be able to create clause requests.
- The Sales user must be able to modify contract variables.
- The Sales user must be able to track changes across different contract versions.

Contract Review & Compliance:

- The Sales user must be able to trigger AI-powered compliance checks.
- The Sales user must be able to change the contract status (e.g., in editing, in review, finalized).

Export, Sharing, and Management:

- The Sales user must be able to access only the contracts they have created.
- The Sales user must be able to export contracts as downloadable files (Word, PDF).

- The Sales user must be able to share contracts via secure links or email attachments.
- The Sales user must be able to archive and delete contracts.

2.3.4.3 Legal User Functionalities

Clause Management:

- The Legal user must be able to review clause requests from Sales.
- The Legal user must be able to refine clause content using AI assistance.
- The Legal user must be able to create, insert, modify, and delete legal clauses.

Contract Editing and Versioning:

- The Legal user must be able to access and create versions of contracts.
- The Legal user must be able to restore previous contract versions.
- The Legal user must be able to edit contracts using a rich-text editor.
- The Legal user must be able to modify contract variables.

Contract Review & Compliance:

- The Legal user must be able to trigger AI-powered compliance checks.
- The Legal user must be able to change the status of a contract (e.g., in editing, in review).

Export, Sharing, and Management:

- The Legal user must be able to access all existing contracts.
- The Legal user must be able to export contracts as downloadable files (Word, PDF).
- The Legal user must be able to share contracts via secure links or email attachments.

2.3.4.4 Administrator Functionalities

User Management:

- The administrator must be able to add, modify, or remove platform users.
- The administrator must be able to manage role-based access permissions.

System Configuration:

- The administrator must be able to configure integration settings and security parameters.
- The administrator must be able to monitor system performance through administrative dashboards.

Note: During this phase of the project, the administrator role had limited operational access to the platform. Specifically, client creation was only accessible through the contract creation workflow, primarily intended for Sales users. Direct client management by administrators had not yet been implemented or scoped as part of the sprint's functional coverage.

2.3.5 Non-Functional Requirements

- **Usability**: High intuitiveness and ease of use for Sales, Legal, and Admin users, promoting efficient adoption across roles.
- **Maintainability**: Modular and well-documented architecture to facilitate smooth updates, debugging, and feature evolution.
- **Security**: Robust access control, secure data handling, and full compliance with applicable legal and organizational standards.
- **System Responsiveness**: Fast UI response, low-latency AI processing, and efficient data retrieval for optimal user experience.
- **Scalability**: Capacity to handle increased user load and data volume while maintaining system performance.

2.3.6 Use Case Diagram

The Use Case Diagram presented in Figure 2.14 visually represents the interactions between the identified actors—Sales Users, Legal Users, and Administrators—and the main functionalities offered by the intelligent contract management platform. It illustrates how each user group engages with distinct and shared system features, clearly depicting the overall workflow from contract drafting and editing to compliance verification, version control, and collaboration activities.

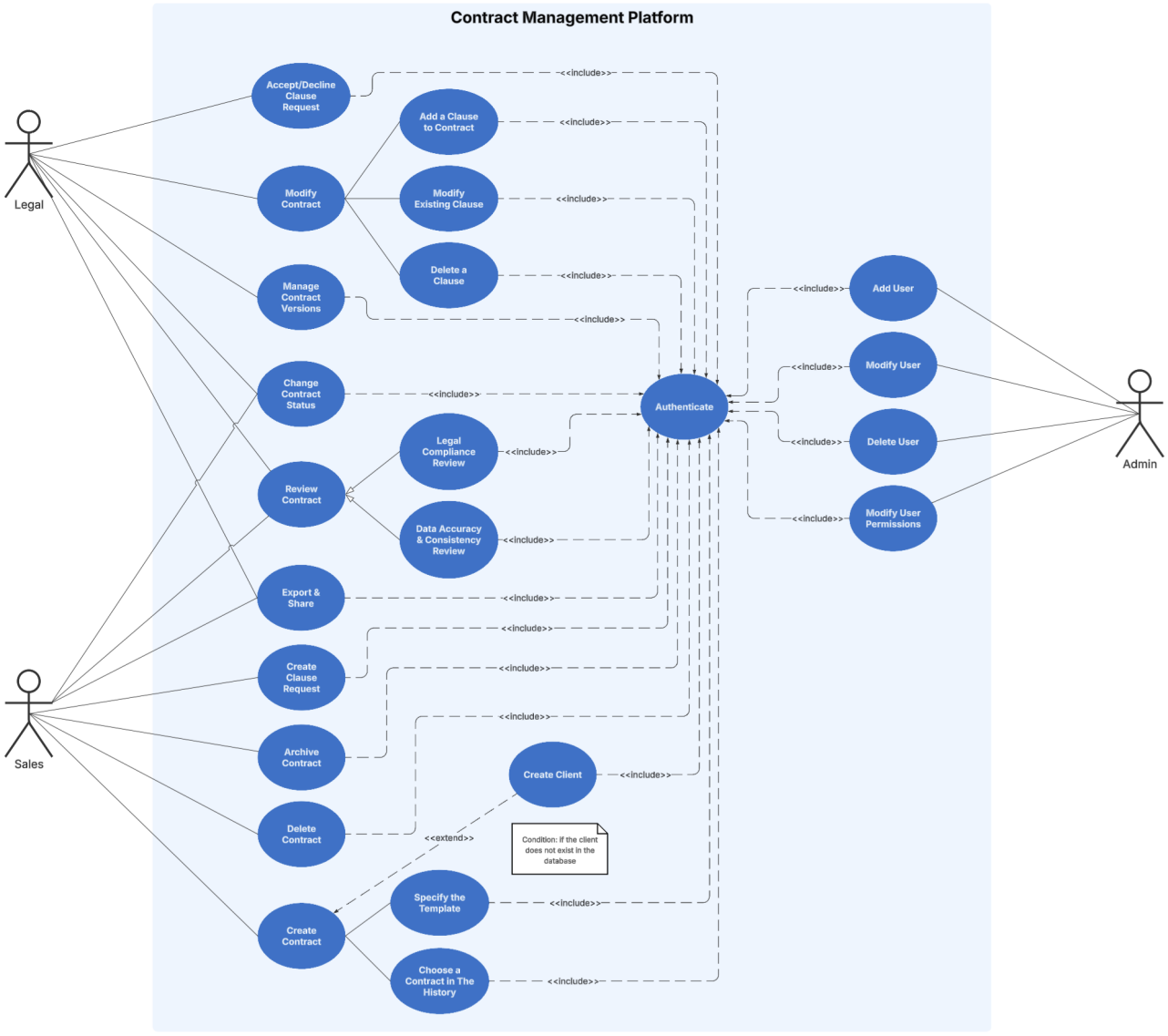

Figure 2.14. Use Case Diagram

2.3.7 Textual Description

To describe the dynamics of a use case, it is essential to enumerate all interactions between the actor and the system in textual form. This description is crucial, as it facilitates clear communication with all stakeholders and ensures a shared understanding of the business terminology involved.

Textual Description of Use Case: Create Contract

UC 1: Create Contract	
Actor	Sales User
Goal	Create a new contract
Preconditions	User must be authenticated
Postconditions	Successful creation of a contract draft
Nominal Scenario	1. User accesses the Contract Repository page. 2. User clicks on "Generate a New Contract." 3. User selects one of the following input methods: voice prompt, image upload, or text prompt. 4. System extracts key contract information. 5. If a new client or mismatched client information is detected, the system prompts the user for confirmation. 6. User confirms key information. 7. System automatically generates the contract step-by-step.
Alternative Scenario	Missing required information – Display an alternative message indicating the specific missing information.

Table 2.5. Textual Description of Use Case: Create Contract

Textual Description of Use Case: Create Clause Request

UC 2: Create Clause Request	
Actor	Sales User
Goal	Create a clause request to add, modify, or delete clauses
Preconditions	User must be authenticated
Postconditions	Clause request submitted and pending Legal User action
Nominal Scenario	1. User accesses the Contract Editing page. 2. User initiates a clause request from the left panel in the clause area or by right-clicking on an article, clause, or sub-clause. 3. User enters the description of the request. 4. User optionally refines the request description using an LLM-powered "Refine" button. 5. User submits the clause request. 6. System notifies the Legal User of the submitted clause request.
Error Scenario	Invalid or incomplete request – System displays an error message indicating required corrections.

Table 2.6. Textual Description of Use Case: Create Clause Request

Textual Description of Use Case: Review Contract

UC 3: Review Contract	
Actor	Legal User or Sales User
Goal	Review a contract for accuracy, consistency, and legal security
Preconditions	User must be authenticated
Postconditions	Contract reviewed with provided suggestions and quality score
Nominal Scenario	1. User accesses the Editor View (Contract Editing page). 2. User selects "Review" in the right panel. 3. User chooses between "Accuracy and Consistency" and "Legal Security" review options. 4. User initiates the review process. 5. System completes the review and provides a quality score (\%). 6. System presents suggestions generated by the LLM for the user to accept or discard.
Error Scenario	Review fails or incomplete – System displays a message indicating the review error and required next steps.

Table 2.7. Textual Description of Use Case: Review Contract

2.3.8 Analysis Sequence Diagram

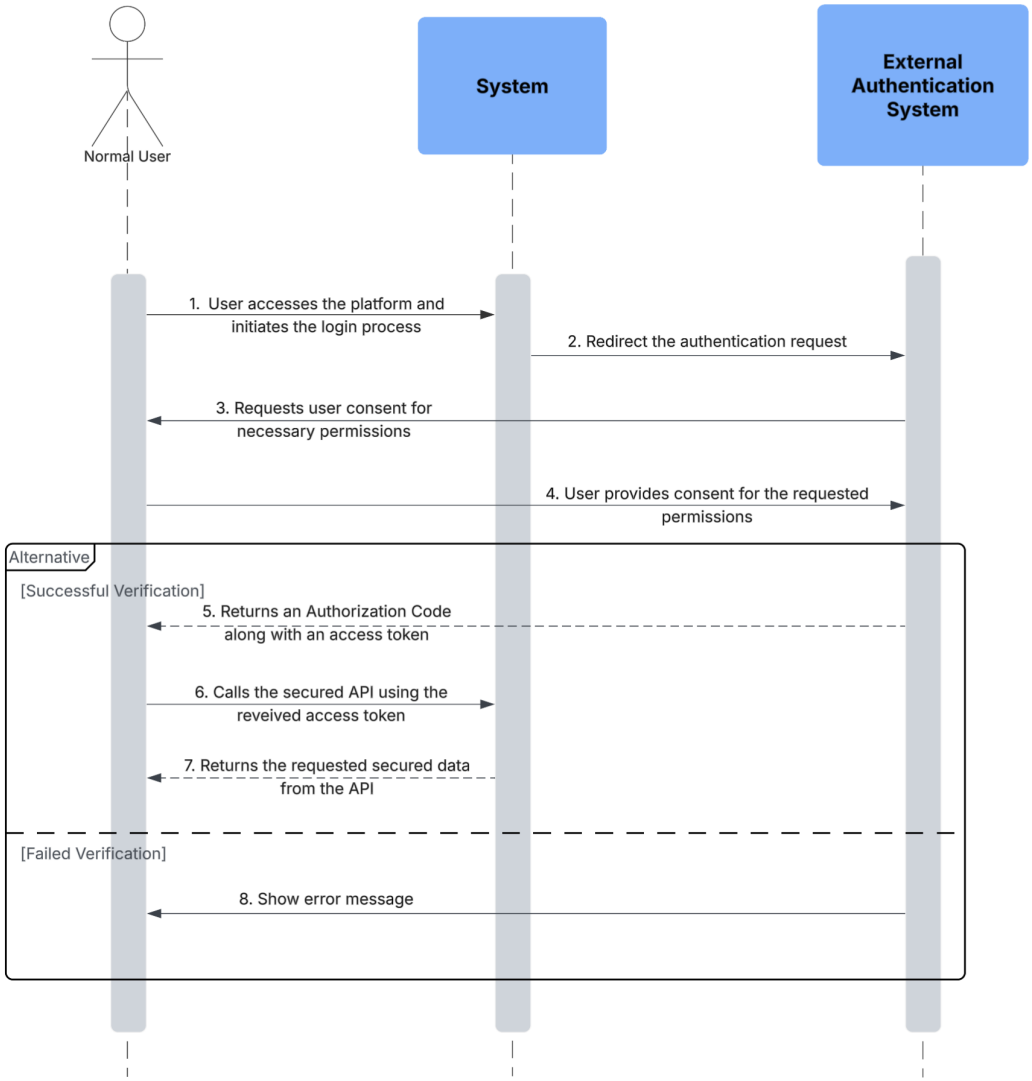

Figure 2.15. Analysis Sequence Diagram

The diagram presented above outlines the interaction flow between a normal user intending to authenticate, the main system, and an external authentication system. Initially, the user accesses the platform and clicks on the login button, which prompts the system to redirect the authentication request to the external authentication system. The external authentication system then requests user consent for specific permissions. Upon granting consent by the user, the authentication system issues an authorization code along with an access token back to the system. Finally, the system utilizes this access token to securely call the API and retrieve the requested secured data.

2.3.9 Constraints and Assumptions

To clearly delineate the operational environment and development boundaries, the following constraints and assumptions have been identified:

Constraints:

- **Enterprise Security Compliance**: The system must strictly comply with the organization's security policies, standards, and regulatory frameworks, ensuring the protection and confidentiality of sensitive contractual and user information.
- **Infrastructure Compatibility**:All technical solutions must be compatible with existing enterprise systems, particularly Azure-based services such as Azure Kubernetes Service (AKS), Azure Entra ID, and Azure PostgreSQL, to ensure seamless integration and minimize disruptions.
- **Cost Efficiency**:The platform must be developed and operated within the allocated budget constraints, adhering to the capacity planning forecasts to optimize resource utilization and cost-effectiveness.

Assumptions:

- **Data Availability and Quality**: It is assumed that structured and high-quality data is consistently available from internal sources and third-party integrations to ensure effective AI-driven decision-making.
- **Reliable Cloud Infrastructure**: It is assumed that the Azure cloud infrastructure provides continuous, high-performance service with minimal downtime, ensuring uninterrupted user experiences and workflows.
- **Stable Internet Connectivity**: It is assumed that end users have stable and continuous internet connectivity, enabling consistent access to cloud-hosted services.
- **User Proficiency**: Users are assumed to have a basic level of digital proficiency, enabling them to interact effectively with the intuitive yet sophisticated interfaces provided by the system.
- **Stakeholder Collaboration**: Continuous engagement from all stakeholders, including legal experts, technical teams, and business users, is expected to ensure alignment and smooth progression throughout the project's lifecycle.

2.4 Conclusion

This chapter detailed the technical rationale behind the project's approach and identified key enabling technologies. It also defined the system requirements that guide the platform's architecture, which will be examined in the following chapter.

Chapter 3

System Architecture and Design

This chapter explores the detailed architecture and software design of the intelligent contract management platform. It begins with a high-level overview of the system's structure, covering frontend, backend, and AI components. The integration of LLMs via LangChain and advanced editing through Tiptap is also discussed. The logical designs of backend and frontend modules are illustrated. Finally, the chapter presents UML diagrams to clarify class structures and workflow interactions.

The architectural elements and components presented in this chapter reflect the broader system context in which the internship was carried out; however, the descriptions are limited to areas in which I was directly involved or contributed during the project.

3.1 Solution's Architecture

3.1.1 Overall Architecture

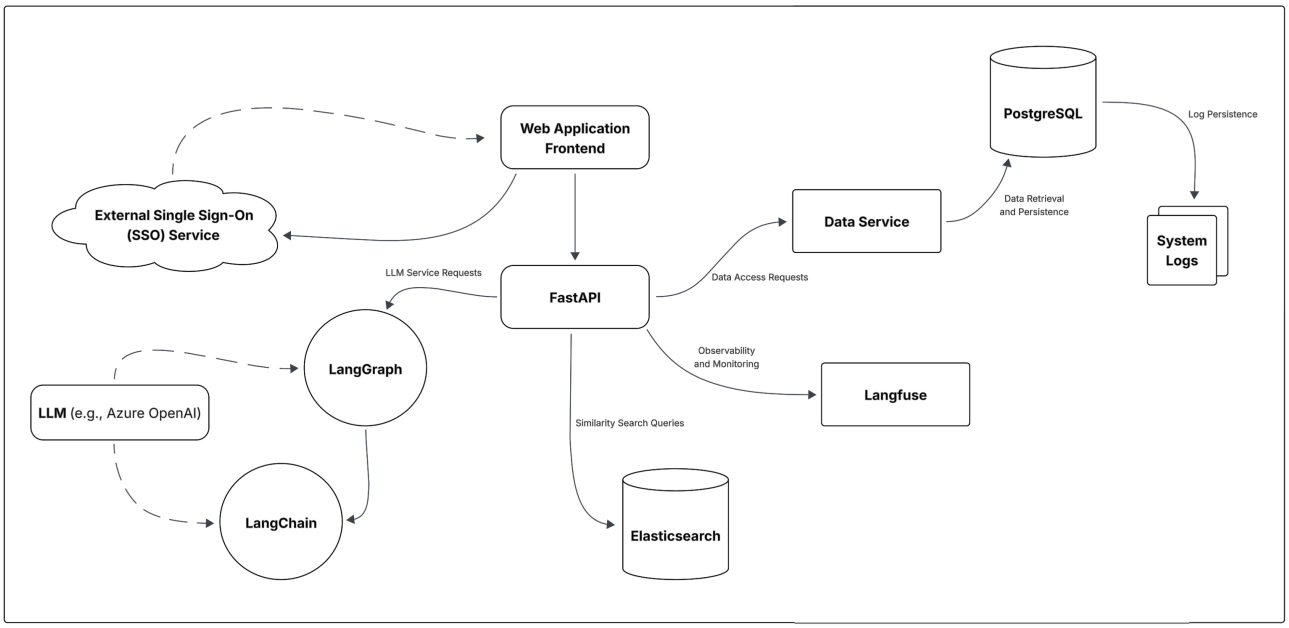

Figure 3.1. Overall Architecture of the Platform

The overall architecture of the platform, as illustrated in Figure 3.1, depicts a structured interaction between users, frontend interfaces, backend services, databases, and AI-driven components. Users interact with the platform primarily through the web application frontend. Depending on the access requirements, users may need to authenticate via an external Single Sign-On (SSO) service to securely access private functionalities or can directly access public features like shared links through URL-based token authentication.

The frontend communicates with the backend services built using the FastAPI framework, where the core business logic resides. This backend interacts extensively with a PostgreSQL database to persist logs and manage structured data efficiently. Elasticsearch is implemented as a vector database responsible for similarity searches, enhancing the platform's data retrieval capabilities.

Interaction with various Large Language Models (LLMs)—including Azure OpenAI, OpenAI GPT-4.1, and Anthropic—is managed via the LangChain framework, abstracting and simplifying communications and integration. LangGraph extends LangChain's capabilities, facilitating the execution of complex, graph-based AI agent workflows, thereby enabling more dynamic interactions and decision-making processes within the platform.

For observability and monitoring purposes, the system integrates Langfuse. Langfuse provides comprehensive tracking of all interactions with LLM APIs, including essential metrics such as API response times, token usage, and associated costs. This observability ensures robust management and operational transparency of the AI services integrated within the platform.

3.1.2 Logical Architecture

This section presents the internal structure of the application codebase. Given that the core mission focused on the implementation of new features across both the backend service and frontend, the focus here will be on the logical design of these two key components.

3.1.2.1 Main Backend Service

In the FastAPI-based backend, we adopted a layered modular architecture for each functional domain. As illustrated in Figure 3.2, each module is composed of a controller responsible for managing HTTP and WebSocket requests, which communicates with a service layer that encapsulates the business logic. The service interacts with the data models and relies on DTOs (Data Transfer Objects) for structured data exchanges across layers.

To abstract and streamline access to persistent storage, a DAO (Data Access Object) pattern is implemented. The DAO interfaces with a dedicated Data Service layer that encapsulates direct database operations. This clear separation of responsibilities ensures high maintainability, testability, and scalability. It also allows for improved consistency in query logic and facilitates secure data access patterns.

This architecture was particularly beneficial for implementing legal use cases and AI-enhanced features, where the backend needed to efficiently orchestrate data retrieval, validation, and transformation between internal components and the frontend interface.

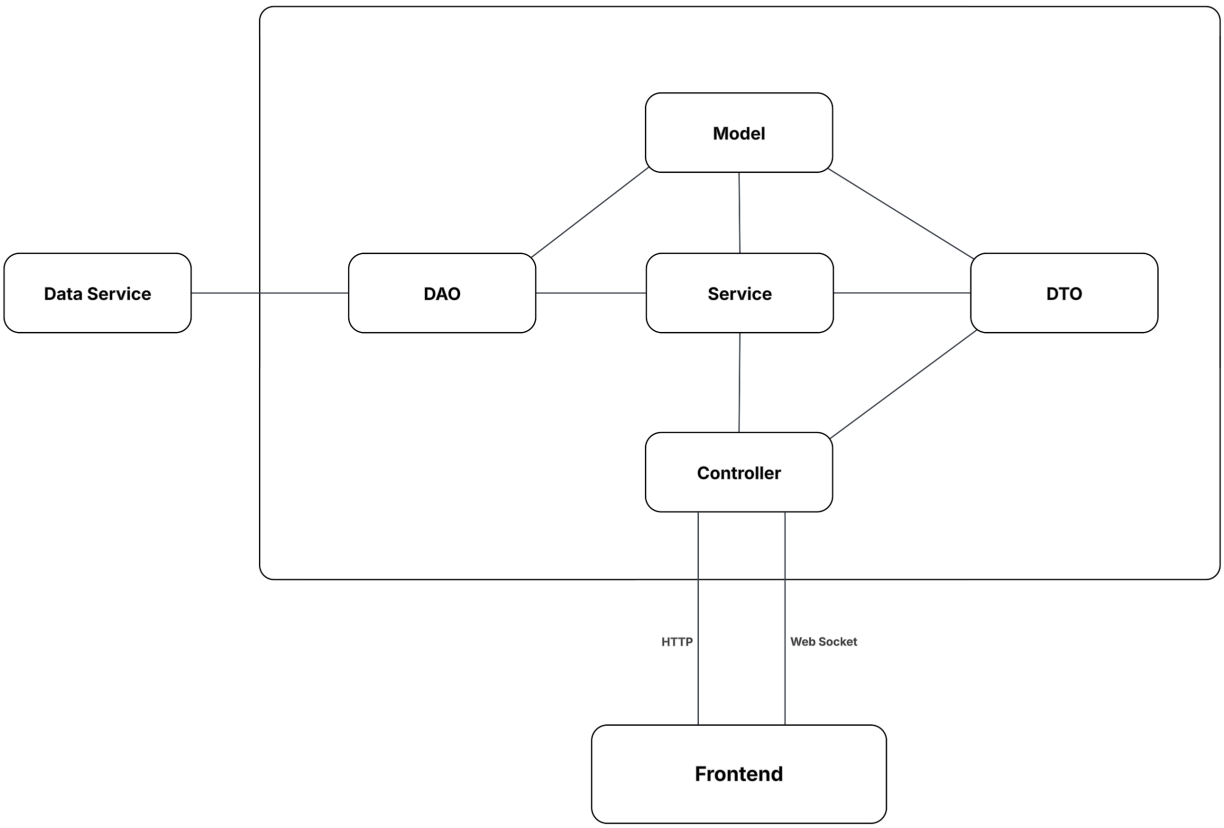

Figure 3.2. Layered Architecture of a Backend Module

3.1.2.2 Frontend Structure

On the frontend side, we implemented a modular structure grounded in the principles of separation of concerns and component reusability. As shown in Figure 3.3, each module consists of components responsible for managing the user interface and interactions, and services that encapsulate domain-specific business logic and handle communication with the backend via HTTP or WebSocket, depending on the context.

The architecture also integrates a dedicated store layer to manage state, especially for dynamic content and session-persistent interactions. DTOs are used to structure and validate data exchanged with the backend, while models represent the internal structure of application entities.

This modular organization promotes code maintainability and scalability, enabling independent development and testing of each module. It also facilitates a smooth integration of new features such as AI-based clause recommendations and real-time collaboration within the contract editor.

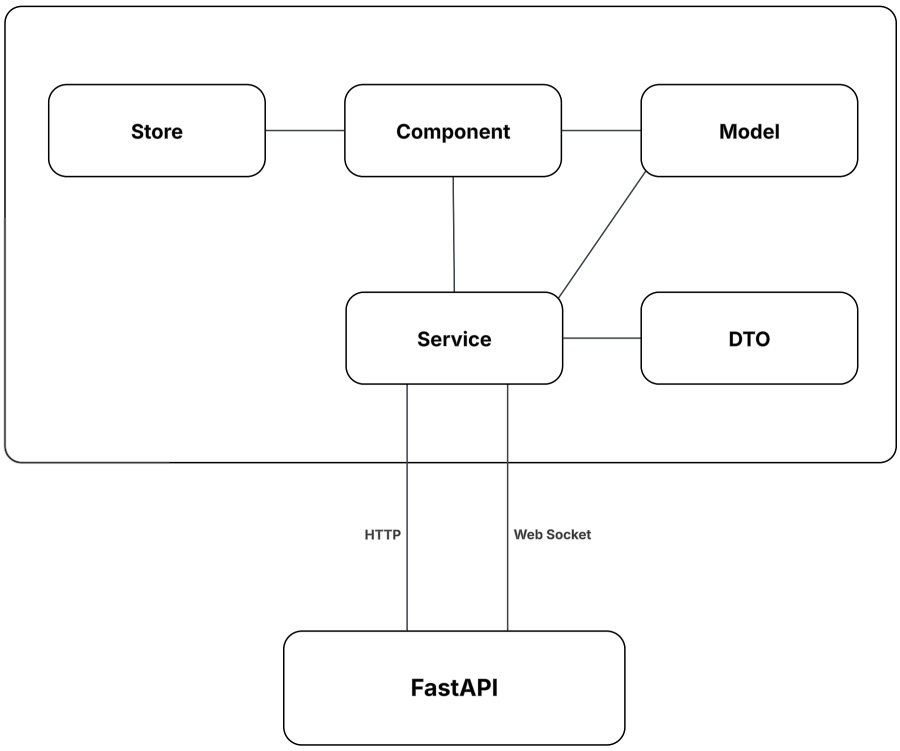

Figure 3.3. Component Architecture of a Frontend Module

3.1.3 LLM Integration

Integration of LLMs within the platform is streamlined using the LangChain framework, which provides an abstraction layer to interact uniformly with multiple LLM providers. As depicted in Figure 3.4, this integration allows the platform to remain flexible, accommodating advancements in LLM technologies with minimal architectural adjustments.

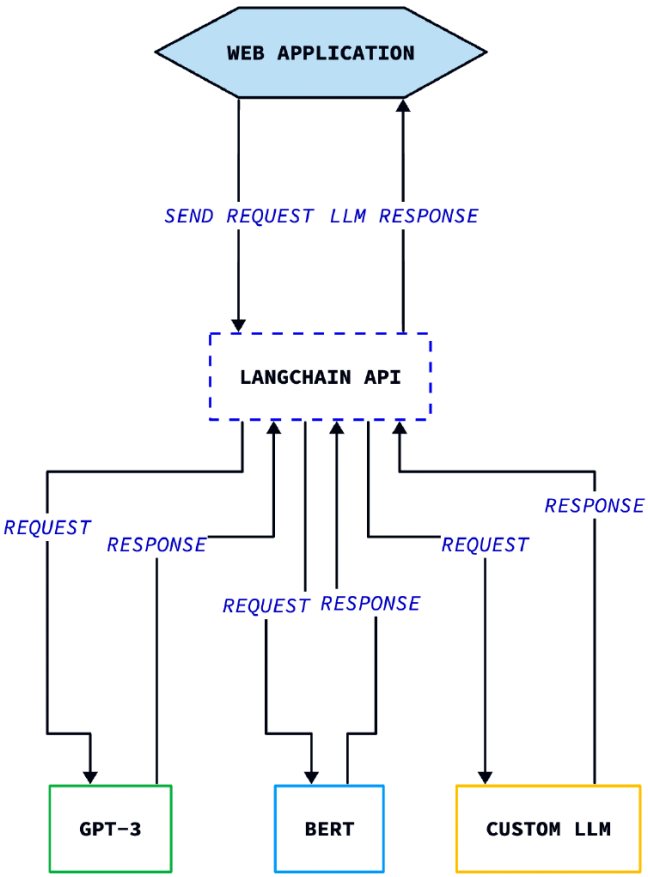

Figure 3.4. LLM Integration via LangChain

LangChain efficiently manages the interaction between the platform and various LLM services, handling data flow optimization, API call management, input/output formatting, and robust error handling mechanisms. It also incorporates rate-limiting functionalities to ensure resource-efficient and cost-effective interactions.

3.1.4 Text Editing Framework: Tiptap

On the frontend, the platform employs the Tiptap editor—an open-source, third-party framework built on top of ProseMirror and designed to provide extensible rich-text editing capabilities, especially within modern React-based applications. Tiptap is integrated as a ready-to-use library rather than being developed in-house, and its adoption ensures robust and reliable document editing features aligned with best practices in frontend architecture.

Within the platform, Tiptap has been tailored specifically for the structured editing of legal documents such as contracts. Unlike traditional rich-text editors, Tiptap is configured to meet the complex formatting and regulatory requirements intrinsic to legal documents. Users can efficiently organize document elements into legally defined sections, clauses, and subsections.

Tiptap supports critical legal editing features, including:

- **Structured Document Editing**: Customized nodes and marks for legally compliant structural elements, allowing users to efficiently manage and format contracts according to legal and organizational standards.

- **Clause and Variable Management**: Specialized nodes for inserting, editing, and managing clauses and dynamic contract variables directly within the document, ensuring precision and consistency.
- **Real-time Collaborative Editing**: Integration with collaborative technologies such as Y.js enables multiple legal professionals and stakeholders to simultaneously edit, review, and approve documents, reflecting changes in real-time and maintaining a clear audit trail of modifications.

This tailored configuration of Tiptap significantly enhances the user experience, providing a robust, intuitive interface optimized specifically for drafting, reviewing, and finalizing complex legal contracts, while leveraging an established and actively maintained framework from the open-source ecosystem.

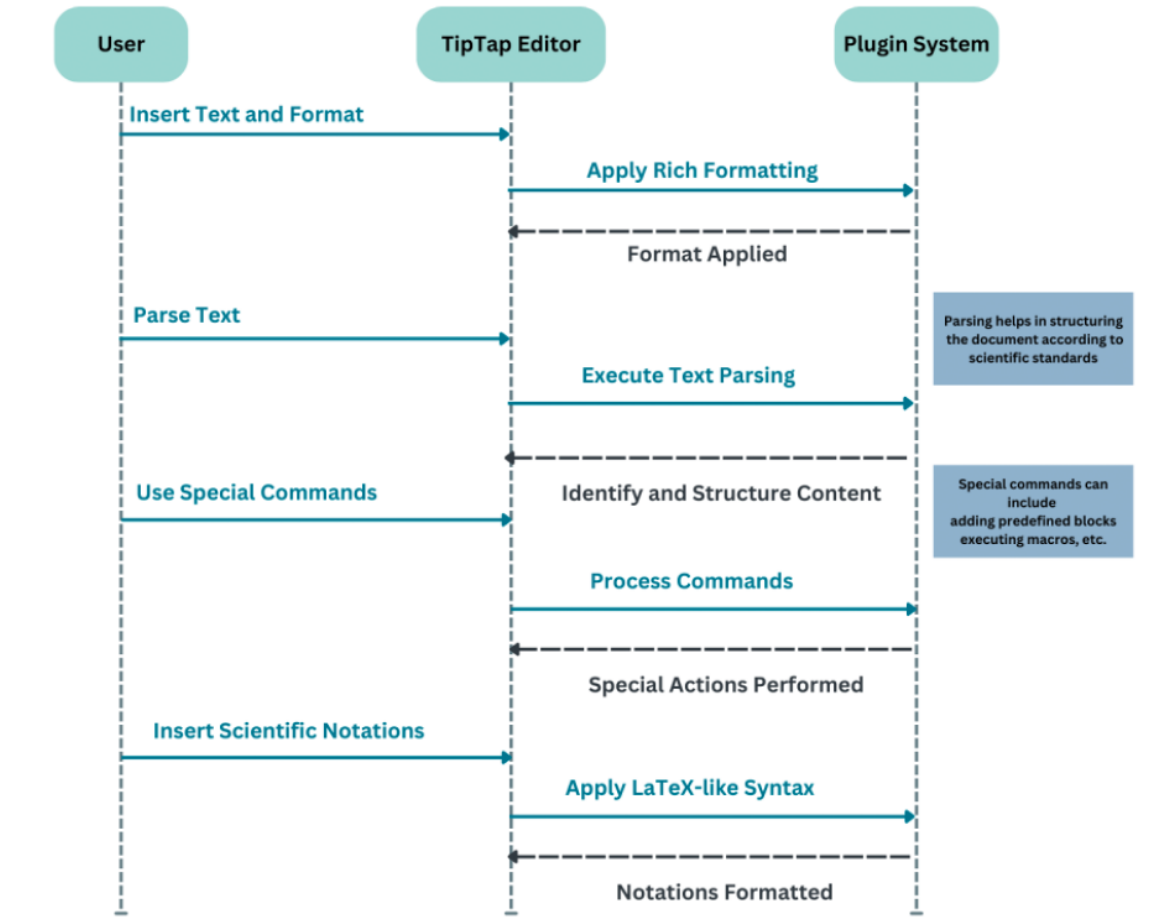

Figure 3.5. Tiptap Text Editing Framework for Legal Documents

3.2 Detailed Design

3.2.1 Class Diagram

The class diagram is a fundamental UML diagram type that illustrates the static structure of the software system, detailing its classes, their attributes, methods, and the relationships among these classes. Figure 3.6 presents the comprehensive class diagram of the intelligent contract management platform developed during this internship.

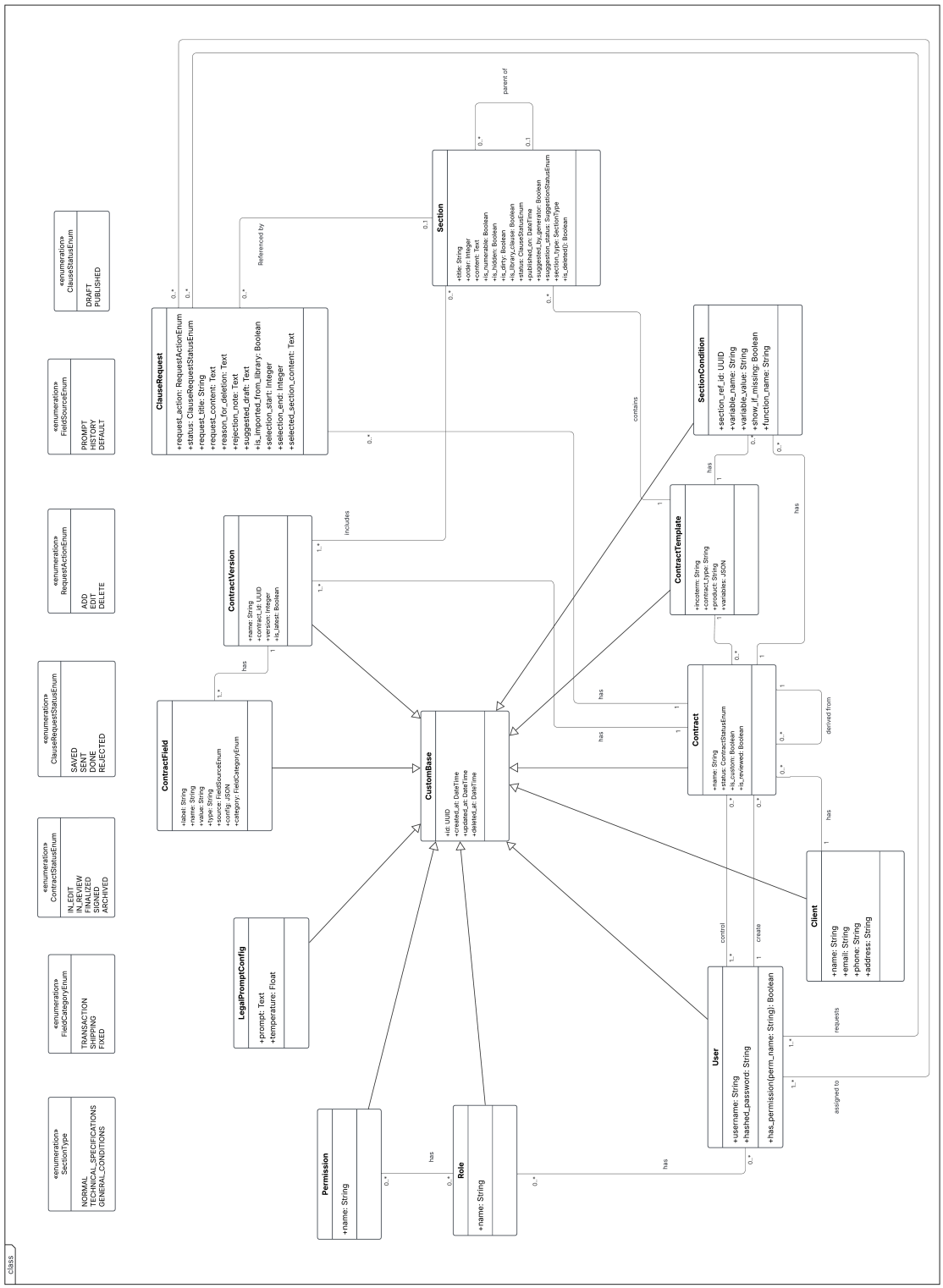

Figure 3.6. Class Diagram of The Platform

All classes in this architecture inherit from the **CustomBase**, providing universal attributes such as a unique identifier (*id*), timestamps for creation, update, and deletion of entries.

The **User** class represents users of the platform, associating each user with specific **Roles** (e.g., Sales or Legal), which determine a set of **Permissions** defining accessible functionalities.

The **Contract** class represents an agreement between a company and its client, distinct from the platform users. Each contract is associated with a specific **Client**, and relies on a **ContractTemplate** based on a combination of product, contract type, and Incoterm. Contracts can also originate by cloning existing contracts.

Contracts feature five distinct statuses represented by the enumeration *ContractStatusEnum*:

- **IN_EDIT**: Initial stage, editable by both Sales and Legal teams.
- **IN_REVIEW**: Editable by Legal, but frozen for Sales, acting as a proxy to finalization.
- **FINALIZED**: Fully reviewed and locked from editing by both teams.
- **SIGNED**: Confirmed and mutually agreed upon by the company and the client.
- **ARCHIVED**: Used for historical purposes based on the company's archiving policies.

Each **Contract** is composed of one or multiple **ContractVersions** to manage iterative changes and improvements. Versions include structured **Sections**, each comprising multiple fields detailed by **ContractField** classes representing variables specific to each contract section.

The **ClauseRequest** class is specifically dedicated to handling clause modification requests made by Sales users to the Legal team, marked by statuses within the *ClauseRequestStatusEnum* (*SAVED*, *SENT*, *DONE*, *REJECTED*). Requests transition from *SAVED* when initially drafted by a Sales user, to *SENT* upon submission to Legal. Depending on the decision by Legal, these requests become *DONE* upon acceptance and implementation, or *REJECTED* if not approved.

This class diagram provides a clear and structured representation, highlighting crucial entities, their interrelations, and functionalities integral to the platform's effective management of digital contract operations.

3.2.2 Detailed Sequence Diagram

Following the analysis phase where chronological interactions between the system and actors were outlined, detailed sequence diagrams have been created. These diagrams precisely illustrate interactions among system components to realize specific use cases. The use cases chosen for detailed representation are **Generate a New Contract**, **Create Clause Request** and **Review Clause Request**.

3.2.2.1 Generate a New Contract

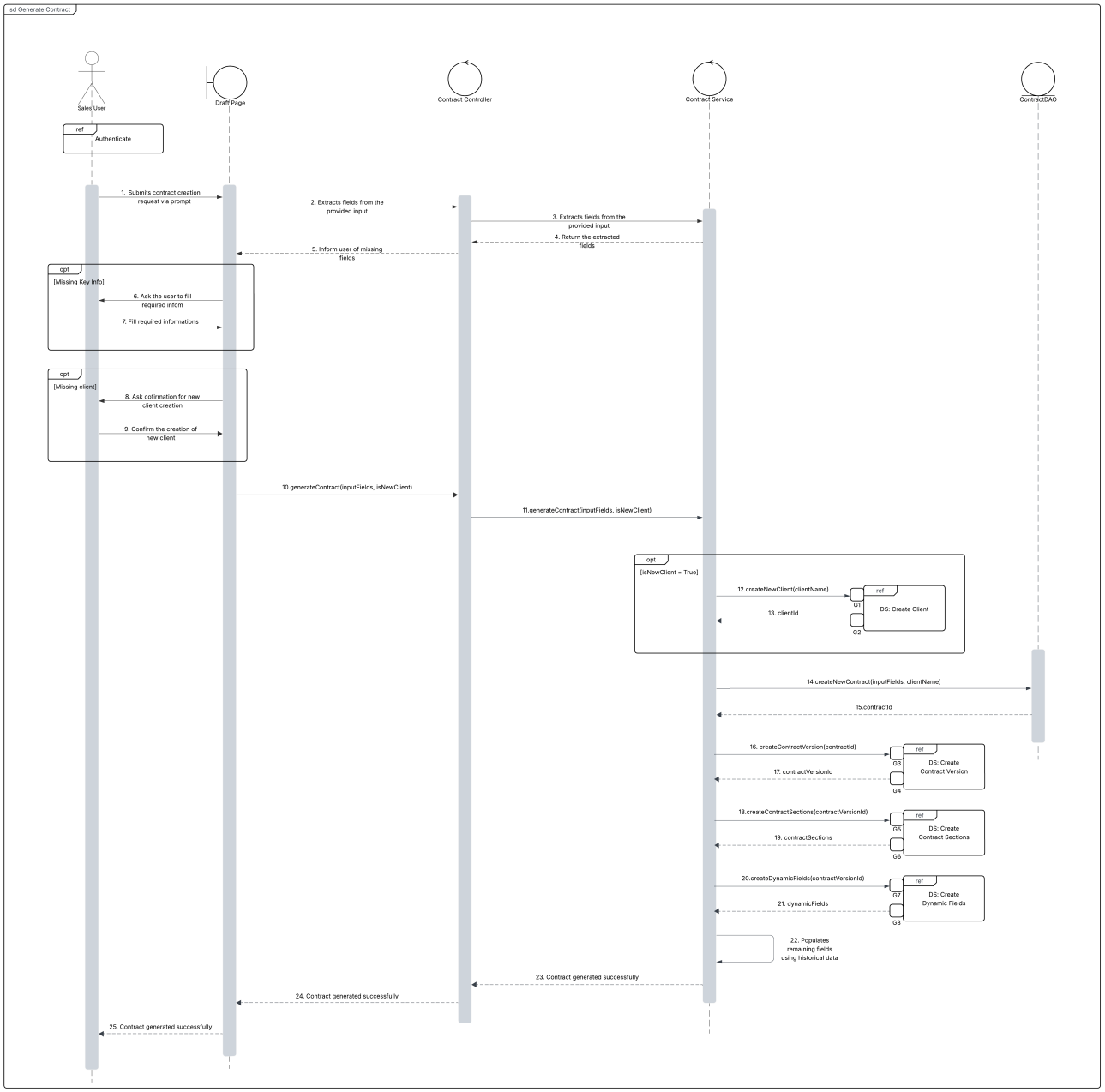

Figure 3.7. Sequence Diagram - Draft Contract

To maintain clarity and reduce complexity, certain subprocesses—such as Create Client and Create Contract Version—are encapsulated within ref frames. The detailed sequence diagrams for these referenced interactions are provided in the following figures:

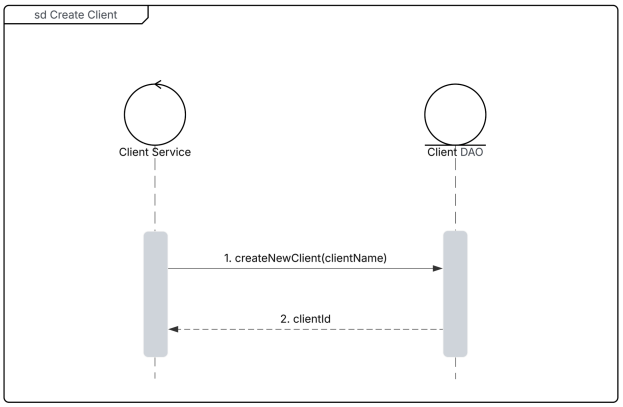

Figure 3.8. Sequence Diagram – Create Client

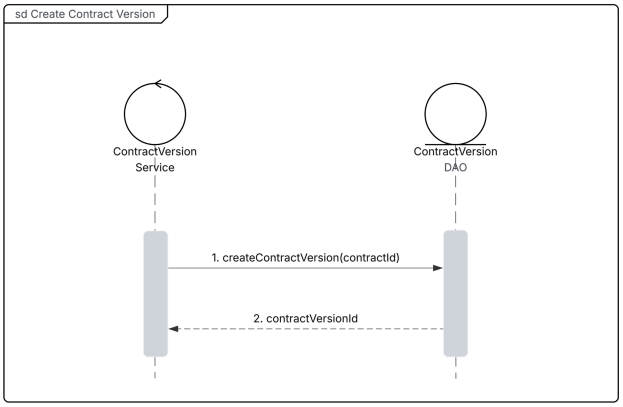

Figure 3.9. Sequence Diagram – Create Contract Version

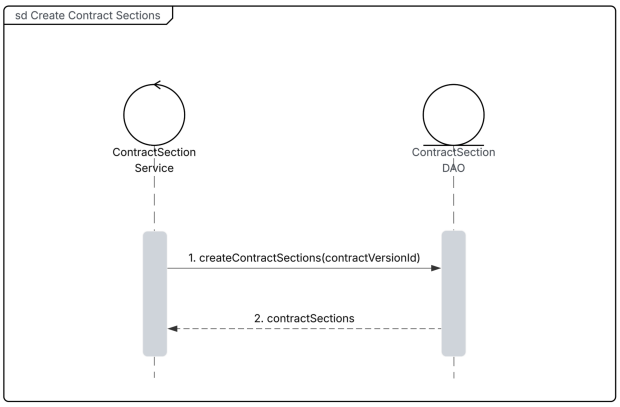

Figure 3.10. Sequence Diagram – Create Contract Sections

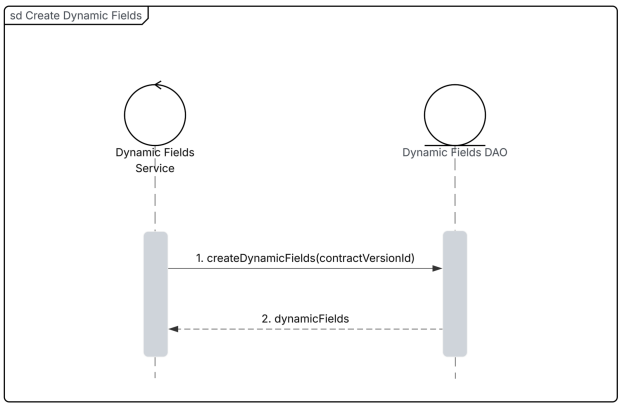

Figure 3.11. Sequence Diagram – Create Dynamic Fields

Figure 3.7 depicts the detailed sequence diagram for generating a new contract. In this scenario, the sales user initiates by logging into the system and providing a prompt through text, voice, or image, detailing essential information needed for the contract. The system then extracts necessary fields such as Incoterm, contract type, product, and client name to identify the appropriate template and client data. If critical information is missing or a new client is identified, the system prompts the user for clarification or confirmation to avoid mismatches.

Upon confirmation, the system creates a new client record if necessary, extracts dynamic fields from the provided prompt, selects the appropriate contract template, and populates the dynamic fields accordingly. Missing fields are filled either by referencing a specific past contract indicated by the user or by defaulting to the client's most recent contract. If the client is new or no relevant contracts are found, fields remain empty. After contract generation, the user is automatically redirected to the contract editing interface.

3.2.2.2 Clause Request Lifecycle

The Clause Request Lifecycle involves coordinated interactions between sales and legal teams, as shown in Figure 3.12 and Figure 3.14. Initially, a sales user creates a clause request by providing a description for modifying, adding, or deleting clauses within a contract. The user can optionally refine this description with assistance from the Refiner Agent, which leverages an LLM to enhance clarity and precision. The sales user can then either save the request for

later or send it directly to the legal team.

Upon submission, the legal team receives a notification of the new clause request. Legal users may reject the request immediately by providing specific feedback or accept it for further processing. Accepted requests prompt legal users to utilize a dedicated editor to craft or modify clauses as requested. Completed clauses may be directly integrated into the active contract by publishing, or saved in the clause library for future use.

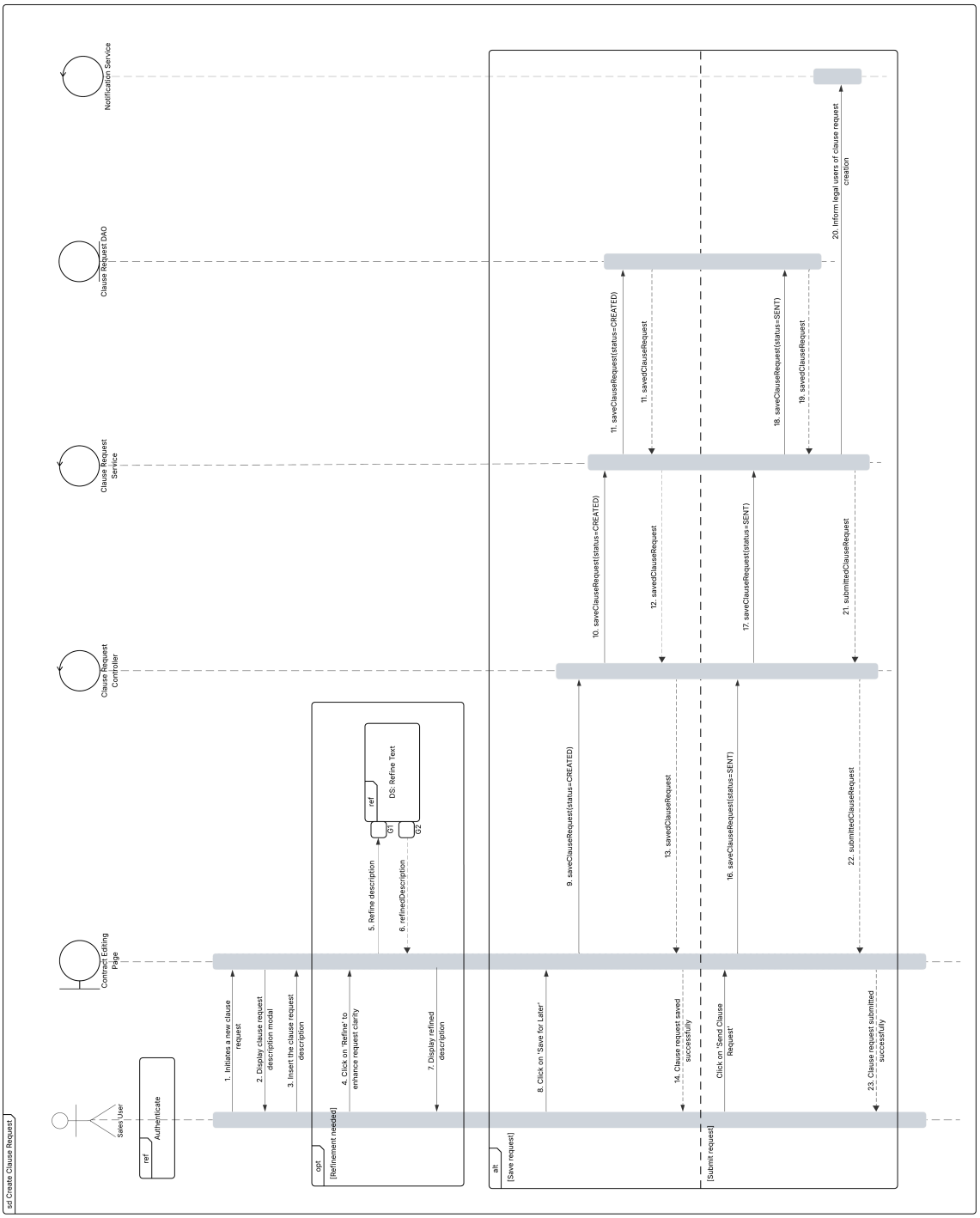

Figure 3.12. Sequence Diagram - Clause Request (Sales Perspective)

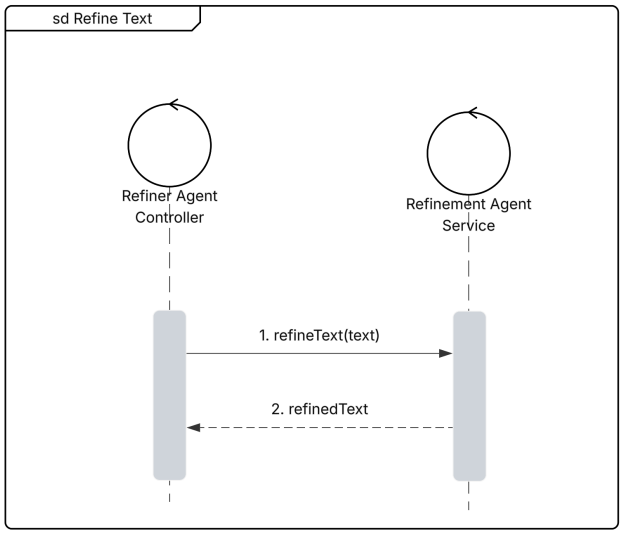

Figure 3.13. Sequence Diagram - Refine Text

After presenting the clause request process from the Sales user's perspective, the following sequence diagram illustrates the corresponding workflow from the Legal user's point of view:

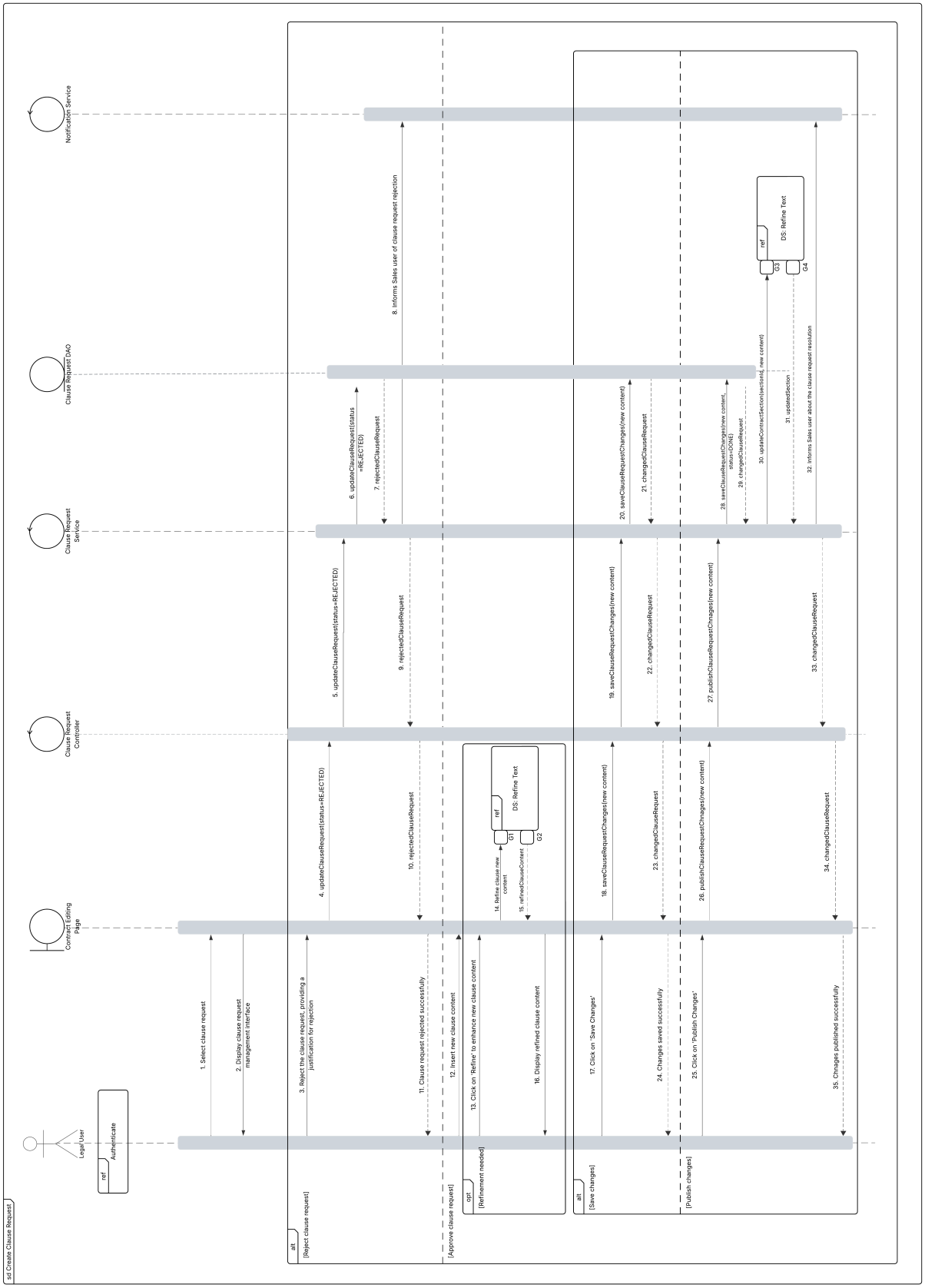

Figure 3.14. Sequence Diagram - Clause Request (Legal Perspective)

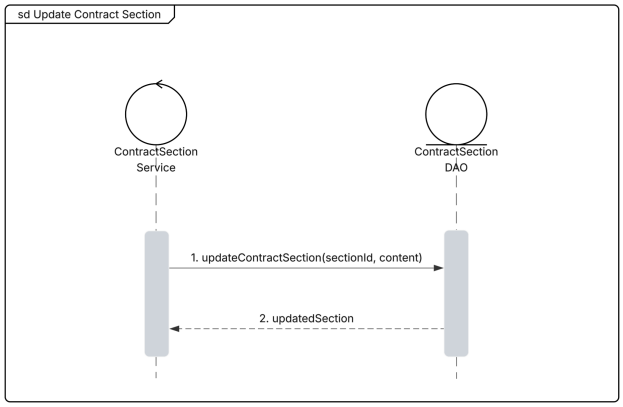

Figure 3.15. Sequence Diagram - Update Contract Sections

These diagrams comprehensively clarify the interactions between system components, highlighting critical processes and dependencies to ensure a smooth and efficient workflow within the intelligent contract management platform.

3.3 Deployment Architecture

The deployment architecture is a core operational layer that supports the delivery, testing, and maintenance of the intelligent contract management platform. It enables rapid feature iteration, automated quality control, and consistent deployment across environments. This section outlines how DevOps practices—particularly Continuous Integration and Continuous Deployment (CI/CD)—are integrated into the project's delivery lifecycle.

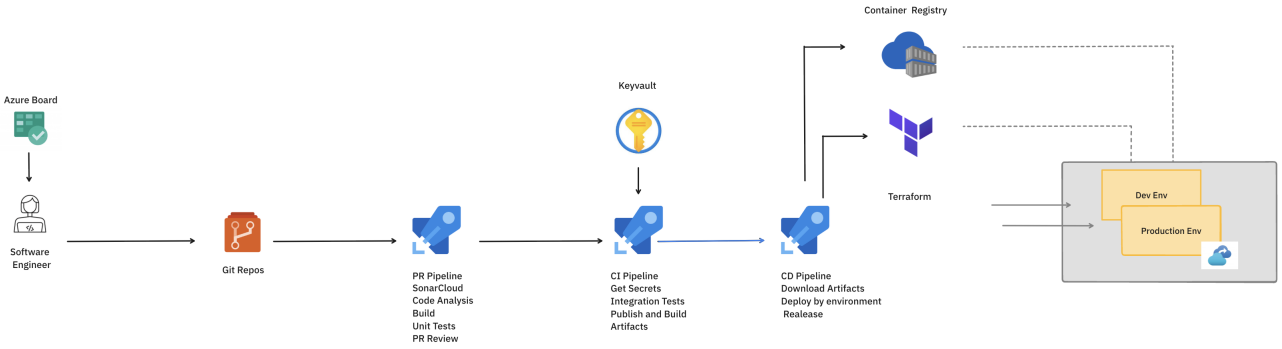

Figure 3.16. Deployment Architecture Overview

The process begins with developers managing tasks via Azure Boards and committing changes to a Git repository. Each push triggers a CI pipeline configured to perform essential validations, including static code analysis using SonarCloud, unit testing, and artifact building. Secrets required during builds are retrieved securely from Azure Key Vault, and integration tests are executed within isolated environments to ensure stability.

After successful validation, build artifacts are promoted to the CD pipeline, where IaC (Infrastructure-as-Code) tools such as Terraform and Helm are used to deploy services to Kubernetes clusters across development and production environments. Docker images are stored in Azure Container Registry, enabling versioned and reproducible deployments. These workflows ensure high confidence in new releases and accelerate the delivery of new features while maintaining system reliability and traceability.

3.4 Conclusion

This chapter presented the platform's system architecture, from frontend/backend logic to AI integration and legal document editing. It concluded with UML diagrams detailing the software's structure and component workflows. The next chapter will focus on implementation, key functionalities, and system validation.

Chapter 4

Implementation

This chapter outlines the implementation of the intelligent contract management platform. It starts with a review of the selected technologies across AI, frontend, backend, and infrastructure. The core functionalities—contract drafting, editing, clause handling, and tracking—are described in detail. It also presents the collaborative and review workflows powered by AI agents. Lastly, it explains the validation process for verifying functional and non-functional requirements.

4.1 Technological Environment

The choice of appropriate technologies is critical for ensuring both performance and maintainability. Below, we outline our selected technologies, emphasizing the reasoning behind each choice.

4.1.1 AI Integration and Workflow Orchestration

4.1.1.1 Azure OpenAI's Large Language Models

Azure OpenAI's Large Language Models (LLMs) were selected for their exceptional performance in generating human-like text and handling complex language tasks. Known for their extensive training on diverse datasets, these models excel in text generation, summarization, and conversational AI, proving essential across multiple application scenarios in our project.

The decision to use Azure OpenAI was influenced by specific client requirements, notably the necessity for data security and compliance. The Azure environment allows operations to be confined locally, ensuring sensitive data, particularly legal contracts, remain secure within the client's network. Additionally, opting for Azure OpenAI provided seamless integration with the client's existing infrastructure, including Azure Repos and Azure Pipelines, thus offering an enterprise-grade ecosystem.

We concluded that Azure OpenAI's models were the most suitable after considering other alternatives such as Google's BERT and Meta's LLaMA.

Figure 4.1. Azure OpenAI Logo
[21]

4.1.1.2 LangChain

LangChain was integrated to enable flexible and model-agnostic communication within our system. Its modular nature allows seamless integration with various LLM providers, offering the flexibility to switch models or combine outputs based on specific tasks. This ensures the system remains adaptable to ongoing developments in LLM technology and can readily incorporate future advancements.

Additionally, LangChain provides essential utilities for managing conversational flows, context tracking, and connections with external data sources, critical for sophisticated generative AI-driven applications. Its use underpins the modularity and scalability of our architecture, effectively handling complex tasks across diverse domains.

LangChain was ultimately selected for its abstraction capabilities after evaluating direct integrations with alternative APIs.

Figure 4.2. LangChain Logo
[22]

4.1.1.3 LangGraph

LangGraph complements LangChain by enabling sophisticated workflow orchestration through a graph-based model. It supports complex, dynamic, and iterative processes within AI workflows, facilitating advanced conditional logic, stateful interactions, and multi-agent collaborations. This capability is especially beneficial for iterative legal document processing tasks, enhancing decision-making capabilities and workflow adaptability.

LangGraph was chosen over other orchestration tools for its native compatibility with LangChain and AI-specific use cases.

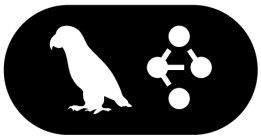

Figure 4.3. LangGraph Logo
[23]

4.1.2 Vector Databases

Elasticsearch was selected as the primary vector database due to its exceptional capabilities in managing high-dimensional vector searches, essential for Retrieval-Augmented Generation (RAG) architectures. Elasticsearch facilitates rapid similarity searches, allowing efficient real-time retrieval of relevant documents based on embedding vectors derived from queries. Its scalability, robustness, and comprehensive indexing capabilities were critical in meeting performance expectations under large data volumes while ensuring minimal latency.

After comparing it with other solutions like FAISS and Milvus, Elasticsearch was determined to be the most enterprise-ready and integrable option.

Figure 4.4. Elasticsearch Logo
[24]

4.1.3 Frontend Technologies and Tooling

4.1.3.1 TypeScript

TypeScript is an open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript that adds optional static typing to the language. By enabling developers to define data types for variables, function parameters, and return values, TypeScript improves code clarity, detects potential errors during development, and enhances the scalability of large codebases.

TypeScript was adopted for frontend development primarily due to its strong type-checking system, which significantly reduces potential bugs and enhances code maintainability. It compiles directly to JavaScript, enabling native browser execution and optimal performance while retaining compatibility with existing JavaScript libraries and frameworks.

We selected TypeScript after evaluating other approaches, concluding it offered the best balance of developer experience, code robustness, and long-term reliability for our frontend needs.

Figure 4.5. TypeScript Logo
[25]

4.1.3.2 React

Developed by Meta (formerly Facebook), React is a popular open-source JavaScript library for building user interfaces, particularly single-page applications. It enables developers to construct interactive UIs using a declarative paradigm and a component-based architecture, where each UI element is encapsulated in a reusable component. Its virtual DOM efficiently updates only the parts of the interface that change, contributing to excellent performance.

This modular architecture supports efficient state management and seamless integration with backend services. Combined with React's widespread community support and flexibility, these features enable the rapid development of dynamic, responsive applications.

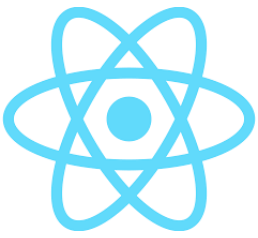

Figure 4.6. React Logo
[26]

4.1.3.3 Tiptap

Built on top of ProseMirror, Tiptap is a modern, open-source rich-text editor framework designed to be headless and highly extensible. It allows developers to craft custom editing interfaces tailored to specific application needs. Unlike conventional WYSIWYG editors, Tiptap decouples the editing engine from the UI layer, offering full control over document structure, formatting behavior, and content validation—making it ideal for structured documents and domain-specific workflows.

Its ability to handle complex legal documents made it a natural fit for our platform. Tiptap supports customized editing experiences, structured clause and variable handling, and real-time collaborative editing. After reviewing alternative editors, its extensibility, legal adaptability, and collaborative compatibility made it the most suitable choice for our use case.

Figure 4.7. TipTap Logo
[27]

4.1.3.4 Testing Libraries

Developed and maintained by Meta, Jest is a comprehensive JavaScript testing framework primarily designed for React applications, though it also supports a wide range of JavaScript and TypeScript projects. It comes equipped with powerful features such as test runners, assertions, mocking capabilities, code coverage analysis, and snapshot testing. Its zero-configuration setup and intuitive syntax make it accessible to developers of all experience levels.

Thanks to its efficiency, ease of configuration, and robust mocking tools, Jest proved ideal for unit and integration testing within our TypeScript and React stack. Among several alternatives, it stood out for its mature ecosystem, fast performance, and seamless integration with modern frontend tooling.

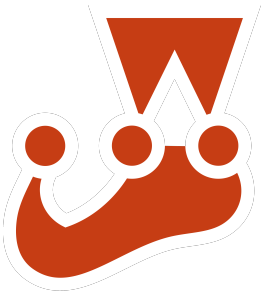

Figure 4.8. Jest Logo
[28]

4.1.4 Backend Technologies and Tooling

4.1.4.1 Python

Python is a high-level, interpreted programming language renowned for its simplicity, readability, and versatility. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python has a rich ecosystem of libraries and frameworks, particularly in the fields of web development, data analysis, and artificial intelligence, making it one of the most widely used languages in both academia and industry.

Its extensive support for machine learning and natural language processing—most notably through libraries like LangChain—made Python the preferred backend language for our project. Its clean syntax and strong community support enabled rapid development, easy maintenance, and seamless AI integration. After evaluating multiple options, Python's unmatched flexibility and robust ecosystem positioned it as the most suitable choice for our needs.

Figure 4.9. Python Logo
[29]

4.1.4.2 FastAPI

FastAPI is a modern, high-performance web framework for building APIs with Python 3.7+, leveraging standard type hints and asynchronous programming via async and await. This enables non-blocking I/O operations and supports highly concurrent API services. One of its standout features is the automatic generation of interactive OpenAPI documentation, making it highly developer-friendly and ideal for rapidly building and testing RESTful services.

Its performance, simplicity, and native async support made FastAPI an excellent fit for developing high-throughput APIs. Combined with robust type validation via Pydantic, the framework promotes clean, maintainable code. After evaluating multiple frameworks, it emerged as the most effective solution for building fast and reliable APIs in Python.

Figure 4.10. FastAPI Logo
[30]

4.1.4.3 Testing Libraries

Pytest is a feature-rich testing framework for Python that supports everything from simple unit tests to complex functional and integration scenarios. It prioritizes readability and scalability, offering fixtures, parameterized tests, and assert rewriting. Its plugin-based architecture and support for parallel test execution make it ideal for testing large and sophisticated codebases.

We adopted Pytest for backend testing due to its concise syntax, versatility, and robust plugin ecosystem. Among various Python testing tools, it proved to be the most extensible and efficient choice, streamlining our workflow for validating the platform's core services.

Figure 4.11. Pytest Logo
[31]

4.1.5 SQL Databases

Given the platform's need for transactional consistency, relational data integrity, and advanced query support, an SQL-based approach was favored over NoSQL. SQL databases better accommodate structured data, complex relationships, and reporting use cases relevant to our application.

PostgreSQL was chosen for its advanced query capabilities, robust support for both structured and unstructured data, high performance, extensibility, and strict adherence to ACID transactions. These attributes were critical for efficiently managing the hybrid data requirements inherent in our GenAI-powered system.

Among various relational databases, PostgreSQL was selected as the most capable and scalable option for our backend architecture.

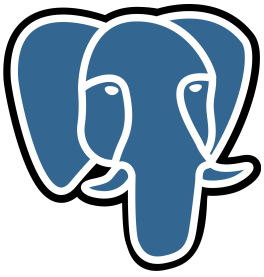

Figure 4.12. PostgreSQL Logo
[32]

4.1.6 API Testing

4.1.6.1 Postman

As a comprehensive API platform, Postman simplifies the creation, testing, and documentation of APIs through its intuitive interface. Within our project, it played a key role in verifying API functionality by streamlining request validation and response analysis. Its robust testing capabilities—including automated testing and request chaining—enabled thorough validation of API behavior.

We selected Postman as our primary API testing tool for its ease of use, extensive features, and superior usability compared to other evaluated options.

Figure 4.13. Postman Logo
[33]

4.1.6.2 cURL

cURL stands us a lightweight yet powerful command-line tool for API testing, offering a way to interact with APIs directly from the terminal. In our project, cURL serves as a solution for quick and efficient API testing, providing control over HTTP requests and responses. While lacking the graphical interface of tools like Postman, cURL compensates with its simplicity and speed.

Due to its flexibility and universal compatibility, cURL complemented our testing stack effectively.

Figure 4.14. cURL Logo
[34]

4.1.7 Static Code Analysis and Quality Assurance

SonarQube is a static code analysis tool designed to detect bugs, identify security vulnerabilities, and evaluate source code quality. In this project, SonarQube was employed to analyze both existing code and newly developed features. The tool facilitated the correction of detected

bugs and security flaws, offering actionable recommendations to enhance code structure and readability, thus ensuring compliance with the client's quality standards.

SonarQube was selected after comparing several tools, as it offered the best combination of analysis depth, usability, and integration capabilities.

Figure 4.15. SonarQube Logo
[35]

4.1.8 CI/CD Pipelines

Azure Pipelines, part of the Microsoft Azure DevOps suite, is a cloud-based CI/CD service that automates the building, testing, and deployment of code across diverse platforms and environments. It integrates seamlessly with popular source control systems and cloud services, offering powerful automation and precise control over deployment workflows.

In our project, Azure Pipelines—along with Azure Repos—was used to streamline continuous integration and deployment processes. This choice aligned with the client's infrastructure, enabling seamless integration, robust version control, and efficient automation. Its native compatibility with the Azure ecosystem made it the optimal solution for our CI/CD and DevOps needs.

Figure 4.16. Azure Pipelines Logo
[36]

4.1.9 Containerization and Orchestration

4.1.9.1 Docker

As an open-source platform, Docker automates the deployment of applications as portable, self-sufficient containers by packaging code, libraries, and dependencies into isolated units that run reliably across various environments. It streamlines the development lifecycle by ensuring

consistent environments from development to production and integrates seamlessly with orchestration tools and cloud platforms.

We selected Docker to containerize applications and their dependencies, ensuring reliability across all stages. Although Podman was considered for its rootless architecture, Docker's widespread adoption, mature ecosystem, and ease of use made it the preferred choice for our project.

Figure 4.17. Docker Logo
[37]

4.1.9.2 Kubernetes

Originally developed by Google and now maintained by the Cloud Native Computing Foundation (CNCF), Kubernetes is an open-source platform for orchestrating containerized applications. It automates deployment, scaling, and management, offering advanced features like service discovery, load balancing, self-healing, and automated rollouts.

Its robust ecosystem and enterprise-grade capabilities made Kubernetes the top choice for orchestration. While alternatives such as Docker Swarm and Apache Mesos were reviewed, Kubernetes stood out as the most powerful and production-ready solution for our deployment needs.

Figure 4.18. Kubernetes Logo
[38]

4.1.9.3 Terraform

Developed by HashiCorp, Terraform is an open-source infrastructure as code (IaC) tool that enables users to define and provision both cloud and on-premises infrastructure using a declarative configuration language (HCL). With support for multiple cloud providers and a strong provider ecosystem, Terraform maintains infrastructure state and ensures safe, efficient changes.

Its flexibility, cloud-agnostic nature, and version control support made Terraform the ideal tool for managing our infrastructure. After comparing it with AWS CloudFormation and Ansible, we concluded that its multi-cloud capabilities and extensibility made it the optimal solution for our IaC strategy.

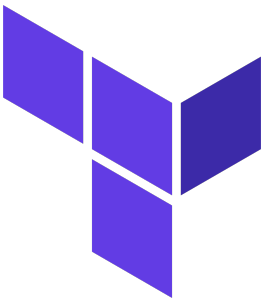

Figure 4.19. Terraform Logo
[39]

4.1.10 IDE (Integrated Development Environment)

Visual Studio Code (VS Code) is a an IDE (Integrated Development Environment) with powerful developer tooling like code editing, debugging, version control integration, and support for various programming languages and frameworks. Its user-friendly interface and intuitive design streamline the coding process, allowing us to write and edit code with ease. Additionally, VS Code offers robust Docker integration, enabling us to manage containers and write YAML files for Kubernetes, simplifying our local deployment and testing workflows.

VS Code was chosen due to its lightweight design, extensive extension ecosystem, and strong support for containerized development workflows.

Figure 4.20. VS Code Logo
[40]

4.2 Implementation

This section is dedicated to presenting the implementation details of the developed solution. Specifically, we outline the interfaces related to contract management functionalities, including contract repository, contract creation, editing, review, notification handling, and history management. The implementation and interfaces associated with other modules were allocated to and handled by different members of the project team, thus falling outside the scope of this section.

4.2.1 Contract Repository

The Contract Repository interface is accessible by both Sales and Legal users, serving as the primary dashboard displaying a list of existing client contracts. Contracts are categorized into three distinct statuses: **In Editing** (editable by both Sales and Legal teams), **In Review** (finalized by Sales and awaiting Legal review), and **Finalized** (reviewed, approved, and ready for client delivery).

Each contract card provides essential details, including the title, client involved, contract type, and Incoterms, and supports specific actions such as:

- **Archive/Delete**: Exclusive to Sales users, this functionality enables users to either *delete* a contract completely from the system, primarily used for erroneous entries, or *archive* contracts, which removes them from the main repository interface but retains them in an archived section for future reference or audits.
- **Logs/Notifications**: Users receive relevant notifications reflecting significant changes, such as contract status updates, ensuring stakeholders remain consistently informed and engaged.

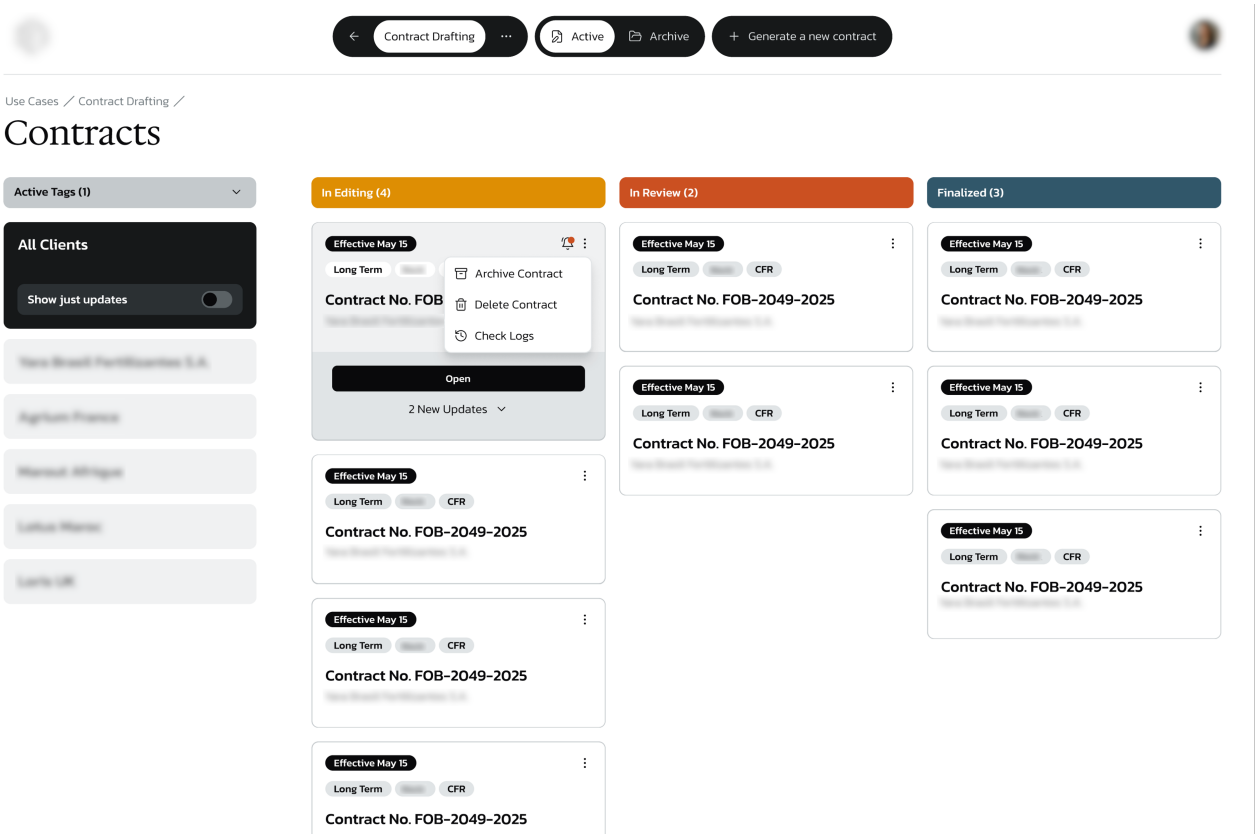

Figure 4.21. Contract Repository Interface

4.2.2 Contract Generation

From the Contract Repository page, Sales users initiate contract creation via three input methods: audio recording, image uploads, or textual prompts. For instance, a user might prompt: *Ädraft a spot contract for rock product with Incoterm CFR for new client theClient-Name¨*, optionally supplemented with additional image-based information.

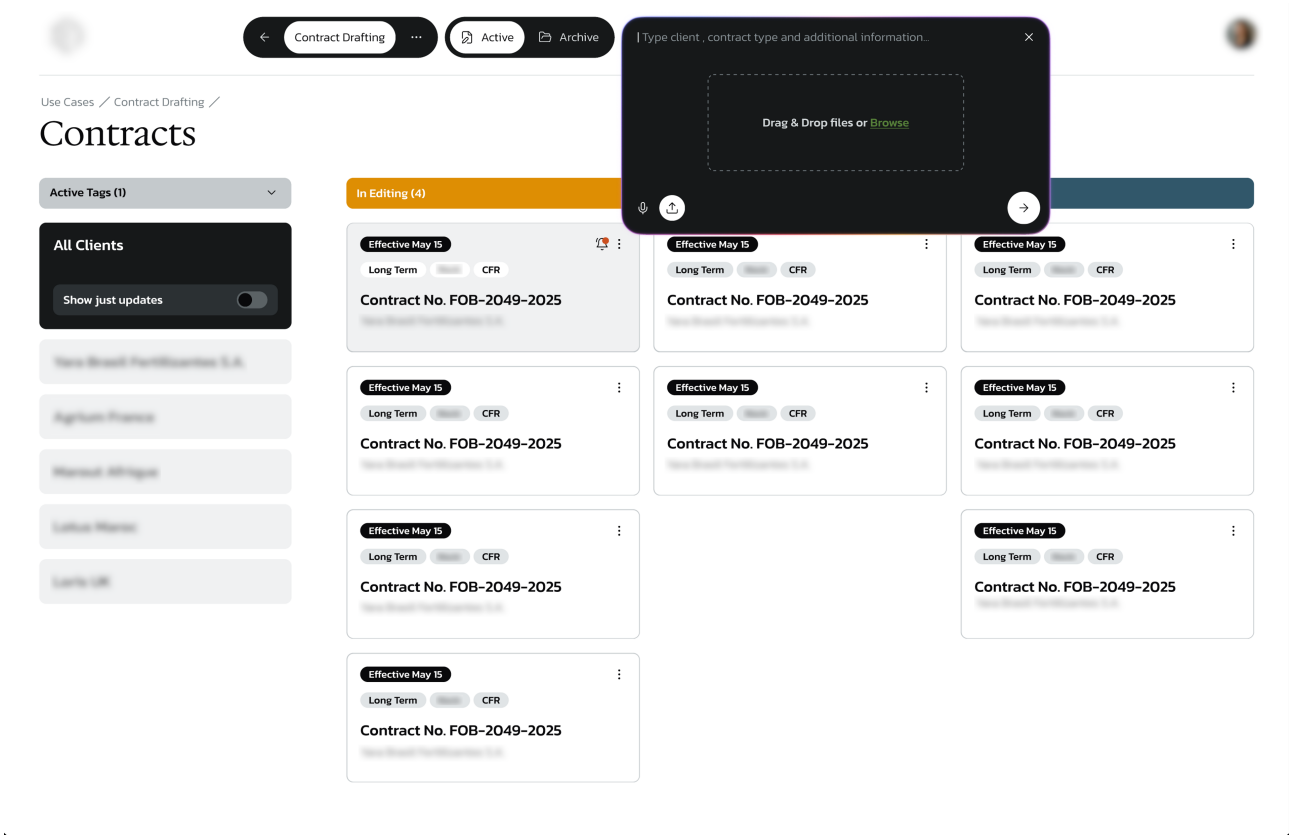

Figure 4.22. Contract Creation Component

The first agent involved is the Generator, which initially checks for the required information: Contract Type, Incoterm, Client, and Product. Each combination of these fields corresponds to a distinct contract template. In the current scenario, the Generator detects that all essential fields are provided but identifies **theClientName** as a new client, not yet registered in the database. Consequently, the user is prompted either to confirm the new client details to avoid potential mismatches or, if the client already exists, to utilize an input field that suggests similar existing client names through Elasticsearch-based logic for selecting the correct one.

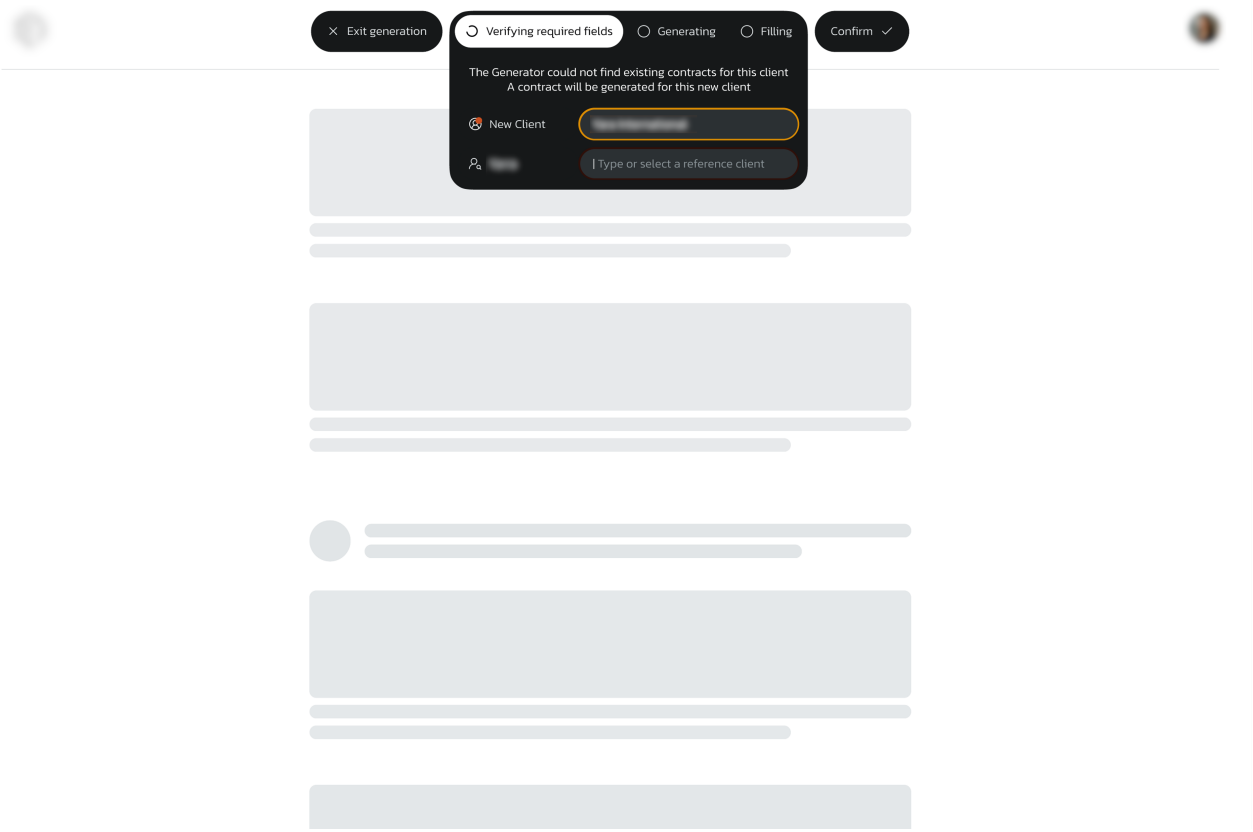

Figure 4.23. Contract Verifying Fields Step

The contract generation process involves the Generator agent, performing several key steps:

- **Checking**: Confirms necessary fields (Contract Type, Incoterm, Client, Product) are provided. For new clients, users confirm client details or select from suggested matches using Elasticsearch.
- **Generating**: Creates contract sections and dynamically fills fields, leveraging historical data for existing clients to suggest relevant articles.
- **Filling**: Populates contract fields into respective database tables (Contracts, Contract Fields, Versions), and generates the initial contract version.

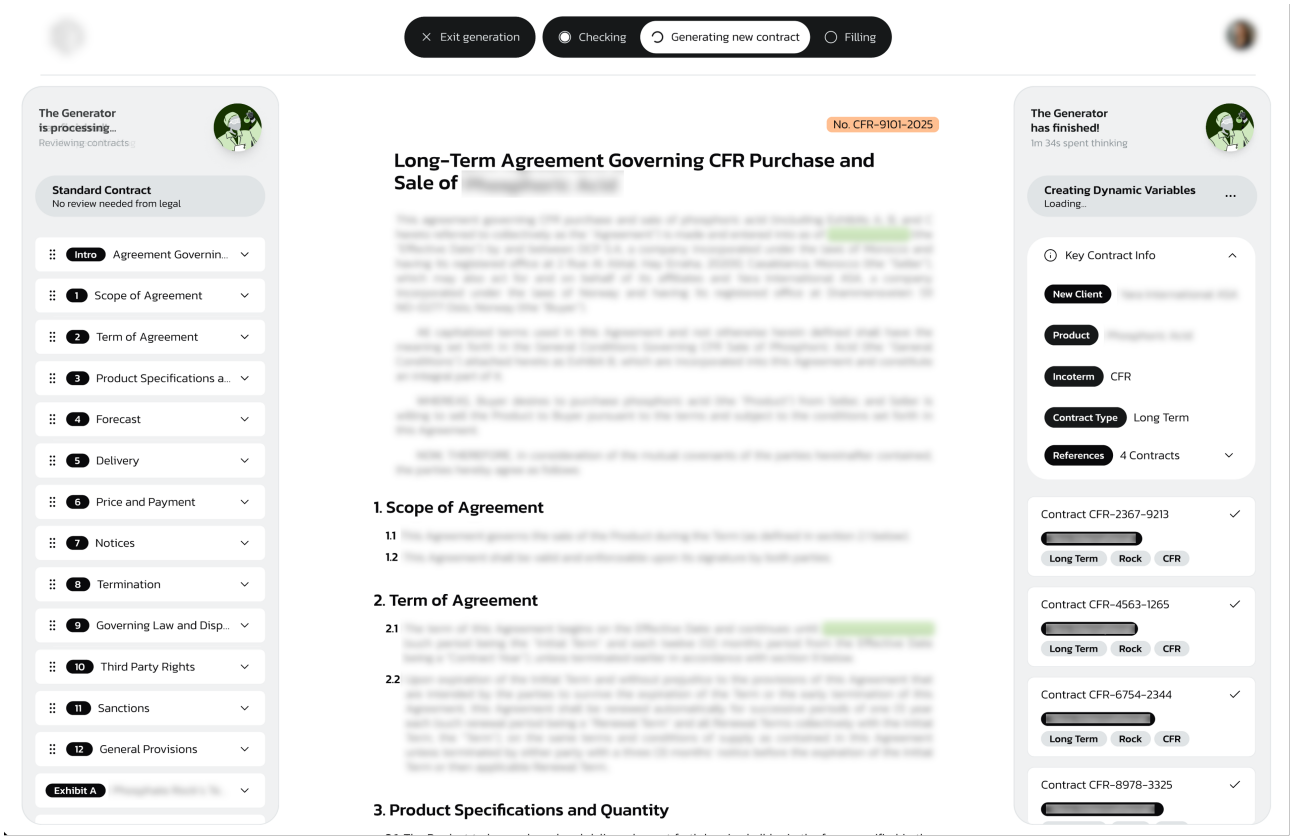

Figure 4.24. Contract Generating Process

4.2.3 Contract Editor

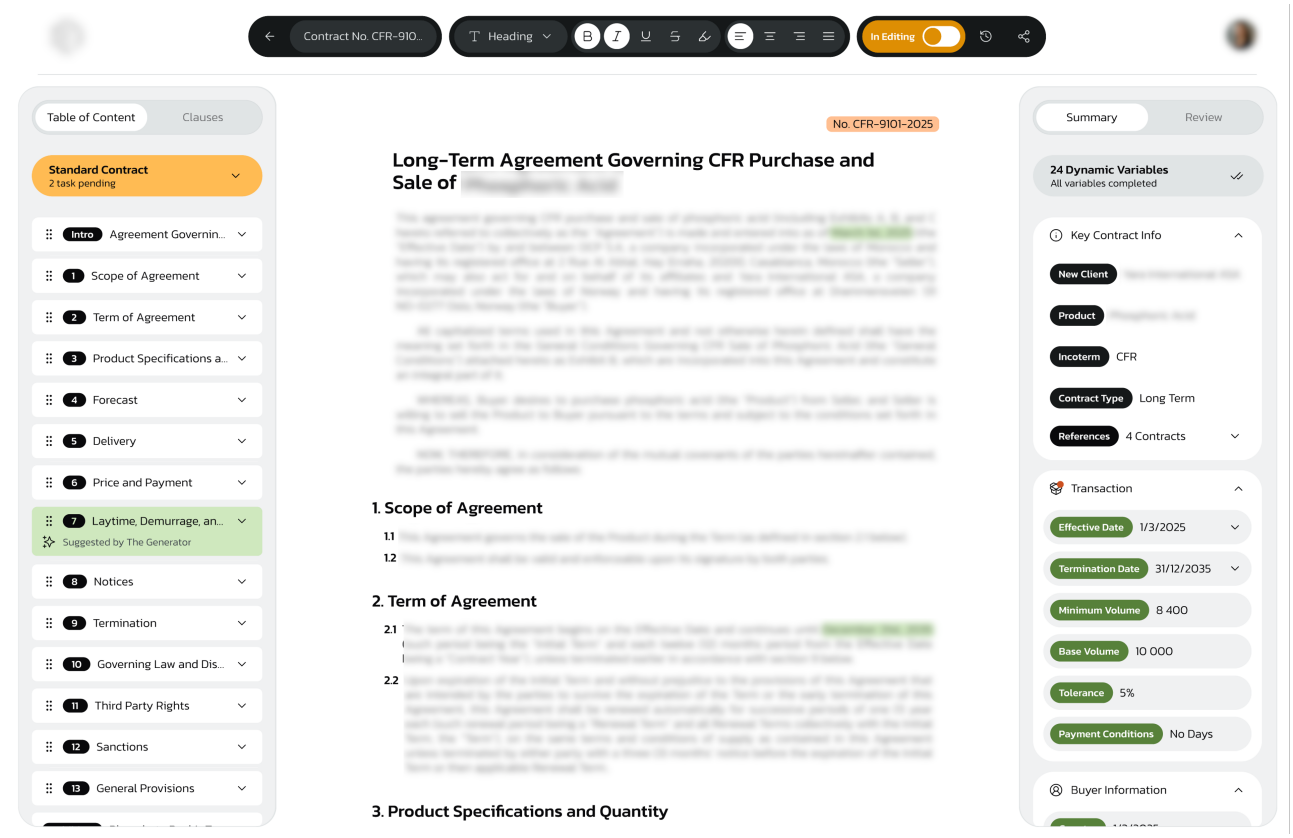

Figure 4.25. Contract Editor Interface

Upon creation or selection from the repository, the contract opens in the Contract Editor interface, structured into four distinct sections:

- **Header:** The header section prominently displays the platform logo on the left and the user profile logo on the right, which provides quick access for modifying user information or logging out. Centrally located within the header, the contract title is editable directly by the user. Navigation controls are positioned alongside the title, including a button to return to the Contract Repository page. Additionally, a formatting toolbar allows users to apply text styles such as bold, italic, or underline. To the right side of the toolbar, three action buttons facilitate contract status updates, history access, and contract sharing options.
- **Left Panel:** This panel provides easy navigation through the document's structured contents with a dynamic table of contents. Additionally, it contains functionalities for managing contract clauses, including viewing existing clauses, requesting new ones, or initiating clause modifications.
- **Right Panel:** The right panel consists of two critical components. First, the summary panel, which lists all required and dynamic fields of the contract, allowing users to edit these directly. Second, the review panel where the Reviewer Agent performs comprehensive reviews, ensuring accuracy, consistency, and compliance with regulatory standards.
- **Editor Body:** At the heart of the interface lies the document editing area, powered by the Tiptap editor. This editor allows Legal users full editing capabilities to draft, modify, and finalize the contract's textual content. For Sales users, the editor body is presented in a read-only mode, ensuring document integrity and maintaining clear responsibilities between roles.

4.2.3.1 AI-Suggested Articles

The Generator agent may suggest relevant articles based on past client contracts or user-specified historical references. Users can accept or reject suggestions, automatically integrating accepted articles into the editor.

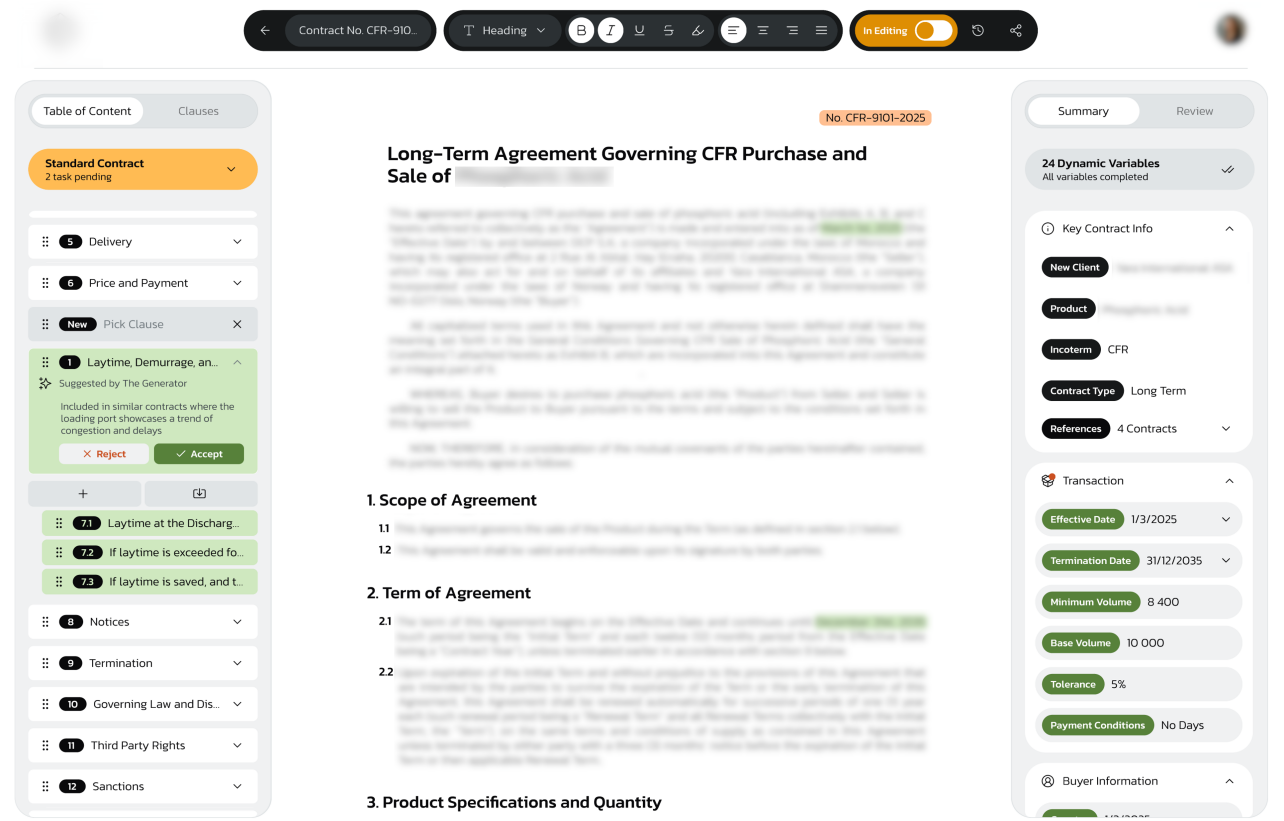

Figure 4.26. Contract Editor with AI-Suggested Article

4.2.3.2 Clause Management

As Sales users are restricted from directly editing contracts, the platform provides a specialized clause management feature to facilitate necessary modifications through clause requests. When a Sales user identifies a need to modify a contract, they initiate a new clause request. This action can be triggered through multiple methods: directly via the "New Clause" button in the left panel, by selecting a specific section within the contract to associate the request with the content and its number, or through interaction with existing comments within the editor interface.

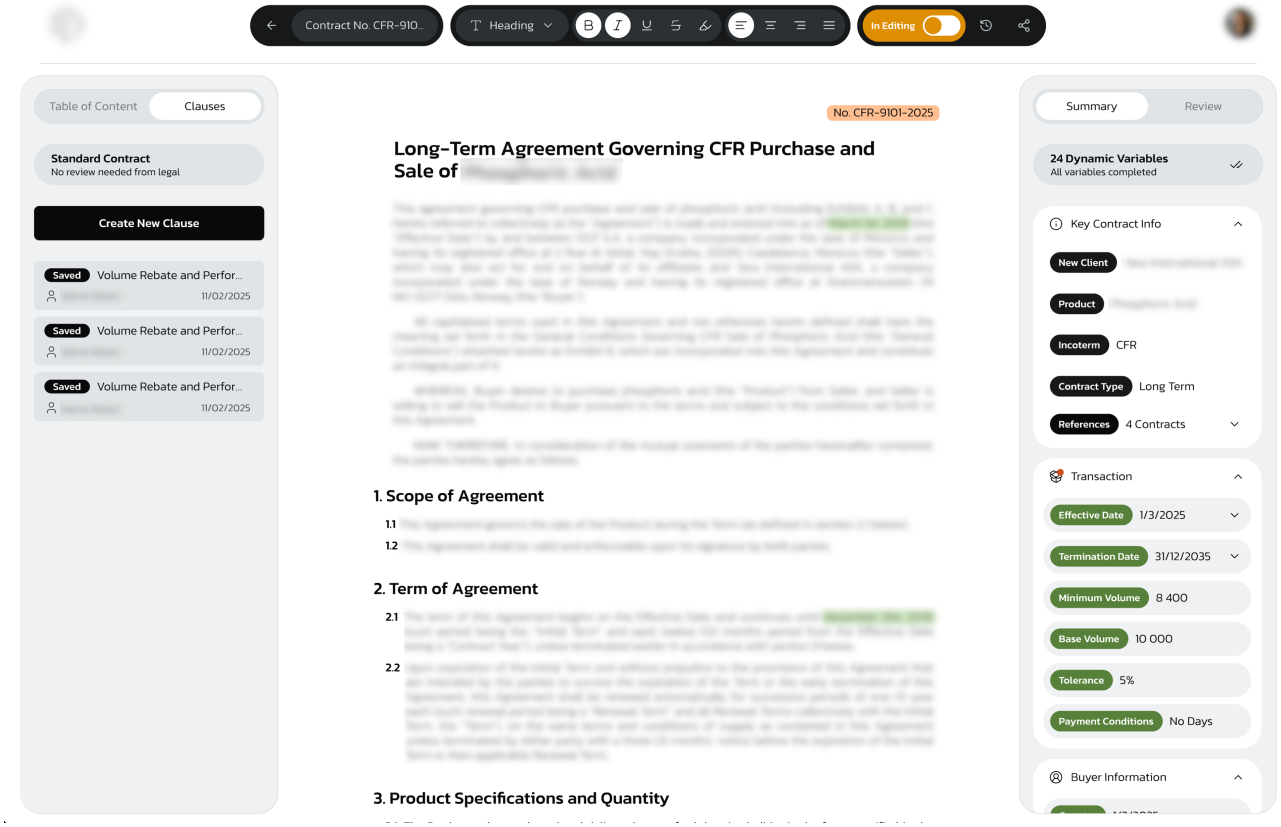

Figure 4.27. Clause Request Interface

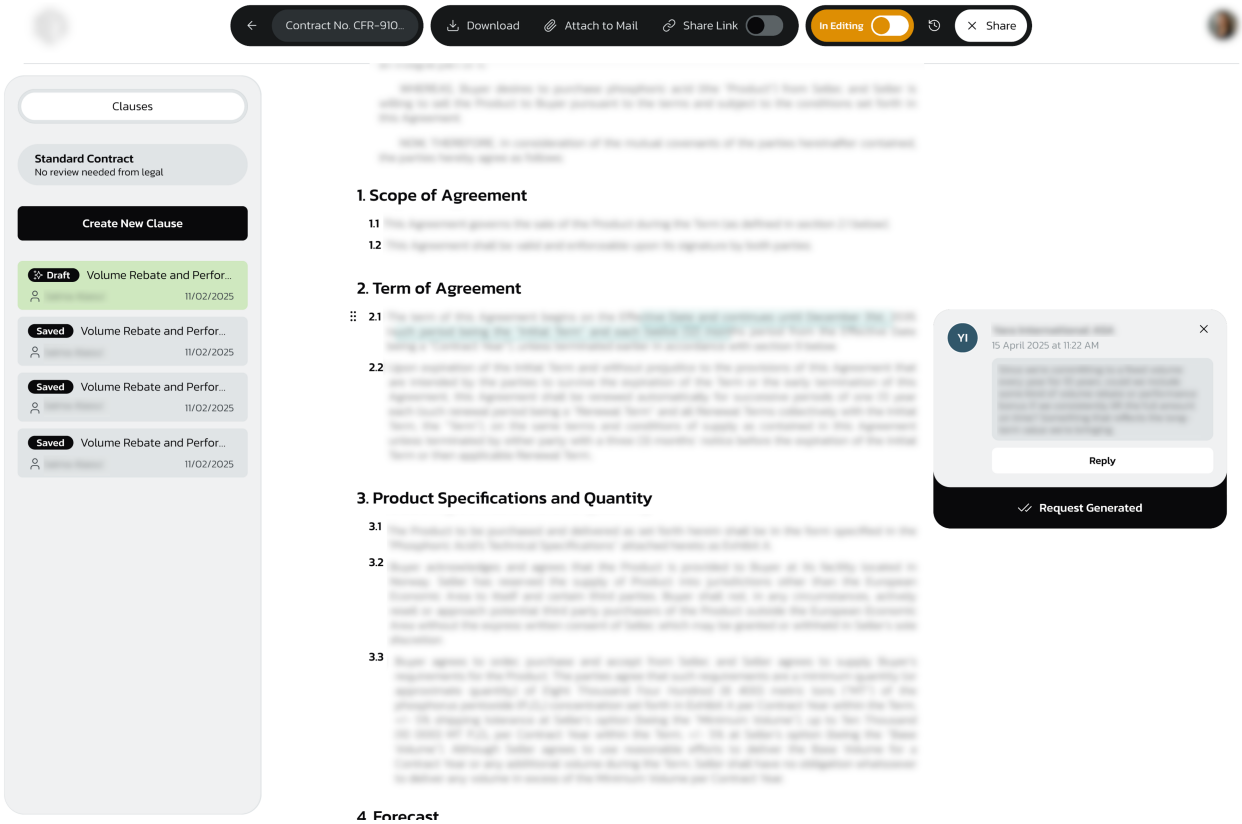

Figure 4.28. Create New Clause from Comment

Upon initiating a clause request, Sales users compose a detailed description of their desired modification. To enhance clarity and precision, the platform provides multiple input options,

including voice recordings, file uploads, and insertion of dynamic fields using the "@" symbol. Additionally, users can refine their descriptions utilizing integrated LLM assistance, ensuring the request accurately communicates the intended modifications.

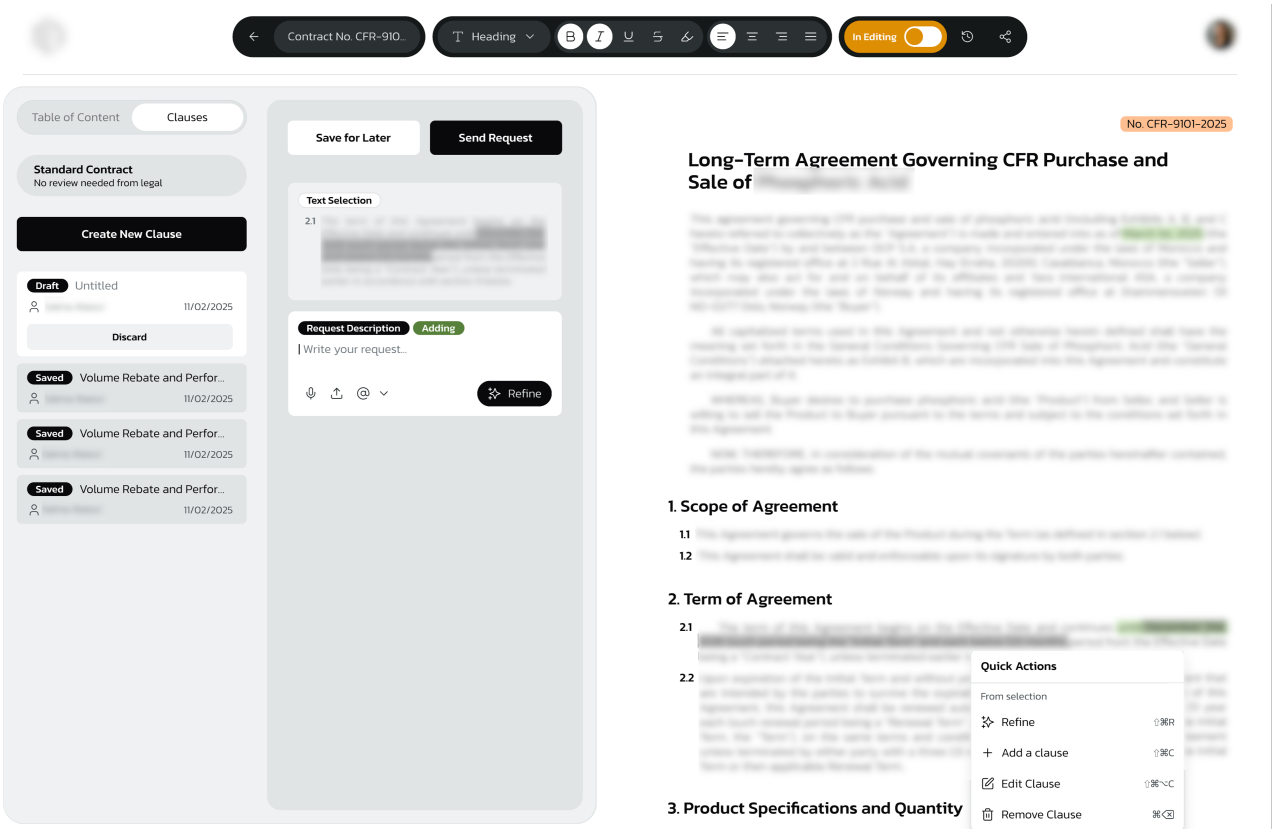

Figure 4.29. Clause Request Description and Enhancement

After submission, the Legal team receives a notification indicating the clause request along with details of the Sales user who submitted it. Upon reviewing the notification, Legal users access the clause management interface to review the Sales user's request comprehensively. This interface provides a dedicated editor enabling Legal users to add, modify, or delete clauses and subclauses directly. The editor also includes the same capabilities provided to Sales users, such as refining with LLM assistance, voice recording, and file uploads. Legal users can finalize their changes immediately or save their edits for later completion.

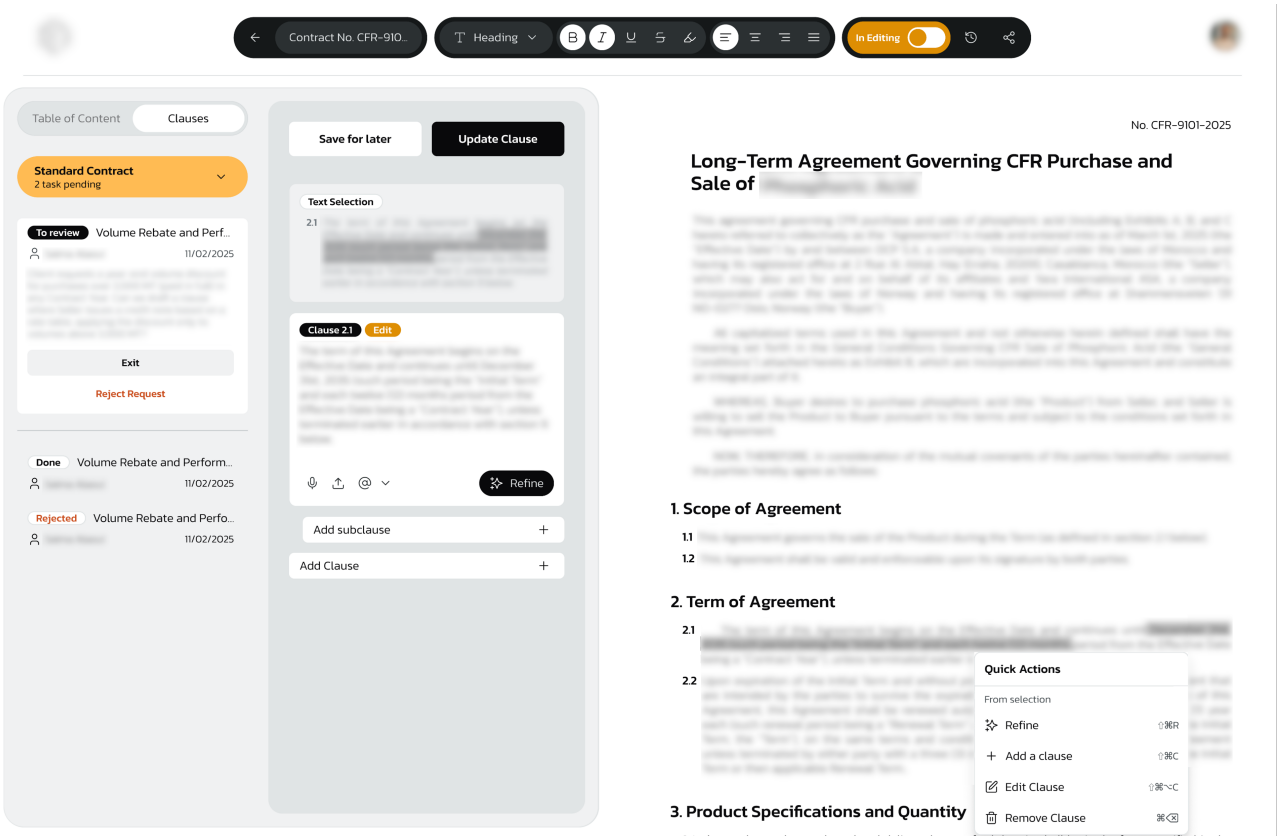

Figure 4.30. Clause Request Review Interface

4.2.4 Contract Review

Upon marking a contract as *"In Review"* by Sales users, Legal users can initiate a thorough review either manually or through AI-driven assistance. Two specialized AI agents are available for this purpose:

- **Reviewer Agent:** Conducts Data Accuracy and Consistency checks, verifying essential details such as dates, amounts, and product specifications. It also evaluates text clarity, grammar, and formatting consistency.
- **Guardian Agent:** Performs comprehensive Legal Security evaluations, ensuring compliance with legal standards, verifying clauses, payment terms, Incoterms responsibilities, and detecting deviations from predefined templates.

Upon execution, the Reviewer agent provides an overall quality score with targeted suggestions to enhance the contract. Legal users can iteratively apply these recommendations, aiming to achieve optimal compliance (100%).

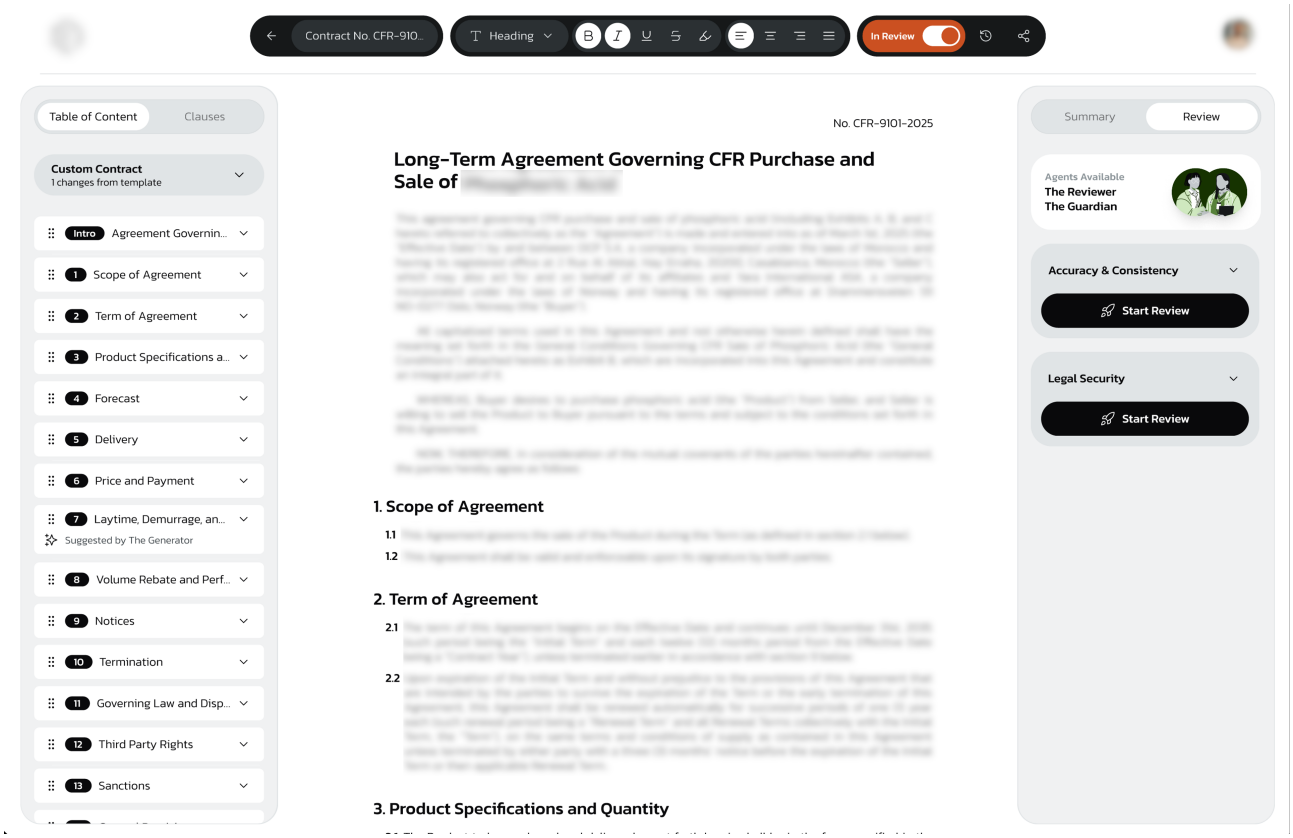

Figure 4.31. Contract Review Panel

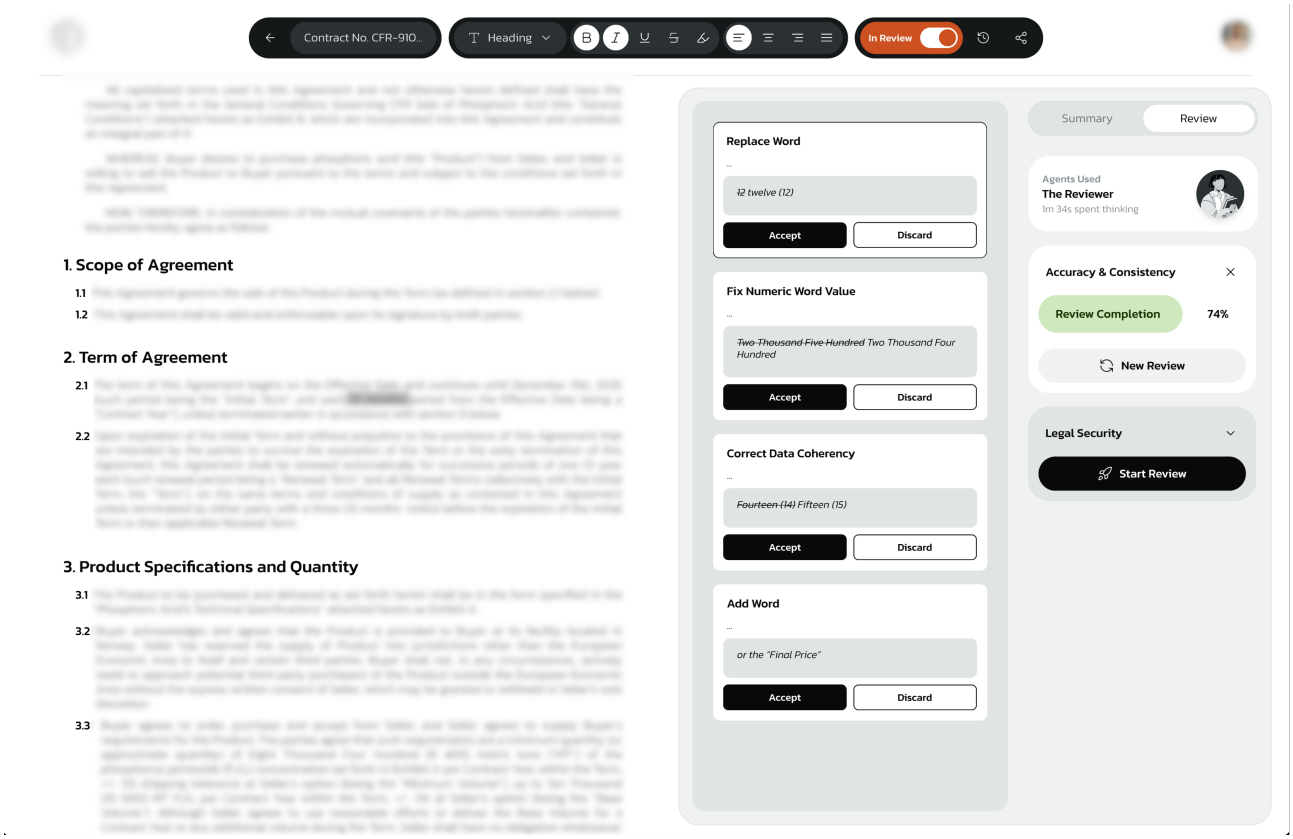

Figure 4.32. Contract Review Results

4.2.5 Contract History

Users access the contract's History mode via the history button on the navigation bar. This mode automatically tracks each contract modification—triggered by status changes, downloads, or manual saves—while also allowing manual creation of specific versions. Each version is listed chronologically in the right panel, facilitating easy navigation and management.

From this interface, users can perform several key actions:

- **Restore Version:** Reverts the entire contract to a previously saved version.
- **Rename Version:** Allows clear labeling and identification of historical contract states.
- **Paragraph-Level Management:** Enables users to individually restore or copy previous paragraphs into the current document. Restoration replaces current content, while copying appends historical content alongside existing text.

The integrated TipTap editor automatically synchronizes all changes, ensuring consistency across all user views.

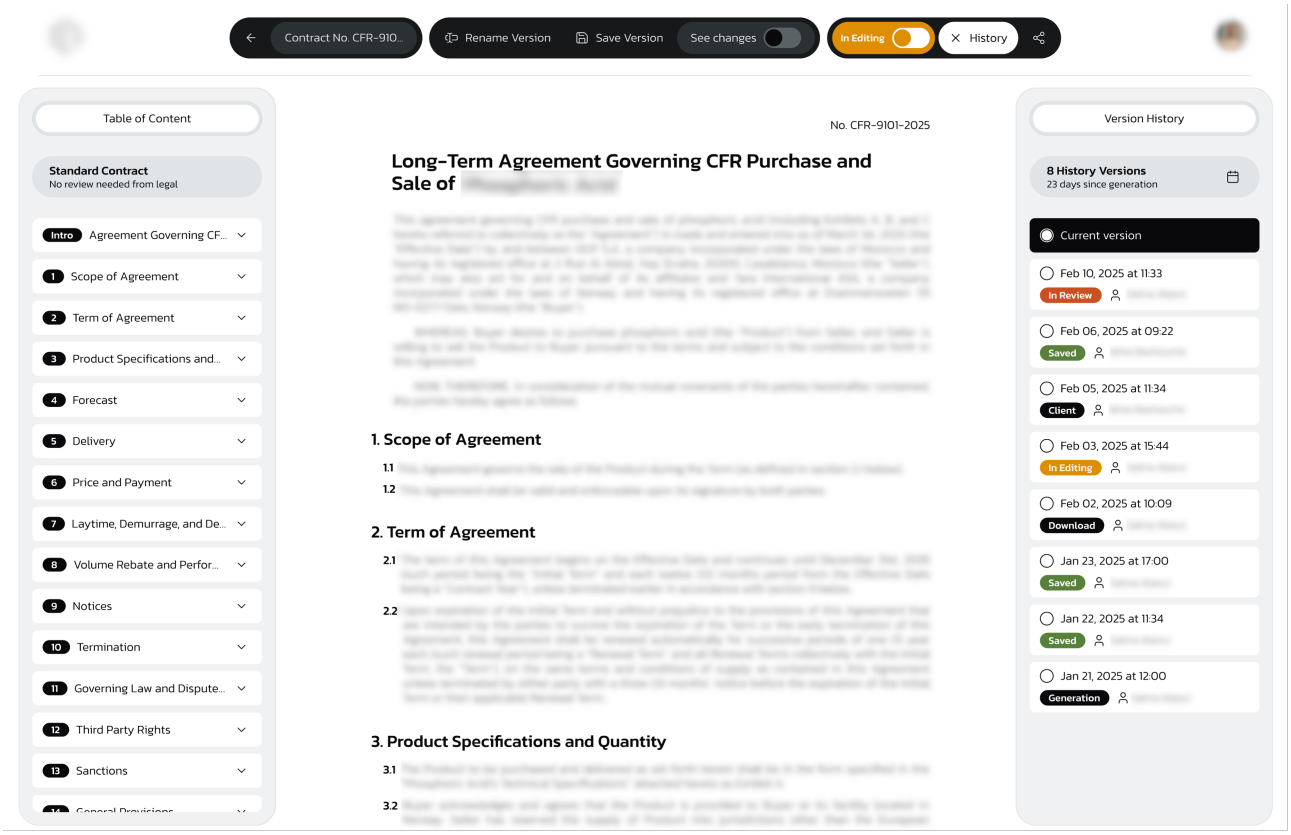

Figure 4.33. Contract History Interface

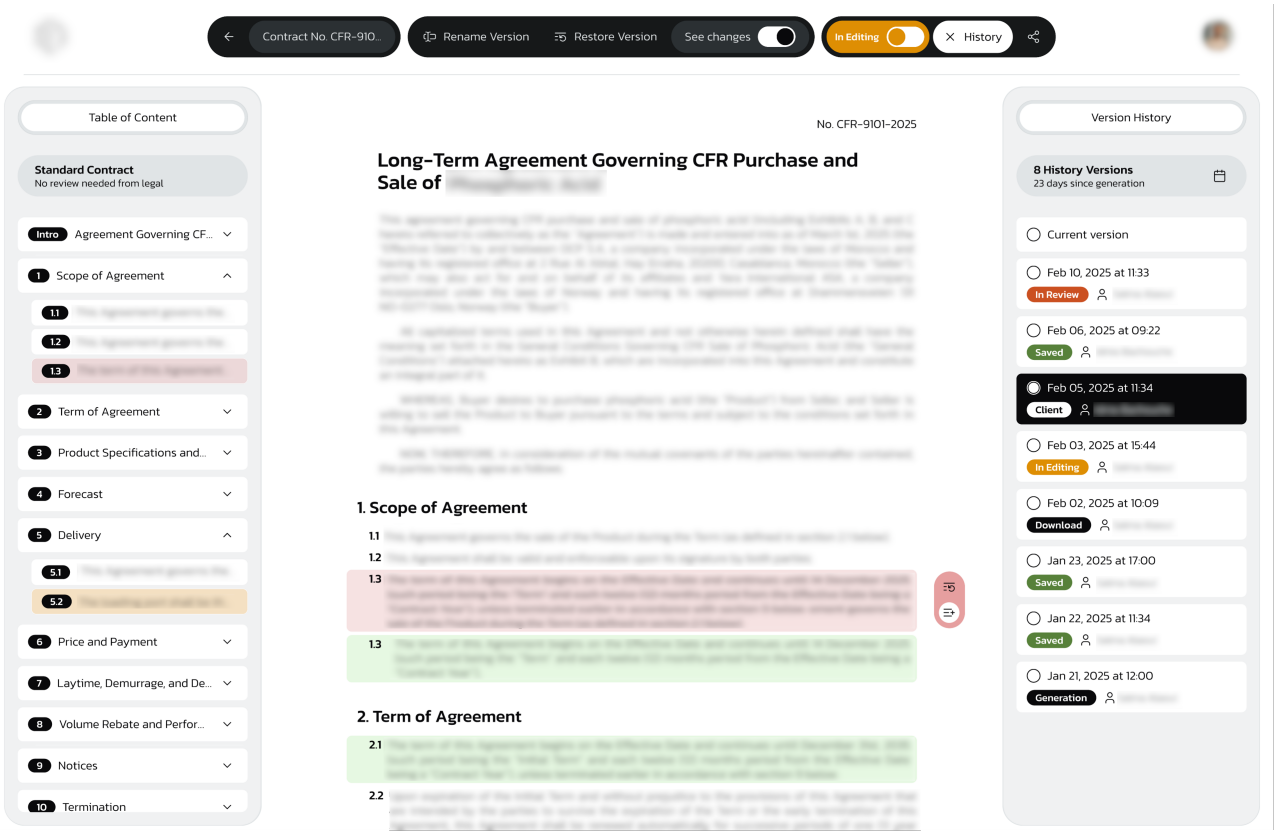

Figure 4.34. Paragraph Version Tracking and Restoration

4.2.6 Finalized Contract

Upon successful reviews and edits, contracts are finalized, transitioning into the finalized status. Finalized contracts are exportable as PDF/Word documents, shareable via secure links or email attachments, enabling seamless client communications.

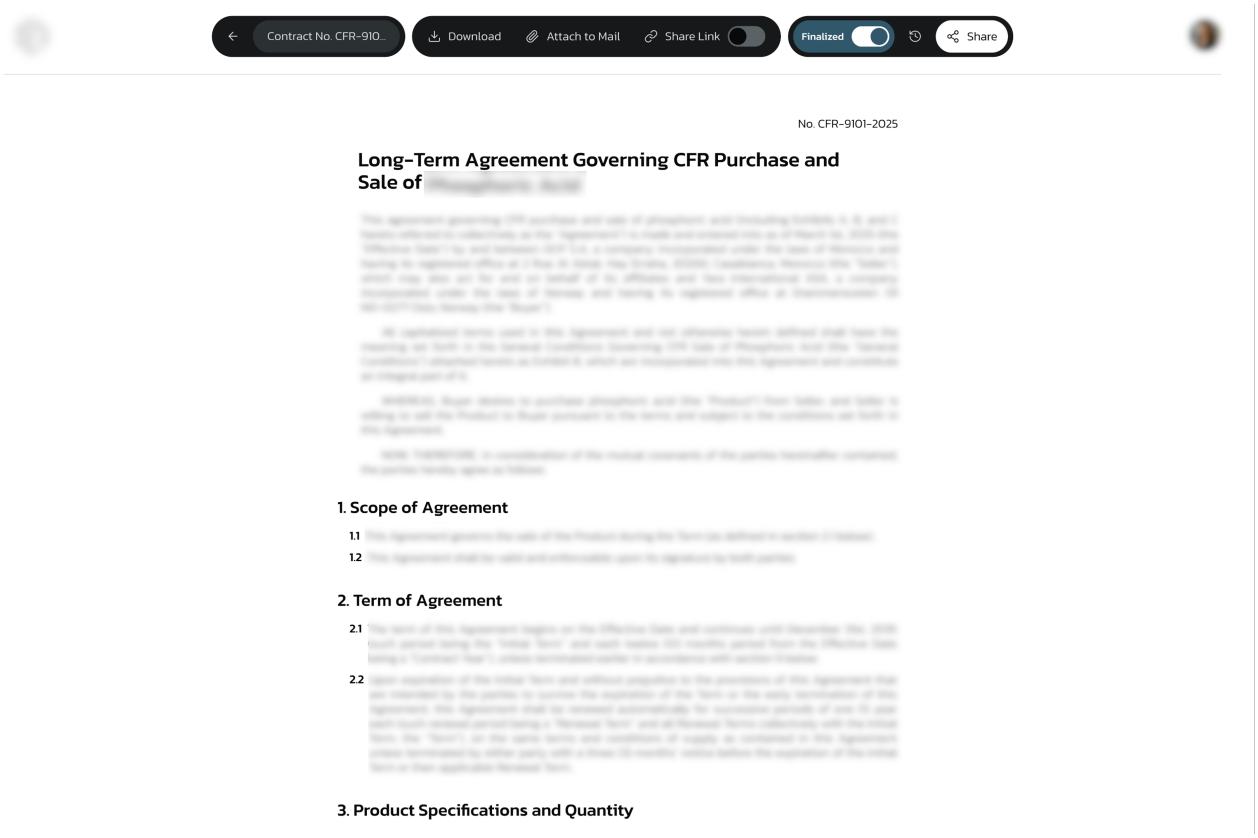

Figure 4.35. Finalized Contract Interface

4.3 Verification of Requirements

The verification of functional and non-functional requirements is an essential phase in ensuring the developed solution aligns with the initial specifications identified during the analysis phase. This section outlines how the implemented solution satisfies the requirements detailed in Chapter 2.

4.3.1 Verification of Functional Requirements

The intelligent contract management solution has undergone partial verification of its functional capabilities based on the requirements defined in Chapter 2. Early-stage validation focused on core functionalities and was conducted through stakeholder walkthroughs, feature-specific testing, and user feedback sessions.

Key functionalities—including contract generation, clause management, sections editing, and AI-assisted contract review—were demonstrated iteratively and reviewed in sessions with Sales and Legal users. These interactions helped validate the practical usability and role-specific behavior of the system from a user standpoint.

A full round of manual and integration testing is planned and will be executed in the remaining phase of the internship. These tests will ensure proper interaction between modules and confirm end-to-end functionality under realistic system conditions.

To provide a transparent overview of current verification progress, the following traceability matrix summarizes the verification method and current status of each functional requirement:

Requirement Description	Verification Method	Status	Remarks
Contract Generation	Stakeholder demo	Verified	Validated with Sales team using 8 templates and past contracts
Clause Management (add/edit/delete)	Sales & Legal user reviews	Verified	Legal team proposed clause tagging enhancement
Editing Contract Dynamic Fields	UI testing by Sales users	Verified	All fields editable as expected
Contract Review & Compliance	Prototype demo & feedback session	Verified	Confirmed using 5 contract samples
Export, Sharing & Management	Feature testing, stakeholder demo	Verified	Export output met all format requirements
Contract Version History	Manual test & backend log check	Verified	Functionality confirmed through logs
Role-Based Access (Sales/Legal)	Access validation	In Progress	UI functional, but backend logging under development

Table 4.1. Verification Status of Functional Requirements

This matrix captures the current verification status, with emphasis on areas already validated and those scheduled for integration testing in the next phase.

4.3.2 Verification of Non-Functional Requirements

The non-functional aspects, crucial for user satisfaction and system reliability, were verified through targeted approaches as follows:

- **Usability:** The platform was designed with an intuitive and familiar interface, inspired by widely used enterprise applications. Initial user feedback, collected from a select group of Sales and Legal users, confirmed that the interface met the usability standards, ensuring a comfortable and efficient user experience.
- **Maintainability:** Regular code reviews and adherence to best practices in software architecture were employed to validate maintainability. The adoption of a modular, service-oriented architecture facilitated easier updates and extensions, allowing new functionalities to be integrated efficiently. Future scheduled unit and integration tests will further ensure system stability and maintainability.
- **Security:** Security verification involved the integration and configuration of Azure Entra ID, ensuring robust authentication and role-based access control. Regular security audits and compliance checks were established to detect and rectify vulnerabilities promptly. Further penetration tests and comprehensive security assessments are planned to maintain and enhance the platform's security posture.
- **System Responsiveness:** The system's responsiveness was evaluated through interactive testing of user-facing components. Smooth navigation, quick interface updates, and prompt visual feedback were observed during typical usage, indicating high responsiveness under normal load conditions.

- **Scalability:** The deployment setup on Docker and Kubernetes allows for efficient horizontal scaling across services. Tests involving multiple users and simultaneous operations confirmed that the platform maintains responsiveness and stability as usage demands increase.

4.4 Conclusion

This chapter detailed the implemented functionalities and justified the technology stack supporting the platform. It also validated the system against the initial specifications, confirming its reliability, scalability, and readiness for enterprise use.

General Conclusion & Perspectives

This report has detailed the design and development of an AI-powered intelligent contract management platform, carried out during my internship at BCG X. The project addresses a key challenge in legal operations: the inefficiency, inconsistency, and compliance risks associated with manual contract workflows, particularly in drafting, validation, and approval processes. These workflows are time-consuming, prone to error, and difficult to standardize across teams.

To respond to this challenge, the developed platform offers an end-to-end solution that supports contract generation from dynamic templates, variable insertion, clause request management, collaborative editing, and legal review. Built with modular components and integrated AI agents, the platform automates critical tasks while maintaining traceability and legal rigor. It also introduces AI-driven features such as clause refinement, compliance checks using LLMs, and structured document modeling aligned with legal conventions.

Technologies used include Azure OpenAI, LangChain, LangGraph, FastAPI, React, and Elasticsearch, enabling scalable integration of autonomous AI agents and robust backend logic. The platform demonstrates how Agentic AI can transform traditional legal processes into structured, efficient, and adaptive workflows—reducing cycle time, improving accuracy, and supporting enterprise-level collaboration in a secure environment.

This experience allowed me to apply and enhance my skills in software architecture, AI integration, and secure systems engineering. Looking ahead, the platform could evolve through predictive clause analytics, stronger modularity, and customizable user interfaces. These advancements would further position the platform as a critical enabler of digital transformation within legal departments and beyond.

Bibliography

[1] Boston consulting group. https://www.bcg.com/bcg/logo/.

[2] Bcg x | the tech build and design division of bcg. https://www.bcg.com/x/.

[3] Executive leadership board. https://theorg.com/org/bcg-x/.

[4] What is scrum? https://www.pm-partners.com.au/insights/the-agile-journey-a-scrum-overview/.

[5] Clickup logo. https://clickup.com/brand/.

[6] Slack logo. https://slack.com/media-kit/.

[7] Zoom logo. https://www.zoom.com/en/about/media-kit/.

[8] Microsoft teams logo. https://www.microsoft.com/en-us/microsoft-teams/group-chat-software/.

[9] Microsoft outlook logo. https://www.microsoft.com/en-us/microsoft-365/outlook/email-and-calendar-software-microsoft-outlook/.

[10] Miro logo. https://miro.com/about/.

[11] Pandadoc logo. https://brandfetch.com/pandadoc.com/.

[12] Harvey ai logo. https://www.harvey.ai/.

[13] Docdraft logo. https://www.docdraft.ai/.

[14] The importance of prompt engineering in maximising benefits of existing llms. https://medium.com/@ah_amin_b/context-is-all-you-need-the-importance-of-prompt-engineering-in-maximising-benefits-of-existing-2d8f982d98ef/.

[15] Ai agents in your workforce. https://www.electronicsforu.com/technology-trends/ai-agents-in-your-workforce/.

[16] Agentic ai vs generative ai: Understanding the key differences and impacts. https://medium.com/@myscale/agentic-ai-vs-generative-ai-understanding-the-key-differences-and-impacts-e4527bb7c4ee/.

[17] Multiagent planning in ai. https://www.geeksforgeeks.org/multiagent-planning-in-ai/.

[18] Langchain platfrom overview. https://x.com/zhanghaili0610/status/1696554153609253033/.

[19] Orchestrating agentic systems. https://medium.com/@raunak-jain/orchestrating-agentic-systems-eb945d305083/.

[20] Tiptap. https://tiptap.dev/.

[21] Announcing a new openai feature for developers on azure. https://azure.microsoft.com/de-de/blog/announcing-a-new-openai-feature-for-developers-on-azure/.

[22] Langchain logo. https://www.langchain.com/.

[23] Langgraph logo. https://www.langchain.com/langgraph/.

[24] Libvirt overview. https://www.elastic.co/elasticsearch/.

[25] Typescript logo. https://www.typescriptlang.org/branding/.

[26] React logo. https://react.dev/.

[27] Tiptap logo. https://tiptap.dev/docs/.

[28] Jest logo. https://freebiesupply.com/logos/jest-logo/.

[29] Python logo. https://www.python.org/community/logos/.

[30] Fastapi logo. https://fastapi.tiangolo.com/.

[31] Pytest logo. https://docs.pytest.org/en/stable/.

[32] Postgresql logo. https://www.postgresql.org/about/press/presskit16/base/.

[33] Postman logo. https://www.postman.com/legal/logo-usage/.

[34] curl logo. https://curl.se/logo/.

[35] Sonarqube logo. https://www.sonarsource.com/brand-identity/.

[36] Azure pipelines logo. https://azure.microsoft.com/en-us/products/devops/pipelines/.

[37] Docker logo. https://www.docker.com/company/newsroom/media-resources/.

[38] Kubernetes logo. https://github.com/kubernetes/kubernetes/blob/master/logo/logo.png.

[39] Terraform logo. https://developer.hashicorp.com/terraform/.

[40] Vscode logo. https://code.visualstudio.com/brand.