Jordan Andrade Custodio da Silva

Luma - Agent Studio: A Scalable Platform for AI Agent Creation : Empowering businesses through streamlined AI agent development and deployment

SÃO PAULO
2025

Jordan Andrade Custodio da Silva

Luma - Agent Studio: A Scalable Platform for AI Agent Creation : Empowering businesses through streamlined AI agent development and deployment

Final Course Project submitted to the Institute of Technology and Leadership (INTELI), to obtain a bachelor's degree in Software Engineering

Advisor: Prof. Filipe Resina

SÃO PAULO
2025

Cataloging in Publication
Library and Documentation Service
Institute of Technology and Leadership (INTELLI)
Data entered by the author.
(Cataloging record with international cataloging data, according to NBR 14724. The record will be completed later, after approval and before the final version is deposited. The completion of the cataloging record is the responsibility of the institution's library.)

Sobrenome, Nome

Título do trabalho: subtítulo / Nome Sobrenome do autor; Nome e Sobrenome do orientador. – São Paulo, 2025.

nº de páginas : il.

Trabalho de Conclusão de Curso (Graduação) – Curso de [Ciência da Computação] [Engenharia de Software] [Engenharia de Hardware] [Sistema de Informação] / Instituto de Tecnologia e Liderança.

Bibliografia

1. [Assunto A]. 2. [Assunto B]. 3. [Assunto C].

CDD. 23. ed.

Acknowledgments

The successful completion of this work would not have been possible without the invaluable support, guidance, and encouragement of numerous individuals and institutions. It is with profound gratitude that I acknowledge their contributions.

My deepest and most sincere appreciation is reserved for my academic supervisor, Professor Filipe Resina. His expertise, insightful feedback, and unwavering commitment were fundamental pillars throughout the entire research and development process. Professor Resina's rigorous mentorship, strategic oversight of the Luma - Agent Studio project, and dedication to ensuring both academic excellence and practical relevance were indispensable. His constructive criticism and intellectual guidance consistently challenged me to refine my concepts, navigate complexities, and elevate the standard of this work.

I owe a special debt of gratitude to the partner company for entrusting me with this challenging and innovative project. The opportunity to address a tangible business problem and develop a functional, scalable platform has been an unparalleled professional and academic experience. My sincere thanks go to all the professionals at Boston Consulting Group (BCG) who provided the initial context, necessary resources, and continuous trust, thereby enabling the transition from a theoretical framework to a concrete technological solution.

Furthermore, I wish to express my heartfelt thanks to the faculty and staff of the Instituto de Tecnologia e Liderança (Inteli). The institution's pioneering project-based learning model provided the essential pedagogical structure and innovative environment that made this applied research not only possible but also deeply formative.

To my colleagues and friends, thank you for the stimulating discussions, collaborative spirit, and shared resilience throughout this intensive journey. Your camaraderie was vital in overcoming challenges and enriching the overall experience.

Lastly, and most importantly, my eternal gratitude goes to my family. To my parents, for their unconditional love, immense sacrifices, and steadfast belief in my path. Your emotional and moral support has been the unwavering foundation upon which this achievement stands. This accomplishment is as much a testament to your encouragement as it is to my efforts.

While any limitations in this work remain my sole responsibility, its merits are undeniably a collective achievement. To all who contributed directly and indirectly, I extend my most genuine thanks.

Resume

Andrade, Jordan. **Luma - Agent Studio: A Scalable Platform for AI Agent Creation**. 2025. nº de folhas. TCC (Undergraduate) - Engenharia de Software, Institute of Technology and Leadership (INTELI), São Paulo, 2025.

This academic project addresses the high cost and resource intensity associated with developing custom Artificial Intelligence (AI) solutions for businesses, which creates a bottleneck in adopting AI-driven tools. The primary objective was to develop Luma - Agent Studio, a scalable, self-service web platform that empowers users, regardless of their technical expertise, to create, customize, and deploy AI agents tailored to specific business contexts. The platform leverages advanced technologies, including Retrieval-Augmented Generation (RAG), Large Language Models (LLMs), and vector databases, to process various document types (PDF, Word, audio, video) and enable context-aware interactions through an intuitive chat interface. The methodology followed an Agile development framework, structured into four ten-week modules comprising bi-weekly sprints. The technical implementation involved a React-based frontend, a FastAPI backend orchestrated with LangChain, and Chroma DB as a vector database, all deployed on a cloud architecture using Infrastructure as Code (Terraform) on AWS. The results encompass the delivery of a functional Minimum Viable Product (MVP) that successfully automates the agent creation workflow, integrates secure multi-tenant authentication, and demonstrates significant potential for reducing the time and financial investment required for AI Proofs of Concept (POCs). The conclusion affirms that the platform effectively democratizes access to customized AI, offering a strategic tool for businesses to enhance operational efficiency and innovation. Future work should focus on expanding LLM agnosticism, implementing advanced agent workflows, and conducting extensive usability testing to refine the user experience.).

Palavras-Chave: Artificial Intelligence; AI Agents; Retrieval-Augmented Generation (RAG); Software Development; Agile Methodology; Cloud Computing.

ABSTRACT

Andrade, Jordan. **Luma - Agent Studio: A Scalable Platform for AI Agent Creation**. 2025. nº de folhas. TCC (Undergraduate) - Engenharia de Software, Institute of Technology and Leadership (INTELI), São Paulo, 2025

This academic project addresses the challenge of high costs and resource intensity in developing custom Artificial Intelligence (AI) solutions for businesses, which creates a significant barrier to the adoption of AI-driven tools. The central objective was to design and implement Luma - Agent Studio, a scalable, self-service web platform that enables users, including those without deep technical expertise, to create, customize, and deploy contextual AI agents. The platform utilizes advanced technologies such as Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs) to process diverse document formats - including PDFs, Word files, audio, and video - and facilitate intelligent, context-aware dialogue through a conversational interface. Development followed an Agile methodology, structured into four ten-week modules with bi-weekly sprints, ensuring iterative progress and continuous stakeholder alignment. The technical architecture comprises a React-based frontend, a FastAPI backend integrated with LangChain for agent orchestration and LlamaIndex for document processing, and Chroma DB as a vector database. The system was deployed on Amazon Web Services (AWS) using Infrastructure as Code with Terraform, incorporating secure JWT-based authentication with strict multi-tenant isolation. The primary result is a fully functional Minimum Viable Product (MVP) that successfully automates the end-to-end workflow of AI agent creation and deployment. The platform demonstrates a substantial reduction in the time and financial investment traditionally required for AI Proofs of Concept (POCs), thereby democratizing access to tailored AI solutions. It is concluded that Luma - Agent Studio represents an effective tool for businesses to enhance operational efficiency, accelerate innovation, and maintain competitiveness in an AI-driven market. Future developments should focus on expanding model agnosticism, implementing advanced data analysis features, and validating the platform through formal usability tests..

Key words: Artificial Intelligence; AI Agents; Retrieval-Augmented Generation (RAG); Agile Development; Cloud Computing.

List of Illustrations (Optional item – NBR14724, item 4.2.1.9)

List of Tables (optional item – NBR14724, item 4.2.1.10)

List of Abbreviations and Acronyms (optional item – NBR14724, item 4.2.1.11)

ACRONYM 1 Full name of acronym 1
ACRONYM 2 Full name of acronym 2
ACRONYM 3 Full name of acronym 3

Summary **(Required item – NBR14724, item 4.2.1.13; NBR 6027)**

1 Introduction

The rapid evolution of Artificial Intelligence (AI), particularly through advancements in Large Language Models (LLMs) and generative AI, has ushered in a transformative era for business operations and strategic innovation. Organizations across sectors recognize the immense potential of AI to automate complex tasks, enhance decision-making, and create new value streams. A prominent and powerful paradigm enabling this is Retrieval-Augmented Generation (RAG), which combines the generative capabilities of LLMs with dynamic access to proprietary, contextual knowledge bases. This synergy allows for the creation of highly specialized AI agents-intelligent systems capable of understanding nuanced queries, retrieving relevant information from internal documents, and generating accurate, context-aware responses. Such agents promise to significantly improve productivity, customer service, and internal knowledge management.

However, a critical bottleneck impedes the widespread adoption of this technology: the high cost, complexity, and resource intensity associated with developing custom AI solutions. Building a functional Proof of Concept (POC) for a contextual AI agent typically requires assembling a multidisciplinary team of data scientists, machine learning engineers, and full-stack developers for several months. This process involves significant investment in research and development, infrastructure setup, data pipeline construction, and continuous iteration. For many companies, especially those without extensive in-house AI expertise, these barriers render custom AI development prohibitively expensive and slow, stifling innovation and competitive agility. This gap between technological potential and practical accessibility defines a salient problem in the contemporary digital landscape.

This academic project, developed in partnership with Boston Consulting Group (BCG), directly addresses this challenge. It focuses on the conception, design, and implementation of Luma - Agent Studio, a scalable and user-centric web platform designed to democratize the creation of sophisticated AI agents. The core proposition of Luma is to transform a traditionally complex and resource-heavy

development process into a streamlined, self-service experience. The platform empowers consultants and client teams-regardless of their deep technical proficiency-to rapidly build, customize, and deploy AI agents tailored to specific business contexts by simply uploading their own documents (PDFs, Word files, audio, video) and defining agent behavior through an intuitive interface.

Therefore, the central objective of this work is to deliver a fully functional Minimum Viable Product (MVP) for the Luma - Agent Studio platform. This objective unfolds into several specific technical and methodological goals: to architect and develop a modular system comprising a React-based frontend and a FastAPI backend integrated with LangChain and LlamaIndex; to implement a secure, multi-tenant cloud infrastructure on AWS ensuring data isolation and scalability; to embed a robust RAG pipeline for processing diverse document formats; and to establish a professional CI/CD workflow for maintainable and iterative development. The project follows an Agile methodology, structured into four development modules, to ensure disciplined execution, continuous stakeholder feedback integration, and the timely delivery of a high-quality, production-ready tool.

The relevance of this project is multifaceted. For the partner organization, BCG, it represents a strategic asset that can drastically reduce the cost and time required to deliver AI POCs to clients, enhancing service agility and value proposition. For the broader business ecosystem, it offers a tangible solution to a pervasive market problem, lowering the barrier to entry for AI adoption. Academically, it serves as a comprehensive case study in applied software engineering, integrating cutting-edge AI frameworks, cloud architecture, and human-computer interaction principles into a single, cohesive system. The following sections of this thesis will detail the methodological framework, architectural design, implementation process, and resulting outcomes of this endeavor, concluding with an analysis of its impact and future potential.

1.1. **Partner Company Context:**

Sector and Corporate Profile

Boston Consulting Group (BCG) operates within the global management consulting sector. It is one of the world's preeminent advisory firms, renowned for its strategic expertise in helping organizations across private, public, and non-profit sectors navigate complex challenges, drive innovation, and achieve sustainable competitive advantage. As a "Big Three" management consultancy, BCG employs thousands of professionals worldwide and maintains a significant presence across all major economic regions. The affected area for this project resides within BCG's Technology & Digital Advantage practice, specifically focusing on the intersection of advanced analytics, artificial intelligence, and digital transformation. This practice is dedicated to helping clients leverage cutting-edge technologies to reinvent their business models, operations, and customer experiences.

Motivation and Strategic Imperative

The Luma - Agent Studio project is profoundly strategic for BCG, addressing a critical nexus of market demand, operational efficiency, and service evolution. Its motivation is threefold:

First, it responds to a surging and unambiguous client demand for practical, scalable AI solutions. Clients across industries are urgently seeking to harness generative AI and RAG technologies to unlock value from their proprietary data, automate knowledge work, and enhance customer interactions. However, they frequently lack the internal expertise, bandwidth, or risk tolerance to build such solutions from the ground up. By developing Luma, BCG positions itself not merely as an advisor but as an enabler and co-creator, providing a tangible, productized tool that accelerates the path from AI strategy to implemented value. This transforms the consulting engagement model from purely strategic recommendation to hands-on, rapid solutioning.

Second, the project is a direct response to an internal operational challenge within BCG's own service delivery. The traditional process of developing a custom AI Proof of Concept (POC) for a client is highly resource-intensive, requiring the assembly of

specialized technical teams and consuming weeks or months of effort. This creates a bottleneck, limiting the number of clients BCG can serve in this domain and extending the time-to-value. Luma - Agent Studio is designed to productize and industrialize this development process. By empowering BCG consultants and client teams to create agents through a configurable platform, the project dramatically reduces the marginal cost and time per POC, enhancing scalability, improving resource allocation, and allowing consultants to focus on higher-value strategic integration rather than foundational technical build.

Finally, this initiative represents a strategic investment in long-term capability building and market positioning. In a competitive landscape where differentiation is key, owning a proprietary, cutting-edge platform for AI agent creation establishes BCG as a forward-thinking leader in applied AI. It moves the firm beyond leveraging third-party tools to developing its own intellectual property and technological assets. This not only strengthens the firm's value proposition but also creates potential new revenue streams through platform licensing or integrated service offerings. In essence, Luma is not just a project deliverable; it is a strategic asset designed to future-proof BCG's service portfolio, enhance its competitive moat, and directly translate the firm's deep industry and strategic knowledge into a scalable, technology-powered advantage.

1.2. **Problem Definition (Corporate Pain Point):**

Detailed Description of the Process and Malfunction

The corporate pain point addressed by this project originates in the end-to-end process for developing custom, context-aware AI agents as client deliverables, specifically Proofs of Concept (POCs) and pilot solutions. The current, pre-Luma process is characterized by a highly manual, bespoke, and resource-intensive workflow that functions more as a series of artisanal projects than a scalable, repeatable service line.

The process typically unfolds as follows: Upon identifying a client need-such as a chatbot for internal knowledge bases, a sales assistant, or a document analysis tool-a BCG case team initiates a scoping phase. This is followed by the assembly of a dedicated technical task force, often comprising data scientists, machine learning engineers, and full-stack developers. This team must then manually execute a complex chain of tasks: data ingestion and cleaning from client-provided documents (in varied formats like PDFs, Word files, and spreadsheets); the design and implementation of a custom data pipeline for chunking and embedding; the integration and fine-tuning of a Large Language Model (LLM) via APIs; the development of a RAG (Retrieval-Augmented Generation) orchestration layer using frameworks like LangChain; the creation of a frontend chat interface; and finally, deployment and integration testing. Each step requires deep specialized knowledge, and the entire sequence is recreated nearly from scratch for each new client or use case.

The core malfunction in this process is its inherent lack of standardization, automation, and democratization. It suffers from significant duplication of effort, as foundational components are rebuilt repeatedly. It creates a critical dependency on scarce, high-cost technical talent (LLM engineers, RAG specialists), forming a bottleneck that constrains throughput. Furthermore, it places a high cognitive and coordination burden on case teams and project managers, who must act as integrators between the client's domain experts and the technical build team. This leads to elongated timelines, elevated costs, and inconsistent outcomes, as the quality of the final deliverable is heavily dependent on the specific composition and availability of the technical team assigned. The process, in essence, fails to leverage economies of scale or productized knowledge, treating each AI solution as a unique construction project instead of a configurable instance of a unified platform.

Current Metrics

The project aims to directly improve the following key performance indicators, which define the baseline of the current, inefficient process:

Average Development Time per POC: The current timeline from project kick-off to delivery of a functional, client-ready AI agent POC spans 8 to 12 weeks. A significant

portion of this time is dedicated to environment setup, foundational coding, and resolving integration issues common to greenfield development.

Resource Cost per POC: The direct financial outlay is substantial, driven by personnel costs. A typical POC requires a dedicated technical team of 3-4 specialists (e.g., data engineers, ML engineers, developers) for the full project duration. At prevailing market rates for this expertise, the baseline cost per POC falls within the range of R$150,000 to R$250,000, exclusive of cloud infrastructure and software licensing.

Technical Resource Utilization (Bottleneck): A qualitative but critical metric is the dependency ratio. Currently, the capacity of the AI/ML service line is directly constrained by the availability of its specialized technical staff. This creates a pipeline where client demand often exceeds available technical bandwidth, leading to delayed project starts or rejected opportunities.

Consultant Autonomy & Efficiency: In the current model, BCG consultants acting as project leads have low autonomy (near-zero) in the technical build phase. They must function as intermediaries, translating client needs for the technical team and managing the feedback loop. This intermediary role is inefficient and can introduce miscommunication, measured indirectly through iteration cycles required to align the delivered agent with client expectations.

Time-to-Value for the Client: From the client's perspective, the elapsed time between articulating a business need and interacting with a tangible AI solution is prohibitively long (aligning with the 8-12 week development cycle). This delay dampens enthusiasm, allows business priorities to shift, and reduces the perceived agility of the consulting engagement.

The Luma - Agent Studio project is fundamentally designed to establish a new baseline for these metrics by productizing the development workflow, thereby targeting drastic reductions in time and cost, elimination of the technical bottleneck, and a marked increase in consultant-led execution speed and autonomy.

1.3. Proposed Solution and Expected Contribution:

The Luma - Agent Studio Platform

The proposed solution to the identified corporate pain point is Luma – Agent Studio, a comprehensive, self-service web platform engineered to industrialize the creation and deployment of contextual AI agents. Luma is not a single-point tool but an integrated computational ecosystem designed to encapsulate and automate the entire complex workflow that is currently performed manually. Its core architecture is built upon a modular, API-first, and cloud-native design philosophy, ensuring scalability, maintainability, and seamless integration.

The platform's computational logic is structured around three primary user-facing components and a robust backend orchestration layer:

Agent Creation Interface (Manager Frontend): A React-based, intuitive web application where BCG consultants or client managers can define an agent through a structured form. This includes naming the agent, providing descriptive instructions (system prompts), configuring behavioral parameters, and selecting which knowledge bases (uploaded document collections) it can access. This interface abstracts away the need to write code for agent logic.

Knowledge Ingestion & Processing Pipeline: Upon document upload (supporting PDF, DOCX, TXT, audio, and video files), the platform automatically triggers a backend pipeline. This pipeline, powered by LlamaIndex, handles text extraction, intelligent chunking, embedding generation using state-of-the-art models, and persistent storage of vectors and metadata in a dedicated Chroma DB vector database. This process transforms unstructured documents into a queryable knowledge base, which is the foundation for the RAG capability.

Conversational Chat Interface (End-User Frontend): A clean, chat-style interface where end-users (e.g., client employees) interact with the deployed AI agents. User queries are processed by the backend, which uses LangChain to orchestrate the RAG flow: retrieving the most relevant document chunks from the vector database, injecting this context into a carefully engineered prompt, and calling a configured

Large Language Model (LLM)—initially OpenAI's models, but designed to be agnostic—to generate a contextual, accurate, and cited response.

The entire system is bound together by a FastAPI backend that serves as the central nervous system, managing business logic, user authentication, session management, and API communications. It is deployed on Amazon Web Services (AWS) using Infrastructure as Code (Terraform) for reliability and scalability, featuring isolated environments, load balancing, and secure data storage. A key innovation is the implementation of a multi-tenant architecture with JWT-based authentication, ensuring strict data isolation between different client companies and role-based access control within them. By productizing the entire stack - from data ingestion and vectorization to agent orchestration and chat deployment - into a unified, browser-accessible platform, Luma transforms AI agent development from a custom software engineering project into a configurable, self-service operation.

Contribution Objective

The primary contribution of the Luma – Agent Studio project is to establish a new, vastly more efficient performance baseline for delivering AI agent solutions. The project is designed to deliver the following quantifiable results, directly countering the current metrics of the problematic process:

Reduction in Average Development Time per POC: The platform aims to reduce the development and deployment timeline for a functional AI agent POC by a minimum of 80%. The target is to move from the baseline of 8-12 weeks down to a streamlined process of 1-2 weeks. This acceleration is achieved by eliminating greenfield development through pre-built, configurable modules and automated pipelines.

Reduction in Resource Cost per POC: By democratizing the creation process and drastically reducing the need for a dedicated senior technical team, the project targets a reduction of approximately 90% in the direct personnel cost per POC. This shifts the cost from the baseline of R$150,000–R$250,000 to a target range of R$15,000–R$25,000. The cost is primarily associated with a consultant's time for configuration and client liaison, rather than deep technical construction.

Elimination of the Technical Bottleneck (Increase in Capacity): The solution aims to decouple project throughput from the availability of specialized AI/ML engineers. The target is to enable any BCG consultant or trained client manager to independently create and deploy a sophisticated agent. This effectively aims to increase the potential delivery capacity of the AI service line by an order of magnitude, moving from a few concurrent projects limited by technical staff to a scalable model limited primarily by client demand and consultant bandwidth.

Increase in Consultant Autonomy & Efficiency: The platform is designed to grant consultants near-total autonomy in the technical build phase for standard agent use cases. The objective is to reduce the intermediary iteration cycles between consultant and technical team to zero for these standard scenarios, thereby improving communication efficiency, reducing misalignment, and accelerating the feedback loop with the end-client.

Acceleration of Client Time-to-Value: The overarching business objective is to dramatically shorten the client's time-to-value. The goal is to enable a client to go from a defined need to interacting with a tailored, knowledge-grounded AI agent in a matter of days, as opposed to months. This rapid demonstration of capability is crucial for maintaining engagement, validating concepts, and accelerating the overall digital transformation journey.

In summary, the quantifiable contribution of Luma is the transformation of AI agent delivery into a high-velocity, high-efficiency, and scalable service, directly measured by reductions of 80-90% in time and cost, and a proportional exponential increase in delivery capacity and strategic agility.

1.4. **Business Objectives:**

The deployment and adoption of the Luma - Agent Studio platform are expected to yield transformative results for Boston Consulting Group (BCG), generating value across operational, strategic, and commercial dimensions. These results are not merely incremental improvements but represent a fundamental shift in the firm's service delivery model and market positioning in the AI domain.

1. Operational Excellence and Scalability

The primary operational result is the industrialization of AI agent delivery. By productizing a previously bespoke and labor-intensive process, BCG will achieve dramatic gains in efficiency. The expected outcome is the establishment of a repeatable, high-velocity service line where AI Proofs of Concept (POCs) and pilot solutions can be spun up on demand. This will directly alleviate the bottleneck created by scarce technical talent, freeing senior AI engineers to focus on frontier research, complex edge cases, and architectural innovation rather than routine POC construction. Consequently, the utilization rate of high-value technical resources will shift from being tied to direct client project execution to a mix of platform enhancement, complex problem-solving, and strategic R&D, thereby optimizing the firm's human capital investment and increasing overall team capacity.

2. Enhanced Client Service and Value Proposition

From a client-facing perspective, Luma will enable BCG to deliver tangible value with unprecedented speed. The expected result is a significant enhancement of the client experience and partnership model. Consultants will be empowered to co-create solutions interactively with clients during workshops or strategy sessions, transitioning almost immediately from ideation to demonstration. This "strategy-to-execution" acceleration strengthens BCG's role as an implementer and not just an advisor, deepening client relationships and increasing stakeholder satisfaction. Furthermore, the ability to rapidly deploy tailored agents allows BCG to address a wider range of client pain points, from departmental knowledge management to customer-facing automation, thereby expanding the firm's serviceable addressable market within existing accounts and making its value proposition more concrete and compelling.

3. Strategic Intellectual Property and Competitive Differentiation

A critical expected result is the creation of a proprietary, defensible technological asset. In a competitive consulting landscape where firms increasingly leverage

similar third-party AI tools, owning a sophisticated, internally-developed platform like Luma provides a distinct competitive moat. It transforms BCG's AI capability from a service (dependent on partner technologies) into a product (a core, owned asset). This platform can become a key differentiator in proposals and pitches, signaling technical depth and innovative capacity. Over time, the data and insights generated from platform usage can feed back into improving BCG's own methodologies and industry insights, creating a virtuous cycle of improvement and specialization.

4. New Commercial and Monetization Pathways

Luma opens avenues for new revenue models and commercial strategies. The expected result includes the potential to:

- Productize and license the platform directly to enterprise clients for internal use, creating a recurring software-as-a-service (SaaS) revenue stream alongside traditional consulting fees.
- Bundle platform access with strategic engagements, offering it as a premium enabler within broader transformation programs.
- Reduce the cost-to-serve for AI-related projects, thereby improving profitability margins on fixed-fee engagements or allowing for more competitive pricing in the market.

5. Cultural and Capability Advancement

Internally, the successful implementation of Luma is expected to foster a culture of product-centric innovation within a traditionally project-centric firm. It will upskill the consulting workforce, equipping a broader population of professionals with "hands-on" AI implementation literacy. This democratization of technology builds internal resilience and future-proofs the organization's talent pool, aligning with the long-term strategic need for more techno-business hybrids.

In conclusion, the expected results for BCG extend far beyond the efficiency gains of a single tool. Luma - Agent Studio is anticipated to catalyze a shift towards a more scalable, product-driven, and technologically advanced service model. It aims to solidify BCG's leadership in applied AI, create new sources of value and competitive advantage, and fundamentally enhance how the firm delivers impact to its clients in the digital age. The platform is positioned not as a final deliverable, but as a

foundational capability that will generate compounding returns in efficiency, client impact, and market leadership for years to come..

1.5. Structure of the thesis/dissertation:

- Brief description of the content of each subsequent chapter.

2 Solution Development

2.1 Applied Rationale

2.1.1 Business Area Rationale:

Relevant Sector Concepts and Market Benchmarking

Several core concepts from BCG's operational and strategic paradigm directly shaped the problem definition and solution approach:

Time to Value (TTV): This is a paramount metric in consulting engagements. It measures the elapsed time between the initiation of a project and the moment the client realizes tangible, measurable benefits. The traditional, bespoke development model for AI POCs inherently results in a prolonged TTV, as value is only delivered after months of technical work. Luma directly attacks this by compressing the TTV for AI solutions from quarters to weeks or even days, aligning with the sector's intensifying focus on rapid, demonstrable impact.

Asset-Based Consulting and Productized Services: The consulting industry is undergoing a significant shift from purely labor-based, custom advisory models toward asset-based or productized service models. This involves creating reusable intellectual property (IP)—frameworks, software platforms, diagnostic tools—that can be deployed across multiple client engagements to increase efficiency, consistency, and scalability. Luma – Agent Studio is a quintessential example of this trend. It transforms the service of "building an AI agent" from a custom project (pure labor)

into a configurable product (an asset), allowing BCG to leverage its investment repeatedly.

Co-Creation and Agile Delivery: Modern consulting emphasizes collaborative, iterative work with clients rather than a detached "analyze-and-report" model. This co-creation ethos requires tools that enable real-time collaboration and rapid prototyping. Similarly, agile delivery within consulting projects involves breaking down initiatives into short sprints with frequent client check-ins. Luma's self-service nature empowers consultants to literally co-create agents with clients in workshops, and its rapid deployment capability fits perfectly within an agile delivery framework, providing working software at the end of each sprint.

Solution Industrialization: This refers to the process of standardizing and automating the delivery of complex solutions to achieve scale and reduce cost. The pain point of manual, non-scalable AI development is, in essence, a failure to industrialize the solution. Luma addresses this by applying principles of industrialization - standardized components (the platform), automated workflows (the RAG pipeline), and a repeatable process (the configuration interface) - to a previously artisanal service line**.**

Market best practices (benchmarking) for the process in question.

An analysis of market best practices for delivering enterprise AI and RAG solutions further validates the architectural and methodological choices behind Luma.

Modular and API-Driven Architecture: Leading technology service providers and software vendors have moved decisively towards modular, microservices-based architectures exposed via well-documented APIs. This practice, observed in platforms from major cloud providers (AWS, Azure, GCP) and AI startups, ensures scalability, independent deployability, and ease of integration. Luma's design—with its separate frontend/backend services, clear API boundaries (FastAPI), and dedicated data services—directly adheres to this best practice, ensuring it is maintainable and can evolve with technology.

Cloud-Native Deployment and Infrastructure as Code (IaC): The market standard for deploying business applications, especially those involving variable compute loads like AI inference, is cloud-native deployment. Best practice dictates leveraging

managed cloud services for scalability, resilience, and security, precisely as done with AWS S3, EC2, and RDS. Furthermore, the use of Terraform for IaC is a recognized best practice for ensuring consistent, auditable, and reproducible infrastructure, moving away from error-prone manual setups. This aligns Luma with enterprise DevOps standards.

Security-First Design with Multi-Tenancy: In a B2B context, particularly when handling sensitive client data, security cannot be an afterthought. Market best practice, as seen in leading SaaS platforms, mandates a security-first design. This includes robust authentication/authorization (JWT), encryption of data at rest and in transit, and, most critically, true multi-tenancy at the application logic level to ensure strict data isolation. Luma's rigorous implementation of company-scoped data access is a direct adoption of this non-negotiable best practice for professional services platforms.

Focus on Developer and End-User Experience (DX/UX): Successful platforms invest heavily in both Developer Experience (DX) and User Experience (UX). For DX, this means comprehensive documentation, interactive API explorers (like Swagger UI from FastAPI), and clean codebases—all priorities in Luma's development. For UX, especially for a self-service tool targeting non-technical users, it means an intuitive, guided, and forgiving interface. The wireframes and implementation of Luma, with their clear workflows, form validation, and contextual help, reflect this user-centric design best practice, which is essential for driving adoption.

Vendor Agnosticism and Open Standards: To avoid lock-in and ensure long-term flexibility, a best practice in enterprise software is to build on open standards and agnostic frameworks. While Luma uses specific technologies (LlamaIndex, Chroma DB), its architecture is designed to be LLM-agnostic and its data pipelines are based on standard concepts (embeddings, vector search). This strategic choice prevents the platform from becoming dependent on a single LLM provider's pricing, performance, or policies, aligning with the prudent, forward-looking investment strategy expected of a firm like BCG.

2.1.2 **Technological rationale for the solution:**

The proposed Luma – Agent Studio solution is architected upon a synthesis of established and emerging concepts, technologies, and methodologies from software engineering, artificial intelligence, and cloud computing. This section reviews these foundational pillars and provides a technical justification for their selection in the specific context of Boston Consulting Group's (BCG) operational environment and strategic objectives.

Retrieval-Augmented Generation (RAG)

Concept: RAG is a hybrid AI architecture that enhances a generative Large Language Model's (LLM) output by dynamically retrieving relevant information from an external knowledge base (like a vector database) and injecting it into the model's context window before response generation.

Relevance to Solution: This is the core conceptual engine of Luma. It directly addresses the business need for AI agents that are knowledgeable about specific, proprietary client content (PDFs, reports, data) without requiring costly and slow model fine-tuning. RAG enables the platform's promise of "context-awareness," allowing a single, general-purpose LLM to provide accurate, cited answers based on any uploaded document corpus.

Vector Databases and Semantic Search

Technology/Concept: A vector database (e.g., Chroma DB, Pinecone, Weaviate) is optimized to store and query high-dimensional vector embeddings—numerical representations of text, images, or audio that capture semantic meaning. Semantic search uses these embeddings to find data points that are conceptually similar to a query, not just lexically matching.

Relevance to Solution: This technology stack is the enabler of the RAG pipeline. When a user uploads a document, LlamaIndex converts its chunks into embeddings stored in Chroma DB. A user's chat query is similarly embedded, and the database performs a nearest-neighbor search to find the most semantically relevant document passages. This provides the precise, high-speed retrieval necessary for a responsive chat interface, a fundamental improvement over traditional keyword-based search within documents.

Microservices & API-First Architecture

Methodology/Concept: This architectural style structures an application as a collection of loosely coupled, independently deployable services that communicate via well-defined APIs (Application Programming Interfaces). It contrasts with a monolithic architecture where all components are intertwined.

Relevance to Solution: Luma adopts this paradigm by separating the frontend applications (React), the backend business logic (FastAPI), and data services (PostgreSQL, Chroma DB). This separation is crucial for scalability (each part can scale independently), maintainability (teams can work on discrete services), and future-proofing (new features or technologies can be integrated as separate microservices). The API-first design, evidenced by FastAPI's auto-generated OpenAPI spec, ensures the platform can be extended or integrated with other BCG or client systems.

Infrastructure as Code (IaC)

Methodology/Concept: IaC is the practice of managing and provisioning computing infrastructure through machine-readable definition files (code), rather than physical hardware configuration or interactive configuration tools.

Relevance to Solution: Using Terraform to define the AWS infrastructure (VPCs, EC2 instances, RDS databases, S3 buckets) brings repeatability, auditability, and disaster recovery to the platform's operational foundation. It eliminates configuration drift between environments and allows the entire production stack to be version-controlled and recreated reliably—a critical requirement for a business-critical asset.

JWT-Based Authentication and Multi-Tenancy

Concept/Technology: JSON Web Tokens (JWT) are an open standard for securely transmitting claims between parties as a JSON object. Multi-tenancy is an architectural principle where a single instance of software serves multiple customer organizations (tenants), with strict logical data isolation between them.

Relevance to Solution: This combination is non-negotiable for a platform serving multiple corporate clients. JWTs provide a stateless, scalable authentication mechanism suitable for a distributed, API-driven application. The programmatic enforcement of multi-tenancy (e.g., all database queries filtered by company_id)

ensures absolute data isolation and confidentiality, which is the bedrock of trust in a professional services context and a prerequisite for compliance with regulations like GDPR and LGPD.

Technical Justification for Key Technology Choice

Why FastAPI over Flask or Django?

For the backend, FastAPI was selected over alternatives like Flask or Django Rest Framework (DRF). The justification hinges on performance, developer experience, and modern standards alignment.

Performance: FastAPI is built on Starlette for the web parts and Pydantic for the data parts, offering asynchronous request handling out-of-the-box. This is superior for the I/O-heavy operations in Luma, such as document processing, LLM API calls, and database queries, as it can handle many concurrent connections efficiently. A synchronous framework like Flask or Django (without async layers) would be a bottleneck.

Developer Experience & Safety: FastAPI's use of Python type hints and Pydantic models provides automatic data validation, serialization, and documentation. This reduces bugs, auto-generates interactive API docs (Swagger UI), and speeds up development - a critical advantage for a single-developer project on a tight timeline.

Suitability for BCG's Environment: BCG's infrastructure teams are familiar with modern, high-performance APIs. Delivering a backend with automatic OpenAPI documentation aligns with professional standards, eases future integration work, and presents a technically sophisticated artifact that reflects well on the firm's engineering capabilities.

Why LlamaIndex over Custom-Built RAG or LangChain's Offerings?

While LangChain offers RAG utilities, LlamaIndex was chosen as the dedicated framework for data ingestion and retrieval.

Specialization and Abstraction: LlamaIndex is hyper-specialized for connecting custom data sources to LLMs. Its abstractions for document loaders, text splitters, index creation, and query engines are more refined and purpose-built for the RAG use case than the more generalized tools in LangChain. This specialization meant less "glue code" was required, accelerating development.

Ecosystem and Performance: LlamaIndex offers deep, optimized integrations with vector databases like Chroma DB and a focus on advanced retrieval techniques. For a platform whose primary value proposition is accurate retrieval from client documents, choosing the most focused and performant tool for that specific task was justified. It provided a more direct and controllable path to a robust RAG system.

Alignment with Project Scope: The initial MVP scope required powerful RAG but not the complex, multi-tool agent orchestration that is LangChain's primary strength. A custom, lightweight orchestration logic coupled with the specialized LlamaIndex proved to be a more streamlined and maintainable stack for the v1 product, avoiding the overhead of a larger, more complex framework.

Why AWS with Terraform over Other Clouds or Manual Deployment?

The choice of AWS managed via Terraform is justified by enterprise readiness, ecosystem maturity, and operational discipline.

Partner Company Environment: BCG, as a global enterprise, likely has established relationships, security compliance frameworks, and in-house expertise related to major cloud providers like AWS. Proposing a solution on AWS reduces friction for adoption, support, and potential handover to an internal ops team. It fits naturally into the company's existing technology landscape.

Managed Services for Scalability and Reliability: Using Amazon RDS for PostgreSQL provides automated backups, patching, and easy scaling—operations that would be burdensome on a self-managed database. S3 offers durable, scalable object storage for uploaded files. Leveraging these managed services offloads undifferentiated heavy lifting, allowing the project to focus on application logic rather than infrastructure management, which is appropriate for a strategic software asset.

Terraform for Governance and Reproducibility: In a corporate environment, infrastructure must be auditable, compliant, and reproducible. Manual deployment or configuration scripts are unsustainable and error-prone. Terraform provides a declarative, version-controlled state file that acts as a single source of truth for the infrastructure. This aligns with enterprise IT governance models, simplifies compliance reporting, and makes the platform's foundation as much a professional deliverable as its code.

In synthesis, the selection of FastAPI, LlamaIndex, and AWS/Terraform was not arbitrary. Each choice was evaluated against criteria of technical superiority for the task, alignment with enterprise-grade operational standards, and fit within the partner's likely technology ecosystem. Together, they form a coherent, justified stack designed to deliver a secure, scalable, and maintainable platform that meets both the immediate functional requirements and the long-term strategic needs of Boston Consulting Group.

2.1.3 **Fundamentals of Management and Development Methods:**

The successful execution of the Luma - Agent Studio project required more than just sound technical choices; it demanded a rigorous and disciplined approach to project management and software development. This approach was carefully aligned with methodologies prevalent in the corporate environment of Boston Consulting Group (BCG) and the broader technology industry. This section reviews the core methodologies employed - Agile, DevOps, and PMBOK principles - and justifies their application within the context of this corporate-academic partnership.

Agile Development Framework (Scrum-based)

Concept: Agile is an iterative and incremental approach to software development that emphasizes flexibility, customer collaboration, and the delivery of working software in short, time-boxed cycles called sprints. While not implementing a full Scrum team (due to the single-developer constraint), the project adopted a Scrum-inspired Agile framework.

Application in the Project: The entire 40-week project timeline was structured into four macro Modules, each functioning as a release milestone. Each module was further decomposed into five two-week Sprints. This structure created a predictable rhythm of planning, execution, review, and adaptation.

Sprint Planning: At the start of each sprint, deliverables were defined based on the product backlog and partner priorities.

Daily Execution: The developer dedicated four hours daily to focused development, simulating a daily stand-up's focus on progress and impediments.

Sprint Review & Retrospective: Each sprint concluded with a demo and alignment meeting with the partner (BCG) to gather feedback, and an internal retrospective to refine the process for the next cycle.

Corporate Environment Alignment: The consulting industry, including BCG, has widely embraced Agile principles for client engagements and internal projects under terms like "Agile delivery" and "sprint-based work." This methodology's focus on adaptability and demonstrable progress is highly compatible with BCG's culture of solving complex, evolving problems. Using an Agile framework ensured the project remained responsive to BCG's evolving strategic needs and provided the partner with continuous visibility and influence over the product's direction, mirroring a true client-consultant collaborative dynamic.

DevOps Culture and Continuous Integration/Continuous Delivery (CI/CD)

Concept: DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to shorten the systems development life cycle and provide continuous delivery of high-quality software. CI/CD is the automated pipeline that enables this.

Application in the Project: While a full DevOps team structure was not present, the project operationalized a DevOps culture and implemented a CI/CD pipeline.

Continuous Integration (CI): Implemented via GitHub Actions. On every push to a feature branch and upon pull request creation, an automated workflow executed.

This workflow ran the frontend test suites (Cypress end-to-end tests) to validate new code did not break existing functionality, enforcing a quality gate before merging.

Continuous Deployment (CD): Achieved through the combination of Infrastructure as Code (Terraform) and automated deployment scripts (deploy.sh). While not fully automated from commit to production, the process established a reliable, scripted, one-command procedure to provision cloud infrastructure and deploy application artifacts. This embodies the CD principle of making deployments predictable, low-risk, and repeatable.

Monitoring & Observability (Foundation): The cloud architecture (AWS) provides built-in monitoring tools (CloudWatch), and the application logs structured events, laying the groundwork for operational observability.

Corporate Environment Alignment: For a firm like BCG, which advises clients on digital transformation, demonstrating mastery of modern software delivery practices is itself a value signal. Implementing a professional CI/CD pipeline shows that the delivered platform is not a prototype but a production-ready asset built with industry best practices. It ensures the solution is maintainable, scalable, and can be seamlessly handed over to an internal or client operations team—a critical consideration for any strategic software asset.

Project Management Framework (PMBOK-Guided)

Concept: The Project Management Body of Knowledge (PMBOK) is a set of standard terminology and guidelines for project management. It outlines processes across ten knowledge areas (Integration, Scope, Time, Cost, Quality, etc.).

Application in the Project: The project did not follow PMBOK prescriptively but was guided by its core principles and knowledge areas, formalizing management aspects often overlooked in pure development-focused projects.

Scope Management: The Project Scope – Product Backlog section (in the initial plan) and the dynamic backlog prioritization process during sprint reviews provided formal scope definition and control mechanisms, preventing uncontrolled scope creep.

Time & Cost Management: The fixed 40-week timeline, divided into modules and sprints, along with the defined daily work allocation (4 hours), established clear time and resource (cost) baselines. The Risk Management table proactively identified threats like scope creep and resource limitations with mitigation plans.

Stakeholder & Communication Management: A formal Communication Management plan was outlined, specifying weekly meetings with the academic supervisor, bi-weekly reviews with the partner, and documented reports. This ensured systematic stakeholder engagement and alignment, a PMBOK cornerstone.

Quality Management: Quality was managed through multiple layers: the definition of Functional Requirements, the CI pipeline's automated testing, and the planned Usability Test Plan. This systematic approach to quality aligns with PMBOK's focus on planning and assuring quality.

Corporate Environment Alignment: BCG's core business is managing complex client projects. Its internal governance, reporting, and delivery expectations are inherently aligned with structured project management disciplines like those PMBOK describes. By consciously integrating these principles—formal risk logs, communication plans, scope documentation—the project adopted the professional language and discipline of a corporate deliverable. This made the project more understandable, predictable, and credible to BCG stakeholders, effectively "speaking their language" of structured project management.

Synthesis and Justification

The interplay of these methodologies was deliberate and synergistic. The Agile framework provided the tactical engine for iterative development and flexibility. DevOps practices provided the technical engine for quality, automation, and operational readiness. The PMBOK-guided management provided the governance engine for structure, risk mitigation, and stakeholder alignment.

This hybrid approach was the most suitable for the partner's environment for two key reasons:

Bridging Academic and Corporate Realms: The project existed at the intersection of an academic timeline and a corporate strategic initiative. Agile provided the flexibility needed for research and discovery, while PMBOK principles provided the formal structure and documentation expected in a corporate context for accountability and review.

Delivering a Strategic Asset, Not Just a Prototype: The goal was a production-viable platform. A purely academic "build-and-demo" approach would be insufficient. Incorporating DevOps (CI/CD, IaC) and formal project management (risk, quality, communication plans) elevated the work from a proof-of-concept to the standard of a manageable, transferable, and supportable corporate software asset. This demonstrates an understanding that for BCG, the long-term maintainability and strategic integration of the tool are as important as its immediate functionality.

2.2 **Specification and Development :**

2.2.1 **Requirements and Specifications:**

The functional architecture of the Luma – Agent Studio platform is defined by a comprehensive set of requirements and specifications. These were derived from the initial problem analysis, ongoing stakeholder feedback, and the strategic objectives of the partner company. This section details the system's functional and non-functional requirements and outlines key user specifications through representative use cases.

Functional Requirements (FR)

Functional Requirements define what the system must do—the specific behaviors and functionalities it must exhibit to satisfy user needs and business objectives.

FR01 - Agent Creation and Management: The system must provide an authenticated web interface that allows users (Consultants/Managers) to create, configure, name, describe, save, and list personalized AI agents.

FR02 - Multi-Format Document Ingestion: The system must allow users to upload documents in PDF, DOCX, TXT, MP3, and MP4 formats through the management interface for processing by the AI agents.

FR03 - RAG-Based Information Retrieval: Upon receiving a user query in the chat interface, the system must process uploaded documents using a Retrieval-Augmented Generation (RAG) pipeline to retrieve the most semantically relevant content chunks as context for response generation.

FR04 - Conversational Chat Interface: The system must provide a real-time chat interface where end-users can interact with deployed AI agents, view conversation history organized by session, and initiate new chats.

FR05 - Multi-Tenant Scalability: The system must support the concurrent operation of multiple independent AI agents and serve multiple users from different client companies simultaneously without performance degradation or data crossover.

FR06 - LLM Agnosticism & Configuration: The system must allow the configuration of agents to use different underlying Large Language Models (LLMs), starting with OpenAI's API, with an architecture that permits future integration of alternative models (e.g., Anthropic Claude, open-source Llama).

FR07 - Authentication and Authorization: The system must implement a secure login system using JWT tokens and enforce role-based access control (RBAC). It must distinguish between Admin/Manager users (access to agent creation) and Standard Users (access only to chat).

FR08 - Structured Data Output: The system must enable the creation of agents capable of generating structured outputs beyond text, specifically images and charts/graphs, based on processed data or instructions.

FR09 - Agent Workflow Definition: The system must allow the configuration of multi-step agent workflows within the creation interface, where users can define sequential instructions or logic for the agent to follow.

FR10 - Granular Document Access Control: The system must allow users setting document visibility to "Private" to restrict access to specific user roles (e.g., "Financial Department") or individual users, ensuring data is accessible only to authorized personnel.

FR11 - Data Persistence and History: The system must persistently store all chat session histories, uploaded file metadata, agent configurations, and user data in a reliable database, allowing retrieval and continuity across sessions.

FR12 - System Testing and Validation: The system must undergo integration testing (via automated CI pipeline) and formal usability testing with defined user profiles to validate functionality and user experience before final deployment.

FR13 - Cloud Deployment: The system must be deployable to a cloud environment (AWS) using Infrastructure as Code, ensuring high availability, scalability, and separation from the development environment.

FR14 - Audit and Isolation: The system must enforce strict data isolation where all data queries (agents, documents, chats) are automatically scoped to the user's company ID, preventing cross-company data access.

FR15 - Feedback Integration: The system must include a mechanism (e.g., a feedback table) to collect user feedback on agent responses, supporting continuous improvement of the platform and specific agents.

Non-Functional Requirements (NFR)

Non-Functional Requirements define how the system performs its functions, encompassing qualities such as performance, security, and usability.

NFR01 - Performance (Latency): The system must deliver a response from the AI agent in the chat interface within 5 seconds for 95% of queries under standard load, ensuring a conversational user experience.

NFR02 - Scalability: The system architecture must be scalable to support a load increase of 300% in concurrent users and document processing volume without requiring architectural redesign, leveraging cloud elasticity.

NFR03 - Availability: The deployed production platform must target an availability of 99.5% (uptime), excluding scheduled maintenance windows, ensuring reliable access for business operations.

NFR04 - Security (Data Confidentiality): The system must encrypt sensitive data (user credentials, document contents) both in transit (using TLS/HTTPS) and at rest (using AES-256 encryption in databases and S3). It must implement robust multi-tenant isolation as per FR14.

NFR05 - Security (Authentication): JWT tokens must have a short expiration time (30 minutes) and be securely validated on every request to protected endpoints. Password storage must use strong, salted hashing (bcrypt).

NFR06 - Usability (Learnability): A user with the role of "Manager" and basic computer literacy must be able to create and configure a new AI agent without external help within 15 minutes of first using the platform, as measured in usability tests.

NFR07 - Usability (Efficiency): An end-user must be able to ask a question and receive an answer from a pre-configured agent within 3 clicks or fewer from the chat homepage.

NFR08 - Maintainability: The codebase must be modular, documented, and accompanied by a CI/CD pipeline, allowing a new mid-level developer to understand the code structure and deploy a bug fix within 1 business day of onboarding.

NFR09 - Portability (Deployment): The system must be deployable to any standard AWS region using the provided Terraform configuration and deployment scripts, without modification to the application code.

NFR10 - Auditability: All user authentication events, document uploads, and agent creation actions must be logged with timestamps and user identifiers to support security audits and compliance reviews.

User Specifications and Use Cases

User specifications are detailed through Use Cases, which describe interactions between actors (users) and the system to achieve a specific goal.

Christopher: The Consultant as Platform Manager

The primary managerial actor, epitomized by the persona of Christopher, interacts with Luma's backend configuration interface. His overarching goal is to transform a client's business need into a functional AI agent with minimal time and technical friction. A quintessential use case involves the rapid assembly of a Proof of Concept for a sales analysis workshop. Upon authenticating into the manager portal, Christopher would navigate to the agent creation module. Here, he defines the agent's identity - for instance, naming it "Q4 Sales Insights Bot" - and articulates its expertise through a natural language system prompt, such as instructing it to perform as an expert sales analyst. Critically, the platform's design allows him to define multi-step workflows declaratively; he could, for example, specify that the agent's final step is to generate a bar chart visualizing sales by region. Upon saving, the system validates the agent's uniqueness and persists this configuration, instantly making the new bot available for deployment. This seamless process, from conception to availability, directly embodies the platform's core value proposition of democratizing and accelerating agent creation.

A further critical specification for the managerial role involves governing data security and access. Consider a scenario where Christopher needs to equip an agent for a client's financial department with sensitive budget reports. After uploading a confidential PDF, the platform's granular access control system allows him to set the document's visibility to "Private." He can then explicitly authorize access only to users associated with the "Financial Department" role. During subsequent queries, the system's Retrieval-Augmented Generation (RAG) pipeline will filter all document retrievals through this permission layer, ensuring strict adherence to the principle of least privilege and maintaining the confidentiality expected in a corporate environment. This capability transforms the consultant from a mere configurator into a data governance steward.

Michael: The End-User in Pursuit of Knowledge

The end-user experience, represented by Michael, a client employee, is defined by simplicity, speed, and contextual accuracy. His interaction is channeled exclusively through the conversational chat interface. A typical use case begins with Michael selecting a pre-configured agent, such as an "Internal Policy Assistant," from his

personalized list. He poses a natural language query - for example, inquiring about the procedure for submitting a remote work request. Behind the scenes, this triggers a sophisticated sequence: the system encodes his query, performs a low-latency semantic search against the vectorized knowledge base of approved HR documents, retrieves the most pertinent text passages, synthesizes them within a prompt for the Large Language Model, and generates a coherent, sourced response. This entire operation must complete within a few seconds to meet usability standards, delivering Michael a precise answer without requiring him to navigate document repositories or understand the underlying technology. Furthermore, the system meticulously maintains the context of this interaction, preserving the full dialogue history for future reference and continuity.

The platform's security model is rigorously tested through negative use cases. Imagine Michael, who works in marketing, attempts to query an agent about the quarterly marketing budget—a piece of information contained in a document restricted to the finance department. The system's retrieval process, bound by the access controls set by the manager, will find no authorized context for his query. Consequently, instead of exposing restricted data or providing a misleading hallucination, the agent is designed to respond with a graceful and non-revealing fallback message, such as stating it lacks access to the necessary information. This interaction not only protects sensitive data but also reinforces user trust in the system's reliability and security boundaries.

The System Administrator: Ensuring Operational Integrity

Beyond end-user functionality, the platform must be operationally robust, a responsibility falling to the System Administrator. This role engages with the platform through infrastructure and deployment workflows. A fundamental use case is the execution of a production deployment following the successful integration of new features into the main code branch. Utilizing the Infrastructure as Code foundation, the administrator executes a consolidated deployment script. This script first reconciles the cloud environment - provisioning and updating necessary resources like EC2 instances and S3 buckets via Terraform - and then proceeds to build the latest frontend application bundles, synchronizing them to the web host. The backend

services are subsequently updated in a coordinated manner. The successful execution of this workflow results in the new version of Luma being live in the production environment, ideally with zero perceptible downtime for active users. This automated, repeatable process encapsulates DevOps best practices, ensuring the platform remains scalable, maintainable, and resilient, thereby supporting its status as a dependable corporate asset rather than a fragile prototype.

In summary, these discursive use cases demonstrate how the Luma - Agent Studio's specifications converge to serve a holistic user experience. They illustrate a platform where strategic configuration, intuitive consumption, and rigorous operational management are interwoven, fulfilling the complex demands of a professional services environment and directly addressing the core business pain points of speed, cost, and security that initially motivated its development.

2.2.2 **Architecture and Technology:**

The Luma - Agent Studio is architected as a cloud-native, modular web application following a refined client-server model with clear service separation, laying the groundwork for a future microservices evolution. This architecture was consciously selected to balance immediate development efficiency against long-term scalability and maintainability demands within a corporate IT context.

The system is logically stratified into three primary layers, each with distinct responsibilities:

Presentation Layer (Client): This layer comprises two dedicated, single-page applications (SPAs) built with React.

- **Manager Frontend**: Served from an Amazon S3 bucket configured for static website hosting, this application provides the interface for BCG consultants and client administrators. It handles all agent creation, configuration, document upload, and system management tasks.
- **Chat Frontend**: Similarly hosted on S3, this application delivers the conversational interface for end-users. It is responsible for rendering chat

sessions, managing message history, and providing a real-time interaction experience.

Both frontends communicate exclusively with the backend layer via a RESTful API over HTTPS, ensuring a clean separation of concerns. They are stateless, with client-side state (like authentication tokens) managed in the browser.

Application Layer (Server): This is the core business logic hub, implemented as a monolithic but well-structured FastAPI application deployed on Amazon EC2 instances behind an Application Load Balancer (ALB). While monolithic in deployment, its internal organization is highly modular, mirroring microservice principles:

- **API Gateway & Business Logic**: The FastAPI app acts as a unified gateway, routing requests, enforcing authentication/authorization via JWT middleware, and executing business workflows (e.g., agent creation, chat orchestration).
- **AI Service Modules**: Within the same process, dedicated modules encapsulate specific AI functionalities. The LlamaIndex service manages the entire RAG pipeline—document loading, chunking, embedding, and vector store interaction. A separate, custom agent orchestration engine interprets stored agent configurations, manages prompt engineering, and handles the execution of defined workflows (e.g., "generate a chart").
- **External Service Integration**: This layer also manages all external communications, primarily with Large Language Model APIs (OpenAI) and other potential cognitive services.

The decision for a modular monolithic backend, rather than a full microservices deployment, was justified for the v1 product due to the single-developer constraint, reduced operational complexity, and the tightly coupled nature of the core AI workflows. However, the clear internal boundaries allow these modules to be extracted into independent microservices as future scale demands.

Data Layer: Following the polyglot persistence pattern, this layer utilizes specialized databases for different data types:

- **Metadata & Transactional Data**: Amazon RDS for PostgreSQL serves as the system of record for all structured, relational data. This includes user profiles, company information, agent configurations, session metadata, chat history, and access control lists. RDS provides managed, ACID-compliant storage essential for business data integrity.
- **Vector Data (Unstructured Knowledge)**: Chroma DB, running on a dedicated EC2 instance, functions as the high-performance vector database. It stores the embedding vectors and associated metadata for all ingested document chunks, enabling the low-latency semantic search that powers the RAG system.
- **Object Storage (Raw Files)**: Amazon S3 provides durable, scalable storage for the original files (PDFs, DOCX, etc.) uploaded by users. It acts as the source-of-truth archive and is accessed by the LlamaIndex pipeline during processing.
- **Integration and Communication Flow**: A standard user query exemplifies the architecture's integration. An end-user's message from the Chat Frontend is sent via HTTPS to the Load Balancer, which directs it to a FastAPI instance. The JWT middleware validates the request and extracts user/company context. The request is routed to the chat controller, which calls the custom agent orchestration module. This module retrieves the agent's configuration from PostgreSQL, then invokes the LlamaIndex service. LlamaIndex queries Chroma DB for relevant context, which is assembled into a final prompt. The orchestration module calls the external LLM API, receives the response, logs the interaction to PostgreSQL, and returns the answer through the API layer to the frontend for display.

Integration into the Company's IT Ecosystem

The architecture of Luma - Agent Studio is designed for seamless and secure integration into BCG's - or any enterprise's-existing IT ecosystem, adhering to

corporate standards for security, governance, and interoperability. Its integration occurs across multiple planes:

Identity and Access Management (IAM) Integration: The platform's authentication system is designed for compatibility with corporate IAM standards. While the MVP implements a standalone JWT-based auth system, its architecture anticipates integration with enterprise identity providers (e.g., Azure Active Directory, Okta) via industry-standard protocols like OAuth 2.0 or SAML 2.0. The internal role_id and company_id mapping would simply be populated from claims provided by the corporate IdP, allowing BCG to enforce its centralized password policies, multi-factor authentication, and user lifecycle management. This design ensures the platform does not become a siloed identity store.

Cloud Infrastructure and Governance: By being built entirely on Amazon Web Services (AWS) using Terraform, Luma aligns natively with the cloud infrastructure strategies of most modern enterprises. The Terraform code can be integrated into the company's existing Infrastructure as Code (IaC) pipelines and version control systems. Deployment can adhere to corporate change management processes, and the infrastructure itself can be placed under the governance of existing cloud management platforms for cost monitoring, compliance auditing (e.g., ensuring encryption standards), and security posture management (e.g., integration with AWS Security Hub).

Data Security and Compliance Integration: The architecture directly supports integration with corporate data security frameworks.

- The application-level multi-tenancy ensures data isolation aligns with contractual and regulatory requirements.
- All data in transit is encrypted via TLS, and encryption at rest in RDS and S3 can utilize corporate-managed AWS KMS (Key Management Service) keys, allowing key rotation and access policies to be controlled by central security teams.
- Comprehensive audit logs (generated by the application and from AWS CloudTrail for infrastructure actions) can be forwarded to a corporate SIEM (Security Information and Event Management) system like Splunk or Sumo Logic, enabling centralized security monitoring and threat detection.

API-First Design for Ecosystem Interoperability: The FastAPI backend, with its auto-generated OpenAPI specification, exposes a well-documented set of RESTful APIs. This API-first approach is pivotal for ecosystem integration. It enables other internal systems at BCG to interact with Luma programmatically. Potential integrations include:

- Pulling usage analytics into a central dashboard or BI tool (e.g., Tableau).
- Triggering agent creation or document ingestion from a client project management system.
- Feeding anonymized interaction data into a central AI model training pipeline.

The platform is positioned not as a closed application but as a service within a larger digital service mesh.

DevOps and SDLC Alignment: The implemented CI/CD pipeline using GitHub Actions can be integrated into the company's standard Software Development Life Cycle (SDLC). The repository can be mirrored to the company's internal Git management system (e.g., GitHub Enterprise, GitLab). The CI pipeline can incorporate corporate-mandated security scans (SAST, SCA) and quality gates. The deployment process, codified in scripts, fits into established release management procedures.

In essence, the architecture of Luma – Agent Studio is corporate-ready by design. It avoids proprietary lock-in at the infrastructure and identity layers, embraces standard enterprise cloud services, and provides the necessary hooks—through APIs, IaC, and standardized protocols - for deep integration into BCG's IT governance, security, and operational frameworks. This ensures the platform can evolve from a project deliverable into a sustainable, governable, and integrated component of the partner's technology portfolio.

2.2.3 **Development and Implementation (MVP):**

The development of the Luma - Agent Studio project was fundamentally guided by Agile methodology, executed with rigor and discipline to navigate the complexities of a solo-developed, business-critical software initiative. This choice was strategic,

directly aligning with the project's core needs for adaptability, continuous delivery, and stakeholder synergy within a dynamic corporate-academic partnership. The methodology was not a nominal label but the operational engine that governed all activities from planning through deployment.

Core Principles and Tailored Implementation

Agile, as manifested in this project, was defined by its foundational values: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan. These principles were operationalized through a tailored framework that accommodated the single-developer constraint while preserving the essence of iterative, value-driven progress.

The entire project arc was structured into a macro-cycle of four 10-week Modules, each representing a coherent phase of capability delivery (e.g., foundational architecture, core agent creation, advanced features, polish and deployment). The true heartbeat of the Agile process, however, was the two-week Sprint. Each module comprised five such sprints, establishing a relentless, predictable rhythm for the work. This time-boxing was instrumental in maintaining focus, creating a sense of urgency, and providing regular, natural breaks for assessment and realignment.

The roles within this streamlined Agile process were condensed yet distinct:

Product Owner/Stakeholder: This role was collectively fulfilled by the partner company (BCG), who provided the business vision and priorities, and the academic supervisor, who ensured technical and pedagogical alignment. They were the ultimate arbiters of the product backlog's content and order.

Development Team: This role was singular, comprising the student developer responsible for all analysis, design, coding, testing, and integration work.

Scrum Master/Developer: In this consolidated model, the developer also assumed the responsibilities of facilitating the Agile process—removing impediments (often logistical or informational), self-managing the sprint workflow, and ensuring adherence to the methodological framework.

The Agile Cycle: Ceremonies and Artifacts in Practice

The methodology was given concrete form through the execution of key Agile ceremonies and the maintenance of living artifacts:

Sprint Planning: At the outset of each two-week sprint, a planning session was conducted. This involved a review of the prioritized Product Backlog—a dynamic list of user stories, functional requirements, and technical tasks derived from the project's objectives. A subset of these items, deemed achievable within the sprint, was selected to form the Sprint Backlog. This selection was informed by the developer's capacity and the strategic importance of the features, ensuring each sprint had a clear, committed goal (e.g., "Implement the document upload API and frontend component").

Sprint Execution and Daily Discipline: During the sprint, development work proceeded based on the sprint backlog. While a formal daily stand-up meeting with a team was not possible, its principle was internalized through a disciplined daily personal sync. Each work session began with a brief reflection: What was accomplished yesterday? What is the focus for today? Are there any blockers? This practice maintained momentum, ensured continuous progress, and allowed for the early identification of issues requiring escalation to the supervisor or partner.

Sprint Review and Demonstration: The most critical ceremonial interface with the stakeholder was the Sprint Review, held faithfully at the end of every two-week cycle in meetings with the partner company. This was not a status report but a working software demonstration. The developer showcased all completed functionalities, from new API endpoints to UI interfaces. This practice provided the partner with tangible evidence of progress, facilitated immediate and contextual feedback, and validated that the development trajectory was meeting business expectations. It transformed the partner from a passive recipient into an active collaborator in the product's evolution.

Sprint Retrospective: Following the review, a brief but important Sprint Retrospective was conducted. This was a personal and analytical look back at the sprint process itself. Questions were posed: Which practices worked well? What challenges were

encountered? How can the workflow, tools, or communication be improved for the next sprint? Insights from this reflection led to concrete adjustments in the subsequent sprint, embodying the Agile commitment to continuous process improvement.

Artifacts and Flow of Work

The flow of work was managed through central Agile artifacts:

Product Backlog: The single, authoritative source for all desired work. It was continuously groomed and reprioritized based on feedback from sprint reviews, new insights, and strategic shifts.

Sprint Backlog: The set of items selected for the current sprint, representing a forecast of functionality to be developed.

Increment: The sum of all completed backlog items at the end of a sprint, culminating in a demonstrable, potentially shippable improvement to the Luma platform.

Justification: Why a Pure Agile Methodology Was Optimal

The selection and rigorous application of a pure Agile methodology for the Luma – Agent Studio project was a deliberate and strategic choice, fundamentally shaped by the nature of the technical challenge and the operational realities of the partnership with Boston Consulting Group (BCG). This approach transcended mere procedural adherence; it became the philosophical and practical lens through which uncertainty was managed, value was continuously delivered, and a complex vision was systematically rendered into functional software. The methodology's inherent flexibility and iterative cadence proved not merely suitable but essential, serving as the dynamic scaffold upon which the project's ambitions were responsibly constructed.

At the heart of this endeavor lay the profound complexity and inherent uncertainty associated with integrating frontier artificial intelligence technologies. The project's core technical pillars - Retrieval-Augmented Generation (RAG), vector databases, and Large Language Model APIs - represent a domain characterized by rapid

evolution and non-deterministic behavior. A traditional, linear development model, such as Waterfall, with its dependence on exhaustive upfront specification and fixed phases, would have been fundamentally mismatched to this reality. Such rigidity would have stifled necessary experimentation, delayed the discovery of technical constraints or opportunities, and risked culminating in a costly, late-stage integration failure. In contrast, the Agile framework, with its foundational embrace of iteration, provided the necessary intellectual and operational space for disciplined exploration. Each two-week sprint became a contained cycle of hypothesis, implementation, and validation. This structure allowed for technical pivots and learning to be incorporated organically, transforming uncertainty from a project risk into a managed variable within a controlled, empirical process.

Furthermore, the Agile methodology was instrumental in forging and sustaining a deep, collaborative alignment with the partner's strategic objectives. BCG's engagement was predicated on the delivery of tangible, business-ready value, not a theoretical artifact. The Agile principle of prioritizing working software over comprehensive documentation was operationalized through the sacrosanct rhythm of bi-weekly sprint reviews and demonstrations. These sessions were far from perfunctory status updates; they were dynamic forums where abstract requirements met concrete functionality. By consistently presenting a functional increment of the platform, the development process demystified progress for the partner, providing them with a continuous and tangible return on their engagement. More critically, this tight feedback loop ensured the product's evolution was constantly calibrated against actual business needs. Specifications were not frozen at the outset but remained living documents, refined through dialogue informed by seeing the software in action. This collaborative, outcome-oriented dynamic elegantly mirrored the very consulting ethos BCG employs with its own clients, fostering a partnership model based on co-creation and shared ownership of the result.

On a practical level, the Agile structure provided an indispensable framework for optimizing efficiency and maintaining discipline within a solo development context. The singularity of the development team posed unique challenges in terms of focus, prioritization, and self-accountability. The imposed discipline of the sprint cycle - with its clear planning, commitment, and review phases - acted as a guardrail against distraction and scope diffusion. The time-boxed nature of sprints created a healthy

sense of urgency and a series of manageable short-term goals, preventing the overwhelming scale of the total project from becoming paralyzing. Even in their adapted, solitary form, Agile ceremonies like the daily personal sync and the sprint retrospective provided a structured mechanism for self-organization and continuous process improvement. This internal governance was crucial for maintaining a sustainable pace and ensuring consistent, high-quality output over the extended project timeline.

Ultimately, the Agile approach institutionalized a capacity for adaptability as a core competitive feature of both the development process and the resulting platform. In a corporate and technological landscape defined by change, the ability to respond gracefully to new information is paramount. The Agile process, with its mechanisms for ongoing backlog grooming and reprioritization, formally embedded this responsiveness. Insights gleaned from market shifts, evolving partner strategy, or emerging technological capabilities could be seamlessly integrated into the project's trajectory at the next planning horizon without triggering a crisis or a fundamental replanning. This built-in flexibility ensured that Luma - Agent Studio would not be a relic of its initial conception but could evolve intelligently alongside the domain it was designed to serve. The methodology thus delivered more than a product; it delivered a resilient and adaptable capability.

In essence, the Agile methodology was far more than a project management technique applied to Luma - Agent Studio; it was the constitutive logic of its creation. It provided the rhythmic discipline to navigate technical novelty, the collaborative protocol to align with business value, the personal framework to sustain solo execution, and the philosophical commitment to embrace change. The successful realization of the platform's Minimum Viable Product stands as a testament to the profound suitability of this approach, demonstrating how principled agility can transform ambitious vision into robust, deliverable reality within the demanding intersection of academic rigor and corporate expectation.

2.2.4 **Testing and Technical Evaluation:**

Given the project's defined scope and practical constraints, the technical validation of Luma - Agent Studio was anchored in two primary, complementary pillars: Integration Testing for systemic integrity and designed Usability Testing for user-centric quality, albeit with a noted limitation in execution.

Integration Testing and Continuous Validation

A comprehensive suite of Integration Tests was developed and rigorously maintained. These tests were not isolated unit tests but focused on validating the critical interactions and data flows between the system's core components - the FastAPI backend, the LlamaIndex processing pipeline, the Chroma vector database, and the PostgreSQL metadata store. For instance, tests were created to ensure that a document upload request correctly triggered the embedding process, stored vectors in Chroma DB, and saved the associated metadata in PostgreSQL with the proper company and permission tags. Similarly, the complete RAG query flow, from chat request to contextualized LLM response, was validated end-to-end. Crucially, this entire test suite was integrated into a Continuous Integration (CI) pipeline via GitHub Actions, configured to execute automatically on every pull request and code push to the main branch. The consistent "passing" status of all integration tests throughout the development cycle served as the primary technical safeguard, confirming that new features and changes did not break existing core functionalities and that the system's architectural seams operated cohesively.

Usability Test Planning and Design Limitation

In parallel, a formal Usability Test Plan was meticulously designed to align with corporate standards for user acceptance. The plan defined clear objectives, participant profiles (junior developers and non-technical users), specific test scenarios, and quantitative/qualitative success metrics, including the System Usability Scale (SUS). The design aimed to empirically validate the platform's intuitiveness and efficiency against key non-functional requirements. However, a critical technical constraint prevented its full execution: the unavailability of a dedicated, test-environment API key for the core Large Language Model (LLM)

service during the planned testing window. Since the platform's fundamental value proposition - generating intelligent, context-aware responses—is dependent on this external service, executing realistic user tasks (e.g., "ask the agent a question about an uploaded document") was rendered impossible. Therefore, while the test plan stands as a robust blueprint for future validation, the actual user testing and collection of performance data (task success rate, time-on-task, SUS scores) could not be completed as intended.

Validation of Systemic Integrity and Technical Requirements

The successful execution of the entire integration test suite provides direct and automated evidence for compliance with a significant portion of the system's Functional and Non-Functional Requirements. The tests validate that the document upload pipeline works (FR02), that the RAG retrieval logic is integrated and functional (FR03), and that the backend API correctly orchestrates chat interactions (FR04). Furthermore, the tests verifying data persistence and correct company-scoped queries demonstrate adherence to data isolation and auditability standards (FR14, NFR10). The fact that these tests pass within the CI pipeline also implicitly supports requirements related to maintainability and deployment automation (FR13, NFR08, NFR09), proving the codebase can be built, integrated, and validated in a consistent, automated manner - a fundamental corporate technical standard.

Acknowledged Gap and Path Forward for User-Centric Validation

The inability to execute the usability tests represents a recognized gap in the current evaluation. While integration tests confirm the system's technical soundness, they cannot fully speak to its experiential quality for end-users. The designed test plan remains a critical project artifact, defining the precise methodology and success criteria for the essential next step: User Acceptance Testing (UAT). Full validation of requirements related to learnability, efficiency (NFR06, NFR07), and ultimate user satisfaction is therefore contingent upon securing the necessary resources (e.g., a test LLM API key) and executing this planned evaluation with representative users.

2.3 [Assessment of Impact and Contribution to the Business]

[In this section, the return on investment of time/resources for implementing the solution should be measured.]

2.3.1 Defining Corporate Success Metrics:

- **Project KPIs (Key Performance Indicators):** Clear definition of the metrics used to measure success (based on item 1.3);
- **Measurement Methodology:** How the data was collected to compare the "Before" and "After" of the solution.

2.3.2 Results and Impact Analysis:

- Presentation of the results of the solution implementation **versus the** *baseline* metrics (item 1.2);
- **Quantitative Analysis:** Demonstration of results (Ex : "The solution reduced operational costs by 15%, saving R$ X per month").
- **Qualitative Analysis:** Gains in agility, user/internal customer satisfaction, or improved decision-making.

2.3.3 Cost-Benefit Analysis:

- Estimated development/implementation costs (labor, licenses, infrastructure) ;
- Calculation of **Return on Investment (ROI)** or other financial indicators relevant to the company (e.g. , *Payback*).

2.3.4 Critical Success Factors and Lessons Learned:

- What factors contributed to or hindered the implementation and its impact on the company's results?

3 [Conclusion]

[In the conclusion, the author states whether or not the objectives set out in the project were achieved, discusses the impacts on the business, presents recommendations for the company on the evolution of the solution (new features, scalability, maintenance), and the knowledge transfer plan and technical documentation for the company's internal team]

References (mandatory item – NBR14724, item 4.2.3.1; NBR 6023)

(# References must comply with the rules of NBR6023 (Preparation of References;
Only the works consulted and cited in the text should be included in the reference list.
 # font size 12
 # prepared using single spacing;
 # aligned to the left margin of the text and without indentation;
 # separated from each other by a single blank line;
For online documents, the web address must be indicated, preceded by the expression "Available at:" and the access date preceded by the expression "Accessed on:";
The author should be indicated by their last name, in capital letters, followed by their first name and other last names, abbreviated or not; authors should be separated by semicolons.
 # When there are up to three authors, all must be listed;
When there are four or more authors, you can list all of them, or only the first one followed by the expression *et al* .

(# Examples:

Book:

[LAST NAME, First Name. **Book Title** : Subtitle. Edition. Publisher Location: Publisher, Publication Date.]

[PRESSMAN, RS; MAXIM, BR **Software Engineering** : A Practitioner's Approach. 9th ed. Porto Alegre: AMGH, 2021.]

[BASS, L.; CLEMENTS, P.; KAZMAN, R.; Software architecture in practice. 4th ed. Boston: Addison-Wesley, 2022.]

Article:

[LAST NAME, First Name. Article Title: Subtitle. **Journal Name** , Publication Location, Volume, Number, Pages, Publication Date. Available at: [electronic address]. Accessed on: [date of last access]]

[SANTOS, EH; BENITES, CS; STATERI, J. Study of artificial intelligence within the scope of PBL teaching. **Contemporary Journal** . [*S. l.*], v. 4, n. 9, p. 01-18, 2024. Available at: https://ojs.revistacontemporanea.com/ojs/index.php/home/article/view/5781/4271 . Accessed on: Oct. 27, 2025.

[VALENTE, JA; BITTENCOURT, II; SANTORO, FM; GARCIA, M.; ISOTANI, S.; GARCIA, A.; HABIMORAD, M. Project-Based Higher Education in Computing: Inteli on the path to innovation. **Brazilian Journal of Informatics in Education** , [*S. l.*], v. 33, p. 605–642, 2025. Available at: https://journals-sol.sbc.org.br/index.php/rbie/article/view/4320. Accessed on: Oct. 27, 2025.]

[VALENTE, JA *et al* . Project-Based Higher Education in Computing: Inteli on the path to innovation. **Brazilian Journal of Informatics in Education** , [*S. l.*], v. 33, p. 605–642, 2025. Available at: https://journals-sol.sbc.org.br/index.php/rbie/article/view/4320. Accessed on: Oct. 27, 2025.]

)

Appendices (Optional item – NBR14724, item 4.2.3.3) (For titles without numerical indicators, see NBR 14724, item 5.2.3)

[Supporting document: text or document prepared by the author, used to support their argument and aid in understanding the main text without compromising the content presented in the body of the work. Ex : tables, reports, questionnaires, programming code, etc.]

Annexes (optional item – NBR14724, item 4.2.3.4)

[Supporting document: text or document not created by the author, but which serves as a basis, proof, or illustration of the content of the work. Ex : tables, reports, laws, manuals, etc.]