SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Data Engineering And Analytics

An LLM-based Decision Support System for Strategic Decision-Making in Business Environments

Author	**Majd Alkayyal**
Examiner	**Prof. Dr. Georg Groh**
Supervisor	**Simon Malberg**
Submission Date	**06.04.2025**

I confirm that this master's thesis in data engineering and analytics is my own work and I have documented all sources and material used.

Munich, 06.04.2025 Majd Alkayyal

TUM School of Computation, Information and Technology
Technical University of Munich

An LLM-based Decision Support System for Strategic Decision-Making in Business Environments

Master Thesis Report

Majd Alkayyal ✉

TUM School of Computation, Information and Technology, Technical University of Munich

✉ majd.alkayyal@tum.de

April 6, 2025

Abstract — Strategic decision-making in business environments is inherently complex, often impeded by human cognitive biases and the limitations of existing artificial intelligence (AI) tools, such as their lack of transparency and structured reasoning. This master's thesis presents StrategicAI, an innovative Decision Support System (DSS) powered by Large Language Models (LLMs), designed to enhance strategic decision-making by addressing these challenges. The research pursues three key objectives: developing StrategicAI with a structured framework encompassing Root Cause Analysis, Solution Exploration, and Solution Validation; evaluating its effectiveness through empirical case studies; and exploring human-AI collaboration to optimize decision outcomes. StrategicAI integrates LLMs with a Human-in-the-Loop interface, enabling a systematic approach that automates critical processes while fostering transparency and collaboration. Evaluations across six diverse business scenarios demonstrate that StrategicAI significantly improves decision quality, achieving a Problem Identification Score (PIS) of 0.79 and a Solution Generation Score (SGS) of 0.66 after refinements, surpassing baseline tools like ChatGPT in key metrics. User testing further elevates performance, with human-AI collaboration yielding a PIS of 0.84 and an SGS of 0.80, highlighting the synergy between human expertise and AI capabilities. This work advances the academic understanding of LLM applications in decision-making and offers a practical tool for businesses, mitigating biases and enhancing data-driven strategies. By setting a new standard for transparent and effective decision support, StrategicAI bridges the gap between AI potential and real-world applicability, paving the way for more resilient organizational outcomes.

Keywords: *Strategic Decision-Making*, *Decision Support System (DSS)*, *Large Language Models (LLMs)*, *Human-AI Collaboration*, *Root Cause Analysis*, *Solution Exploration*, *Solution Validation*, *Transparency*, *Design Science Research (DSR)*, *StrategicAI*, *Problem Identification*, *Solution Generation*, *Usability*

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background and Context

Strategic decision-making is a cornerstone of organizational success, empowering leaders to navigate complex choices that define a company's future. These decisions—spanning market entry, technological adoption, resource allocation, and strategic partnerships—are marked by complexity, uncertainty, and profound impacts on performance. The stakes are high: poor decisions can lead to financial ruin, while well-timed choices can secure a competitive edge. For instance, Netflix's pivot to streaming in 2007 propelled its market value beyond $200 billion by 2020, showcasing the rewards of strategic foresight ("Netflix Market Cap 2010-2024 | NFLX", n.d.). In contrast, Kodak's failure to embrace digital photography—despite its invention of the technology—resulted in a valuation drop from $31 billion in 1997 to under $300 million by 2012, culminating in bankruptcy (Anthony, 2016). These cases underscore strategic decision-making as a critical driver of organizational survival and growth.

Empirical research consistently highlights the pivotal role of decision-making processes in achieving organizational objectives. Approaches such as intuitive, analytical, or collaborative methods significantly influence outcomes, particularly in dynamic, competitive environments (Sinnaiah et al., 2023). However, managers encounter substantial barriers that impair their ability to formulate effective strategic choices. A primary challenge is *information overload*, where decision-makers are inundated with data from internal reports, market trends, and regulatory updates. This overwhelming volume often hampers their capacity to distill actionable insights, leading to delayed or flawed decisions. Shahrzadi et al. (2024) demonstrate in their 2024 scoping review that information overload severely restricts managers' ability to filter critical data, thereby complicating the strategic process.

Another critical issue is the tendency to *overlook key factors during root cause analysis*, resulting in incomplete or erroneous problem diagnoses. While pinpointing the origins of an issue is foundational to effective strategy, cognitive limitations, and suboptimal methodologies often obscure vital insights. Organizational factors, including political influences and inadequate investigative rigor, exacerbate this problem. Although focused on healthcare, Peerally et al. (2017) reveal systemic issues—such as flawed analysis and political pressures—that parallel challenges in business settings, where similar dynamics can distort problem identification.

Even when issues are accurately identified, *organizational barriers frequently stifle creativity in solution generation*. Bold, innovative strategies demand imaginative thinking, yet constraints like time pressure, limited autonomy, and risk aversion restrict managers' creative scope. Consequently, organizations may default to conventional or suboptimal solutions. Meinel et al. (2012) argue that such barriers represent significant impediments, advocating for environments that foster rather than inhibit creative thought.

Further complicating matters, *cognitive biases*—including confirmation bias and overconfidence—skew the validation of potential solutions. These biases prompt managers to favor options aligning with preconceived notions or to overestimate their feasibility, often at the expense of superior alternatives. In their 2021 systematic review, Acciarini et al. (2021) illustrate how such biases, especially amid organizational change, compromise strategic assessments and diminish decision quality. The cumulative economic impact of these challenges is notable, with evidence suggesting that flawed strategic decisions can erode profitability by as much as 3%—a substantial loss for large enterprises (Rathindran, 2018).

In response to these decision-making challenges, recent developments in artificial intelligence technology offer promising solutions. Specifically, Large Language Models (LLMs) represent a significant advancement in decision support tools. These models can process extensive information, recognize patterns, and generate meaningful insights without requiring specialized programming for each task (Brown et al., 2020; Radford et al., 2019). As organizations face increasingly unpredictable and complex business environments, LLMs can serve as valuable tools for strategic decision-makers. Their capacity to analyze large datasets allows them to function as virtual advisors that can propose alternative problem definitions, identify diverse solution paths, and develop explanations for various stakeholders (Csaszar et al., 2024). From generating initial concepts to testing possible scenarios, LLMs can enhance strategic decision processes by complementing human judgment in ways that traditional computer systems cannot. However, several important challenges remain: ensuring these systems

provide clear explanations for their suggestions, maintaining reliability in their outputs, and reducing potential errors when applied to critical business decisions (Eigner & Händler, 2024).

These four impediments—information overload, overlooked factors in root cause analysis, barriers to creativity, and cognitive biases—constitute the primary obstacles to effective strategic decision-making. Artificial Intelligence (AI), particularly Large Language Models (LLMs), presents a compelling solution. With their ability to process vast datasets, identify patterns, and generate insights, LLMs hold significant potential (Hackathorn et al., 2024). Real-world applications underscore this value: Amazon leverages AI to drive 35% of its revenue through supply chain and recommendation systems ("How artificial intelligence will transform decision-making", 2023), while JP Morgan enhances risk assessment using LLMs ("The Impact of Artificial Intelligence on Financial Decision Making", 2024). Such models could assist managers by filtering data, enriching analysis, and expanding solution exploration (Mankins, 2023).

Nevertheless, existing LLM-based tools exhibit critical shortcomings. Their opaque nature—often functioning as "black boxes"—leaves managers uncertain about the rationale behind recommendations, undermining trust in high-stakes scenarios (Novelli et al., 2023). Additionally, these tools struggle to integrate structured frameworks, such as root cause analysis or scenario planning, which are essential for strategic tasks (Felzmann et al., 2020). Moreover, they seldom capitalize on human expertise, limiting their role to passive support rather than active collaboration. This research seeks to address these deficiencies by developing an LLM-based Decision Support System (DSS) that directly confronts both human and technological limitations.

1.2 Motivation and Research Problem

This study is motivated by the persistent challenges undermining strategic decision-making—information overload, cognitive biases, incomplete root cause analysis, and stifled creativity—alongside the unrealized potential of AI to mitigate them. Strategic decisions, characterized by complexity, ambiguity, and risk, carry profound consequences. The collapse of Kodak amid digital innovation exemplifies the perils of misjudgment (Anthony, 2016), whereas Netflix's streaming triumph illustrates the benefits of adaptability ("Netflix Market Cap 2010-2024 | NFLX", n.d.). Human limitations—overburdened managers, biased reasoning, overlooked insights, and constrained creativity—necessitate innovative solutions. LLMs offer promise by analyzing data, simulating scenarios, and alleviating cognitive burdens, yet current tools lack transparency, obscuring the basis of their outputs (Novelli et al., 2023).

Furthermore, these systems often fail to incorporate structured methodologies—like root cause analysis or scenario planning—that managers depend on for complex problem-solving (Felzmann et al., 2020). Human expertise, with its contextual depth and experience, remains underutilized, relegating AI to a supplementary rather than synergistic role. This research addresses these shortcomings, posing the question: How can LLMs be engineered to provide transparent, reliable decision support while collaborating with human insight to enhance strategic outcomes? The objective is to develop an advanced LLM-based DSS that bridges these gaps.

1.3 Research Questions

The following central research question guides this investigation:

- How can an LLM-based Decision Support System (DSS) be designed to function as an effective AI consultant, enhancing organizational strategic decision-making?

To address this, two sub-questions are proposed:

- How can the DSS be structured to deliver transparent and comprehensible explanations of its recommendations, thereby fostering trust and usability among decision-makers?
- How can the DSS be engineered to ensure reliability, minimize output errors, and maintain consistency across diverse strategic contexts?

These questions prioritize transparency and reliability, emphasizing the system's role as a collaborative partner.

1.4 Research Contributions

This study advances the application of LLMs in strategic decision-making through the following contributions:

1. **Design and Development of StrategicAI:** We introduce StrategicAI, a novel LLM-based Decision Support System that embeds structured frameworks for root cause analysis, solution exploration, and solution validation. Unlike opaque "black box" AI tools, StrategicAI delivers transparent reasoning and actionable recommendations that decision-makers can trace, comprehend, and trust.

2. **Reliability Enhancement Framework:** We devise mechanisms to bolster LLM reliability in strategic contexts, incorporating certainty quantification, validation processes, and consistency checks. This framework mitigates limitations in existing systems, ensuring robust decision support amid complex organizational challenges.

3. **Human-AI Synergistic Feature Set:** Through empirical testing in real-world settings, we establish features that enable seamless human-LLM collaboration in strategic decision-making. These include guided tree construction, chat agents, data retrieval processes, real-time feedback, and interactive validation processes. Comparative analysis with tools like ChatGPT provides evidence of StrategicAI's efficacy in overcoming strategic decision challenges.

2 Theoretical Background

Strategic decision-making is pivotal to organizational success, shaping long-term direction amid complexity and uncertainty. This section lays the theoretical groundwork for enhancing such decisions and addressing human challenges like incomplete data. It explores Strategic Decision-Making and its three-step framework—Root Cause Analysis, Solution Exploration, and Solution Validation—followed by Decision Theory for rational tools under uncertainty, the Decision Quality Framework for robust choices, Issue Trees and Hypothesis Trees for structured problem-solving, and Human in the Loop UI/UX for effective human-AI collaboration. Together, these frameworks underpin an LLM-based tool to improve strategic decisions, blending theory with practical application.

2.1 Strategic Decision-Making

Strategic decision-making involves high-level choices that shape an organization's direction over the long term. These decisions, such as launching a new product, entering an international market, or adopting sustainable practices, are not routine but rather align with the organization's mission and vision (Collis & Rukstad, 2013). For example, Apple's strategic choice to integrate its products—iPhones, Macs, and Apple Watches—into a cohesive ecosystem aimed not only to sell devices but also to retain customers and secure market leadership (Lakhani, 2016). Such decisions are bold, forward-looking, and tied to the organization's long-term aspirations.

The importance of strategic decision-making lies in its role as a unifying framework. It directs the allocation of resources—financial, human, and temporal—ensuring alignment with overarching goals despite daily operational challenges (Porter, 1996). Effective strategic decisions, such as Netflix's transition from DVD rentals to streaming, can propel an organization ahead of competitors and foster sustained success. Conversely, flawed decisions, like Kodak's reluctance to embrace digital photography as film declined, can lead to failure (Lucas & Goh, 2009). Thus, strategic decision-making is critical not only for survival but also for fostering growth, maintaining relevance, and balancing short-term gains with long-term objectives.

However, strategic decision-making poses significant challenges for humans due to the complexity and uncertainty of the business environment. Markets evolve unpredictably, competitors introduce unexpected disruptions, and technological advancements accelerate rapidly (Collis & Rukstad, 2013). Additionally, human decision-makers face cognitive biases, such as overconfidence—overestimating one's foresight—or a tendency to prioritize immediate rewards over sustainable outcomes (Collis & Rukstad, 2013). Emotional factors further complicate the process; for instance, teams may resist significant changes due to perceived risks or disruptions

to established routines. Moreover, reconciling the diverse interests of stakeholders—investors, employees, and customers—adds further difficulty. To address these challenges, a structured approach is essential.

This thesis proposes a three-step framework to enhance strategic decision-making: Root Cause Analysis, Solution Exploration, and Solution Validation. These steps provide a systematic method to navigate complex problems and arrive at informed decisions. As illustrated in Figure 1, the strategic decision-making process consists of these three key steps, each building upon the previous to ensure a comprehensive and effective approach.

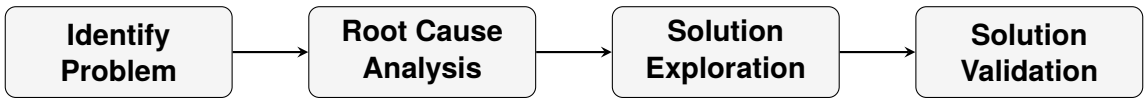

Figure 1 Flowchart of the strategic decision-making process in StrategicAI, comprising three key steps: Root Cause Analysis, Solution Exploration, and Solution Validation.

- **Root Cause Analysis (RCA):** This step focuses on identifying the underlying causes for a problem rather than merely addressing its symptoms. For instance, if sales decline, RCA investigates whether the cause lies in ineffective marketing, product shortcomings, or other factors (Rooney & Heuvel, 2004). This is vital because unresolved root causes lead to recurring issues. Consider a factory experiencing frequent equipment failures: RCA might reveal inadequate maintenance schedules as the source, preventing larger disruptions (Kumar & Kumar, 2004). Practical tools for RCA include the Five Whys method—repeatedly asking "why" to uncover deeper causes (e.g., "Why did sales drop? Due to poor advertising. Why poor advertising? Incorrect audience targeting.")—and Fishbone Diagrams, which categorize causes (e.g., people, processes, technology) in a visual format (Choo et al., 2007).

- **Solution Exploration:** After identifying the root causes of a problem, this phase involves generating and evaluating potential solutions aligned with organizational goals. For example, if customer retention falls, options might include refining the product, intensifying marketing efforts, or introducing a loyalty program (Mumford et al., 1991). This step is crucial as it encourages innovative thinking and prevents premature commitment to suboptimal ideas, as exemplified by Netflix's shift to streaming while DVDs remained dominant (Johnson & Suskewicz, 2009). Techniques such as brainstorming allow teams to propose diverse ideas, while SWOT analysis evaluates internal strengths (e.g., brand reputation) against external threats (e.g., emerging competitors). Scenario Planning, which explores possible future conditions (e.g., market growth or decline), further informs decision-making (Schoemaker, 1995).

- **Solution Validation:** This final step tests the feasibility and effectiveness of a proposed solution before full implementation. For instance, a new marketing strategy might be piloted in a single region to assess its impact (Thomke, 2001). Validation is essential to minimize risks—such as launching an unappealing product—and to build confidence among stakeholders. In software development, companies prototype applications and refine them based on user feedback to ensure a successful launch (Cooper & Kleinschmidt, 1995). Methods include pilot testing (small-scale trials), benchmarking (comparisons with industry standards), and simulations to forecast outcomes (Leonard-Barton, 1992).

These three steps—Root Cause Analysis, Solution Exploration, and Solution Validation—form an integrated process that mitigates the inherent uncertainties of strategic decision-making (Collis & Rukstad, 2013). For this thesis, an LLM-based tool is being developed to automate and enhance this framework. Acting as an intelligent assistant, it will analyze data, propose solutions, and validate outcomes while minimizing the biases and emotional influences that often hinder human decision-makers.

2.2 Decision Theory

Decision Theory provides a rigorous framework for making rational choices under uncertainty, making it a cornerstone for strategic decision-making where outcomes are unpredictable and stakes are high. It is broadly

categorized into two branches: *Normative Decision Theory*, which outlines how decisions should ideally be made by maximizing expected utility in a rational, mathematical manner, and *Descriptive Decision Theory*, which examines how decisions are actually made, often influenced by cognitive biases and emotions (Stanford Encyclopedia of Philosophy, 2023). For strategic purposes, the normative approach is particularly valuable, offering structured tools to guide high-level choices, though descriptive insights help address human limitations in practice (Collis & Rukstad, 2013).

Strategic decision-making involves long-term, high-impact choices—such as launching a new product or entering a global market—that shape an organization's future (Lakhani, 2016). These decisions are fraught with uncertainty due to evolving markets, technological disruptions, and human biases like overconfidence or short-termism (Collis & Rukstad, 2013). Decision Theory connects directly to this context by providing a systematic way to evaluate options, manage risks, and optimize outcomes, serving as a bridge to the proposed three-step framework: Root Cause Analysis (RCA), Solution Exploration, and Solution Validation. Its relevance lies in enhancing each step with analytical rigor, ensuring decisions are not only reactive but also proactive and aligned with organizational goals (ScienceDirect, 2023).

2.2.1 Integration with Strategic Decision-Making Processes

Decision Theory integrates seamlessly with the three processes outlined in this thesis, offering practical tools to navigate uncertainty and improve decision quality.

Root Cause Analysis (RCA) RCA identifies the underlying causes of problems, such as a sales decline, rather than treating symptoms (Rooney & Heuvel, 2004). Decision Theory enhances this process by introducing probabilistic reasoning—e.g., using Bayesian analysis to assign likelihoods to potential causes based on data like customer feedback or operational metrics (ScienceDirect, 2023). For instance, a factory facing equipment failures might use Decision Theory to weigh the probability of inadequate maintenance versus design flaws, guiding resource allocation to the most likely root cause. This ensures RCA is not just diagnostic but also decision-oriented, preventing recurring issues efficiently.

Solution Exploration This phase generates and evaluates solutions, such as improving product quality or ramping up marketing to address customer retention (Mumford et al., 1991). Decision Theory provides powerful frameworks here:

- **Decision Trees:** These map options (e.g., "launch a product" or "wait") and outcomes (e.g., "high sales" or "failure"), assigning probabilities and payoffs to calculate expected values. A retailer might use this to assess opening a new store, balancing costs against revenue forecasts (The Decision Lab, 2023).
- **Cost-Benefit Analysis:** This quantifies pros (e.g., increased revenue) and cons (e.g., investment costs) to rank solutions. A carmaker deciding between electric and gas models could use it to compare production costs and market demand (Schoemaker, 1995).
- **Scenario Planning:** This explores future states—like economic growth or downturns—to test solution robustness. An airline planning fleet upgrades might assess fuel price scenarios, ensuring flexibility (Focused Momentum, 2023).

These tools, rooted in Decision Theory, make Solution Exploration systematic, reducing reliance on intuition and fostering innovative, data-driven options (Nooraie, 2021).

Solution Validation Validation tests solutions before full rollout, such as piloting a marketing campaign in one region (Thomke, 2001). Decision Theory supports this by designing validation protocols and interpreting results under uncertainty. For example, statistical hypothesis testing can determine if a pilot's success is significant, while decision trees model whether to scale up based on cost and performance thresholds (Leonard-Barton, 1992). A software firm prototyping an app might use Decision Theory to decide if user feedback justifies a launch, minimizing risks and building stakeholder confidence. This step ensures solutions are not only viable but also optimized for real-world impact.

2.2.2 Why Decision Theory Matters

Decision Theory's connection to strategic decision-making is critical because it addresses the inherent uncertainties and biases that plague human judgment (Collis & Rukstad, 2013). By integrating with RCA, Exploration, and Validation, it transforms these processes into a cohesive, analytical pipeline. For instance, it enables an LLM system to automate decision-making tasks—building decision trees from market data, running cost-benefit analyses instantly, or simulating scenarios like "What if a competitor cuts prices?"—enhancing speed and precision (Collis & Rukstad, 2013). This synergy not only mitigates flaws like overconfidence but also empowers organizations to make bold, informed strategic moves, as seen in Netflix's streaming pivot or Apple's ecosystem strategy (Lakhani, 2016).

Process	What It Does	Why It Matters	Tools
Root Cause Analysis	Digs into why problems happen (e.g., sales drop)	Stops issues from recurring, saves trouble	Diagnostic Issue Trees, Five Whys, Fishbone, Pareto
Solutions Exploration	Dreams up fixes (e.g., new product or marketing)	Sparks ideas, avoid bad calls	Solution Issue Trees, Brainstorming, Scenario Planning
Solutions Validation	Tests if solutions work (e.g., pilot a campaign)	Cuts risks, builds confidence	Hypothesis Trees, Pilots, Prototyping, Benchmarking

Table 1 Key Processes in Strategic Decision-Making Enhanced by Decision Theory

2.3 Decision Quality Framework

The Decision Quality Framework, developed by Carl Spetzler and colleagues at the Strategic Decisions Group (SDG), provides a systematic methodology for enhancing decision-making processes, as outlined in their seminal work, *Decision Quality: Value Creation from Better Business Decisions* (Spetzler et al., 2016). This framework is particularly significant in strategic contexts where complexity, uncertainty, and high stakes demand robust decision-making. This subsection examines the framework's definition, structure, significance, practical application, and contributions to strategic decision-making, focusing on its integration with Root Cause Analysis (RCA), Solution Exploration, and Solution Validation. It also evaluates its strengths, limitations, and comparative standing against other methodologies.

Definition and Structure The Decision Quality Framework is a six-step process designed to ensure decisions are both robust and value-driven. Rooted in decades of decision science research and practical application at SDG, it comprises the following interconnected components:

- **Appropriate Frame**: Clearly defining the decision problem or opportunity.
- **Creative Alternatives**: Generating a diverse set of feasible options.
- **Relevant and Reliable Information**: Collecting accurate data to inform choices.
- **Clear Values and Trade-offs**: Identifying priorities and evaluating trade-offs.
- **Sound Reasoning**: Applying logical analysis to assess options.
- **Commitment to Action**: Ensuring stakeholder alignment and implementation readiness.

Relation to Analytical Processes The framework integrates seamlessly with analytical methodologies:

- **Root Cause Analysis (RCA)**: Enhances problem framing by identifying underlying issues.
- **Solution Exploration**: Supports the generation of creative alternatives.
- **Solution Validation**: Ensures rigorous evaluation through data and reasoning.

Significance in Strategic Decision-Making The Decision Quality Framework is critical in mitigating risks and optimizing outcomes in complex, uncertain environments. It is especially valuable for strategic business decisions—such as market entry or profitability—where alignment and long-term value creation are paramount. Fostering comprehensive analysis and stakeholder consensus drives sustainable success.

2.4 Issue Trees

Issue trees are an important tool for problem-solving in management consulting, particularly within business management. This section will thoroughly examine the two primary types—Diagnostic and Solution Issue Trees—covering their definitions, construction methods, utility in Root Cause Analysis and Solution Exploration, and their role in strategic decision-making.

2.4.1 Issue Trees in Consulting and Their Role in Making Strategic Decisions

Issue trees, also known as logic trees, are visual diagrams used to decompose complex problems into smaller, manageable components. They are widely utilized in management consulting to structure problems and facilitate systematic analysis, particularly in firms like McKinsey and BCG. Their role in strategic decision-making includes problem structuring, enhancing client communication, prioritizing efforts, and enabling data-driven decisions. The MECE principle (Mutually Exclusive, Collectively Exhaustive) ensures comprehensive and non-redundant decomposition, enhancing decision-making efficiency (Rasiel & Friga, 2001).

2.4.2 Diagnostic Issue Trees

Diagnostic Issue Trees or Why Trees are designed to identify the root causes of a problem, addressing "why" questions. They start with a problem statement and break it down into potential causes, guiding consultants toward data-driven root cause analysis.

Definition and Purpose A Diagnostic Issue Tree begins with a specific problem, such as "Decreasing Profitability," As shown in Figure 2 and decomposes it into major components like "Decreasing Revenues" and "Increasing Costs." Its purpose is to systematically explore all possible causes, ensuring no aspect is overlooked (MConsultingPrep, 2024).

Construction Construction involves:

1. Defining the problem (e.g., "Why are sales declining?").
2. Breaking it into MECE sub-issues (e.g., "Price" and "Volume").
3. Decomposing further (e.g., "Volume" into "Customer Acquisition" and "Retention").

The tree is typically drawn vertically or horizontally (the Case Interview, 2025).

Example of Node Decomposition Consider "Decreasing Profitability", see figure 2:

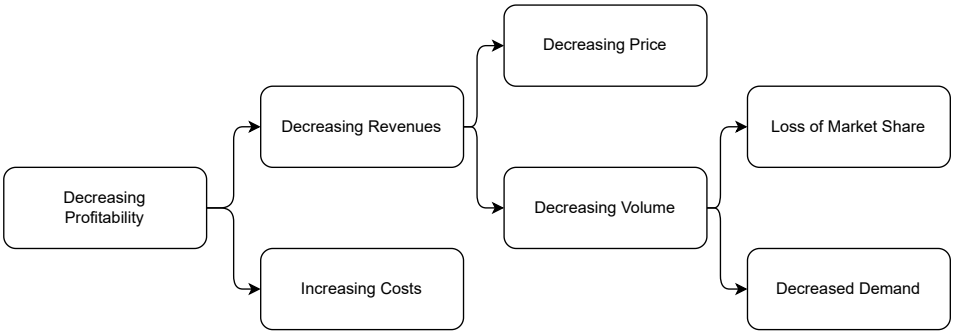

Figure 2 Example of a Diagnostic Issue Tree decomposing "Decreasing Profitability" into causes.

Types of Decompositions Decomposition methods include:

- **Functional:** By departments (e.g., marketing, operations).
- **Causal:** By cause-effect (e.g., competition, pricing).
- **Geographical:** By regions or markets.

These ensure context-specific analysis (Nogueira, 2024).

Utility in Root Cause Analysis In Root Cause Analysis, Diagnostic Issue Trees guide data collection to pinpoint issues. For example, if "Loss of Market Share" is identified, consultants can analyze competitor actions or customer preferences.

2.4.3 Solution Issue Trees

Solution Issue Trees or **How Trees** focus on exploring solutions, addressing "how" questions. They start with a goal and decompose it into actionable strategies, aiding in solution exploration and prioritization.

Definition and Purpose A Solution Issue Tree begins with an objective, such as "Increase Market Share," and breaks it into strategies like "Improve Product Quality" or "Increase Marketing Spending." It aims to evaluate and prioritize solutions based on feasibility and impact (Serrano, 2024).

Construction Construction involves:

1. Defining the goal (e.g., "How to increase market share?").
2. Breaking it into MECE strategies (e.g., "Marketing," "Distribution").
3. Decomposing further (e.g., "Marketing" into "Advertising," "Promotions").

Example of Node Decomposition Consider "Increase Market Share", see figure 3:

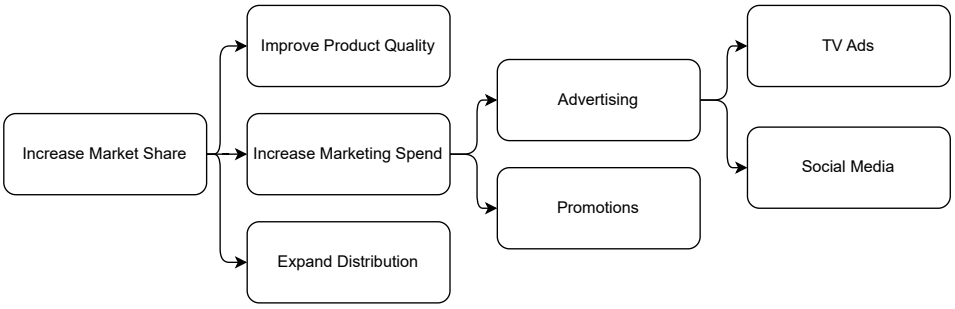

Figure 3 Example of a Solution Issue Tree decomposing "Decreasing Profitability" into causes.

Types of Decompositions Decomposition methods include:

- **Process-Based:** By steps (e.g., advertising, sales process).
- **Strategic:** By approaches (e.g., cost reduction, innovation).
- **Customer Segments:** By demographics or behavior.

These tailor solutions to the context (Nogueira, 2024).

Utility in Solution Exploration In Solution Exploration, Solution Issue Trees brainstorm and assess strategies. For example, "Advertising" can be evaluated for cost and reach, prioritizing high-impact options (the Case Interview, 2025).

2.4.4 Why They Should Be Used

Issue trees handle complexity systematically, ensure comprehensive coverage via MECE, enhance client communication, and facilitate data-driven decisions. They are critical for resource prioritization in consulting projects (Cheng, 2024).

Surprising Detail: Lack of Academic Research Despite their prominence, there is a notable scarcity of academic papers on issue trees, with knowledge primarily from practitioner resources like books and online articles.

2.5 Hypothesis Trees

Hypothesis Trees are key tools for making strategic decisions, helping to test ideas and ensure choices are sound. This subsection defines Hypothesis Trees, explains how they are built and used, and shows their link to Solution Issue Trees in strategic contexts.

2.5.1 What Are Hypothesis Trees?

Hypothesis Trees are diagrams that split a specific idea, or hypothesis, into smaller parts to test its truth. Unlike Issue Trees, which explore broad problems or solutions, Hypothesis Trees focus on one question, such as "Will more marketing increase sales?" They support strategic decisions by testing ideas with data, ensuring reliable choices (Kenney, 2024).

Their main purposes are:

- **Testing Ideas:** They verify if a hypothesis, like a strategic move's impact, holds up.
- **Validating Options:** They check if proposed strategies can achieve goals, providing clear feedback.

2.5.2 How Hypothesis Trees Aid Strategic Decisions

In strategic decision making, hypotheses follow solution issue trees, which suggest ways to meet goals (e.g., "increase market share"). For each option, a Hypothesis Tree tests its value. This data-driven method ensures that decisions are based on evidence, not guesses.

Consider the option "increase marketing." The hypothesis "More marketing boosts sales" is tested with past sales or customer data. This helps decision-makers:

- Rely on facts to make decisions.
- Focus resources on effective strategies.
- Explain the decisions clearly to stakeholders.

2.5.3 Building a Hypothesis Tree

Creating a Hypothesis Tree is simple and structured:

1. **State the Hypothesis:** Define the idea, e.g., "Expanding to new markets will increase market share."
2. **Break It Down:** Split it into testable parts like market potential, readiness, and costs.
3. **Ask Questions:** Form questions like "Is the market growing?"
4. **Gather Data:** Use reports or surveys to answer.

This method, outlined by (Stratechi, 2024), keeps the tree focused and useful.

Example of a Hypothesis Tree For a goal of growing market share with the strategy "expand to new markets," the hypothesis is "Expanding to new markets will increase market share." The tree might be:

Level	Details
Hypothesis	Expanding to new markets will increase market share
Part 1	Market Potential: Market size, competitors
Part 2	Company Readiness: Distribution, product fit
Part 3	Financial Feasibility: Cost, expected profit

Table 2 Example Hypothesis Tree for Market Expansion

Each part has questions like "What is the competitor share?" answered with data, ensuring a full test (Kenney, 2024).

2.5.4 Why Hypothesis Trees Matter in Strategic Decisions

Hypothesis Trees are valuable because they:

- **Simplify Testing:** Break big ideas into small, clear tests.
- **Ensure Evidence-Based Choices:** Use data over assumptions.
- **Save Effort:** Prioritize key tests first.
- **Boost Clarity:** Show how strategies were checked.

As (PrepLounge, 2024) notes, this makes strategic planning more effective and trusted.

2.5.5 Link to Solution Issue Trees

Solution Issue Trees offer strategies/solutions, like "improve products" or "more marketing," to reach goals such as "increase market share." Hypothesis Trees then test each one. For example, "Will better products increase sales?" is checked with feedback and trends. This builds on Solution Issue Trees, turning options into proven decisions (Kenney, 2024), enhancing strategic outcomes.

2.6 Human in the Loop UI/UX for AI-Based Decision Support Systems

This subsection examines Human in the Loop (HITL) in AI-powered Decision Support Systems (DSS) for strategic business decision-making, focusing on UI/UX design enhancements to foster effective human-AI collaboration. It explores the critical role of HITL, the complementary nature of human and AI capabilities, and proposes UI/UX strategies supported by recent academic research and practical examples.

2.6.1 Conceptual Framework

Human in the Loop (HITL) integrates human expertise with AI systems to improve accuracy and trustworthiness in AI-powered DSS. As defined by Q. Zhang et al. (2022), HITL involves humans providing oversight, validation, and context to AI-generated insights, forming a feedback loop where users input problems and evaluate suggestions. This is illustrated in Figure 4. The framework leverages AI's capacity to process large datasets and detect patterns, complemented by human judgment, creativity, and ethical oversight (Dellermann et al., 2020). This synergy is essential for strategic decisions, such as market expansion or resource allocation, ensuring outcomes are data-driven and contextually appropriate.

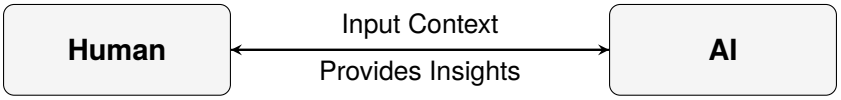

Figure 4 Human-in-the-Loop interaction: humans input problems and evaluate AI suggestions in a feedback loop.

2.6.2 Importance of HITL in Strategic Decision-Making

Strategic decisions in business, characterized by long-term impacts and uncertainty, require both analytical rigor and human insight (Rousseau, 2018). AI-powered DSS excel at trend analysis and outcome prediction, yet lack contextual understanding and ethical judgment, making HITL indispensable. Humans validate AI outputs, aligning them with organizational goals and ethical standards, as seen in financial strategies where AI suggests investments and humans ensure regulatory compliance (Shank et al., 2019). HITL mitigates AI biases and enhances trust, with Monarch (2022) emphasizing that human oversight ensures reliability and transparency in high-stakes scenarios like supply chain optimization (Shahriar, 2024).

2.6.3 Complementary Roles of Humans and AI

Humans and AI complement each other by combining their strengths. AI automates data processing, identifying patterns such as market trends from customer feedback (SmythOS, 2024), while humans excel in judgment, creativity, and stakeholder engagement (Jarrahi, 2018). This collaboration enables AI to handle routine tasks, freeing humans for strategic work, such as assessing cultural fit in planning (Parry et al., 2016). The resulting feedback loop, where AI learns from human inputs, improves decision-making efficiency (Lyons & Havig, 2017), balancing data-driven insights with human contextual analysis (Bolton et al., 2018).

2.6.4 UI/UX Design Enhancements for Effective Collaboration

UI/UX design is critical for enabling seamless HITL in AI DSS. The following strategies, supported by academic research and examples, enhance human-AI interaction:

- **Natural Language Interaction**: Interfaces should allow queries in everyday language (e.g., "What are our sales trends?"). This is something we see in every LLM provider like the ChatGPT web app.
- **Explainability**: AI must provide clear explanations of recommendations, using methods like LIME or SHAP. For instance, a DSS for investment decisions could display visual breakdowns of predicted returns, enabling users to adjust parameters (Adadi & Berrada, 2018).
- **Feedback Mechanisms**: Tools for correcting AI outputs refine performance iteratively. A case study on 3D model processing showed artists using a drag-and-drop interface to adjust AI suggestions, improving accuracy (Q. Zhang et al., 2022).

2.6.5 Empirical Insights and Case Studies

Empirical evidence highlights HITL's value. Dellermann et al. (2020) found transparency and AI literacy critical for strategic contexts, while Oxford Law Blogs (2024) noted that HITL boosts adoption but risks accuracy if oversight is inconsistent, underscoring UI/UX's role. In practice, 1000minds (2023) showed users refining AI suggestions in decision-making, improving outcomes through iterative feedback.

3 Related Work

To inform the development of an LLM-based Decision Support System (DSS) for strategic business decision-making, this literature review examines foundational research across AI-driven decision support, structured problem-solving, and LLM explainability. By exploring LLMs' emerging roles in strategy generation and evaluation alongside principles from the Decision Quality (DQ) framework, this review identifies critical methodologies for enhancing decision accuracy, transparency, and adaptability. Tree-of-thoughts and Issue Trees also provide structural models for decomposing complex problems, supporting a rigorous approach to hypothesis testing and solution evaluation. Collectively, these studies offer critical insights that shape a robust, adaptable DSS tailored for complex business contexts.

3.1 Role of Large Language Models in Strategic Decision-Making (SDM)

The use of Large Language Models (LLMs) in strategic decision-making (SDM) has garnered significant interest due to their potential to enhance various decision-making processes traditionally managed by human strategists. A recent study by Csaszar et al. (2024) explores how LLMs, specifically models like GPT-3.5 and GPT-4, could support the SDM process by generating, evaluating, and augmenting strategic insights, thus positioning AI as an influential tool in business contexts where complex decision-making is essential. Their findings align with our objective of developing a Decision Support System (DSS) that utilizes LLMs to assist in generating hypotheses, evaluating potential solutions, and supporting business managers and CEOs in strategic decision-making.

LLMs for Generating Strategies in Strategic Decision Making (SDM)

Csaszar et al. (2024) argues that LLMs can effectively be utilized in SDM by generating credible, contextually relevant strategies. Their study provides empirical evidence through a controlled experiment where an LLM-generated business plan was compared with plans crafted by human entrepreneurs within a European startup accelerator program. The LLM produced business plans that incorporated strategic components such as problem definition, solution development, and market analysis, closely aligned with those generated by human experts. Furthermore, their results reveal that the LLM-generated plans were often rated as highly as, or even higher than, those created by entrepreneurs. Evaluators—experienced investors—rated the LLM-generated strategies favorably, with these plans showing a significant ability to attract investor interest.

These findings provide crucial validation for our thesis. The capability of LLMs to generate strategic ideas that are credible enough to capture the attention of industry experts aligns well with our use of LLMs to generate potential solutions hypotheses. Our DSS leverages LLMs similarly to propose alternative solutions to specific business problems, helping managers consider a broad spectrum of strategic options.

LLMs for Evaluating Strategic Alternatives

In addition to generating strategies, Csaszar et al. (2024) demonstrates that LLMs can serve a valuable role in evaluating strategic plans. Through an experiment involving a startup competition, the authors tested the LLM's capacity to assess business plans compared to evaluations by human judges, including experienced venture capitalists and angel investors. Their results showed a positive correlation between the scores generated by the LLM and those of human judges, indicating that LLM evaluations often agreed with expert opinion on essential criteria such as team composition, market opportunity, financial viability, and long-term potential.

This ability to accurately evaluate strategic alternatives further supports our thesis objective of using LLMs to evaluate potential solutions in a business context. Our DSS system intends to utilize LLMs to assess each alternative's applicability to a company's unique profile and goals. This integration of AI in evaluation aligns with Csaszar et al.'s findings, underscoring the viability of LLMs to complement or even partially substitute human evaluators in preliminary decision-making stages.

Enhancing Cognitive Processes in SDM: Search, Representation, and Aggregation

One of the significant contributions of Csaszar et al.'s study lies in their analysis of how AI, especially LLMs, can enhance the cognitive processes foundational to SDM—namely, search, representation, and aggregation. They emphasize that AI can significantly extend the "search" function within SDM by rapidly generating a wide array of potential solutions and allowing for a more thorough exploration of strategic possibilities than is feasible with human decision-makers alone. This expansion of search capability aligns directly with our goal of utilizing LLMs in our DSS to assist in decomposing complex business problems into manageable components. In our application, LLMs are central to exploring potential alternative solutions and potential causes of a problem.

Alignment with Our Research Objectives

The insights provided by Csaszar et al. offer a valuable theoretical and empirical foundation for our work. Their findings support the integration of LLMs in SDM, particularly for generating and evaluating strategies, and align closely with our DSS objectives. LLMs' demonstrated capacity to generate credible strategies and evaluate alternatives with a level of accuracy comparable to human experts strengthens our thesis premise that LLMs can serve as effective AI consultants in strategic decision-making. Additionally, their discussion on AI's enhancement of cognitive processes in SDM validates our approach to using LLMs to facilitate the decomposition of problems and the generation of structured hypotheses through our Issue and Hypothesis trees.

Overall, Csaszar et al.'s study provides robust evidence for the potential of LLMs to transform SDM and affirms the relevance of our thesis in leveraging these advancements. Their work reinforces the value of using AI as a complementary tool and an integral component of a dynamic, human-AI collaborative framework for strategic decision-making.

3.2 STRUX Framework for Decision-Making with Structured Explanations

The STRUX framework, introduced by Lu et al. (2024), presents a novel LLM-based approach aimed at enhancing decision-making transparency through structured explanations. Designed specifically for applications in financial decision-making, STRUX applies its model to stock investment forecasting by analyzing earnings call transcripts. The framework generates a structured fact table from the input data, where relevant factors are categorized as favorable or adverse, each assigned a strength level to denote its influence on the decision. This format not only organizes key facts in an accessible manner but also aids in evaluating their impacts on decision outcomes.

The structured fact table generated by STRUX is central to its decision-making transparency, as it summarizes and organizes key details from the financial data, mirroring a goal in our own project. Inspired by this approach, our system intends to generate a fact table for each uploaded data file, leveraging these facts to support explanations for root cause analyses and hypothesis validations, similar to STRUX's application in investment decision forecasting.

STRUX's iterative "self-reflection" process is another critical component. By reflecting on previous decisions and recognizing any misjudgments in the fact selection and evaluation, the framework is able to refine its decision-making process without relying on human feedback. This self-reflection process enhances the model's accuracy in identifying and prioritizing the most relevant factors. Lu et al. further emphasize the importance of fine-tuning the model specifically for decision-making tasks, ensuring that it prioritizes highly pertinent facts from the generated fact table and maintains decision accuracy through each iteration.

In comparison to other LLM-based decision-making frameworks, STRUX distinguishes itself by presenting structured rather than narrative explanations, which mitigates the risk of overwhelming users with unstructured data. This approach contrasts with frameworks like DeLLMa (Liu et al., 2024), which uses classical decision theory but lacks STRUX's detailed tabular explanation model. This tabular approach enables users to see a balanced perspective of both positive and negative factors influencing a decision, thus enhancing transparency. Experimental results show that STRUX outperforms strong baselines, such as Llama3 and GPT-4o-mini (Hurst et al., 2024), achieving superior accuracy and F-scores in stock investment prediction.

Overall, STRUX's structured explanations represent a significant step toward practical decision-making with LLMs, and its approach aligns with our objectives in creating a Decision Support System (DSS) for business environments. Adopting a fact table similar to STRUX enables our DSS to provide users with accessible and traceable explanations for strategic decisions, thereby enhancing transparency and accountability in LLM-driven decision-making.

3.3 Explainability for Large Language Models: A Survey

The paper titled "Explainability for Large Language Models: A Survey" by Zhao et al. (2024) provides an extensive review of methods for enhancing explainability in Transformer-based large language models (LLMs). It highlights explainability's critical role in understanding how and why specific predictions are made. In this context, explainability is essential for building user trust and aiding developers in interpreting and improving model behavior.

Explainability is particularly important in LLMs, where model complexity often results in a lack of transparency—commonly known as the "black-box" problem. Explainability enables end-users to understand the reasoning behind predictions, fostering more informed decision-making. For developers, it serves as a diagnostic tool, helping identify and address biases, ultimately supporting the development of more robust, ethical, and accurate models (Yuksekgonul et al., 2023). This need for transparency is especially pertinent in strategic decision support systems (DSS) managers use, where LLMs impact high-stakes decisions.

The survey categorizes explainability methods using two predominant LLM paradigms: fine-tuning and prompting. In the fine-tuning paradigm, models undergo pre-training followed by adjustments on domain-specific data, allowing explanations to focus on task-specific behaviors (Rogers et al., 2020). In contrast, the prompting paradigm—used in models like GPT-4o (Hurst et al., 2024), LLaMA, and Qwen—enables zero-shot or few-shot learning without additional training. As our DSS leverages general-purpose LLMs with prompting, the survey's focus on prompting-based explainability is highly relevant.

Achieving effective explainability in LLMs involves several challenges. LLM complexity often leads to transparency issues, making it difficult to explain internal decision-making processes meaningfully. Zhao et al. note that large models may produce nonsensical or unexpected outputs (hallucinations) due to incomplete representations of underlying concepts (Weidinger et al., 2021). Additionally, as model sizes increase, traditional interpretability methods, such as feature attribution and attention-based techniques, become less feasible.

An essential aspect of explainability in LLMs is uncertainty quantification, which helps evaluate model reliability by estimating prediction confidence. For closed-source LLMs where logits may not be accessible, confidence elicitation techniques offer an alternative by assessing uncertainty without probability scores. In our DSS, we incorporate uncertainty quantification by associating a confidence level with each explanation and action. For instance, the DSS might suggest "enter a new market" with an explicit confidence percentage and a rationale, giving users a clearer sense of the model's certainty.

Consistency-based approaches, such as self-consistency (Wang et al., 2023), also contribute to uncertainty estimation by examining variation across multiple responses to repeated prompts. This technique supports reliable decision-making in our DSS, where consistency can be measured by comparing response similarity across prompt variations.

Zhao et al. also discuss the evaluation of explanations based on faithfulness and plausibility:

Faithfulness measures the alignment of an explanation with the actual reasoning process of the model. Many LLM-generated explanations lack faithfulness, meaning they may not reflect the true decision-making path leading to a prediction. In our DSS, evaluating faithfulness is vital, especially when addressing complex

strategic problems. We can assess it by comparing LLM explanations against known ground truths or employing chain-of-thought decomposition to validate reasoning steps.

Plausibility assesses how logical and understandable an explanation appears to users, regardless of whether it reflects the model's internal reasoning. Plausibility can foster user trust by aligning explanations with human reasoning, even if the model's underlying processes differ. In our DSS, we will evaluate plausibility and faithfulness through an LLM-based Preference Evaluation on 4 different metrics including plausibility and faithfulness.

3.4 Issue Trees and the Transformative Role of Large Language Models in Decision Support

Business problems can feel overwhelming, like a puzzle with too many pieces. Issue Trees offer a clear way to solve them by breaking big challenges into smaller, easier parts. According to Parniangtong, 2017 in *Problem-Solving Approach to Business Strategy*, these trees help companies find the real reasons behind issues, not just quick fixes. For example, instead of only boosting sales with discounts, a company might fix deeper problems like poor product quality.

Issue Trees work by organizing a problem into main issues and smaller sub-issues. This setup follows a rule called MECE—mutually exclusive and collectively exhaustive—which means no overlap between parts and all possibilities are covered. If sales are low, an Issue Tree might split into areas like customer demand, pricing, or production costs, then break those down further. This method uses logical thinking, either starting from the problem (deductive) or building up from details (inductive), to guide decision-makers toward solid answers.

3.4.1 How Large Language Models Improve Issue Trees

While Issue Trees are helpful, people using them can miss important details. They might be too busy, biased, or unsure where to look. This is where LLMs, make a difference. LLMs can look at huge amounts of information quickly and suggest ideas that humans might not think of.

For instance, Benary et al., 2023 studied how LLMs like ChatGPT and others help doctors choose cancer treatments. The LLMs didn't always agree with experts, but they offered new ideas—like different drug mixes—that doctors hadn't considered. This shows LLMs can add value by expanding what Issue Trees cover, making them more complete and useful.

3.4.2 Combining Human and Machine Strengths for Better Decisions

Using LLMs with Issue Trees creates a strong team: human experience plus machine power. People can forget options or run out of time, but LLMs keep suggesting possibilities. Imagine a manager planning how to sell more products. An LLM might point out using social media in a new way, something the manager hadn't thought of yet.

This teamwork makes sure no key idea is missed. LLMs support the Issue Tree process by offering data-based suggestions, while people decide what fits best. Together, they help find solutions that fix the real problem and open up new paths forward, improving decisions clearly and thoroughly.

3.5 Advanced Structural Reasoning Paradigms in LLMs

Recent years have witnessed remarkable advancements in prompt engineering techniques designed to enhance the reasoning capabilities of Large Language Models (LLMs). These techniques have evolved from simple instruction-based prompting to sophisticated structural approaches that organize the reasoning process into various topologies. As Besta et al. (2024) highlight in their comprehensive survey, these structural reasoning paradigms—including chains, trees, and graphs—have emerged as powerful methods for guiding LLMs through complex problem-solving tasks ranging from mathematical reasoning to creative writing and planning. This section explores three prominent structural reasoning techniques that have demonstrated significant improvements in LLM performance across various reasoning tasks: Tree of Thoughts (ToT), Probabilistic Tree-of-thought Reasoning (ProbTree), and Self-Consistency Prompting. By organizing the reasoning process into

explicit structures rather than treating it as an unstructured black box, these approaches enable more systematic exploration of solution spaces, better error recovery, and improved handling of complex, multi-step problems.

3.5.1 Tree of Thoughts Prompting Strategy

The Tree of Thoughts (ToT) prompting strategy, introduced by Yao et al. (2023), represents a significant advancement in leveraging LLMs for complex problem-solving through the use of tree structures. Unlike the Chain of Thought (CoT) method (Z. Zhang et al., 2022), which guides LLMs along a linear sequence of reasoning steps, ToT organizes the reasoning process into a hierarchical tree, enabling a more deliberate and exploratory approach. Each node in the tree represents a "thought," defined as a coherent language sequence that serves as an intermediate step toward solving a problem, while edges connect these thoughts to form multiple branching paths. This structure allows LLMs to generate diverse reasoning trajectories, evaluate their potential, and backtrack when necessary, closely mimicking human strategies for tackling tasks that require planning, look-ahead, and adaptability.

The ToT framework operates through two core mechanisms: *propose prompts* and *value prompts*. The proposed prompt generates a set of candidate thoughts based on the current state, effectively creating branches in the tree that represent different possible next steps. For example, in a mathematical reasoning task, these thoughts might include alternative methods like algebraic manipulation or trial-and-error substitution. The value prompt then assesses the quality or promise of each thought, assigning a score or ranking to guide the LLM in selecting which branches to pursue further. This evaluation process can be implemented through heuristic scoring or by prompting the LLM to self-assess, such as asking, "How likely is this step to lead to the solution?" The tree structure is navigated using search algorithms like breadth-first search (BFS) or depth-first search (DFS), where BFS explores all thoughts at a given level to ensure broad coverage, and DFS delves deeply into a promising path before backtracking if it proves unfruitful (Yao et al., 2023).

The use of tree structures in ToT fundamentally enhances problem-solving by providing a systematic way to explore the solution space. For instance, in a puzzle-solving scenario, such as arranging colored balls based on logical clues, the LLM might start with an initial thought (e.g., "Eliminate red balls first"), then branch into subsequent thoughts (e.g., "Check blue ball constraints" or "Test green ball positions"). As the tree grows, the LLM evaluates each path, pruning those that violate constraints and expanding those that align with the clues, eventually converging on the correct solution. This ability to maintain and traverse multiple reasoning paths distinguishes ToT from CoT, which lacks the flexibility to explore alternatives or recover from early missteps. The tree-based approach also supports strategic lookahead, allowing the LLM to anticipate future steps and adjust its reasoning dynamically, a critical feature for tasks like game-playing or optimization problems.

Recent empirical evidence underscores ToT's effectiveness.Yao et al. (2023) demonstrated that ToT can improve LLM performance by up to nine times over standard prompting on benchmarks like mathematical reasoning and puzzle-solving. This leap in capability stems from its capacity to handle tasks where initial decisions significantly impact outcomes, a challenge for linear prompting methods. Additionally,Hulbert (2023) highlights ToT's practical applications, such as in creative writing and complex scenario analysis, where it generates diverse and nuanced outputs by branching into multiple creative directions. For example, in a vacation planning task, ToT might explore branches like national parks, beaches, and mountains, evaluating each for nature-centric appeal, resulting in richer recommendations.

In summary, ToT leverages tree structures to transform LLMs into more versatile problem-solvers, offering a framework that combines exploration, evaluation, and adaptability. Its novelty lies in its departure from linear reasoning, enabling LLMs to tackle complex, multi-step problems with a level of deliberation previously unattainable in standard prompting techniques.

3.5.2 Probabilistic Tree-of-thought Reasoning (ProbTree)

ProbTree, or Probabilistic Tree-of-thought Reasoning, is introduced as an advanced method for addressing knowledge-intensive complex questions by enhancing the reasoning capabilities of large language models (LLMs). It overcomes limitations in traditional chain-of-thought (CoT) reasoning, such as negative retrieval and limited sight, by employing a tree-based structure known as the query tree. Each node in this tree represents a

sub-question, and reasoning is conducted probabilistically from leaf to root nodes. The approach integrates both parametric knowledge embedded within the LLM and external knowledge retrieved from additional sources, making it particularly suited for open-domain complex question answering (Cao et al., 2023).

Answering Techniques in ProbTree ProbTree employs multiple answering strategies tailored to the hierarchical structure of the query tree:

- **Closed-book QA:** This technique utilizes the LLM's internal parametric knowledge to generate answers without relying on external retrieval, suitable for questions where the required information is likely encoded within the model.
- **Open-book QA:** Here, the LLM augments its reasoning with external knowledge retrieved from sources, addressing gaps in the model's knowledge, particularly for up-to-date or domain-specific information.
- **Aggregated Reasoning for Non-leaf Nodes:** For non-leaf nodes, ProbTree synthesizes answers from child nodes, considering their confidence scores, to formulate a response for the parent node, enabling recovery from local errors through broader context.

The selection between Closed-book and Open-book QA for leaf nodes is determined by confidence scores, ensuring the most reliable answer is chosen.

Confidence Calculation and Certainty Certainty in ProbTree is quantified through confidence scores derived from explanation logits, which are the log probabilities of the generated explanations or answers. For aggregated reasoning at non-leaf nodes, the confidence score incorporates both the quality of the decomposition into sub-questions and the reasoning at each node. This probabilistic framework ensures a consistent measure of reliability across the tree.

Query Tree Construction The query tree is constructed by translating a complex question into a hierarchical structure, where the root node represents the original question, and each non-root node is a sub-question of its parent. Leaf nodes are atomic questions that cannot be further decomposed. The process involves few-shot prompting of the LLM to output the tree structure, with the quality of decomposition evaluated probabilistically to ensure effective reasoning.

3.5.3 Self-Consistency Prompting

Self-Consistency Prompting by (Wang et al., 2023) is a technique in prompt engineering for large language models (LLMs) designed to enhance the accuracy and reliability of their outputs by leveraging multiple reasoning paths. Introduced as an advancement over traditional chain-of-thought (CoT) prompting, it addresses the limitations of single-response generation by sampling diverse reasoning trajectories and selecting the most consistent answer through a consensus mechanism

Definition and Mechanism The method operates by generating multiple responses to a given query, each accompanied by a distinct reasoning path, and then employing majority voting to select the most frequent answer. For example, consider the query: "When I was 6, my sister was half my age. Now I'm 70, how old is my sister?" The LLM might produce several reasoning paths, with the majority correctly concluding the sister is 67 years old, thus overriding erroneous single-path calculations. This process involves:

1. **Generation of Multiple Reasoning Paths:** The model is prompted multiple times, typically using few-shot CoT examples, to produce varied step-by-step reasoning sequences.
2. **Selection via Majority Voting:** The most consistent answer across these paths is chosen as the final output, based on the assumption that correct answers are more likely to recur.

Benefits and Applications Self-Consistency Prompting offers several advantages, particularly for complex reasoning tasks:

- **Improved Accuracy:** By aggregating multiple responses, it reduces errors inherent in single-path reasoning, as demonstrated by significant performance gains on benchmarks like GSM8K (+17.9%) and SVAMP (+11.0%).
- **Enhanced Complex Task Handling:** It excels in multi-step arithmetic and commonsense reasoning, such as solving equations or interpreting social scenarios.
- **Robustness to Biases:** The consensus mechanism mitigates model-specific biases, enhancing reliability.

An illustrative example is an arithmetic problem: "If a store has 10 apples and 8 oranges, and sells 6 apples and 4 oranges, how many fruits are left?" Multiple paths may yield 8 as the majority answer, correcting any single-path miscalculations.

Empirical Evidence The technique's efficacy was rigorously evaluated by Wang et al. Wang et al. (2023), showing substantial improvements across diverse benchmarks, as summarized in Table 3. Extensions like Universal Self-Consistency (USC) further adapt the method for free-form generation tasks (Chen et al., 2023).

Benchmark	Improvement (%)
GSM8K	+17.9
SVAMP	+11.0
AQuA	+12.2
StrategyQA	+6.4
ARC-challenge	+3.9

Table 3 Performance Improvements with Self-Consistency Prompting (Wang et al., 2023)

This method's ability to enhance LLM performance makes it a notable contribution to the field of natural language processing, particularly for applications requiring precise reasoning.

4 Research Methodology

4.1 Design Science Research

Design Science Research (DSR), as introduced by Hevner et al. (2004), is a methodological framework widely embraced within the Information Systems (IS) discipline. It emphasizes the creation and evaluation of innovative artifacts to address practical challenges, distinguishing itself from behavioral science's focus on theory development and testing. DSR adopts a constructive approach, prioritizing the design of solutions—such as constructs, models, methods, and instantiations—that deliver tangible utility to users or organizations. Drawing from engineering principles and the sciences of the artificial (Simon, 1996), DSR transforms existing conditions into preferred states through deliberate design processes. This paradigm integrates knowledge generation with actionable outcomes, posing the question of "what is effective" rather than "what is true," making it particularly suited to the dynamic, technology-driven nature of IS.

A key element of DSR is its structured framework for IS research, illustrated in Figure 5 (Hevner et al., 2004). This framework encapsulates the interplay between the research context, the iterative design process, and the dual requirements of practical relevance and theoretical rigor. It begins by situating research within a contextual environment comprising stakeholders (e.g., users and managers with distinct roles and capabilities), organizational settings (e.g., strategies, structures, and processes), and technological systems (e.g., infrastructure and applications). This contextual analysis identifies real-world problems that the research aims to address, ensuring alignment with practical needs.

The framework then outlines an iterative research cycle at its core, involving the development of artifacts or theories, their assessment against defined criteria, and subsequent refinement based on evaluation outcomes.

This cyclical process reflects DSR's commitment to continuous improvement, enabling the creation of solutions—such as decision support tools or database systems—that enhance efficiency or performance. For this study, this iterative approach will guide the artifact's design and optimization, ensuring it evolves to meet its objectives effectively.

To ensure practical impact, the framework emphasizes evaluation methods that connect research outcomes to tangible needs. These methods include analytical validation, case studies, controlled experiments, field observations, and simulations, each assessing the artifact's utility in solving identified problems. This thesis will leverage such techniques to demonstrate the artifact's effectiveness, aligning with DSR's focus on relevance. Simultaneously, the framework demands methodological and theoretical rigor, drawing on an established knowledge base of constructs, models, methods, and prior instantiations, as well as robust methodologies like data analysis and validation standards. This ensures that the research is scientifically credible, a principle that will underpin the artifact's development and evaluation in this work.

Finally, the framework highlights the application of research outcomes in real-world settings, testing their feasibility and effectiveness. Successful applications contribute new insights or methods back to the IS knowledge base, fostering a feedback loop that advances the field. In this thesis, the artifact will be implemented or demonstrated in an appropriate context, with findings aimed at enriching design theory or practice. By adhering to this framework, as depicted in Figure 5, this research will balance innovation with rigor, producing an artifact that addresses a significant IS challenge while contributing to scholarly knowledge (Hevner et al., 2004).

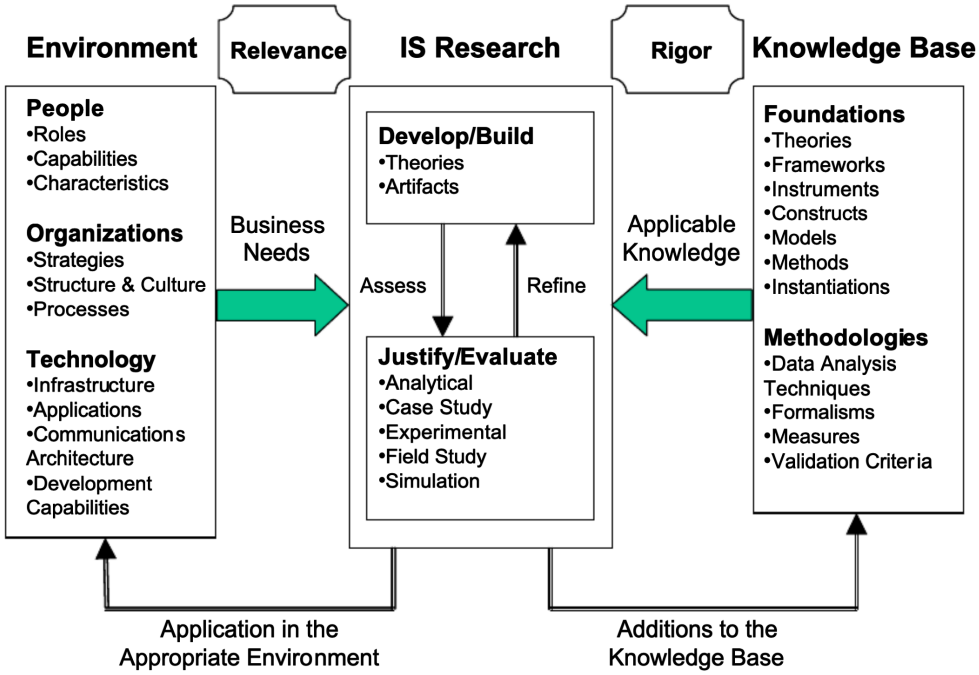

Figure 5 Information Systems Research Framework (adapted from Hevner et al. (2004))

To ensure the rigor and validity of DSR, Hevner et al. (2004) proposes seven foundational guidelines that steer the research process:

1. **Design as an Artifact**: The research must yield a purposeful artifact—be it a system, method, or model—crafted to resolve a specific, identified issue.

2. **Problem Relevance**: The artifact must tackle a meaningful challenge within its domain, ensuring its practical significance.

3. **Design Evaluation**: The artifact's effectiveness, quality, and utility must be substantiated through robust evaluative methods, such as empirical studies or simulations.

4. **Research Contributions**: The study should provide distinct advancements, whether in the artifact's design, theoretical foundations, or evaluation approaches.

5. **Research Rigor**: The construction and assessment of the artifact must rest on solid theoretical and methodological foundations to guarantee credibility.

6. **Design as a Search Process**: The design effort involves an iterative exploration, refining solutions through successive trials and feedback loops.

7. **Communication of Research**: Outcomes must be clearly disseminated to both technical and managerial audiences, enhancing the dissemination of knowledge.

These principles distinguish DSR from routine development by demanding novelty, scholarly rigor, and impactful contributions, a standard this thesis will uphold.

Building on this, Peffers et al. (2007) offers a six-activity methodology to operationalize DSR, providing a systematic yet adaptable structure:

1. **Problem Identification and Motivation**: The research starts by defining a specific problem and justifying its significance, often through literature reviews or stakeholder input, to establish its relevance.

2. **Objective Definition**: Clear goals are articulated for the artifact, specifying what it must accomplish to address the problem effectively.

3. **Design and Development**: The artifact is iteratively crafted, blending creativity with technical expertise and prior knowledge, and refined through ongoing assessments.

4. **Demonstration**: The artifact's ability to solve the problem is showcased, potentially through prototypes, simulations, or pilot implementations, to confirm its feasibility.

5. **Evaluation**: Rigorous techniques—such as experiments, field studies, or analytical methods—assess the artifact's performance, often prompting further refinements.

6. **Communication**: Findings are shared with academic and practitioner communities, contributing to the IS knowledge base and guiding future efforts.

This structured approach will shape this thesis, ensuring a balanced focus on theoretical grounding and practical utility (Peffers et al., 2007).

DSR's strength lies in its capacity to address the IS problems - characterized by shifting requirements and complex interdependencies (Rittel & Webber, 1984)—through adaptive, iterative solutions. Historical examples, such as database management systems, underscore its transformative potential (Hevner et al., 2004). However, challenges remain, including the risk of prioritizing technical novelty over theoretical depth (Tichy, 1998) and the limited longevity of outcomes due to rapid technological evolution (Hevner et al., 2004). Despite these, DSR's emphasis on innovative, rigorously assessed artifacts ensures its relevance, a foundation this research leverages to deliver impactful IS solutions.

4.2 Artifact Development Process: A Design Science Research Approach

This subsection elaborates the development process of "StrategicAI" a web-based application designed to enhance strategic decision-making for companies, adhering to the Design Science Research (DSR) methodology as outlined by (Hevner et al., 2004; Peffers et al., 2007). The process systematically addressed a real-world problem through a structured, iterative approach, incorporating theoretical and practical inputs to produce a purposeful artifact. Focusing on four DSR activities—problem identification, objective definition, design and development, and demonstration—this narrative details the methodological journey, avoiding technical specifics of the app's components.

4.2.1 Problem Identification and Motivation

The development of the decision-support tool started by recognizing a key issue in business environments: managers often struggle to make effective strategic decisions due to large amounts of data and unclear methods. This problem leads to shallow solutions that focus on symptoms instead of root causes, slowing down decision-making, wasting resources, and increasing reliance on costly external consultants (McAfee & Brynjolfsson, 2012). This challenge matters because it affects a company's ability to stay competitive and profitable—poor or delayed decisions can harm market standing or financial health (Careers, 2023). As data continues to grow rapidly, this issue has worsened, with businesses finding it hard to pull useful insights from overwhelming information.

This problem was uncovered through academic research and real-world input. A detailed literature review explored online articles and studies, such as (McAfee & Brynjolfsson, 2012), which explains how "big data" overwhelms managers, and (Scientist, 2023), which notes that few analytics leaders see value in their data efforts. Alongside this, informal interviews were held with the thesis supervisor, an experienced consultant, and peers in consulting, project management, and decision-making roles. These talks highlighted practical struggles: reluctance to dive into complex data, reliance on guesswork to find problems, and missing key details due to confusion or mental overload. For example, a project manager friend mentioned losing track of important problem parts mid-process, while the supervisor pointed out clients' frustration with messy data. Combining academic findings with these real-life insights confirmed the problem's importance, motivating the creation of a tool to simplify decision-making and aligning with DSR's focus on solving meaningful challenges (Hevner et al., 2004).

4.2.2 Defining Solution Objectives

Once the problem was clear, the next step was to set specific goals for the app, meeting DSR's need to define solution aims. The main purpose was to help decision-makers make faster, more accurate choices with fewer mistakes. The detailed objectives were:

- **Root Cause Identification:** The app should understand business issues and find their true causes, much like a consultant's diagnostic skills. This uses the Diagnostic Issue Tree framework from the Theory Background section.
- **Solution Exploration:** The app should suggest practical solutions for company challenges or goals by blending company data with public historical data from online sources, ensuring solutions fit the situation. This relies on the Solution Issue Tree framework.
- **Solution Validation:** The app should assess strategy options (e.g., "Will this boost profits?") with data-driven insights. It checks if a solution solves a problem, meets a goal, or tests a company hypothesis, using the Hypothesis Tree framework.
- **Clear Data Presentation:** The app should turn complicated data into simple, easy-to-read fact lists, similar to the STRUX approach (**<empty citation>**) but based on company or online data. This helps support or challenge causes and solutions from the tree frameworks, making data less intimidating for managers.
- **Usability for Non-Experts:** The app should have an easy-to-use interface for managers without technical skills, widening its use. It includes Human-In-the-Loop methods to simplify navigation and understanding.

These goals tackle the main issues: reducing data overload with clear displays, focusing on root causes instead of symptoms, and simplifying processes with better usability (Scientist, 2023). They were shaped by practical needs from literature and interviews—like the supervisor's call for quick, clear analysis—and by theories like the Decision Quality Framework, which supports structured decisions for better outcomes (Spetzler et al., 2016). Consulting tools, such as issue tree analysis, also guided the goals to match professional practices

(FourWeekMBA, 2023). This mix of real-world and theoretical foundations supported the app's development, fitting DSR's goal-focused approach.

Design and Development: Iterative and Grounded The app's design and development followed DSR's iterative style, using an Agile method with one-week sprints. Every Friday, progress was shown to the thesis supervisor in feedback sessions, where new features, interface updates, and overall performance were reviewed and improved. This cycle of building, showing, and tweaking reflected DSR's idea of design as a search process, exploring solutions step-by-step. The work lasted several months, based on the project timeline, with each sprint moving the app closer to its goals.

Additionally, informal interviews with consulting professionals, such as those from PwC and KPMG, and the thesis supervisor (a McKinsey consultant), were held once or twice based on their availability. These sessions showcased the app's features, collecting notes and feedback on what they'd want from such a tool. This input, alongside the supervisor's regular reviews, helped shape the app to meet real consulting needs.

Demonstration in Development Demonstration, a core part of DSR, happened throughout development as a simple evaluation experiment in weekly sprint reviews with the supervisor. These meetings displayed the app's growing abilities—like turning dense data into clear insights, finding root causes in fake profit-drop scenarios, or testing strategy ideas. Held in a controlled setting, each review used example tasks to mimic managers' work, letting the supervisor check new features, usability, and goal alignment. Feedback varied: early sessions might call for better data visuals, while later ones improved analysis or navigation. This ongoing process proved the app's value and practicality, leading to changes like sharper visuals or simpler steps, matching DSR's focus on showing problem-solving power (Hevner et al., 2004). Though formal testing was saved for later, these reviews linked the development to real use, showing the app's promise.

In summary, the development process followed DSR by tackling a key business issue, setting clear and grounded goals, building a web-based tool with consulting features through iterations, and proving its worth with regular reviews. Further evaluation will be covered later, and when discussing app features, we'll connect back to DSR for insights on design choices and guidelines.

4.3 Evaluation & Experimental Setup

The evaluation of *StrategicAI* was meticulously crafted to align with the Design Science Research (DSR) methodology, as delineated by Hevner et al. (2004), and the Information Systems (IS) research framework. Central to this evaluation is DSR Guideline 3: *Design Evaluation*, which mandates the rigorous demonstration of an artifact's utility, quality, and efficacy through well-executed methods. Our approach tests *StrategicAI*'s ability to address the identified problem—supporting managers in making informed strategic decisions—via a multifaceted evaluation strategy. This strategy encompasses both automated assessments against a baseline and user-centric testing of human-LLM collaboration, reflecting the IS framework's emphasis on balancing practical relevance with theoretical rigor. By situating the evaluation within real-world case studies and organizational contexts, we ensure alignment with DSR Guideline 2: *Problem Relevance*, while contributing novel insights to the IS knowledge base per Guideline 4: *Research Contributions*.

This subsection outlines the experimental design, detailing the dataset preparation, evaluation setups, metrics, and an overview of conducted experiments. These elements collectively demonstrate *StrategicAI*'s performance, usability, and innovative tree-based methodology, adhering to DSR's iterative build-and-evaluate cycle and the IS framework's contextual grounding.

4.3.1 Dataset Preparation

The evaluation leverages six case studies, detailed in Table 4, representing diverse industries such as chemicals, manufacturing, and diagnostics. These cases, sourced from reputable academic collections (e.g., Darden Casebook, MIT Sloan), provide rich, realistic scenarios with predefined problems and solutions as ground truth. Each case includes a company background, task description, and internal data formatted as `data.txt` files, enabling consistent input across evaluation methods. This dataset preparation aligns with DSR Guideline

5: *Research Rigor*, ensuring a robust, replicable foundation for testing *StrategicAI*'s utility across varied organizational contexts, as emphasized by the IS framework.

Case ID	Type	Industry	Source
CavalierChem (Sticky Surfactants)	Profitability	Chemicals	Darden Casebook (2021–2022)[1]
Whizzy Wilco (A Golden Ticket)	Profitability	Manufacturing	Darden Casebook (2021–2022)[1]
Canyon Capital Partners	Profitability	Private Equity	Darden Casebook (2021–2022)[1]
Papyr Co	Profitability	Industrial Goods	Duke MBA Consulting Club Casebook[2]
Tres Burritos	Profitability	Restaurant	NYU Stern School of Business[3]
Quest Diagnostics	Operations Management	Diagnostic Laboratories	MIT Sloan[4]

[1] Darden School of Business Case Collection, University of Virginia (2021–2022), https://drive.google.com/file/d/1HrzPiMKPZjdc-yeFe1nH8fMaH1L1C2QW/view

[2] Duke University Fuqua School of Business, MBA Consulting Club Casebook, https://drive.google.com/file/d/1KX2pxkQdWSVcT_UNEepSxc2mk3xYCIrh/view

[3] New York University, Stern School of Business Case Repository, https://drive.google.com/file/d/1MKhqj27wTZ6u3bVHdw_PvLENAkxYdAEy/view

[4] Massachusetts Institute of Technology, Sloan School of Management, https://mitsloan.mit.edu/teaching-resources-library/quest-diagnostics-a-improving-performance-call-centers

Table 4 Case Study Information

4.3.2 Automated Evaluation Setup

To rigorously assess *StrategicAI*'s automated performance (DSR Guideline 3), we designed an experimental setup comparing its outputs to a baseline model, ChatGPT, both powered by GPT-4o via OpenAI APIs. This isolates the efficacy of *StrategicAI*'s tree decomposition methodology—an innovative construct per DSR Guideline 1: *Design as an Artifact*—while controlling for LLM variance. The evaluation focuses on the "LLM-Alone" mode, where *StrategicAI* autonomously generates root causes and solutions via its AutoRun feature, testing a fully implemented instantiation.

For each case study, the process entails:

1. **Company Profile Creation**: Embedding the company background in *StrategicAI*.
2. **Task Definition**: Specifying a "problem" task using the case description.
3. **Data Upload**: Processing internal data via the file fact extraction feature.
4. **AutoRun Execution**: Identifying root causes by automatic "Why Tree" building and building "How Trees," generating solutions, and validating them automatically using "Hypothesis tree".
5. **Output Export**: Exporting root causes and solutions with explanations in JSON format.

ChatGPT receives identical inputs and is prompted to produce analogous outputs, termed "LLM-Tree" for *StrategicAI* to highlight its tree-based approach versus ChatGPT's standard LLM methodology. This direct comparison supports DSR Guideline 4 by showcasing the novelty of tree-based decision-making, contributing to the IS knowledge base.

4.3.3 User Testing Setup

Given *StrategicAI*'s human-in-the-loop capability, a user testing session evaluated its "Human-LLM Collaboration" mode, further fulfilling DSR Guideline 3's evaluation mandate. Six participants with diverse backgrounds (e.g., computer science, medicine, MBA), were each assigned a unique case study (see Table 5 for participant details). They performed root cause analysis (RCA) and solution exploration collaboratively with the LLM, testing usability and practical utility in a human-operated context, aligning with DSR's focus on real-world applicability.

Location	Age	Education	Experience	Case Worked On
Oman	27	BSc Computer Science	Full-Stack Development	Quest Diagnostics
Germany	24	MSc in Data Engineering and Analytics	3D Computer Vision for Autonomous Driving and Robotics	Papyr Co.
Germany	24	MSc in Data Engineering and Analytics	Medical AI	Whizzy Wilco
Canada	28	MEng Entrepreneurship Innovation	Sales & Project Engineer	CavalierChem
USA	29	Doctor of Medicine	Psychiatry Resident	Tres Burritos
USA	34	MBA	Senior Project Development Manager	Canyon Capital Partners

Table 5 Participant Information

Post-session, participants completed a survey including:

- **System Usability Scale (SUS)**: Quantifying usability (Brooke, 1996).
- **Feature-Specific Questions**: Gathering qualitative feedback on the main feature's clarity and effectiveness.

This dual approach—automated and user-based—embodies the IS framework's iterative cycle, assessing *StrategicAI*'s technical accuracy and user experience (Hevner et al., 2004). It demonstrates feasibility (Guideline 1) and relevance (Guideline 2), setting the stage for robust evaluation metrics.

4.3.4 Evaluation Metrics

To assess *StrategicAI*'s performance, we developed custom metrics—Problem Identification Score (PIS) and Solution Generation Score (SGS)—aligned with DSR Guideline 3's call for rigorous, well-defined methods (Hevner et al., 2004). These metrics evaluate root causes and solutions against ground truth, incorporating:

- **F1 Score**: Harmonic mean of precision and recall for accuracy.
- **Relevance Score (RS)**: Cosine similarity of embeddings (via `all-MiniLM-L6-v2` using Hugginface) for semantic alignment. This is the maximum cosine similarity for each root cause identified with respect to the ground truth problems and the same goes for the solutions.
- **Explanation Relevance**: Cosine similarity of explanation embeddings for interoperability. This is the maximum cosine similarity for each root cause explanation identified with respect to the ground truth problems explanations and the same goes for the solutions

The final score is a weighted sum:

$$\text{Final Score} = 0.35 \times \text{F1} + 0.3 \times \text{RS} + 0.35 \times \text{Explanation Relevance} \quad (1)$$

This balances accuracy and explanatory quality, reflecting DSR's multi-faceted evaluation requirement. Qualitative feedback from user testing, including SUS scores, complements these metrics, ensuring practical utility per the IS framework.

4.3.5 Overview of Experiments

The evaluation comprises five experiments, briefly summarized below, each testing distinct aspects of *StrategicAI*'s performance and aligning with DSR's iterative process (Guideline 6: *Design as a Search Process*) (Hevner et al., 2004):

1. **Single-Agent AutoRun Evaluation**: Tested the initial AutoRun feature's baseline performance against ChatGPT across six case studies, using PIS and SGS. It established foundational efficacy, identifying areas for refinement (e.g., reliability and robustness of the outputs).
2. **Multi-Agent AutoRun Evaluation**: Assessed an enhanced AutoRun feature with multi-agent refinements, improving PIS and SGS, demonstrating iterative improvement per DSR Guideline 6.
3. **Supplementary Evaluation: Recall-Based Metric**: Analyzed *StrategicAI*'s tree structures (Why and How Trees) for recall, revealing comprehensive ground truth capture, enhancing utility insights (Guideline 3).
4. **LLM Preference Evaluation**: Compared *StrategicAI* and ChatGPT outputs using LLM judges across plausibility, faithfulness, comprehensiveness, and non-redundancy.
5. **User Testing Session for Human-LLM Collaboration**: Evaluated the collaborative mode with participants.

These experiments collectively substantiate *StrategicAI*'s utility, quality, and efficacy, fulfilling DSR Guideline 3 and advancing the IS field through rigorous, innovative evaluation (Guideline 4). The dual automated-user approach reflects the IS framework's contextual interplay, ensuring relevance to managerial needs and scholarly contribution.

4.3.6 Alignment with DSR Guidelines and IS Framework

- **Guideline 1: Design as an Artifact**: *StrategicAI* is a purposeful instantiation, rigorously tested in automated and collaborative modes.
- **Guideline 2: Problem Relevance**: The evaluation targets a critical IS challenge—strategic decision-making—validated across case studies from top management schools.
- **Guideline 3: Design Evaluation**: Diverse methods (experiments, metrics, user testing) demonstrate utility and quality.
- **Guideline 4: Research Contributions**: Novel LLM-based web application for strategic decision-making in the business environments.
- **Guideline 5: Research Rigor**: Robust dataset and methodologies from LLMs literature and management blogs to ensure credibility.
- **Guideline 6: Design as a Search Process**: Iterative experiments refine *StrategicAI*'s performance.
- **Guideline 7: Communication of Research**: Results are presented for technical and managerial audiences, as per this thesis.

The IS framework's contextual grounding is evident in case study selection and user testing, while the iterative cycle is mirrored in experiment progression, ensuring a balance of rigor and relevance.

5 An LLM-Based Decision Support System: StrategicAI

5.1 Introduction to StrategicAI

Strategic decision-making is essential for businesses to succeed, but it is often a difficult task for managers and CEOs. Research shows that decisions based on data lead to better results (McAfee & Brynjolfsson, 2012). However, even though companies have access to plenty of data, turning it into useful strategies remains a challenge. Large Language Models (LLMs) have shown great skill in understanding and processing information (Brown et al., 2020). Yet, their use in helping with big, long-term business decisions is still limited. This gap is what inspired the creation of StrategicAI.

StrategicAI is a web-based new Decision Support System (DSS) designed to assist business leaders by using LLMs in a practical way. Unlike older tools that mainly handle numbers or follow set patterns, StrategicAI tackles the real, everyday problems managers face—like understanding why profits are falling or deciding if a new plan will work. The goal was to build something that makes tough decisions easier and faster. It takes the power of LLMs and applies it to the kind of strategic thinking businesses need, offering a fresh approach to solving problems.

5.2 System Overview

Following the introduction's focus on StrategicAI as a tool to turn data into actionable strategies using Large Language Models (LLMs), this subsection outlines how the system operates in practice. StrategicAI is a web-based Decision Support System (DSS) designed to assist managers with strategic decision-making. It guides users through a structured process that begins with setting up a company profile and moves into creating tasks to address problems, goals, or hypotheses. This approach ensures that analyses and solutions are customized to each company's context, making decision-making both practical and effective.

The process starts with user registration. Managers sign up with an email and password and then log in. If it's their first login, StrategicAI directs them to create a company profile, a step illustrated in Figure 6. This flowchart shows how the system collects basic details—company name, industry, and country—and then conducts an online search to gather more information about the company. If insufficient data is found online, the system prompts users with simple questions about their business, such as its operations, profits, or competitors. Users can skip these questions and proceed directly to the homepage. Later, they can refine the profile by adding text or uploading files like internal reports. The LLM processes these inputs—extracting key details from text and files—and updates the company profile accordingly. This profile is crucial, as it informs all subsequent analyses, ensuring decisions reflect the company's specific situation.

The core of StrategicAI is its task feature, where decision-making unfolds. Tasks are central entities that capture a problem (e.g., "Why are sales dropping?"), a goal (e.g., "Grow revenue by 10%"), or a hypothesis (e.g., "Will a new product work?"). To begin, users create a new task by opening a dialog shown in Figure 7. This screenshot displays a text area—already filled with an example like "My profits are dropping and I don't know why"—where users describe their concern. The dialog also includes a flag to include company profile data (useful for company-specific issues but optional for general questions) and a task type selection. Task types define the analysis path and include:

- **Problem-Based Tasks**: For company challenges, like declining profits. These involve three stages: a Diagnostic Issue Tree ("Why Tree") to identify root causes (e.g., poor marketing), a Solution Issue Tree ("How Tree") to explore solutions (e.g., increase advertising), and a Hypothesis Tree to validate solutions (e.g., "Will ads boost sales?").

- **Goal-Based Tasks**: For objectives, like revenue growth. These start with a How Tree to brainstorm strategies, followed by a Hypothesis Tree to test them.

- **Hypothesis-Based Tasks**: For yes/no questions about a known solution (e.g., "Will this solution work?" or even simple questions out of the company context like "Are Tomatoes considered fruits"), using only Hypothesis Tree.

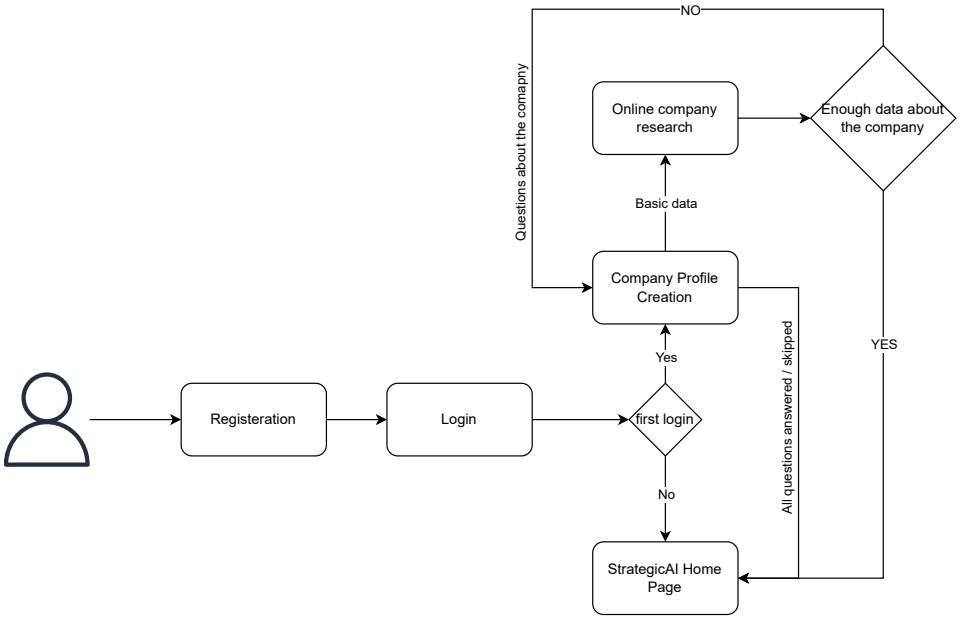

Figure 6 Flowchart of the company creation process, from registration and login to profile setup with basic data, online search, and optional questions, leading to the homepage.

If users are unsure of the task type, they can select "I don't know," and the LLM recommends one—e.g., "This seems like a problem task because you're asking why"—with a certainty score (e.g., 85%) and an explanation.

After creating a task, users arrive at the task details page displaying the task description and the results—like root causes, solutions, or answers—derived from the trees within the task. Initially, this page would contain only the task description entered by the user see Appendix 27 These trees drive StrategicAI's analysis: Why Trees pinpoint problem origins, How Trees map out solutions, and Hypothesis Trees where the hypothesis testing is inspired by ProbTree's answering techniques (Cao et al., 2023), confirm solutions with yes/no logic. This structured progression turns vague issues into clear outcomes.

StrategicAI adapts to user preferences with three operational modes:

- **Human-Alone**: For experts who build trees manually, step-by-step, without LLM assistance—similar to working on papers.
- **LLM-Alone**: An AutoRun feature takes over, constructing trees, searching online, integrating company data, and selecting the task outcomes while users observe the process live.
- **Human-LLM Collaboration**: The preferred mode, where users and the LLM collaborate—co-building trees, retrieving data from upload files or online sources, getting potential candidates for root causes and solutions, and receiving next-step suggestions and asking/answering questions through a Chatbot and finally making the final decision would fall on the human since humans are better in judgment.

This flexibility suits everyone, from consultants to novices. Tasks can range from complex company issues ("Why are profits down?") to simple queries ("Are tomatoes fruits?"), with the company data flag allowing analysis tied to—or independent of—the profile, as shown in the task creation dialog (Figure 7).

5.3 System Architecture

The system architecture of StrategicAI is a modular, scalable framework engineered to provide decision-support capabilities for strategic business challenges. It consists of three primary components: a backend web service, a frontend interface, and a MongoDB database. The backend service is seamlessly integrated with external services for language modeling and information retrieval. A defining characteristic of this architecture is that all

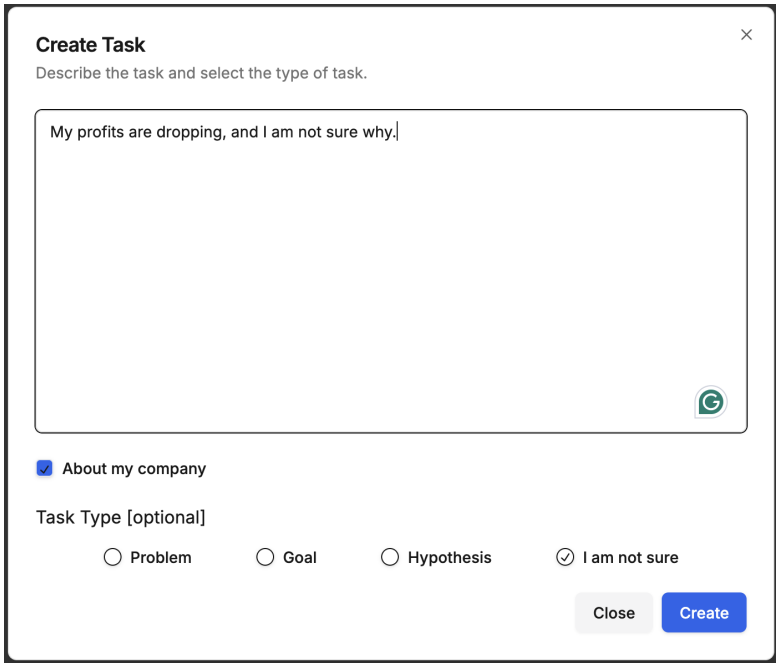

Figure 7 Screenshot of the task creation dialog, showing a text area with "My profits are dropping and I don't know why," a flag for including company information, and task type selection.

components are **Dockerized**, operating as containers to streamline deployment and ensure consistency across diverse environments. This subsection elaborates on each component, and their interactions (illustrated in Figure 8), with the database schema detailed in Figure 9. Additionally, the web search and retrieval systems are designed with flexibility in mind, enabling straightforward integration with various providers and methods to adapt to emerging technologies.

5.3.1 Dockerization Overview

Docker, a containerization platform, encapsulates applications and their dependencies into lightweight, portable containers. In StrategicAI, the backend, frontend, MongoDB, and Mongo Express (for database management) are each Dockerized, running as isolated containers that can be built and executed on any Docker-compatible system. This approach delivers several advantages:

- **Easy Setup**: The entire system can be deployed with a single command (e.g., `docker-compose up`), removing the need for intricate configuration. We added a shell script as well that would run and deploy the app based on the environment for development and production: `./start.sh dev|prod`
- **Consistency**: Containers guarantee identical behavior across development, testing, and production environments, regardless of the host system.
- **Portability**: StrategicAI operates on any Docker-supported platform, including Windows, macOS, and Linux, broadening its accessibility.
- **Scalability**: Docker enables scaling of individual components (e.g., adding backend instances) to meet increased demand.

By harnessing Docker, StrategicAI ensures that developers and business users alike can effortlessly set up and run the system, enhancing its practicality and adoption potential.

5.3.2 Backend Web Service

The backend web service, implemented using FastAPI, serves as the central processing hub of the system, orchestrating task management and facilitating integration with external services. FastAPI was selected for its

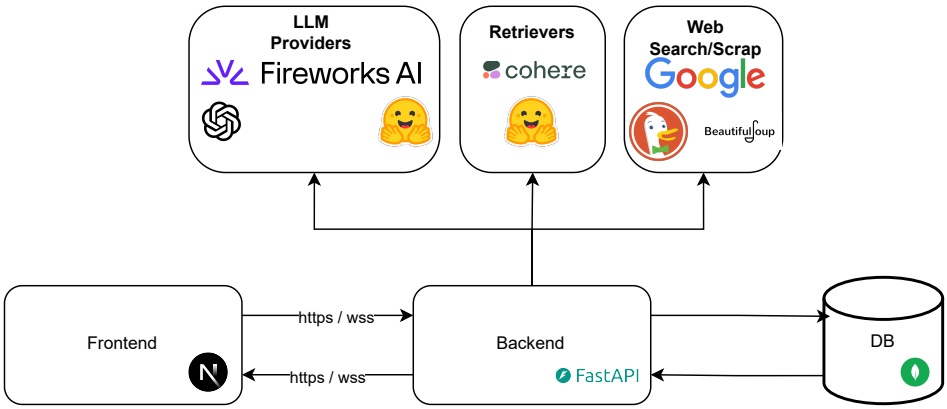

Figure 8 Overall System Architecture

asynchronous programming capabilities, seamless compatibility with Python-based natural language processing (NLP) libraries, and robust data validation features. The backend is encapsulated within a Docker container, bundling all requisite dependencies (e.g., Python, LLM APIs), which ensures portability and consistency across deployment environments.

A key feature of the backend is its adoption of WebSockets, enabling real-time, bidirectional communication with the client application. This functionality supports several critical operations:

- **Notifications**: Instantaneous updates, such as "Task analysis started," inform users of ongoing background processes, enhancing situational awareness.
- **Tree Updates**: Modifications to the Why, How, or Hypothesis Trees are reflected in the client application in real time, ensuring data consistency.
- **Transparency**: Messages like "Fetching competitor data" provide visibility into backend operations, fostering user trust in AI-driven insights.

This real-time communication paradigm reduces server load compared to traditional polling mechanisms, supports collaborative workflows, and improves the overall user experience. These enhancements directly address the research question of building trust with users by transitioning from an opaque "black box" model to a transparent "open box" system, where operational processes are exposed and comprehensible.

The backend integrates a diverse array of external services to support its functionality, with a modular design that ensures flexibility and tunability. For large language model (LLM) interactions, the system leverages multiple APIs:

- **OpenAI APIs**: These provide access to models such as GPT-4o and GPT-4o-mini, with API keys securely stored in an `.env` file within the backend environment.
- **Fireworks API**: This service supports a variety of LLMs, including LLaMA, Qwen, and DeepSeek, broadening the range of available models.

The selection of an LLM for a given task is determined by the client application, which specifies the model name in the request headers via the LLM Config dialog. The backend interprets this header at the start of each LLM-dependent call, dynamically routing the request to the appropriate service (e.g., OpenAI or Fireworks). If no model is specified, the system defaults to GPT-4o-mini, balancing performance and cost-effectiveness. This user-driven configurability enhances the system's adaptability to diverse use cases.

For search capabilities, the backend incorporates multiple APIs:

- **Google API**: A paid service offering comprehensive web search functionality.

- **DuckDuckGo API** (via LangChain): A free alternative prioritized as the default due to its cost efficiency and privacy-focused design.

Web scraping is facilitated using BeautifulSoup, enabling the extraction of structured data from web pages. Search results and extracted facts are further processed using Cohere, which provides semantic search capabilities and reranking of results. Cohere's embedding models are employed to enhance the relevance of retrieved information, making it the default choice for semantic search due to its superior performance in embedding quality.

Additionally, the backend integrates Hugging Face embedding models for fact retrieval from processed documents. These embeddings support efficient similarity searches, complementing Cohere's capabilities. The modular architecture ensures that these services—OpenAI, Fireworks, Google, DuckDuckGo, Cohere, and Hugging Face—are easily interchangeable. Default configurations (e.g., DuckDuckGo for search, Cohere for semantic search, GPT-4o-mini for LLM tasks) are employed when the client does not specify preferences, while backend tunability allows administrators to adjust these settings as needed. The exception is the LLM selection, which remains user-controlled via client headers, aligning with the system's emphasis on user agency.

This modular, service-oriented design not only supports scalability and maintainability but also aligns with the research objective of enhancing transparency and trust. By exposing configurable options and real-time feedback, the backend empowers users to tailor the system to their needs while maintaining visibility into its operations.

5.3.3 Frontend Implementation

The frontend of the application is developed using NextJS and TypeScript, providing a robust, efficient, and contemporary interface for user interaction. NextJS enhances performance through its server-side rendering capabilities, enabling faster page loads and improved search engine optimization. TypeScript complements this framework by introducing static typing, which enhances code reliability and maintainability by catching errors during development. The user interface is styled using TailwindCSS and ShadCN, offering a modern, responsive design with utility-first styling and pre-built components tailored for accessibility and aesthetics.

The frontend application, along with its dependencies (e.g., Node.js), is containerized using Docker. This approach ensures consistency across development and production environments and facilitates rapid deployment on any Docker-enabled system. Communication with the backend is achieved through RESTful APIs and WebSockets, enabling seamless data exchange and real-time features such as notifications and dynamic updates to the tree structure.

A critical component of the client-side application is the tree rendering and construction functionality, implemented using the ReactFlow library. ReactFlow is a highly customizable package designed for building interactive graph-based interfaces. It provides extensive support for styling, node and edge management, and dynamic connections, making it an ideal choice for visualizing and manipulating tree structures within the application. The library's flexibility allows for tailored representations of nodes and edges, coupled with efficient state management to handle user interactions.

State management within the application is handled using Redux, a predictable state container for JavaScript applications. Redux centralizes the application's state, ensuring a single source of truth that simplifies debugging and testing. It employs a unidirectional data flow, where state changes are dispatched as actions and processed by reducers, resulting in a scalable and maintainable architecture suitable for complex, interactive applications like this one.

For backend communication, the application utilizes Axios, a promise-based HTTP client for JavaScript. Axios simplifies API requests by providing an intuitive syntax, automatic transformation of JSON data, and robust error handling. To enhance functionality, custom interceptors have been integrated into Axios calls. These interceptors append metadata about the user-specified Large Language Model (LLM), configured via the LLM Config dialog within the application. This feature enables the frontend to dynamically route requests to the appropriate LLM supported by the system, offering users flexibility in processing tasks without altering the core request logic.

This combination of technologies—NextJS, TypeScript, ReactFlow, Redux, and Axios—results in a high-performance, modular, and user-centric frontend, seamlessly integrated with the backend to deliver a cohesive experience.

5.3.4 Database

The database is a cornerstone of StrategicAI, tasked with storing and managing data related to companies, tasks, trees, nodes, and analysis and decision outcomes. MongoDB, a flexible NoSQL database, is employed for its ability to accommodate varied data structures and relationships. The database schema, depicted in Figure 9, supports the system's operations while permitting dynamic field additions as needed. As MongoDB does not enforce a fixed schema, many fields are included to facilitate data retrieval, making the design adaptable and efficient.

The Entity Relationship Diagram (ERD) in Figure 9 is self-explanatory, but below we clarify key collections and specific fields, along with their purposes:

- **Company Collection**:
  - **Profile**: An object storing basic company details, such as name, industry, and country. It includes a `dynamic_field`, generated by the LLM to capture additional data discovered when updating the company profile via text or files uploaded in the company profile. This field's fully dynamic nature allows for flexible, on-the-fly data storage.
- **Tree Collection**:
  - **Payload**: A string containing the company information and task description, serving as the contextual foundation for tree construction. Depending on the tree type, it may include specific details—e.g., the root cause for "how" trees or the solution being validated for "hypothesis" trees—enhancing its utility in prompts.
  - **Related Node ID**: An identifier linking the tree to the node from which it was derived. For "why" trees, this field is null, as they are the initial tree type. For "how" and "hypothesis" trees, it references nodes in "why" or "how" trees, respectively. A corresponding field in the Node Collection enables rapid bidirectional navigation between trees and nodes.
- **Node Collection**:
  - **Node Types**: In "how" and "why" trees, nodes are classified as "root" or "normal." In "hypothesis" trees, they are "rhypo" (root hypothesis) or "hypo" (normal hypothesis question node). This distinction accommodates ReactFlow's rendering requirements, allowing different options and styles for hypothesis nodes.
  - **Is Part of Decision**: A boolean flag highlighting nodes integral to the tree's final decision (e.g., root causes or valid solutions), used to enhance UI/UX by visually distinguishing these nodes.
  - **Required Data**: A list of data points necessary to validate a node, exclusive to hypothesis tree nodes, supporting their validation process.
  - **Main Fields**:
    - **Text**: The node's primary label, varying by tree type: a potential cause in "why" trees, a potential solution in "how" trees, or a yes/no question in "hypothesis" trees.
    - **Explanation**: In "why" and "how" trees, this justifies the node's placement (e.g., why it's a cause or solution). In "hypothesis" trees, it answers the yes/no question in the text field.
    - **Certainty**: A percentage reflecting confidence in the node's position and explanation, sourced directly from LLM outputs.
- **Tree Analysis Collection**:

  - **History**: A list of objects tracking all actions (e.g., decomposing, adding, editing, or deleting nodes, online searches, or file lookups) and the Chatbot that we call it (Collaboration Agent) interactions (user questions and responses). This history renders and preserves the chat log for a specific tree analysis.

- **Tree Decisions Collection**:
  - Stores the conclusive outcomes of tree analyses: identified root causes for "why" trees, selected solutions for "how" trees, and the final answer for "hypothesis" trees. Represented as a list of objects, typically nodes with explanations for "why" and "how" trees, or the definitive answer for "hypothesis" trees.

The database supports real-time updates and interactions via WebSockets in the backend. As a Docker container, MongoDB ensures consistent data management across environments. Mongo Express, also Dockerized, offers a web-based interface for administrative tasks like viewing or updating records.

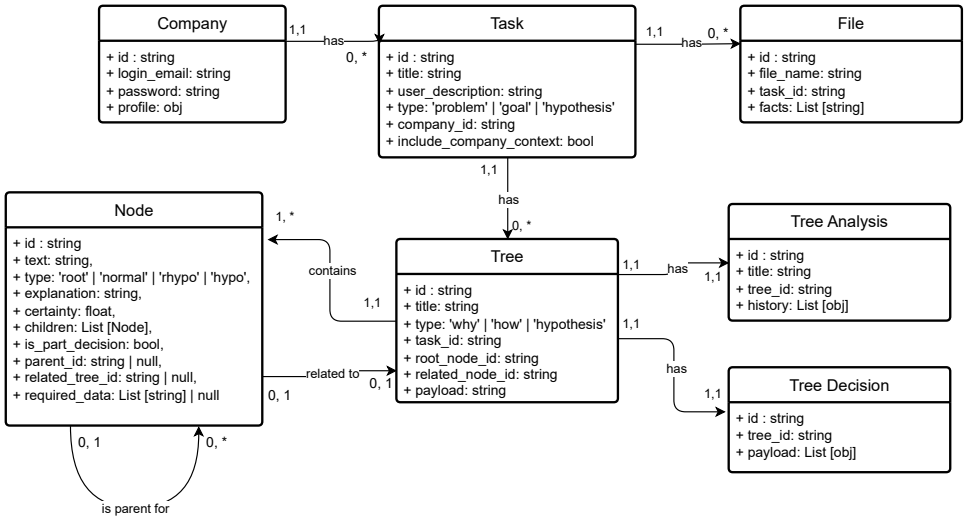

Figure 9 Database Schema

5.3.5 LLM Integration

StrategicAI adopts an LLM-agnostic architecture, enabling the seamless integration of large language models (LLMs) from multiple providers, including OpenAI and Fireworks. These dependencies are encapsulated within the backend's Docker container, ensuring portability and facilitating dynamic model switching based on task-specific requirements. LLM outputs—such as analysis results or updates to the Why, How, or Hypothesis Trees—are transmitted to the client application in real time via WebSockets, providing users with immediate decision-making support. This real-time delivery enhances the system's responsiveness and aligns with its goal of offering actionable insights efficiently.

As of the submission of this thesis, StrategicAI supports a diverse set of LLMs, detailed in Table 6. The table lists each model alongside its provider and, where applicable, the associated cost per million tokens, reflecting the pricing structure of the Fireworks API. For OpenAI models (e.g., GPT-4o, GPT-4o-mini), specifying the model name in the frontend's LLM Config dialog suffices to invoke the corresponding service, leveraging the OpenAI API. Similarly, Fireworks-hosted models (e.g., DeepSeek, LLaMA, Qwen) are identified by their unique identifiers, which are passed from the client application to the backend via request headers. This design

ensures flexibility, allowing users to select the most suitable model for their needs without requiring backend reconfiguration.

Provider	Model
OpenAI	GPT-4o
OpenAI	GPT-4o-mini
DeepSeek (Fireworks)	DeepSeek R1
DeepSeek (Fireworks)	DeepSeek V3
LLaMA (Fireworks)	LLaMA-3.3-70B-Instruct
LLaMA (Fireworks)	LLaMA-3.2-3B-Instruct
LLaMA (Fireworks)	LLaMA-3.1-8B-Instruct
Qwen (Fireworks)	Qwen-2.5-72B-Instruct

Table 6 Supported Large Language Models as of Thesis Submission

To ensure consistency and maintainability, the backend implements a service-oriented architecture for LLM integration. Each provider (e.g., OpenAI, Fireworks) is supported by a dedicated service that inherits from a base LLM service class. This base class defines a standardized interface, mandating a uniform structure across all provider-specific implementations. This abstraction enables the system to handle model invocations agnostically, routing requests to the appropriate service based on the model name specified in the client headers. The modular design not only simplifies the addition of new LLMs but also ensures that the system remains adaptable to future advancements in language model technology.

The LLM-agnostic approach, combined with real-time WebSocket communication, underscores StrategicAI's commitment to flexibility and user-centric design. By supporting a broad range of models (as shown in Table 6) and abstracting their integration through a unified service framework, the system empowers users to tailor its capabilities to specific tasks while maintaining operational efficiency and scalability.

5.3.6 Search and Retrieval Systems

The search and retrieval systems in StrategicAI are designed with modularity and flexibility at their core, enabling seamless substitution of providers or methods to adapt to evolving technological landscapes. Currently, the system integrates two primary search services: the Google API and DuckDuckGo, the latter accessed via LangChain. These services inherit from a base search service class, allowing the search engine to be dynamically selected based on configuration settings. By default, DuckDuckGo is employed due to its cost-free nature, whereas the Google API, despite offering robust search capabilities, is constrained by a free-tier limit of 100 queries per day. Search queries yield a list of snippets, URLs, and titles, forming the initial dataset for further processing.

The retrieval pipeline, a critical component for Retrieval Augmented Generation (RAG), begins with refining these search results. Cohere's reranking functionality is utilized to prioritize relevance, leveraging its semantic understanding to reorder the list. The top three results are then selected for downstream processing, as illustrated in Figure 10. These results are passed to a web scraper service, implemented using BeautifulSoup, which extracts the HTML content from the corresponding web pages. The scraped content is subsequently processed by a fact extractor agent—discussed in detail later in this thesis—which decomposes the text into atomic facts.

The retrieval system then employs a similarity-based approach to identify the most pertinent facts in response to the user's query. This system is backend-agnostic and supports multiple embedding models to capture semantic meaning, outperforming traditional keyword-based methods (Reimers & Gurevych, 2019). Two primary embedding strategies are implemented:

- **Cohere Embedding Model**: The default configuration utilizes Cohere's `embed-english-v3.0` model, which generates embeddings for both the query and extracted facts. Parameters are tuned to ensure embeddings reflect not only contextual similarity but also the factual relevance to the query. Cosine similarity is calculated between the query and fact embeddings, with facts exceeding a default threshold

of 0.5 returned as results. This model's fine-tuned design enhances its ability to discern nuanced semantic relationships.

- **Hugging Face Embedding Model**: An alternative approach employs the `sentence-transformers/all-MiniLM-L6-v2` model from Hugging Face. This model, while effective for word-level and syntactic similarity, is less specialized than Cohere's offering. Consequently, a higher cosine similarity threshold of 0.85 is applied to ensure precision in fact retrieval. This model serves as a fallback option, with Cohere preferred as the default due to its superior semantic performance.

The complete search and retrieval workflow—from query submission to fact ranking—is depicted in Figure 10, highlighting the interplay between search, reranking, scraping, and retrieval stages.

For scenarios involving pre-existing data, such as facts extracted from files stored in the database (processed by the fact extractor agent upon upload), the pipeline bypasses the search and scraping phases. Instead, it directly invokes the similarity-based retrieval service, applying the same embedding and thresholding techniques to retrieve relevant facts. This dual-purpose design ensures the retrieval system is versatile, supporting both real-time web-based queries and database-driven fact retrieval.

The modular architecture of the search and retrieval systems, coupled with their reliance on advanced embedding models and reranking tools, ensures high adaptability and precision. By defaulting to cost-effective and semantically robust options (e.g., DuckDuckGo and Cohere), the system balances efficiency and performance while remaining configurable to meet diverse operational needs.

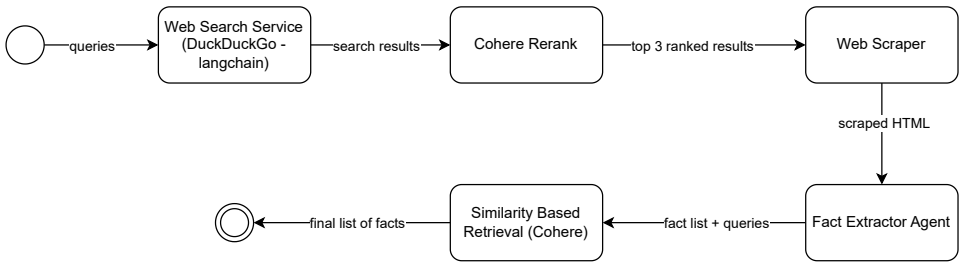

Figure 10 Workflow of the Search and Retrieval Process

5.3.7 Modular Design

The architecture of StrategicAI is built on a modular framework, allowing individual components—such as large language models (LLMs), search engines, and retrieval systems—to be independently developed, tested, and upgraded. This design ensures the system remains flexible and responsive to technological advancements and changing business requirements. For example, the LLM integration is abstracted through a service-oriented architecture, enabling seamless switching between providers like OpenAI and Fireworks. This flexibility means that new LLMs can be incorporated by simply adding a new service class that adheres to the base LLM interface, future-proofing the system against emerging models.

Benefits Beyond Adaptability

Beyond adaptability, modularity simplifies maintenance and debugging. Developers can work on individual components without needing to understand the entire codebase, speeding up development cycles and reducing errors. For users, this translates to a reliable system with minimal downtime during updates, as changes to one module do not affect others. Security is also enhanced: Docker's containerization limits the impact of potential breaches. If one module is compromised, the isolation prevents lateral movement, protecting the system's integrity.

Practical Examples

Consider the integration of LLMs: StrategicAI supports multiple providers (e.g., OpenAI's GPT-4o and Fireworks' LLaMA models) through a modular service layer. Switching providers or adding a new one, like a hypothetical future model from xAI, requires no architectural overhaul—just a new service implementation.

Similarly, the search system can toggle between Google and DuckDuckGo APIs, with the ability to integrate new providers like Bing by updating the search service class.

Challenges and Solutions

Modularity introduces challenges, such as ensuring smooth communication between components. StrategicAI addresses this with well-defined APIs and WebSocket protocols, standardizing interactions and enabling real-time data exchange. Dependency management is another hurdle; Docker mitigates this by encapsulating each module's environment, ensuring consistency across deployments.

Future-Proofing and User Impact

The design prepares StrategicAI for future innovations, such as integrating advanced embedding models or new retrieval methods, without requiring significant rework. From a user perspective, while modularity operates behind the scenes, it ensures a system that evolves seamlessly—delivering new features and improved performance without disrupting workflows.

In summary, StrategicAI's modular architecture, powered by Docker's containerization, provides a robust foundation for adaptability, scalability, maintenance, and security. This design ensures the system remains a cutting-edge tool for strategic decision-making.

5.4 Core Functionalities

StrategicAI represents a pioneering approach to decision support, leveraging the capabilities of Large Language Models (LLMs) to address the complex, strategic challenges business leaders face. Its core functionalities—Root Cause Analysis (RCA), Solution Exploration, and Solution Validation—are designed to transform raw data into actionable insights, guiding managers through a structured problem-solving and strategy formulation process. Central to these functionalities is the Fact Extraction Feature, which processes diverse data inputs into a suitable format for analysis. Each functionality operates within three execution flows—Human-Alone, LLM-Alone, and Human-LLM Collaboration—offering flexibility to accommodate varying levels of user expertise and preference. This subsection provides an in-depth exploration of these components, detailing their operational mechanisms, the roles of specialized agents, their alignment with established business frameworks, and the practical execution flows that drive their effectiveness. Figures illustrating key processes are referenced throughout to enhance comprehension.

5.4.1 Fact Extraction

The agent that is responsible for this feature is the Fact Extractor Agent, which serves as the foundational element of StrategicAI, enabling all subsequent analyses by converting user-uploaded files and scraped HTML from online search into a usable, data-driven format. When a user creates a task—whether problem-based (e.g., "Why are sales dropping?"), goal-based (e.g., "Grow revenue by 10%"), or hypothesis-based (e.g., "Will a new product work?")—they can upload files of any type, such as internal reports, financial statements, PDFs, or competitor analyses. The agent processes these files by first extracting their content as a string, then employing an LLM-powered process to generate a list of context-rich atomic facts. For example, a quarterly report stating "Profits declined by 15% due to supply chain disruptions" might yield facts like "Profit decline reached 15% in Q1" and "Supply chain issues impacted operations."

The use of atomic facts offers several advantages:

- **Efficient Retrieval**: Each fact is a standalone unit of meaning, eliminating the need for document segmentation. This allows the system to quickly retrieve relevant data during tree-based analyses, such as pinpointing "Supply chain issues" as a potential cause without parsing entire files.
- **Storage Simplicity**: By storing facts rather than full documents in the MongoDB database (as detailed in sub-section 9), StrategicAI optimizes storage efficiency, retaining only essential information. This is particularly beneficial for companies managing extensive datasets.
- **Contextual Clarity**: Unlike raw text excerpts, atomic facts are crafted to be self-explanatory, enabling users or agents to understand their significance without revisiting the source. For instance, "Company A has marketing budget cut by 20%" conveys immediate meaning compared to a vague sentence fragment.

In practice, the Fact Extractor Agent ensures that all uploaded files—whether internal company data or external industry reports—are seamlessly integrated into the decision-making pipeline. These facts are stored in the database and utilized across all functionalities, forming the evidence base for RCA's root cause identification, Solution Exploration strategy development, and Solution Validation hypothesis testing. This preprocessing step underscores StrategicAI's commitment to data-driven decision-making, bridging the gap between raw information and actionable strategies.

5.4.2 Root Cause Analysis

Root Cause Analysis (RCA) is a critical feature of the StrategicAI system, specifically designed to uncover the fundamental causes of challenges faced by organizations. The RCA process is initiated when a user defines a problem, which is categorized under the 'problem' task type within the system. The main goal of RCA is to identify the root causes of the defined problem using a structured approach. This approach involves the use of a Diagnostic Issue Tree, also known as a 'Why' Tree, which allows for a systematic investigation of potential causes. The process begins with the creation of a tree object, where the root node represents the primary problem to be analyzed.

Diagnostic Issue Tree Structure The Diagnostic Issue Tree is structured with a root node that poses a 'why' question, encapsulating the core problem being investigated. This tree is built hierarchically, where each node signifies a potential cause of the problem. Nodes can have child nodes, which represent sub-causes contributing to their parent cause. This hierarchical decomposition persists until the causes can no longer be subdivided, reaching what are termed atomic sub-causes. The primary aim is to pinpoint a subset of these nodes that constitute the root causes, supported by empirical evidence derived from data. Each node in the tree possesses specific attributes, including a label and an explanation. The explanation is particularly crucial as it provides a justification, backed by internal or external data, to determine whether the node should be considered a root cause.

The standard workflow for conducting RCA within StrategicAI involves a series of structured steps:

1. **Construction of the Diagnostic Issue Tree**: This initial step involves building the hierarchical tree structure, starting from the root node and decomposing it into potential causes and sub-causes.
2. **Data Collection**: Relevant data is gathered from various sources, which may include internal resources such as company files or external resources like online databases and websites. This data is used to assess the validity of each node in the tree.
3. **Updating Node Explanations**: The explanations associated with each node are refined and updated based on the insights gained from the collected data. This ensures that the rationale for each potential cause is grounded in empirical evidence.
4. **Selection of Root Causes**: Finally, nodes that have explanations supported by substantial evidence are identified and selected as the root causes of the problem.

Execution Flows for Root Cause Analysis To accommodate varying user needs and resource constraints, StrategicAI provides three distinct modes for executing the RCA process:

- **Human-Alone Mode**: This mode is designed for expert users who prefer to conduct the RCA independently, relying solely on their expertise.
- **Human-LLM Collaboration Mode**: This mode integrates the capabilities of Large Language Model (LLM)-powered agents to assist users throughout the RCA process, enhancing decision-making and efficiency.
- **LLM-Alone Mode (AutoRun)**: This fully automated mode allows the system to perform the entire RCA process without human intervention, providing rapid insights.

Each of these modes is tailored to different scenarios, ranging from complete human control to full automation.

Human-Alone Mode In the Human-Alone mode, the RCA process is entirely driven by the user's expertise. StrategicAI functions as a supportive platform, providing tools for constructing and manipulating the Diagnostic Issue Tree. Users can add, modify, or remove nodes as they see fit and ultimately select the root causes based on their professional judgment. This mode does not incorporate any algorithmic or AI-based assistance, relying exclusively on human knowledge and experience.

Human-LLM Collaboration Mode The Human-LLM Collaboration mode represents a hybrid approach, combining human expertise with the advanced capabilities of LLM-powered agents. This mode is central to StrategicAI's RCA functionality, as it leverages AI to augment human decision-making processes. The agents involved are specifically designed to support various aspects of RCA and are also utilized in the subsequent Solution Exploration phase. The primary features and their associated agents are detailed below.

Node Decomposition: This feature is essential for expanding the Diagnostic Issue Tree. The **Decomposition Agent** is responsible for taking a selected node, which represents a potential cause, and generating its immediate sub-causes, which are then added as child nodes in the tree. Employing a few-shot learning methodology, the agent iteratively decomposes the causes until they can no longer be broken down further. This systematic approach ensures a comprehensive exploration of all possible causes, often revealing sub-causes that might not be immediately apparent to human users.

Node Verification: This feature enhances the reliability of the RCA tree by validating the nodes and their associated sub-trees using evidence-based reasoning. It involves the collaboration of three distinct agents: the **Tree Builder Agent**, the **Researcher Agent**, and the **Fact Extractor Agent**. The verification process is initiated by the user through a dialog interface, as illustrated in Figure 11. In this interface, users can choose the data sources to be used for validation, which may include uploaded files, online resources, or a combination of both.

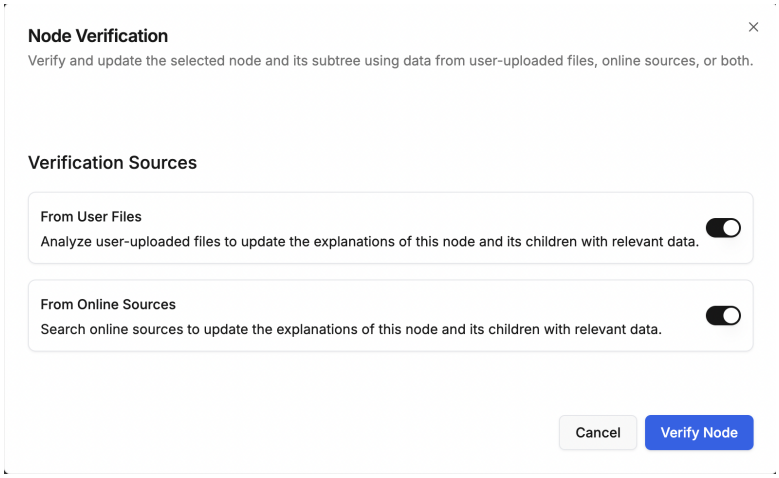

Figure 11 Screenshot of the node verification dialog, where users select data sources (uploaded files, online searches, or both) for validating nodes.

The **Researcher Agent** plays a pivotal role in gathering pertinent information from the selected data sources. When uploaded files are chosen, the agent utilizes the labels of the nodes as search queries to identify and retrieve facts that exhibit high semantic similarity to the node labels. These facts are then compiled into a context string, complete with references to the original files. In the case of online sources, the agent generates appropriate search queries and employs a search tool, such as DuckDuckGo, to retrieve relevant web pages. The search results are ranked using Cohere's ranking algorithm, and the content from the top three websites is scraped. Subsequently, the **Fact Extractor Agent** extracts key facts from this content. All the collected data, whether from files or online sources, is then forwarded to the **Tree Builder Agent**. This agent is responsible for refining the tree structure by adding new nodes, editing existing ones, or removing nodes that are not supported by the evidence.

Get Potential Candidates: This feature aids users by identifying nodes that are likely to be root causes, thereby focusing their attention on the most promising areas of the tree. The **Selector Agent** assesses all

nodes in the tree, along with their explanations, and prioritizes those that are supported by strong, data-driven evidence. To enhance the reliability of its selections, the agent utilizes a self-consistency mechanism. This involves performing the selection process across three separate iterations and then determining the intersection of the selected nodes from these iterations. The resulting set of nodes represents the most consistently identified potential root causes, guiding users to either verify these nodes further or directly designate them as root causes.

Collaboration Agent (Chatbot): Implemented as an interactive chatbot, the **Collaboration Agent** offers multifaceted support throughout the RCA process. It is accessible via a user-friendly chat interface, facilitating seamless interaction between the user and the AI system, in line with Human-in-the-Loop principles. The agent is capable of performing a variety of functions, including:

- Answering user queries related to the current state of the Diagnostic Issue Tree. - Recommending specific actions, such as adding, editing, or deleting nodes, to improve the tree's structure and accuracy. - Executing approved actions on the tree, similar to the capabilities of the Tree Builder Agent. - Generating and executing search queries to retrieve additional information. - Suggesting relevant data files for upload to enhance the evidence base.

Moreover, the Collaboration Agent provides strategic guidance by identifying nodes that require further evidence or verification. It can also respond to open-ended questions posed by the user, such as inquiries about potential root causes, drawing upon the current state of the tree and the available data.

LLM-Alone Mode (AutoRun) The LLM-Alone mode, also referred to as AutoRun, offers a fully automated approach to RCA, ideal for users who require quick insights without manual intervention. In this mode, a series of agents operate in a predefined, deterministic sequence to complete the entire RCA process. The workflow is illustrated in Figure 12 and consists of the following steps:

1. **Tree Construction**: The Decomposition Agent initiates the process by constructing an initial Diagnostic Issue Tree. This involves decomposing the root node into two layers of sub-causes, creating a preliminary structure for further analysis.

2. **Tree Adjustment**: The Tree Builder Agent then refines this initial tree by incorporating specific information about the company and the context of the task. This step may involve modifying existing nodes, adding new ones, or removing irrelevant ones to better align the tree with the available contextual data.

3. **Tree Verification**: The verification process is applied to the entire tree, starting from the root node. This step utilizes the Node Verification feature, which integrates data from both user-uploaded files and online sources. The Researcher Agent and Fact Extractor Agent are employed to gather and extract relevant information, which is then used by the Tree Builder Agent to update the tree accordingly.

4. **Root Cause Selection**: Finally, the Selector Agent identifies the most promising nodes as root causes. Using a self-consistency approach, the agent performs multiple iterations of selection and designates the intersecting set of nodes as the final root causes for the automated RCA process.

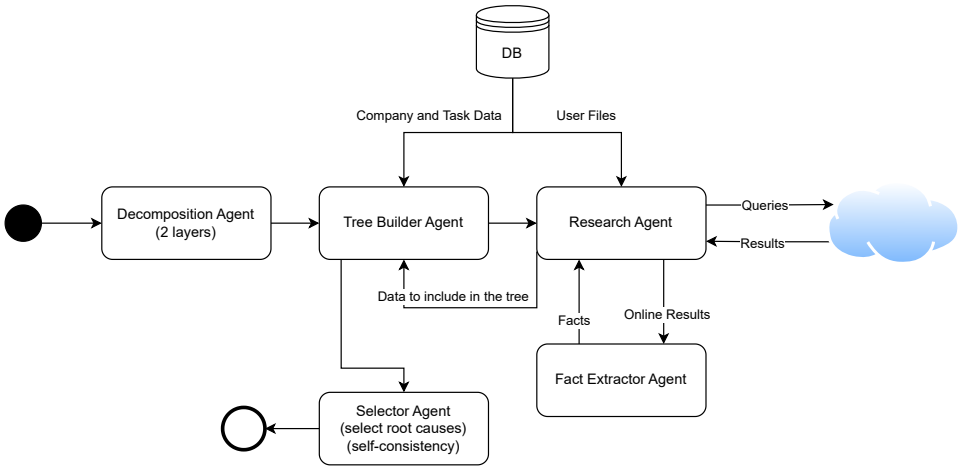

Figure 12 Flowchart of the Root Cause Analysis process using AutoRun, detailing tree construction, data retrieval, explanation updates, and root cause selection.

Selection Criteria for Root Causes In all three execution modes, the selection of root causes is fundamentally based on the quality and robustness of the explanations associated with each node. Nodes that possess explanations grounded in data-driven evidence, which can be verified through internal or external sources, are given priority. In situations where direct evidence is lacking, the Selector Agent leverages contextual knowledge about the company, the specific task, and the current state of the tree to identify nodes with high potential to be root causes. This approach ensures that the outcomes of the RCA process are both actionable and firmly rooted in the available data and context.

Conclusion Upon the completion of the RCA process, the identified root causes, represented by the selected nodes in the Diagnostic Issue Tree, form the basis for the next phase: Solution Exploration. In this subsequent phase, 'How' Trees are developed for each root cause to explore potential solutions. The integration of human expertise with LLM-powered agents in StrategicAI facilitates a robust, adaptable, and efficient RCA process. This hybrid approach caters to a wide range of user requirements and time constraints, ensuring that the analysis is both thorough and timely.

5.4.3 Solution Exploration

Solution Exploration represents the second major feature of StrategicAI, enabling users to identify potential solutions for root causes identified during Root Cause Analysis (RCA) or to address tasks categorized as 'Goal' types. This process leverages Solution-Based Issue Trees, also known as 'How Trees,' to systematically explore ways to resolve a problem or achieve an objective. Unlike the 'Why Trees' used in RCA, the root node of a How Tree is formulated as a 'how' question (e.g., "How can this problem be solved?" or "How can this goal be achieved?"), with subsequent nodes representing potential solutions or strategies.

The Solution Exploration process mirrors RCA in many aspects, particularly in its use of tree structures and collaborative tools. However, key differences distinguish it, particularly in tree construction, solution selection, and post-selection processing. These differences are elaborated below, with other aspects assumed to align with RCA unless specified.

How Tree Construction In Solution Exploration, users construct a How Tree to outline potential solutions. The tree-building process, whether executed via the Human-LLM Collaboration or LLM-Alone (AutoRun) mode, begins with the decomposition of the root node using the **Decomposition Agent**, which is the same agent from the RCA with a different behavior tailored to the solution decomposition. This decomposition yields a first layer of standalone solutions that directly address the root 'how' question. Subsequent decompositions of these solution nodes produce sub-nodes representing either sequential steps to implement a solution or alternative

options to achieve it. For example, a root node such as "How can customer retention be improved?" might decompose into solutions like "Enhance customer support" or "Introduce loyalty programs." Decomposing "Enhance customer support" further could yield steps (e.g., "Train support staff" and "Reduce response times") or options (e.g., "Expand live chat" and "Offer 24/7 phone support").

In contrast to RCA, where root causes may reside at any tree level, the final solutions in a How Tree are typically selected from the first layer beneath the root. Deeper layers provide supporting details—steps or options—utilized later in implementation planning.

Solution Selection and Validation The selection of solutions diverges significantly from RCA. While RCA prioritizes data-driven explanations to identify root causes, Solution Exploration emphasizes validation of proposed solutions. Users may manually designate a first-layer node as a solution based on intuition or expertise. However, StrategicAI recommends employing the Solution Validation feature for a more robust evaluation. This feature constructs a hypothesis tree for a selected solution node, formulating a testable hypothesis (e.g., "Implementing a loyalty program will increase customer retention by 10%"). The system then conducts a hypothesis test, leveraging internal data (e.g., user-uploaded files) or external sources (e.g., online research), and returns a Boolean (true/false) result indicating the solution's feasibility. This validation serves as a reliable indicator of a solution's applicability, though it remains optional.

Implementation Guide Generation Once a node is selected as a solution (with or without validation), it and its sub-tree are processed by the **Writer Agent**. This agent generates a detailed implementation guide tailored to the user's needs. The guide outlines a step-by-step plan if the sub-tree contains sequential steps or presents a comparative analysis of options if the sub-tree includes alternatives. For instance, a solution like "Introduce loyalty programs" with sub-nodes "Offer discounts" and "Provide exclusive benefits" might result in a guide detailing how to design and launch these initiatives. The richness of the guide depends on the depth and quality of the sub-tree, underscoring the importance of thorough decomposition beyond the first layer.

Collaboration Features and AutoRun Solution Exploration inherits RCA's collaboration features, including Node Decomposition, Node Verification (adapted here as Solution Validation), the Collaboration Agent (chatbot), and Get Potential Candidates. These operate similarly, with adjustments for the solution-focused context. For example, the Decomposition Agent generates solutions and their steps/options, while the Selector Agent identifies high-potential solution nodes from the first layer based on explanations or validation results.

In the LLM-Alone (AutoRun) mode, the process follows a deterministic sequence akin to RCA's AutoRun:

1. Construct an initial How Tree, decomposing the root node into a first layer of solutions and a second layer of steps/options.
2. Refine the tree using task context and company data.
3. Validate first-layer nodes by generating and testing hypothesis trees.
4. Flag nodes with a 'true' hypothesis outcome as solutions.
5. Generate implementation guides for selected solutions via the Writer Agent.

The key distinction from RCA's AutoRun lies in the exclusive selection of first-layer nodes and the integration of hypothesis testing prior to solution designation.

Conclusion Solution Exploration complements RCA by transitioning from problem identification to actionable resolution. By constructing How Trees, validating solutions through hypothesis testing, and generating implementation guides, StrategicAI empowers users to address identified root causes or pursue defined goals effectively. The feature's reliance on familiar collaboration tools and its adaptability to both manual and automated workflows ensure flexibility and efficiency, aligning with the broader objectives of StrategicAI.

5.4.4 Solution Validation

Solution Validation represents the final core functionality within the application. Its primary objective is to evaluate the feasibility of proposed solutions, either by validating solution nodes derived from a "How Tree" or by addressing yes/no questions posed in hypothesis-based tasks. This feature ensures that solutions are strategically sound by providing a definitive "True" or "False" answer to the root question, accompanied by an explanation of the reasoning.

The input to this process is a yes/no question, which can take one of the following forms:

- **Hypothesis Task**: A standalone yes/no question (e.g., "Will advertising boost sales?").
- **Goal Task with How Tree**: "Does Solution A help in achieving Goal A?"
- **Problem Task with How and Why Trees**: "Does Solution A help in solving Problem B?"

The output is a definitive answer to the root question—either "True" or "False"—along with a justification based on the analysis, enabling users to assess whether a solution is viable.

Methodology: Hypothesis Tree Construction The validation process begins with the construction of a Hypothesis Tree, which merges principles from business hypothesis testing frameworks (e.g., as seen in *The Lean Startup* by Ries, 2011) and the Query Tree concept from the ProbTree paper (Cao et al., 2023). The root node of the tree is the initial yes/no question provided by the user. This question is recursively decomposed into a set of sub-questions that adhere to the MECE (Mutually Exclusive, Collectively Exhaustive) principle. These sub-questions are designed such that their combined answers fully resolve the parent question.

For example, the root question "Will advertising boost sales?" might be broken down into:

- "Is the target audience reachable through advertising?"
- "Is the advertising budget sufficient to achieve the desired reach?"

This decomposition continues until the sub-questions reach an "atomic" level—questions that are simple enough to be answered directly through general knowledge, a basic online search, or user-provided data. The process is managed by a **Hypothesis Tree Builder Agent**, which constructs the tree and attaches a "Required Data" list to each node. This list contains specific data points or query-like strings (e.g., "market reach data" or "advertising budget figures") necessary to answer the question at that node. These data requirements guide the subsequent hypothesis testing phase.

Hypothesis Testing Process Once the Hypothesis Tree is constructed, the user initiates the hypothesis testing phase, which is inspired by the query resolution approach in the ProbTree paper. The ultimate goal is to determine the answer to the root node question by systematically resolving all nodes in the tree. This is achieved through a **post-order traversal**, where leaf nodes are answered first, and their responses propagate upward to resolve non-leaf (parent) nodes, ultimately reaching the root.

Each node's question is answered using up to four distinct techniques, with leaf nodes employing three and non-leaf nodes using all four. The techniques differ in the context provided by the underlying Large Language Model (LLM) to generate answers. After applying these techniques, the most certain answer is selected based on a calculated certainty score. The process is as follows:

1. **Techniques for Answering Nodes**:
    - **Closed Book**: The LLM answers the question solely based on its internal knowledge, without external context. For example, for "Is advertising effective?" it might respond, "Yes, advertising typically increases sales," relying on its training data.
    - **Open Book**: The question and its "Required Data" list are used as search queries to perform an online search (e.g., via web or X posts). Retrieved facts (e.g., "Studies show advertising increases sales by 10-20%") are provided as context to the LLM to generate an answer.

- **User Files**: Users can upload additional files (e.g., PDFs, text documents, or images) before testing begins. These files are processed as a single corpus by a **Fact Extractor Agent**, which uses the node's question and "Required Data" list as queries to extract relevant facts (e.g., "Last campaign increased sales by 8%"). These facts serve as context for the LLM.
- **Child Aggregation (Non-Leaf Nodes Only)**: For non-leaf nodes, the answers from child nodes (already resolved due to post-order traversal) are used as context. Each child's "official" answer (the one with the highest certainty) is aggregated to inform the parent node's response.

2. **Certainty Calculation**: For each technique, the LLM generates an answer, and its certainty is quantified using the mean of the logarithmic probabilities of the output tokens. The formula is:

$$\text{Certainty} = \frac{1}{n} \sum_{i=1}^{n} \log p_i$$

where:

- p_i is the probability of the i-th output token in the LLM's response,
- n is the total number of tokens in the response.

This score reflects the LLM's confidence in its answer. Leaf nodes are evaluated using Closed Book, Open Book, and User Files (three techniques), while non-leaf nodes include Child Aggregation (four techniques). The technique yielding the highest certainty score provides the "official" answer for that node.

3. **Traversal and Resolution**:
   - **Leaf Nodes**: Answered using the three techniques (Closed Book, Open Book, User Files). The highest-certainty answer is selected.
   - **Non-Leaf Nodes**: Answered using all four techniques. The post-order traversal ensures child nodes are resolved first, allowing their best answers to be aggregated as context for the parent. The highest-certainty answer is then chosen.
   - **Root Node**: The final True/False answer is determined, with an explanation tracing the most certain path through the tree.

Outcome Determination The root node's answer—"True" or "False"—is the culmination of the hypothesis testing process. An accompanying explanation highlights the reasoning, based on the most certain answers from the tree's nodes. For instance, if "Will advertising boost sales?" resolves to "True," the explanation might state: "The audience is reachable (Open Book, 95% certainty), and the budget is sufficient (User Files, 90% certainty), leading to a positive outcome."

This result serves as an additional indicator of a solution's viability, integrating seamlessly with the Solution Exploration phase to ensure strategic decisions are grounded in robust analysis.

User Interaction and File Integration Before initiating the hypothesis test, users can upload supplementary files to enhance the evidence base. These files are processed by the Fact Extractor Agent, which extracts relevant facts to support the User Files technique. This feature ensures that domain-specific data (e.g., internal reports or past performance metrics) can influence the validation process, making it adaptable to unique user contexts.

Solution Validation combines structured tree-based reasoning with advanced LLM capabilities to deliver reliable assessments of strategic feasibility. By decomposing complex questions into manageable sub-questions, leveraging multiple data sources, and quantifying certainty, it provides users with a clear, evidence-based verdict on their proposed solutions.

Integration and Strategic Value StrategicAI's functionalities form a cohesive pipeline: RCA identifies root causes (e.g., "Reduced demand"), Solution Exploration crafts responses (e.g., "Increase advertising"), and Solution Validation confirms viability (e.g., "Yes, advertising will boost sales"). Supported by the Fact Extractor Agent and flexible execution flows, this system leverages LLMs to deliver evidence-based strategies, aligning with real-world frameworks like McKinsey's Issue Trees and lean hypothesis testing. As of March 02, 2025, StrategicAI stands as a robust tool for business leaders, enhancing decision-making efficiency and effectiveness across diverse industries.

5.5 Discussion of Design Choices and Decision-Making Process

The design of StrategicAI is meticulously crafted to address the multifaceted challenges inherent in strategic decision-making, as identified in the introduction. By leveraging the capabilities of Large Language Models (LLMs) within a structured framework, the system not only mitigates human limitations but also enhances the decision-making process through transparency, reliability, and collaborative intelligence. This subsection elucidates how the specific features and processes of StrategicAI directly confront the issues of information overload, cognitive biases, incomplete root cause analysis, and barriers to creativity, thereby providing a robust solution to the research problem.

5.5.1 Addressing Information Overload

Information overload poses a significant barrier to strategic decision-making, overwhelming managers with vast datasets that obscure actionable insights (Shahrzadi et al., 2024). StrategicAI counters this through its **Fact Extraction Feature**, which processes diverse inputs—such as uploaded files and online search results—into atomic facts. These standalone, context-rich units (e.g., transforming "Profits declined by 15% due to supply chain disruptions" into "Profit decline reached 15% in Q1" and "Supply chain issues impacted operations") reduce cognitive load by presenting only essential information. This feature, detailed in the Core Functionalities subsection, ensures managers can focus on decision-relevant data without parsing entire documents.

Complementing this, the **structured tree-based analyses**—Diagnostic Issue Trees (Why Trees), Solution Issue Trees (How Trees), and Hypothesis Trees—organize information hierarchically. For instance, a Why Tree decomposes a problem like "Why are sales dropping?" into manageable sub-causes, allowing users to address specific branches rather than the entire dataset. The **Collaboration Agent (Chatbot)**, accessible in the Human-LLM Collaboration mode, further mitigates overload by providing on-demand assistance, answering queries, and suggesting next steps based on the task context. These features collectively streamline data navigation, ensuring StrategicAI delivers concise, actionable insights tailored to the company profile established during system setup.

5.5.2 Mitigating Cognitive Biases

Cognitive biases, such as confirmation bias and overconfidence, distort strategic decision-making by skewing solution validation (Acciarini et al., 2021). StrategicAI addresses these through design choices that emphasize transparency and deliberative reasoning, drawing on Nobel Prize winner Daniel Kahneman's seminal work, *Thinking, Fast and Slow* (Kahneman, 2011). Kahneman revolutionized our understanding of human behavior by distinguishing between fast, intuitive thinking—prone to biases—and slow, rational thinking, which fosters objectivity. StrategicAI's features align with the latter, countering biases through structured, evidence-based processes.

The system's **transparency and explainability** are evident in the tree structures, where each node (e.g., a cause in a Why Tree or a solution in a How Tree) includes a detailed explanation justifying its placement. In the Hypothesis Tree, answers like "True" to "Will advertising boost sales?" are accompanied by reasoning derived from data-driven paths, enabling managers to scrutinize recommendations rather than accept them intuitively. This transparency reduces confirmation bias by exposing the evidential basis, preventing managers from favoring preconceived notions.

Additionally, the **certainty quantification** in Hypothesis Trees calculates confidence scores using logarithmic probabilities of LLM output tokens (e.g., Certainty = $\frac{1}{n}\sum_{i=1}^{n}\log p_i$), providing a quantitative measure of reliability. This feature, outlined in the Solution Validation subsection, counters overconfidence by grounding decisions in assessed probabilities rather than unchecked assumptions. The **Human-LLM Collaboration Mode** further mitigates biases by encouraging a deliberative process: managers co-build trees, validate suggestions, and interact with the Collaboration Agent, slowing decision-making to align with Kahneman's rational thinking paradigm. This collaborative, transparent approach ensures decisions are critically evaluated, minimizing bias-driven errors.

5.5.3 Enhancing Root Cause Analysis

Incomplete root cause analysis, often due to overlooked factors, undermines strategic decisions (Peerally et al., 2017). StrategicAI's **Diagnostic Issue Tree (Why Tree)** addresses this by systematically decomposing problems into potential causes and sub-causes. For example, a root node like "Why are profits dropping?" branches into nodes such as "Poor marketing" or "Supply chain issues," with further sub-causes explored hierarchically until atomic levels are reached. This exhaustive structure, detailed in the Root Cause Analysis subsection, ensures comprehensive factor consideration.

The **Node Verification Feature** enhances this process by validating nodes with empirical evidence from internal files (processed by the Fact Extractor Agent) and external online sources (retrieved via the Researcher Agent). This evidence-based approach, supported by the Tree Builder Agent's refinements, ensures causes are not speculative but data-driven. The **Get Potential Candidates** feature, powered by the Selector Agent, further streamlines analysis by prioritizing nodes with strong evidence using a self-consistency mechanism across multiple iterations. These features, operational in both Human-LLM Collaboration and LLM-Alone modes, transform root cause analysis into a rigorous, thorough process, reducing oversight risks.

5.5.4 Fostering Creativity in Solution Generation

Organizational barriers, such as time constraints and risk aversion, often stifle creativity in solution generation (Meinel et al., 2012). StrategicAI's **Solution Exploration** feature, utilizing How Trees, counters these by encouraging expansive thinking. A root node like "How can customer retention be improved?" decomposes into solutions (e.g., "Enhance customer support" or "Introduce loyalty programs") and further into steps or options, as described in the Solution Exploration subsection. This structured yet flexible framework promotes the exploration of multiple avenues, overcoming conventional thinking constraints.

The **Implementation Guide Generation** feature, executed by the Writer Agent, produces detailed plans for selected solutions, offering a customizable blueprint (e.g., steps to launch a loyalty program). This structure supports adaptation and innovation, easing time pressures by providing actionable starting points. The **Human-LLM Collaboration Mode** enhances creativity further: managers interact with the Collaboration Agent to explore alternatives and refine ideas, leveraging LLM suggestions to spark innovation. By integrating these features, StrategicAI dismantles barriers, empowering managers to devise bold, effective strategies.

5.5.5 Conclusion

StrategicAI's design choices are strategic responses to the core challenges of decision-making. The Fact Extraction Feature and tree-based analyses address information overload by distilling data into manageable insights. Transparency, certainty quantification, and collaborative modes mitigate cognitive biases, aligning with Kahneman's rational thinking model. Diagnostic Issue Trees and verification processes enhance root cause analysis, ensuring thoroughness. Solution Exploration and implementation guides foster creativity by providing structured flexibility. Collectively, these features—rooted in the system's architecture, functionalities, and execution flows—equip managers to navigate complexity with confidence, fulfilling the research objective of enhancing strategic outcomes.

6 Experiments and Evaluations

In this section, we present a comprehensive evaluation of *StrategicAI*, designed to assess its effectiveness in identifying root causes of business problems and generating actionable solutions to support strategic decision-making. The evaluation is structured around four distinct experiments: (1) the **Single-Agent AutoRun Evaluation**, which establishes the baseline performance of the initial AutoRun feature; (2) the **Multi-Agent AutoRun Evaluation**, which tests enhancements to the AutoRun feature using a multi-agent approach; (3) the **LLM Preference Evaluation**, which compares *StrategicAI* outputs against ChatGPT using LLM judges; and (4) the **User Testing Session for Human-LLM Collaboration**, which evaluates usability and effectiveness in a human-in-the-loop context. These experiments leverage six diverse case studies, ensuring a robust analysis across varied business scenarios, and employ both similarity-based and LLM-based evaluation methods to provide a multifaceted assessment of *StrategicAI*'s capabilities. An overview of these experiments is illustrated in Figure 13, offering a visual representation of the evaluation framework.

Our evaluation methodology is firmly anchored in the **Design Science Research (DSR)** framework, as outlined by Hevner et al. (2004), aligning particularly with **Guideline 3: Design Evaluation**. Through a combination of rigorous, well-executed methods—including controlled experiments, analytical metrics, and user testing—we demonstrate *StrategicAI*'s utility, quality, and efficacy in addressing the identified problem of enhancing managerial decision-making. The use of case studies which were a combination of real-world cases and cases designed to teach consultants in business schools ensures **Guideline 2: Problem Relevance**, grounding the evaluation in practical, organizational contexts. Furthermore, the iterative progression from single-agent to multi-agent evaluations and supplementary analyses embodies **Guideline 6: Design as a Search Process**, reflecting a systematic refinement of the artifact. By integrating automated assessments with human collaboration, this evaluation not only tests *StrategicAI*'s technical performance but also its practical applicability, aligning with the IS framework's emphasis on contextual relevance and theoretical rigor. This comprehensive approach sets the stage for a detailed exploration of *StrategicAI*'s capabilities and its potential impact on decision support systems.

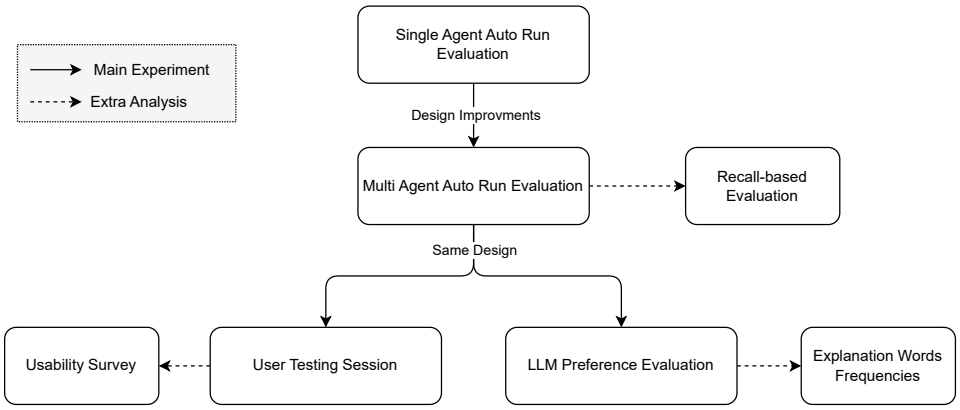

Figure 13 Overview of the experiments

6.1 Single-Agent AutoRun Evaluation

The Single-Agent AutoRun Evaluation tests the initial version of StrategicAI's AutoRun feature, where a single agent handles all tasks, including identifying root causes, constructing "how trees," and generating and validating solutions. This evaluation establishes the baseline performance of StrategicAI across six case studies, using metrics such as Problem Identification Score (PIS) and Solution Generation Score (SGS), and compares it to ChatGPT.

6.1.1 Experiment Setup

The experiment utilized six case studies, each representing a unique business scenario with predefined problems and solutions established as ground truth. These case studies were previously introduced in Section 4 of this thesis. For each case study:

- **StrategicAI Process**: We executed the AutoRun feature to identify the root causes of the problems. For each identified root cause, a "how tree" was constructed, and the AutoRun was subsequently applied to generate corresponding solutions, which were then validated automatically.
- **ChatGPT Comparison**: We obtained outputs from ChatGPT for the same case studies, using identical inputs to ensure a fair comparison.
- **Objective**: The primary goal was to evaluate how well StrategicAI's AutoRun feature, leveraging the LLM-Tree framework, matches the ground truth in identifying problems and generating solutions, compared to ChatGPT's performance.

This setup is critical as it establishes the baseline methodology, which remains consistent with the second experiment, differing only in the system changes introduced later.

6.1.2 Evaluation Metrics

The performance of StrategicAI and ChatGPT was assessed using two primary metrics that we talked about in the Research Methodology section and we are repeating for the ease of reading:

- **Problem Identification Score (PIS)**: A composite score evaluating the accuracy, relevance, and explanation quality of the identified problems.
- **Solution Generation Score (SGS)**: A composite score evaluating the accuracy, relevance, and explanation quality of the generated solutions.

These scores were derived from the following sub-metrics, calculated separately for problems and solutions:

- **Precision**: The proportion of model outputs that correctly match ground truth items.
- **Recall**: The proportion of ground truth items correctly identified by the model.
- **F1 Score**: The harmonic mean of precision and recall, providing a balanced measure of accuracy.
- **Relevance Score**: The average of the maximum similarity between model outputs and ground truth items, assessing content alignment.
- **Explanation Relevance**: The average of the maximum similarity between the explanations of model outputs and ground truth explanations, assessing the quality of supporting details.
- **Final Score**: A weighted combination of F1 score (weight: 0.35), relevance score (weight: 0.3), and explanation relevance (weight: 0.35), calculated as:

$$\text{Final Score} = 0.35 \times \text{F1} + 0.3 \times \text{Relevance Score} + 0.35 \times \text{Explanation Relevance} \quad (2)$$

The PIS is the final score for the identified problems, and the SGS is the final score for the generated solutions. These metrics were computed using two distinct evaluation approaches: a similarity-based method and an LLM-based method, each differing in how matches between model outputs and ground truth are determined. The LLM-based approach, in particular, utilized multiple LLM judges to enhance the reliability of the evaluation, with their results averaged to produce the final metrics.

6.1.3 Results

Similarity-Based Approach In the similarity-based evaluation, we employed embeddings to measure the textual similarity between model outputs and ground truth items, enabling the identification of matches for each model output relative to the ground truth. The methodology is outlined as follows:

- **Embedding Model**: Text and explanation embeddings were generated using the pre-trained model `sentence-transformers/all-MiniLM-L6-v2`.
- **Similarity Calculation**: Cosine similarity was computed between the embeddings of model outputs and ground truth items, yielding a similarity matrix.
- **Matching Logic**: A greedy algorithm was applied to pair each model output with the ground truth item exhibiting the highest similarity score, provided it exceeded a threshold of 0.5. To avoid duplication, each ground truth item was matched to at most one model output.
- **Metrics Calculation**: Precision, recall, and F1 score were derived from these matches. Additionally, relevance score and explanation relevance were computed as the average of the maximum similarity scores for outputs and explanations, respectively, independent of the threshold.

This approach provides a quantitative assessment of textual similarity, emphasizing structural and lexical alignment between outputs and ground truth.

The performance of StrategicAI (denoted as "llm-tree") and ChatGPT, averaged across six case studies, is presented in Table 7. The results elucidate the efficacy of the similarity-based approach in evaluating model outputs.

Method	Metric Type	Precision	Recall	F1 Score	Relevance Score	Explanation Relevance	Final Score
ChatGPT	PIS	0.43	**1.00**	**0.60**	**0.75**	**0.80**	**0.71**
ChatGPT	SGS	**0.43**	0.76	**0.53**	**0.78**	**0.77**	**0.69**
llm-tree	PIS	**0.49**	0.92	**0.60**	0.73	0.75	0.69
llm-tree	SGS	0.22	**0.78**	0.33	0.72	0.74	0.59

Table 7 Averaged performance metrics for StrategicAI (llm-tree) and ChatGPT using the similarity-based approach in the single-agent AutoRun evaluation.

LLM-Based Approach The LLM-based approach utilized multiple large language models (LLMs) to achieve a more robust semantic matching of model outputs against ground truth items, offering a comprehensive evaluation framework. The methodology is detailed as follows:

- **Models Used**: Four distinct LLMs served as judges: `qwen2p5-72b-instruct`, `llama-v3p3-70b-instruct`, `llama-v3p1-405b-instruct`, and `mixtral-8x22b-instruct`, accessed via the Fireworks.AI API.
- **Prompt Construction**: Each LLM was equipped with a system prompt defining its role as an expert evaluator, alongside a user prompt containing company background, task description, relevant data, ground truth, and model outputs.
- **Matching Logic**: Each LLM independently determined matches based on semantic comprehension, accounting for both explicit matches (direct equivalence) and implicit matches (conceptual equivalence). The process entailed processing the prompts to produce a list of ground truth items aligned with model outputs.
- **Metrics Calculation**: Precision, recall, and F1 score were calculated for each LLM judge based on their identified matches. Relevance score and explanation relevance were computed using the similarity-based method (cosine similarity of embeddings) to ensure consistency across evaluation approaches. Final

performance metrics were obtained by averaging the results from all four LLM judges across precision, recall, F1 score, relevance score, explanation relevance, and final score, applied separately for each method (StrategicAI and ChatGPT) and metric type (problems and solutions). This averaging enhanced the reliability and robustness of the evaluation.

This approach was designed to address potential limitations in the similarity-based method, particularly in capturing the semantic nuances of StrategicAI outputs relative to the ground truth, while maintaining consistency in relevance and explanation scoring.

The results of the LLM-based evaluation, averaged across the four LLM judges (`qwen2p5-72b-instruct`, `llama-v3p3-70b-instruct`, `llama-v3p1-405b-instruct`, and `mixtral-8x22b-instruct`), are presented in Table 8. These findings reflect the semantic matching performance of StrategicAI ("llm-tree") and ChatGPT across the six case studies.

Method	Metric Type	Precision	Recall	F1 Score	Relevance Score	Explanation Relevance	Final Score
ChatGPT	PIS	0.43	**1.00**	0.60	**0.75**	**0.80**	**0.71**
ChatGPT	SGS	**0.41**	**0.78**	**0.52**	**0.78**	**0.77**	**0.69**
llm-tree	PIS	**0.50**	0.93	**0.61**	0.73	0.75	0.69
llm-tree	SGS	0.16	0.50	0.23	0.72	0.74	0.55

Table 8 Averaged performance metrics for StrategicAI (llm-tree) and ChatGPT using the LLM-based approach, evaluated by four LLM judges in the single-agent AutoRun evaluation.

6.1.4 Discussion

The evaluation results from both the similarity-based and LLM-based approaches consistently demonstrate that ChatGPT outperforms *StrategicAI* in the Single-Agent AutoRun Evaluation, particularly in terms of the final scores for both the Problem Identification Score (PIS) and the Solution Generation Score (SGS). This finding holds across the averaged metrics presented in Tables 7 and 8, highlighting key differences in the models' approaches and their effectiveness in addressing the case studies.

For ChatGPT, the outputs were generated using a zero-shot prompt that included the company and case context along with internal data. This process resulted in a comprehensive list of problems, each accompanied by detailed explanations, followed by a corresponding list of solutions directly tied to the identified problems. This straightforward, context-driven approach enabled ChatGPT to achieve higher overall performance, with a perfect recall of 1.00 for PIS in both evaluation methods, indicating it captured all ground truth problems, and superior SGS metrics (e.g., precision of 0.41–0.43 and recall of 0.76–0.78) compared to *StrategicAI*.

In contrast, *StrategicAI*'s methodology involved a structured process: identifying root causes, constructing "how trees" for each root cause, and generating solutions from these trees. While this approach aimed to provide a systematic, data-driven framework, the results reveal significant limitations in its current implementation. For PIS, *StrategicAI* achieved a slightly higher precision (0.49–0.50 vs. 0.43) but a lower recall (0.92–0.93 vs. 1.00) compared to ChatGPT. This suggests that while *StrategicAI* was more selective in identifying problems, it missed some ground truth issues that ChatGPT captured. Notably, both models exhibited similar behavior in listing problems that were plausible within the case context but not always precise or directly aligned with the ground truth. Approximately half of the identified problems were correct, with the remainder being contextually reasonable but not specific to the ground truth, undermining the goal of data-driven problem identification.

The disparity is more pronounced in the SGS, where *StrategicAI*'s precision (0.16–0.22) and recall (0.50–0.78) were substantially lower than ChatGPT's (precision 0.41–0.43, recall 0.76–0.78). This performance gap can be attributed to the design of *StrategicAI*'s solution generation process. For each identified root cause—including those that were incorrect—a "how tree" was constructed, and the single agent selected two to three solutions per tree. Consequently, any erroneous root cause led to the generation of additional irrelevant solutions, inflating the output volume and reducing precision. This proliferation of solutions highlights a critical flaw in the current AutoRun feature, as it fails to filter out inaccuracies early in the process.

Further analysis of the single agent's behavior reveals additional shortcomings. The agent was equipped with a comprehensive set of tools intended to support data-driven decision-making, such as accessing data files and

validating solutions through hypothesis testing. However, it frequently bypassed these critical steps, proceeding directly from tree construction to final outputs without leveraging available data or performing validation. For problem identification, this resulted in a reliance on contextual plausibility rather than evidence-based insights, mirroring ChatGPT's approach but without the same breadth of coverage. For solution generation, the lack of validation—such as hypothesis testing—allowed unverified solutions to persist, further degrading performance.

Despite these challenges, there is evidence of potential in *StrategicAI*'s tree decomposition approach. The slightly higher precision in PIS (e.g., 0.50 vs. 0.43 in the LLM-based approach) suggests that structuring problem identification into a "why tree" and decomposing solutions into a "how tree" can enhance focus on accurate issues. This structured framework could theoretically improve decision-making by systematically exploring root causes and solution pathways, provided the process is fully executed.

To address the observed limitations, *StrategicAI*'s AutoRun feature requires significant refinement. Enforcing mandatory use of data-access tools and validation steps, such as hypothesis testing, would ensure that decisions are grounded in evidence rather than assumptions. By reducing the number of extraneous problems identified, this would decrease the number of "how trees" generated, leading to fewer, more focused solutions. Similarly, incorporating rigorous solution validation would filter out irrelevant outputs, potentially increasing both precision and recall for SGS. These adjustments would align the system more closely with its data-driven intent, distinguishing it from ChatGPT's broader, less constrained approach.

In conclusion, while *StrategicAI*'s structured methodology shows promise, particularly in problem identification precision, its current implementation is hindered by the single agent's tendency to skip essential steps. This results in lower overall performance compared to ChatGPT, which benefits from a simpler, context-driven strategy. Enhancing *StrategicAI*'s adherence to a complete data-driven process could unlock the potential of the LLM-Tree framework, improving its effectiveness as a decision support tool.

6.2 Multi-Agent AutoRun Evaluation

The Multi-Agent AutoRun Evaluation extends the framework established in the Single-Agent AutoRun Evaluation, incorporating enhancements to StrategicAI's AutoRun feature. This evaluation assesses the system's performance in identifying root causes and generating actionable solutions across the same six case studies, utilizing both similarity-based and LLM-based evaluation approaches. The primary advancement is the transition from a single-agent process to a multi-agent architecture, where distinct agents handle sequential tasks to improve accuracy and effectiveness.

6.2.1 Experiment Setup

The Multi-Agent AutoRun Evaluation was conducted using the same six case studies as in the Single-Agent Evaluation, each representing a unique business scenario with predefined ground truth problems and solutions, as introduced in Section 4. The evaluation maintains consistency with the prior methodology, with the following adaptations:

- **StrategicAI Process**: The AutoRun feature was executed with a multi-agent approach, assigning specialized agents to tasks such as tree construction, data incorporation, root cause identification, and solution validation. This modular design ensures a structured workflow across the process.
- **Objective**: The goal was to evaluate the enhanced AutoRun feature's ability to match ground truth in problem identification and solution generation, leveraging the multi-agent framework, and to compare its performance against ChatGPT.

This setup builds on the baseline methodology, differing only in the implementation of the multi-agent system.

6.2.2 Evaluation Metrics

The performance of StrategicAI and ChatGPT was assessed using the Problem Identification Score (PIS) and Solution Generation Score (SGS), consistent with the Single-Agent Evaluation. These composite scores,

derived from precision, recall, F1 score, relevance score, and explanation relevance, were calculated using both similarity-based and LLM-based approaches, as detailed in the Single-Agent Evaluation methodology.

6.2.3 Results

Similarity-Based Approach The performance metrics for StrategicAI (denoted as "llm-tree") and ChatGPT, evaluated using the similarity-based approach and averaged across the six case studies, are presented in Table 9.

Method	Metric Type	Precision	Recall	F1 Score	Relevance Score	Explanation Relevance	Final Score
ChatGPT	PIS	0.43	**1.00**	0.60	0.75	0.80	0.71
ChatGPT	SGS	**0.43**	**0.76**	**0.53**	**0.78**	0.77	**0.69**
llm-tree	PIS	**0.64**	**1.00**	**0.77**	**0.77**	**0.82**	**0.79**
llm-tree	SGS	0.34	0.72	0.45	0.75	**0.80**	0.66

Table 9 Averaged performance metrics for StrategicAI (llm-tree) and ChatGPT using the similarity-based approach in the Multi-Agent AutoRun Evaluation.

LLM-Based Approach The performance metrics for StrategicAI (llm-tree) and ChatGPT, evaluated using the LLM-based approach with four LLM judges (`qwen2p5-72b-instruct`, `llama-v3p3-70b-instruct`, `llama-v3p1-405b-instruct`, and `mixtral-8x22b-instruct`) and averaged across the six case studies, are presented in Table 10.

Method	Metric Type	Precision	Recall	F1 Score	Relevance Score	Explanation Relevance	Final Score
ChatGPT	PIS	0.43	**1.00**	0.60	0.75	0.80	0.71
ChatGPT	SGS	0.41	0.78	0.52	**0.78**	0.77	0.69
llm-tree	PIS	**0.59**	0.94	**0.71**	**0.77**	**0.82**	**0.78**
llm-tree	SGS	**0.44**	**0.90**	**0.57**	0.75	**0.80**	**0.70**

Table 10 Averaged performance metrics for StrategicAI (llm-tree) and ChatGPT using the LLM-based approach, evaluated by four LLM judges in the Multi-Agent AutoRun Evaluation.

6.2.4 Discussion

The Multi-Agent AutoRun Evaluation highlights significant enhancements in StrategicAI's performance over its single-agent predecessor. By restructuring the AutoRun feature into a series of specialized agents, each tasked with a specific step—such as tree construction, data integration, root cause analysis, and solution validation—the system achieves greater precision and effectiveness. This sequential, modular approach results in more specific problem identification and richer, more detailed solution explanations, addressing key limitations observed in the single-agent setup, such as skipped steps and irrelevant outputs. The structured workflow ensures thorough execution of each phase, enhancing the system's ability to align outputs with ground truth. Additionally, the multi-agent design improves integration with human expertise, facilitating more effective human-LLM collaboration and increasing its practical utility in decision-making contexts. Overall, this transition strengthens StrategicAI's capabilities, making it a more robust tool for strategic decision support.

6.2.5 Multi-Agent AutoRun Evaluation Supplementary Evaluation: Recall-Based Metric

To further explore StrategicAI's capabilities beyond the final scores reported in the Multi-Agent AutoRun Evaluation, a supplementary evaluation was conducted focusing exclusively on a recall-based metric. This evaluation was motivated by the observation that StrategicAI's Solution Generation Scores (SGS) in Multi-Agent AutoRun Evaluation, while improved, remained slightly lower than ChatGPT's (0.66 vs 0.69 in similarity-based evaluations). The goal was to assess whether StrategicAI's internal tree structures—specifically the "why"

and “how” trees generated by the AutoRun feature—capture the ground truth problems and solutions more comprehensively than the final outputs suggest, thereby demonstrating its latent potential.

In this supplementary experiment, the same six case studies from Multi-Agent AutoRun Evaluation were analyzed, but instead of evaluating the final problem identification and solution outputs, the focus shifted to the intermediate tree structures produced by StrategicAI’s AutoRun. The “why” trees, which map root causes, were evaluated against ground truth problems, while the “how” trees, which outline solution pathways, were assessed against ground truth solutions. These tree structures, consisting of nodes and their explanatory content, were extracted and reviewed by the same four LLM judges used in the LLM-based approach. Each judge was tasked with identifying matches—either explicit mentions or clear references—between the tree nodes (including their explanations) and the ground truth elements for each case study. The recall was then calculated separately for problem identification (PIS) and solution generation (SGS) as the proportion of ground truth items captured within the respective tree structures, averaged across the six case studies and four judges.

The results, visualized in Figure 14, reveal exceptionally high recall values. For problem identification, StrategicAI achieves an average recall of 0.9792, indicating that nearly all ground truth root causes are represented within the “why” tree structures. For solution generation, the average recall is 0.8854, suggesting that a substantial majority of ground truth solutions are embedded within the “how” trees. These figures contrast sharply with the final SGS scores from Multi-Agent AutoRun Evaluation (0.66 and 0.70), highlighting a key insight: StrategicAI’s tree-building process captures a broader range of relevant information than is reflected in its final, filtered outputs.

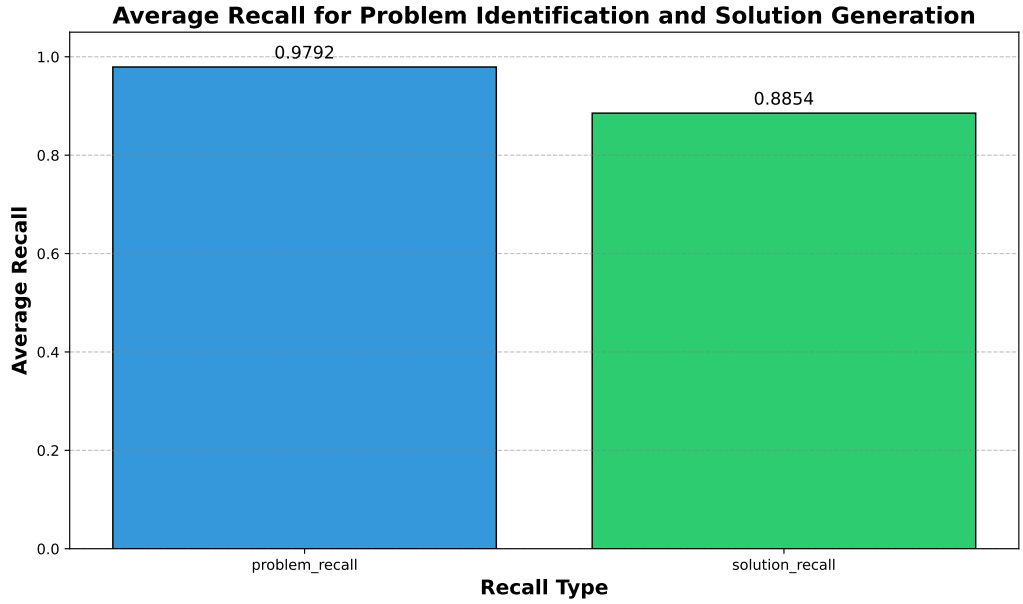

Figure 14 Average recall scores for StrategicAI’s problem identification (PIS) and solution generation (SGS) across six case studies, evaluated using a recall-based metric on “why” and “how” tree structures by four LLM judges in the supplementary evaluation of the Multi-Agent AutoRun Evaluation.

Insights and Implications These recall scores underscore the robustness of StrategicAI’s tree-construction mechanism, a core component of the refined AutoRun feature introduced in the Multi-Agent AutoRun Evaluation. The near-perfect recall for problem identification (0.9792) suggests that the multi-agent approach—where separate agents construct the tree, integrate data, and identify root causes—effectively encapsulates the full scope of ground truth problems. Similarly, the high recall for solution generation (0.8854) indicates that also the agents, tasked with building and validating “How” trees, successfully incorporate the most viable solutions, even if the final output selection process reduces precision (e.g., 0.34 in similarity-based SGS and 0.44 in LLM-based SGS). This discrepancy between tree-level recall and final-score precision points to a bottleneck in the solution-filtering phase rather than a failure to generate appropriate candidates.

A critical implication emerges when considering human interaction with StrategicAI. The high recall within the trees suggests that the tool retains a wealth of actionable insights within its intermediate structures. In a real-world scenario, a human user engaging with StrategicAI could explore these trees interactively, leveraging their comprehensive coverage to refine outputs manually. This human-in-the-loop approach could potentially yield SGS scores surpassing the automated AutoRun results and even ChatGPT's consistent performance (0.69 across both evaluation methods). For instance, a user could prioritize precision by selecting only the most relevant nodes, addressing the over-generation issue noted in Multi-Agent AutoRun Evaluation's discussion.

6.3 LLM Preference Evaluation

This subsection presents a supplementary evaluation experiment designed to compare the outputs of LLM-Tree and ChatGPT, extending beyond the quantitative scores previously reported for problem identification and solution generation. Unlike earlier analyses that emphasized numerical metrics, this study adopts a preference-based evaluation framework utilizing the LLM-as-a-judge methodology. Consistent with prior experiments, four large language models (LLMs) serve as judges, ensuring continuity in evaluative standards.

The evaluation assesses outputs across four meticulously defined metrics: **plausibility**, **faithfulness**, **comprehensiveness**, and **non-redundancy**. Plausibility examines the logical coherence and reasoning flow of the outputs. Faithfulness evaluates their alignment with established world knowledge, offering insights into the models' reasoning processes. Comprehensiveness measures the breadth and depth of coverage in identifying problems or proposing solutions. Non-redundancy assesses the extent to which outputs avoid redundancy, adhering to the Mutual Exclusivity and Collective Exhaustiveness (MECE) principle.

6.3.1 Experimental Design

The evaluation framework involves presenting outputs from LLM-Tree and ChatGPT, derived from six case studies, to four LLM judges. Problem identification and solution generation are treated as distinct tasks. To eliminate model-specific bias, outputs are anonymized as Answer A and Answer B, and judges select the superior output for each metric, accompanied by an explanation reserved for supplementary analysis. To further enhance impartiality, the output assignments are reversed, and the evaluation is repeated. This process generates two evaluations per metric per case per judge (4 metrics $\times$ 2 assignments $\times$ 2 tasks), totaling 16 evaluations per case. Across six cases and four judges, this results in 384 evaluations. Final preference percentages are computed by averaging across all judges and metrics, providing a robust comparison of model performance for problem identification and solution generation separately.

6.3.2 Output Preparation

Initially, outputs were formatted as lists of JSON objects. However, early trials revealed a bias favoring outputs with a higher quantity of problems or solutions, skewing preferences. To address this, Grok 3 from xAI was employed to convert each output into a concise summary paragraph, replacing the JSON lists in the evaluation. This adjustment minimized quantity-based bias, yielding more balanced and meaningful results. As detailed in Table 11, ChatGPT's problem identification summaries average 172.17 tokens, surpassing LLM-Tree's 157.83 tokens, while LLM-Tree's solution summaries are notably longer (312.83 tokens) than ChatGPT's (154.0 tokens). These differences highlight distinct stylistic and depth-related characteristics between the models, potentially influencing judges' preferences.

	Problems	**Solutions**
ChatGPT	**172.17**	154.0
LLM-Tree	157.83	**312.83**

Table 11 Average token counts in the summaries of problems and solutions for ChatGPT and LLM-Tree.

6.3.3 Results for Problem Identification

The results for problem identification preferences are illustrated in Figure 15, a bar graph depicting the percentage of instances each model's output was preferred across the four metrics. ChatGPT demonstrates a significant advantage in comprehensiveness, with a preference rate of 70.83% compared to LLM-Tree's 29.17%. In non-redundancy and plausibility, ChatGPT maintains a slight lead, with 54.17% versus LLM-Tree's 45.83% for both metrics. Faithfulness shows no discernible preference, with both models equally favored at 50.0%.

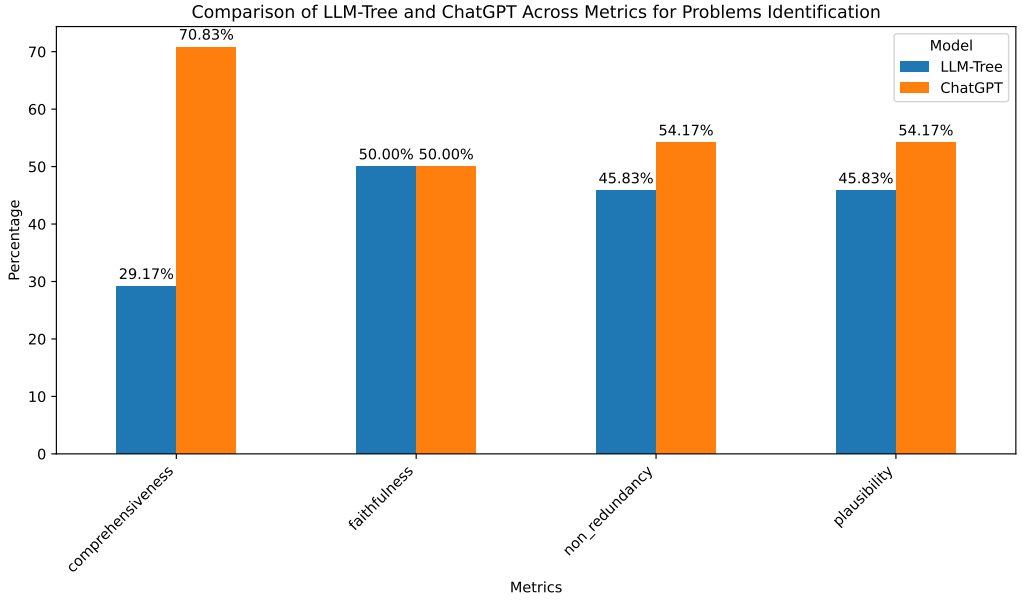

Figure 15 Bar graph illustrating preference percentages for LLM-Tree and ChatGPT across four metrics in problem identification.

6.3.4 Discussion for Problem Identification

The marked preference for ChatGPT in comprehensiveness (70.83% vs. 29.17%, a 41.66% difference) suggests that LLM judges favor outputs with broader scope and inclusivity. ChatGPT's tendency to enumerate diverse issues—spanning financial, operational, and strategic domains—likely underpins this preference. Conversely, LLM-Tree, embedded within the StrategicAI framework, prioritizes data-driven root causes over exhaustive coverage, selecting specific nodes from a diagnostic issue tree with explanatory annotations. This focus on actionable insights rather than comprehensive enumeration aligns with its strategic intent but results in a lower comprehensiveness score.

In non-redundancy, ChatGPT's modest edge (54.17% vs. 45.83%, an 8.34% difference) indicates that its problem lists are perceived as more distinct. This may stem from ChatGPT generating outputs within a single prompt, facilitating effective redundancy management. LLM-Tree's tree-based structure, while enabling deep exploration, may introduce overlap as nodes decompose, occasionally compromising MECE adherence and increasing perceived redundancy.

For plausibility, ChatGPT's slight advantage (54.17% vs. 45.83%, an 8.34% difference) suggests its problems are deemed more believable or contextually apt, possibly due to richer contextual framing. In contrast, faithfulness reveals no preference (50.0% each), indicating both models exhibit equivalent fidelity to the input context. This equivalence suggests that while accuracy is critical, it does not differentiate the models, shifting evaluative emphasis to attributes like scope and coherence.

6.3.5 Results for Solution Generation

The results for solution generation preferences, presented in Figure 16, reveal a pronounced preference for LLM-Tree across all metrics. LLM-Tree achieves unanimous preference in comprehensiveness and faithfulness

(100.0% vs. 0.0% for ChatGPT), a strong lead in non-redundancy (75.0% vs. 25.0%), and a substantial advantage in plausibility (89.58% vs. 10.42%).

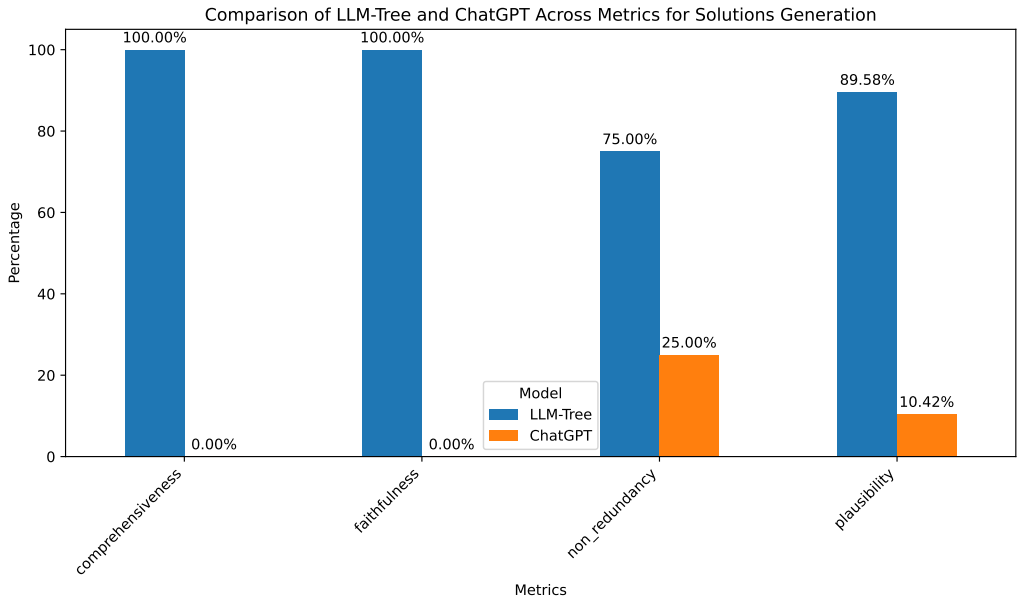

Figure 16 Bar graph illustrating preference percentages for LLM-Tree and ChatGPT across four metrics in solution generation.

6.3.6 Discussion for Solution Generation

LLM-Tree's dominance in solution generation arises from three key attributes. Firstly, its solutions exhibit greater depth and detail, often articulated as actionable steps or implementation guidelines, contrasting with ChatGPT's broader, less specific recommendations. This distinction drives LLM-Tree's perfect scores in comprehensiveness and faithfulness (100.0%), supported by its Solution Diagnostic Tree and writer agent, which elaborate on selected nodes to produce practical outputs.

Secondly, LLM-Tree generates a higher quantity of solutions per problem through a multi-angle approach. For each root cause identified in a 'Why' tree, a 'How Tree' is constructed, with the selector agent potentially choosing multiple nodes, yielding a richer solution set. This abundance reinforces its 100.0% comprehensiveness preference but occasionally introduces overlap, reducing non-redundancy to 75.0% compared to ChatGPT's 25.0%.

Thirdly, LLM-Tree employs precise, structured language, often incorporating technical or industry-specific concepts, enhancing credibility and contextual alignment. This precision contributes to its high plausibility (89.58%) and faithfulness (100.0%) scores, contrasting with ChatGPT's vaguer phrasing, which underscores LLM-Tree's superior performance.

6.3.7 LLM Preference Explanations Analysis

To explain the judges' preferences, an analysis of the most frequent words in their explanations was conducted since we asked the LLMs to explain their choices, categorized by task (problem identification and solution generation) and metric. The top three words per metric, derived from bubble chart data (Figure 17), are examined below.

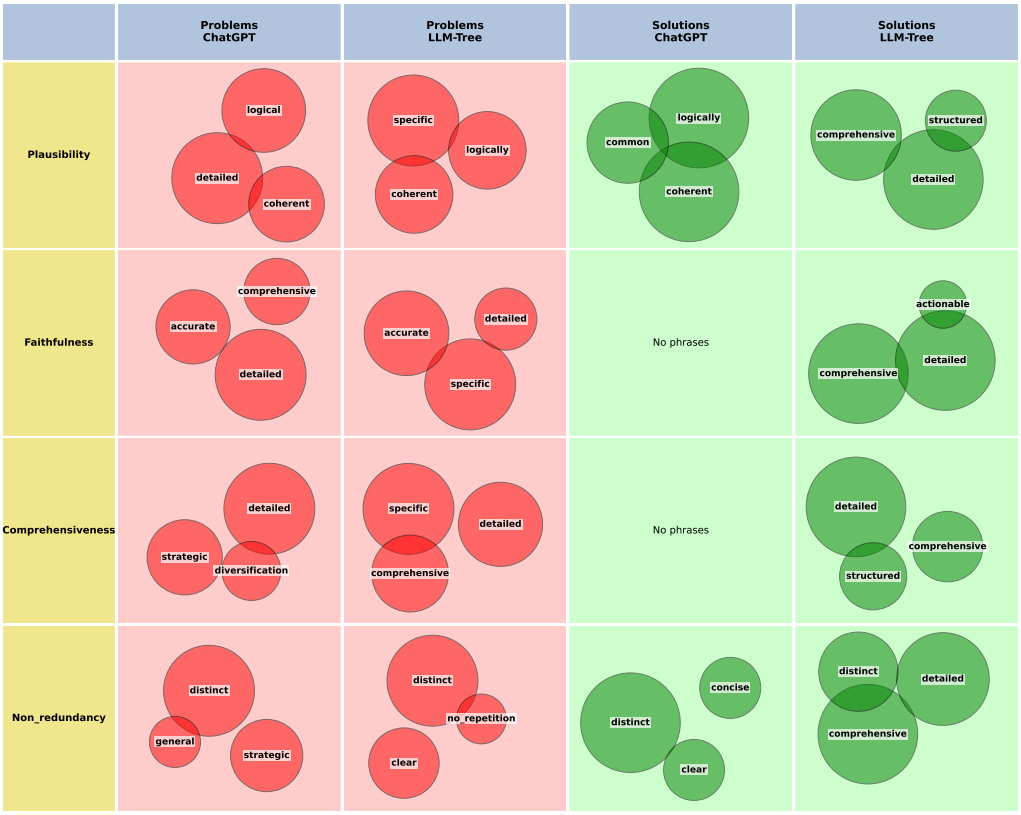

Figure 17 Bubble chart visualizing the frequency of words in judges' preference explanations.

Problem Identification

ChatGPT:

- *Plausibility*: detailed, logical, coherent
- *Faithfulness*: detailed, accurate, comprehensive
- *Comprehensiveness*: detailed, strategic, diversification
- *Non-redundancy*: distinct, strategic, general

LLM-Tree:

- *Plausibility*: specific, coherent, logical
- *Faithfulness*: specific, accurate, detailed
- *Comprehensiveness*: specific, detailed, comprehensive
- *Non-redundancy*: distinct, clear, no repetition

ChatGPT's comprehensiveness lead (70.83% vs. 29.17%) aligns with "detailed," "strategic," and "diversification," reflecting a preference for its broader scope. LLM-Tree's "specific" and "detailed" indicate a focus on targeted insights. In non-redundancy, ChatGPT's slight edge (54.17% vs. 45.83%) corresponds to "distinct" and "strategic," while LLM-Tree's deeper outputs may introduce overlap. Plausibility favors ChatGPT's "detailed" over LLM-Tree's "specific," suggesting richer context enhances believability. Faithfulness parity (50.0%) reflects similar "accurate" and "detailed" descriptors.

Solution Generation

ChatGPT:

- *Plausibility*: coherent, logically, common
- *Faithfulness*: (no notable words)
- *Comprehensiveness*: (no notable words)
- *Non-redundancy*: distinct, clear, concise

LLM-Tree:
- *Plausibility*: detailed, comprehensive, structured
- *Faithfulness*: detailed, comprehensive, actionable
- *Comprehensiveness*: detailed, structured, comprehensive
- *Non-redundancy*: comprehensive, detailed, distinct

LLM-Tree's unanimous preference in comprehensiveness and faithfulness (100.0%) and strong leads in non-redundancy (75.0%) and plausibility (89.58%) are supported by "detailed," "comprehensive," and "structured," highlighting its thorough, actionable outputs. ChatGPT's "common" in plausibility suggests generic solutions, while its lack of notable words in faithfulness and comprehensiveness underscores LLM-Tree's superiority.

Insights

In problem identification, ChatGPT excels in comprehensiveness due to its diversified approach, while LLM-Tree targets specific insights. In solution generation, LLM-Tree's detailed, structured outputs overshadow ChatGPT's simpler solutions, explaining the preference disparity. These qualitative insights enhance the interpretation of quantitative results.

6.4 User Testing Session for Human-LLM Collaboration in StrategicAI

To evaluate the Human-LLM-Collaboration feature of StrategicAI, a user testing session was conducted. This feature, identified as the third and most critical execution flow of the application, is designed to assist human experts in decision-making processes by leveraging the capabilities of LLMs. The testing aimed to assess the usability and effectiveness of this collaborative framework through real user interactions.

6.4.1 Methodology

Each participant engaged in a one-on-one testing session with me. To ensure consistency, all participants received identical preparatory information through a dedicated webpage. This webpage provided explanations of the tree structures utilized in StrategicAI and detailed descriptions of the features available for use during the testing. Individual accounts were created for each participant, and the necessary data files for their respective case studies were pre-uploaded to facilitate the experiment.

The six case studies were prepared, and each participant was assigned a unique case study to work on during the testing session, ensuring no overlap in the scenarios tested. This approach provided a comprehensive evaluation of the application's capabilities across different contexts.

Participants were instructed to utilize specific features of the application, including Agent Chat, Node Level Interaction (Node Decomposition, Node Verification), and the functionality to generate potential causes or solutions. However, they were explicitly prohibited from using the AutoRun feature of the LLM-Tree, as the focus was on evaluating the collaborative dynamics between the human user and the LLM. All of them used GPT-4o as an LLM so we can compare their results to the second experiment the only variable here between the LLM-Tree and ChatGPT is the user.

Upon completion of their respective case studies, participants completed a usability survey. This survey included the standard System Usability Scale (SUS) questions to quantitatively assess the usability of StrategicAI. Additionally, it gathered qualitative feedback on the participants' experiences and opinions regarding the application's features.

The outcomes of this user testing session, including the participants' performance on the case studies and their survey responses, will be analyzed and presented in the following sections. This analysis will include a comparison with the results of the second experiment and an evaluation of the System Usability Score to assess the overall usability of StrategicAI.

6.4.2 Analyzing Results of Human-LLM Collaboration User Testing

This subsection evaluates the outcomes of a user testing session designed to assess the Human-LLM Collaboration feature in the StrategicAI application. Six participants with diverse professional backgrounds engaged

with unique case studies from the same set used in prior experiments, utilizing features such as Agent Chat, Node Decomposition, Node Verification, and generation of potential causes or solutions. The AutoRun feature was disabled to emphasize human-LLM interaction. Participants averaged approximately one hour per session, with no time limit imposed. Their common workflow involved reviewing company data and task descriptions per the provided guide, exploring uploaded files and facts, decomposing nodes, verifying potential root causes or solutions via tools or agent chats, and selecting nodes, occasionally using candidate generation to validate decisions. Some noted the dummy company context limited online verification utility.

Results and Discussion The performance of the Human-LLM Collaboration feature (denoted as "human-llm-tree") was evaluated using both similarity-based and LLM-based approaches, consistent with the methodologies of the second experiment, allowing direct comparison with ChatGPT and the automated LLM-Tree (StrategicAI's AutoRun feature). Results are presented across the six case studies, with ChatGPT serving as a consistent baseline and LLM-Tree reflecting the refined AutoRun from Multi-Agent AutoRun Evaluation.

Similarity-Based Approach Results In the similarity-based evaluation, embeddings from `sentence-transformers/all-MiniLM-L6-v2` and cosine similarity with a 0.5 threshold were used to assess matches between model outputs and ground truth, as in prior experiments. Results are detailed in Table 12.

Method	Metric Type	Precision	Recall	F1 Score	Relevance Score	Explanation Relevance	Final Score
ChatGPT	PIS	0.43	**1.00**	0.60	0.75	0.80	0.71
ChatGPT	SGS	0.43	0.76	0.53	**0.78**	0.77	0.69
llm-tree	PIS	0.64	**1.00**	0.77	0.77	**0.82**	0.79
llm-tree	SGS	0.34	0.72	0.45	0.75	**0.80**	0.66
human-llm-tree	PIS	**0.88**	**1.00**	**0.92**	**0.80**	0.78	**0.84**
human-llm-tree	SGS	**0.85**	**0.83**	**0.82**	0.77	**0.80**	**0.80**

Table 12 Averaged Performance Metrics for ChatGPT, LLM-Tree, and Human-LLM-Tree Using the Similarity-Based Approach in the User Testing Evaluation

Human-LLM-Tree achieves a PIS of 0.84, surpassing ChatGPT's 0.71 and LLM-Tree's 0.79, driven by a precision of 0.88 (vs. 0.43 and 0.64) and perfect recall (1.00). For SGS, Human-LLM-Tree scores 0.80, exceeding ChatGPT's 0.69 and LLM-Tree's 0.66, with a precision of 0.85 (vs. 0.43 and 0.34) and recall of 0.83 (vs. 0.76 and 0.72). Relevance (0.77–0.80) and explanation relevance (0.78–0.80) remain competitive across methods, but the human-involved precision gain highlights reduced over-generation compared to LLM-Tree's automated outputs and ChatGPT's output. Moreover, as shown in Table 13, the precision for both problems and solutions is high across all individual cases that the human testers worked on. For instance, cases such as Papyr Co., Tres Burritos, and Whizzy Wilco achieved perfect precision scores for both problems and solutions, underscoring the reliability of human intervention in enhancing overall accuracy. This improvement is largely attributed to the fact that human testers were highly focused on the data within each case, carefully evaluating the validity of each problem and eliminating any issue that lacked data-driven evidence. By ensuring that all identified problems were grounded in concrete information, human testers effectively minimized false positives and enhanced the precision of the results.

Case Id (Company Name)	Problems Precision	Solutions Precision
Canyon Capital Partners	1.00	0.33
CavalierChem	0.50	1.00
Papyr Co.	1.00	1.00
Quest Diagnostics	0.75	0.75
Tres Burritos	1.00	1.00
Whizzy Wilco	1.00	1.00

Table 13 Problems and Solutions Precision for Each Case

LLM-Based Approach Results The LLM-based evaluation averaged results from four judges: `qwen2p5-72b-instruct`, `llama-v3p3-70b-instruct`, `llama-v3p1-405b-instruct`, and `mixtral-8x22b-instruct`, assessing semantic matches as in Multi-Agent AutoRun Evaluation. Results are presented in Table 14

Method	Metric Type	Precision	Recall	F1 Score	Relevance Score	Explanation Relevance	Final Score
ChatGPT	PIS	0.43	**1.00**	0.60	0.75	0.80	0.71
ChatGPT	SGS	0.40	0.76	0.50	**0.78**	0.77	0.68
llm-tree	PIS	0.59	0.94	0.71	0.77	**0.82**	0.77
llm-tree	SGS	0.40	**0.83**	0.52	0.75	**0.80**	0.69
human-llm-tree	PIS	**0.79**	0.92	**0.84**	**0.80**	0.78	**0.81**
human-llm-tree	SGS	**0.65**	0.69	**0.65**	0.77	**0.80**	**0.74**

Table 14 Averaged Performance Metrics for ChatGPT, LLM-Tree, and Human-LLM-Tree Using the LLM-Based Approach in the User Testing Evaluation, Evaluated by Four LLM Judges

Human-LLM-Tree records a PIS of 0.81, outperforming ChatGPT's 0.71 and LLM-Tree's 0.77, with precision at 0.79 (vs. 0.43 and 0.59) and recall at 0.92 (vs. 1.00 and 0.94). For SGS, it achieves 0.75, ahead of ChatGPT's 0.68 and LLM-Tree's 0.69, with precision of 0.65 (vs. 0.40 for both) and recall of 0.69 (vs. 0.76 and 0.83). The precision improvement reflects human refinement, though recall slightly lags LLM-Tree, possibly due to conservative node selection.

Usability Survey Results A usability survey with 14 questions assessed user experience. The first 10, based on the System Usability Scale (SUS), yielded an average score of 71.2. SUS scores range from 0 to 100, representing a percentile ranking rather than a percentage, with 68 as the average. A score of 71.2, above this benchmark, indicates above-average usability, roughly at the 70th percentile, suggesting StrategicAI outperforms 70% of evaluated systems.

The final four questions targeted key features, with responses visualized in Figure 18:

- **Uploaded File with Facts**: 50% found it somewhat useful, 50% very useful, valuing its role as an initial data resource despite dummy company limitations.
- **'Why' Tree**: 100% rated it very useful, praising tools like node splitting and verification for root cause analysis.
- **'How' Tree**: 100% found it very useful, appreciating its solution exploration and selection capabilities.
- **'Hypothesis' Tree**: 50% deemed it somewhat useful, 50% very useful, indicating confidence in solution testing, though less critical than other trees.

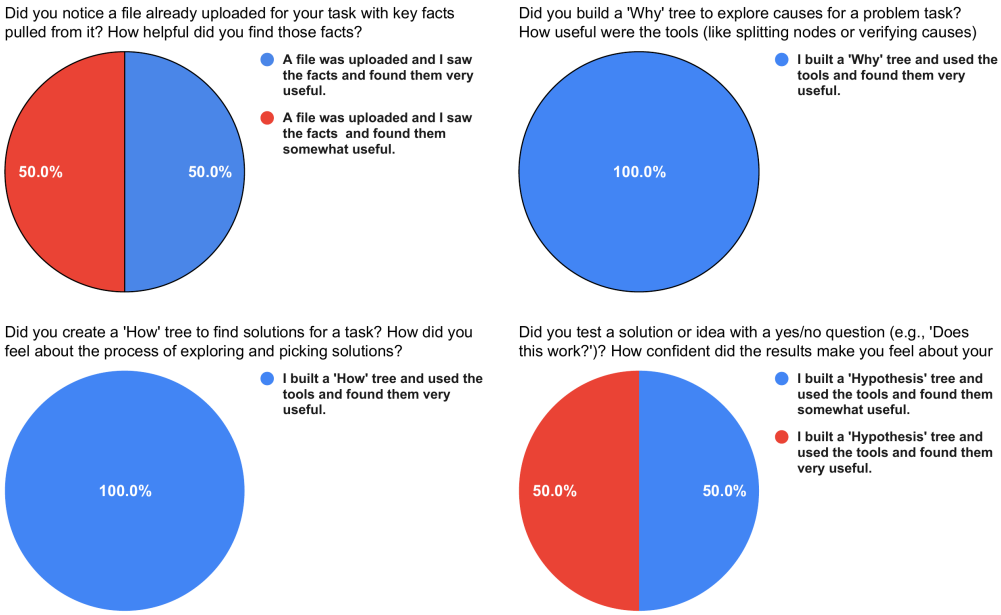

Figure 18 User responses to feature-specific questions in the usability survey for the Human-LLM Collaboration testing session.

These findings align with the high PIS and SGS scores, reinforcing the practical efficacy of tree-based tools in collaborative settings.

Discussion and Comparison with Multi-Agent AutoRun Evaluation Compared to Multi-Agent AutoRun Evaluation (Tables 9 and 10), Human-LLM-Tree significantly enhances performance over LLM-Tree. In the similarity-based approach, PIS improves from 0.79 to 0.84 (precision 0.64 to 0.88), and SGS rises from 0.66 to 0.80 (precision 0.34 to 0.85), surpassing ChatGPT's consistent 0.71 and 0.69. The LLM-based approach shows PIS increasing from 0.77 to 0.81 (precision 0.59 to 0.79) and SGS from 0.69 to 0.75 (precision 0.40 to 0.65), again exceeding ChatGPT (0.71 and 0.68). These gains stem from human oversight mitigating LLM-Tree's precision issues, as seen in Multi-Agent AutoRun Evaluation's solution over-generation. The supplementary recall analysis (0.9792 PIS, 0.8854 SGS) from Multi-Agent AutoRun Evaluation suggests tree structures capture extensive ground truth, which human users effectively refine, aligning with survey praise for the 'Why' and 'How' trees. This positions Human-LLM-Tree as a superior hybrid approach, balancing automation and expertise.

The user testing analysis demonstrates that Human-LLM Collaboration in StrategicAI outperforms both ChatGPT and the automated LLM-Tree, achieving higher PIS (0.84 sim, 0.81 llm) and SGS (0.80 sim, 0.75 llm) through enhanced precision. The SUS score of 71.2 and strong feature approval, particularly for tree tools, affirm its usability and effectiveness. This evaluation complements the Multi-Agent AutoRun Evaluation by highlighting the synergy of human and LLM capabilities, advancing StrategicAI's utility in strategic decision-making.

7 Conclusion

This section concludes the thesis by addressing the research questions introduced at the outset, summarizing the principal contributions of the study, evaluating its limitations, and proposing avenues for future investigation. The work presented herein ties back to the initial problem statement, demonstrating how the research objectives have been met and contributing to both theoretical and practical domains.

7.1 Final Answers to Research Questions

RQ1: How can an LLM-based Decision Support System (DSS) be designed to function as an effective AI consultant, enhancing organizational strategic decision-making?

StrategicAI enhances organizational strategic decision-making by integrating structured frameworks—Root Cause Analysis (Why Trees), Solution Exploration (How Trees), and Solution Validation (Hypothesis Trees)—into a modular, scalable LLM-based DSS. This design tackles key challenges:

- **Information overload** is reduced by the Fact Extraction Feature, which distills data into atomic facts, and tree structures that organize insights hierarchically.
- **Cognitive biases** (e.g., confirmation bias, overconfidence) are mitigated by the slow, deliberate thinking process, as emphasized in the research methodology. The step-by-step construction of trees, requiring users to systematically explore root causes and validate solutions with data, engages System 2 thinking, preventing intuitive leaps.
- **Incomplete root cause analysis** is addressed by the Why Trees' systematic decomposition and Node Verification Feature, ensuring thorough causal exploration.
- **Barriers to creativity** are overcome through How Trees' expansive solution generation, aided by the Collaboration Agent's suggestions.
- **Technological limitations** are resolved: lack of transparency via tree-based reasoning, inability to integrate frameworks by embedding consulting-inspired structures, and underutilization of human expertise through Human-LLM Collaboration mode.

Evaluations across six case studies show a Problem Identification Score (PIS) of 0.79 and Solution Generation Score (SGS) of 0.66 in automated mode, improving to 0.84 and 0.80 in collaborative scenarios, confirming the design's efficacy.

RQ2: How can the DSS be structured to deliver transparent and comprehensible explanations of its recommendations, thereby fostering trust and usability among decision-makers?

StrategicAI provides transparent and comprehensible explanations through its tree-based design, addressing the opacity of existing LLM tools:

- **Lack of transparency** is resolved by exposing data-driven justifications at each tree node.
- **Cognitive biases** are mitigated by the slow thinking process outlined in the research methodology. Tracing the reasoning from root causes to solutions via tree structures allows users to critically evaluate each step, reducing biases like confirmation bias.
- **Information overload** is alleviated by presenting concise, justified insights.
- **Underutilization of human expertise** is addressed by the Collaboration Agent's natural language explanations, enabling interactive clarification.

Real-time WebSocket updates enhance visibility, and certainty scores in Hypothesis Trees reduce overconfidence by offering objective confidence measures. User feedback, with 100% rating tree tools as "very useful," highlights their clarity and trust-building impact.

RQ3: How can the DSS be engineered to ensure reliability, minimize output errors, and maintain consistency across diverse strategic contexts?

StrategicAI ensures reliability and consistency through a multi-agent architecture and adaptable frameworks:

- **Incomplete root cause analysis** is tackled by the multi-agent system's specialized task assignments, minimizing oversight.
- **Cognitive biases** are mitigated by the slow thinking process from the research methodology. The multi-agent breakdown of decision-making into discrete, validated steps—using tree structures to reach root causes and solutions—promotes deliberate analysis, reducing errors like premature judgment.

- **Information overload** is countered by integrating data from multiple sources into structured trees.
- **Inability to integrate frameworks** is resolved by adaptable, consulting-inspired tree designs.
- **Underutilization of human expertise** is addressed via Human-LLM Collaboration mode, leveraging human oversight.

Evaluations demonstrate recall rates of 0.9792 for problems and 0.8854 for solutions, with improved scores in collaborative modes, affirming dependability across diverse strategic scenarios.

7.2 Summary and Importance of the Work

This study developed StrategicAI, a web-based DSS that leverages large language models (LLMs) to convert raw data into actionable strategic insights. By mitigating challenges such as information overload, cognitive biases, and limited creativity. StrategicAI addresses a critical gap in organizational decision-making. The significance of this research is twofold. Theoretically, it enriches the understanding of AI-supported decision-making by introducing a transparent, framework-driven approach. Practically, it equips organizations with a robust tool to tackle complex strategic challenges. The empirical evaluations, particularly those involving human-AI collaboration, underscore the system's potential to elevate decision quality and contribute to organizational success.

7.3 Limitations of the Research

Despite its contributions, this research is subject to certain limitations, which present opportunities for further exploration:

- **Scope of Case Studies**: The evaluation was confined to six case studies. Although these were diverse but the case with profitability type prevailed,so, they may not fully encapsulate the spectrum of strategic decision-making scenarios. Expanding the scope could strengthen the generalizability of the findings.
- **Certainty Score Calculation**: Inconsistencies in the meaning of certainty scores were observed, suggesting a need for refinement in the methodological approach to better reflect decision confidence and the certainty of the node being a problem or a solution and not only about its content and location in the tree.

These limitations do not undermine the research's value but rather delineate areas warranting additional investigation.

7.4 Future Work

Future research offers exciting opportunities to extend this study by addressing the identified limitations and exploring innovative enhancements that will elevate StrategicAI's capabilities. The promising results achieved thus far provide a strong foundation, underscoring the system's potential to become an even more powerful and versatile tool for strategic decision-making. Continued development in the following areas will ensure that StrategicAI remains at the forefront of AI-driven decision support systems:

- **Refining Certainty Score Calculations**: A key avenue for improvement lies in enhancing the reliability and precision of certainty scores. Future work could explore the integration of advanced probabilistic models, machine learning techniques, or domain-specific adaptations tailored to the decision-making context. Collaborating with experts in statistics or decision theory could yield more robust methodologies, ensuring that certainty scores consistently and accurately reflect decision confidence. This refinement will strengthen StrategicAI's ability to provide dependable insights, further solidifying its value to users.
- **Improving Usability and UI/UX**: Enhancing the application's usability is a critical direction with tremendous potential. This could involve integrating full Create, Read, Update, Delete (CRUD) functionality to empower users with greater control over data and decision workflows. Additionally, refining the user

interface and user experience (UI/UX) through extensive user testing and feedback sessions will make StrategicAI more intuitive and accessible. Iterative design improvements, informed by real-world user interactions, can address pain points, cater to diverse user expertise levels, and ensure accessibility standards are met. These enhancements will transform the application into a seamless and enjoyable tool, broadening its appeal and practical impact.

- **Enhancing Referencing and Linking Features**: Transparency and trust are cornerstones of StrategicAI's design, and future work can take this further by developing a comprehensive referencing and linking system. Currently, the system includes basic source references (e.g., "Source: data.txt") when editing explanations. Building on this, future efforts could enable direct access to specific facts or data points from uploaded files, allowing users to click through to the original information for verification. Additionally, integrating hyperlinks to online sources or search results would provide immediate access to external references, enhancing transparency. This could involve automatic citation generation or embedding interactive elements within the tree structures, enabling users to trace the origin of each piece of information seamlessly. Such advancements will reinforce user confidence and position StrategicAI as a leader in transparent AI systems.

- **Expanding System Capabilities**: The modular and scalable nature of StrategicAI opens the door to exciting expansions, such as connecting to multiple data sources like company drives or cloud storage. This would allow the system to pull files directly from organizational repositories, streamlining data integration and ensuring the use of up-to-date information in decision-making. Furthermore, introducing new tree structures tailored to quantitative data—such as nodes that perform calculations, visualize trends, or integrate with analytics tools—would enable StrategicAI to handle numerical analyses more effectively. These enhancements would create a more holistic decision-support ecosystem, catering to both qualitative and quantitative aspects of strategic planning. Future efforts could also develop specialized modules for different sectors (e.g., healthcare, finance, or logistics), amplifying the system's applicability across diverse strategic contexts.

These proposed developments build upon the robust foundation established in this thesis, highlighting the immense promise of StrategicAI. The system's current achievements—demonstrated through empirical evaluations and its innovative design—signal a clear path forward for continued innovation. Refining certainty calculations, enhancing usability and UI/UX, improving referencing features, and expanding capabilities will transform StrategicAI into a more comprehensive, effective, and user-centric tool. The need for further work is not a limitation but a testament to the system's potential, positioning it as a dynamic platform ripe for growth and refinement. The insights gained from this research illuminate a bright future for AI-driven decision support, with StrategicAI poised to lead the way in delivering transparent, reliable, and impactful strategic solutions.

In conclusion, this thesis demonstrates that an LLM-based DSS can enhance strategic decision-making when designed with structured frameworks and human collaboration. StrategicAI advances the theoretical understanding of AI-supported decision processes while delivering a practical solution that enables organizations to make informed, data-driven decisions in complex environments. By addressing the challenges outlined in the introduction, this research establishes a foundation for future innovations in AI-driven decision support, setting a new benchmark for transparency, reliability, and effectiveness in strategic decision-making.

References

1000minds. (2023). What is human-in-the-loop (hitl) in ai-assisted decision-making. *1000minds Articles*.

Acciarini, C., Brunetta, F., & Boccardelli, P. (2021). Cognitive biases and decision-making strategies in times of change: A systematic literature review. *Management Decision*, *59*(10), 2386–2408.

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, *6*, 52138–52160.

Anthony, S. D. (2016). Kodak's downfall wasn't about technology. *Harvard Business Review*.

Benary, M., Wang, X. D., Schmidt, M., Soll, D., Hilfenhaus, G., Nassir, M., Daniel, B., & Klauschen, F. (2023). Leveraging large language models for decision support in personalized oncology. *Nature Precision Oncology*, *7*(121).

Besta, M., Memedi, F., Zhang, Z., Gerstenberger, R., Piao, G., Blach, N., & Hoefler, T. (2024). Demystifying chains, trees, and graphs of thoughts. *arXiv preprint arXiv:2401.14295*.

Bolton, C., Machová, V., Kovacova, M., & Valaskova, K. (2018). The power of human-machine collaboration: Artificial intelligence, decision support, and process optimization. *Journal of Business and Management*, *10*, 22–35.

Brooke, J. (1996). Sus: A "quick and dirty" usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & A. L. McClelland (Eds.), *Usability evaluation in industry* (pp. 189–194). Taylor Francis.

Brown, T., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Cao, S., Zhang, J., Shi, J., Lv, X., Yao, Z., Tian, Q., & Hou, L. (2023). Probabilistic tree-of-thought reasoning for answering knowledge-intensive complex questions. *arXiv preprint arXiv:2311.13982*.

Careers, T. B. (2023). 6 big challenges managers and organizations face with data.

Chen, X., Jiang, L., Weng, Y., Zhang, Y., Lin, Z., Zhang, Y., & Li, C. (2023). Universal self-consistency for large language model generation [Accessed: February 20, 2025]. *arXiv preprint arXiv:2311.17311*.

Cheng, V. (2024). Issue tree. *CaseInterview*.

Choo, A. S., Linderman, K. W., & Schroeder, R. G. (2007). Method and psychological effects on learning behaviors and knowledge creation in quality improvement projects. *Management Science*, *53*(3), 437–450.

Collis, D. J., & Rukstad, M. G. (2013). Making better strategic decisions. *Harvard Business Review*, *91*(7/8), 100–107.

Cooper, R. G., & Kleinschmidt, E. J. (1995). Benchmarking the firm's critical success factors in new product development. *Journal of Product Innovation Management*, *12*(5), 374–391.

Csaszar, F. A., Ketkar, H., & Kim, H. (2024). Artificial intelligence and strategic decision-making: Evidence from entrepreneurs and investors. *arXiv preprint arXiv:2408.08811*.

Dellermann, D., Calma, A., Lipusch, N., Weber, T., Weigel, S., & Ebel, P. (2020). On the current state of combining human and artificial intelligence for strategic organizational decision making. *Business Research*, *13*, 875–919.

Eigner, I., & Händler, T. (2024). Determinants of llm-assisted decision-making: Transparency, trustworthiness, and prompt engineering [Examines challenges in deploying LLMs, focusing on transparency, trustworthiness, and effective prompt design.]. *Information Systems Research*, *35*(2), 278–299.

Felzmann, H., Fosch-Villaronga, E., Lutz, C., & Tamò-Larrieux, A. (2020). Transparency and explainability of ai systems: From ethical guidelines to requirements. *Information and Software Technology*, *159*, 107205.

Focused Momentum. (2023). Strategic planning [Accessed: 20 February 2025].

FourWeekMBA. (2023). 43 consulting frameworks and methodologies you need to know.

Hackathorn, J., Ashdown, B. K., & Rife, S. C. (2024). Large language models present new questions for decision support. *Journal of Decision Systems*, 1–18.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75–105.

How artificial intelligence will transform decision-making [Accessed: 2023-10-01]. (2023).

Hulbert, J. (2023). The revolutionary approach of tree-of-thought prompting in ai. *Medium*.

Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Kivlichan, I., et al. (2024). Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

The impact of artificial intelligence on financial decision making [Accessed: 2023-10-01]. (2024).

Jarrahi, M. H. (2018). Artificial intelligence and the future of work: Human-ai symbiosis in organizational decision making. *Business Horizons*, *61*(4), 577–586.

Johnson, M. W., & Suskewicz, J. (2009). How to disrupt yourself. *Harvard Business Review*, *87*(10), 50–58.

Kahneman, D. (2011). *Thinking, fast and slow*. Farrar, Straus; Giroux.

Kenney, L. (2024). Issue tree: The complete guide with examples 2024. *My Consulting Offer*.
Kumar, U., & Kumar, D. (2004). Maintenance strategy selection and system effectiveness: A case study. *Journal of Quality in Maintenance Engineering*, *10*(4), 254–261.
Lakhani, K. R. (2016). Apple's ecosystem strategy. *MIT Sloan Management Review*, *57*(4), 1–5.
Leonard-Barton, D. (1992). The factory as a learning laboratory. *Sloan Management Review*, *34*(1), 23–38.
Liu, O., Fu, D., Yogatama, D., & Neiswanger, W. (2024). Dellma: A framework for decision making under uncertainty with large language models. *arXiv preprint arXiv:2402.02392*.
Lu, Y., Hu, Y., Foroosh, H., Jin, W., & Liu, F. (2024). Strux: An llm for decision-making with structured explanations. *arXiv preprint arXiv:2410.12583*.
Lucas, H. C., & Goh, J. M. (2009). Disruptive technology: How kodak missed the digital photography revolution. *The Journal of Strategic Information Systems*, *18*(1), 46–55.
Lyons, J. B., & Havig, P. R. (2017). Transparency in a human-machine context: Approaches for fostering shared awareness. *International Journal of Human-Computer Interaction*, *33*(4), 317–325.
Mankins, M. (2023). How large language models reflect human judgment. *Harvard Business Review*.
McAfee, A., & Brynjolfsson, E. (2012). Big data: The management revolution. *Harvard Business Review*, *90*(10), 60–68.
MConsultingPrep. (2024). Issue tree in consulting: A complete guide (with examples). *MConsultingPrep*.
Meinel, C., et al. (2012). Work environment barriers prohibiting creativity. *Procedia - Social and Behavioral Sciences*, *40*, 642–648.
Monarch, R. (2022). Human-in-the-loop machine learning.
Mumford, M. D., Mobley, M. I., Uhlman, C. E., Reiter-Palmon, R., & Doares, L. M. (1991). Process analytic models of creative capacities. *Creativity Research Journal*, *4*(2), 91–122.
Netflix market cap 2010-2024 | nflx [Accessed: 2023-10-01]. (n.d.).
Nogueira, B. (2024). Issue trees: The definitive guide with in-depth examples. *Crafting Cases*.
Nooraie, M. (2021). *Strategic management, decision theory, and decision science* [Accessed: 20 February 2025]. Springer.
Novelli, C., Casolari, F., Rotolo, A., Taddeo, M., & Floridi, L. (2023). Transparency and accountability in ai systems: Safeguarding wellbeing in the age of algorithmic decision-making. *Frontiers in Human Dynamics*, *6*, 1421273.
Oxford Law Blogs. (2024). Putting a human in the loop: Increasing uptake, but decreasing accuracy of automated decision-making. *Oxford Business Law Blog*.
Parniangtong, S. (2017). Problem-solving approach to business strategy. In *Competitive advantage of customer centricity* (pp. 69–87).
Parry, K., Cohen, M., & Bhattacharya, S. (2016). Rise of the machines: A critical consideration of automated decision making. *Organizational Dynamics*, *45*(3), 171–178.
Peerally, M. F., Carr, S., Waring, J., & Dixon-Woods, M. (2017). The problem with root cause analysis. *BMJ Quality & Safety*, *26*(5), 417–422.
Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77.
Porter, M. E. (1996). What is strategy? *Harvard Business Review*, *74*(6), 61–78.
PrepLounge. (2024). All you need to know about the issue tree. *PrepLounge*.
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.
Rasiel, E. M., & Friga, P. N. (2001). The mckinsey mind: Understanding and implementing the problem-solving tools [Available at: https://archive.org/details/the-mc-kinsey-mind-understanding-and-implementing-the-problem-solving-tools-and-$_2$02312]. *McGraw-Hill*.
Rathindran, S. (2018). Reduce the staggering costs of poor operational decisions [Accessed: 2023-10-01].
Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3982–3992.
Rittel, H. W., & Webber, M. M. (1984). Planning problems are wicked problems (N. Cross, Ed.).

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, *8*, 842–866.
Rooney, J. J., & Heuvel, L. N. V. (2004). Root cause analysis for beginners. *Quality Progress*, *37*(7), 45–53.
Rousseau, D. M. (2018). Making evidence-based organizational decisions in an uncertain world. *Organizational Dynamics*, *47*(3), 135–141.
Schoemaker, P. J. H. (1995). Scenario planning: A tool for strategic thinking. *Sloan Management Review*, *36*(2), 25–40.
ScienceDirect. (2023). Decision theory [Accessed: 20 February 2025].
Scientist, D. (2023). Challenges and solutions in big data management.
Serrano, M. (2024). Issue trees: How to use them in case interviews? *IGotAnOffer*.
Shahriar, S. (2024). Ai in supply chain optimization: A review [Forthcoming].
Shahrzadi, Z., et al. (2024). Causes, consequences, and strategies to deal with information overload: A scoping review. *International Journal of Information Management Data Insights*, *4*(2), 100050.
Shank, D. B., DeSanti, A., & Maninger, T. (2019). When are artificial intelligence versus human agents faulted for wrongdoing? *International Journal of Human-Computer Studies*, *125*, 44–53.
Simon, H. A. (1996). *The sciences of the artificial* (3rd). MIT Press.
Sinnaiah, T., Adam, S., & Mahadi, B. (2023). A strategic management process: The role of decision-making style and organisational performance. *Journal of Work-Applied Management*, *15*(1), 37–50.
SmythOS. (2024). Human-in-the-loop ai: Enhancing decision-making with human oversight. *SmythOS Blog*.
Spetzler, C., Winter, H., & Meyer, J. (2016). *Decision quality: Value creation from better business decisions*. John Wiley & Sons.
Stanford Encyclopedia of Philosophy. (2023). Decision theory [Accessed: 20 February 2025].
Stratechi. (2024). How to develop your hypothesis tree: Free ppt template. *Online Article*.
The Decision Lab. (2023). Decision theory [Accessed: 20 February 2025].
the Case Interview, H. (2025). Issue trees: Step-by-step guide with examples (2025). *Hacking the Case Interview*.
Thomke, S. H. (2001). Enlightened experimentation: The new imperative for innovation. *Harvard Business Review*, *79*(2), 66–75.
Tichy, W. F. (1998). Should computer scientists experiment more? *Computer*, *31*(5), 32–40.
Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2023). Self-consistency improves chain of thought reasoning in language models [Accessed: February 20, 2025]. *International Conference on Learning Representations (ICLR)*.
Weidinger, L., Mellor, J., Uesato, J., Lancaster, J., Rauh, M., Griffin, C., Kenton, Z., Gabriel, I., Hughes, C., Kasirzadeh, A., et al. (2021). Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.
Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.
Yuksekgonul, M., et al. (2023). Post-hoc interpretability for neural nlp: A survey. *arXiv preprint arXiv:2301.12345*.
Zhang, Q., et al. (2022). The human in the infinite loop: A case study on revealing and explaining human-ai interaction loop failures. *arXiv preprint arXiv:2207.12761*.
Zhang, Z., Zhang, A., Li, M., & Smola, A. (2022). Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.
Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., Wang, S., Yin, D., & Du, M. (2024). Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, *15*(2), 1–38.

8 Appendices

StrategicAI Screenshots:

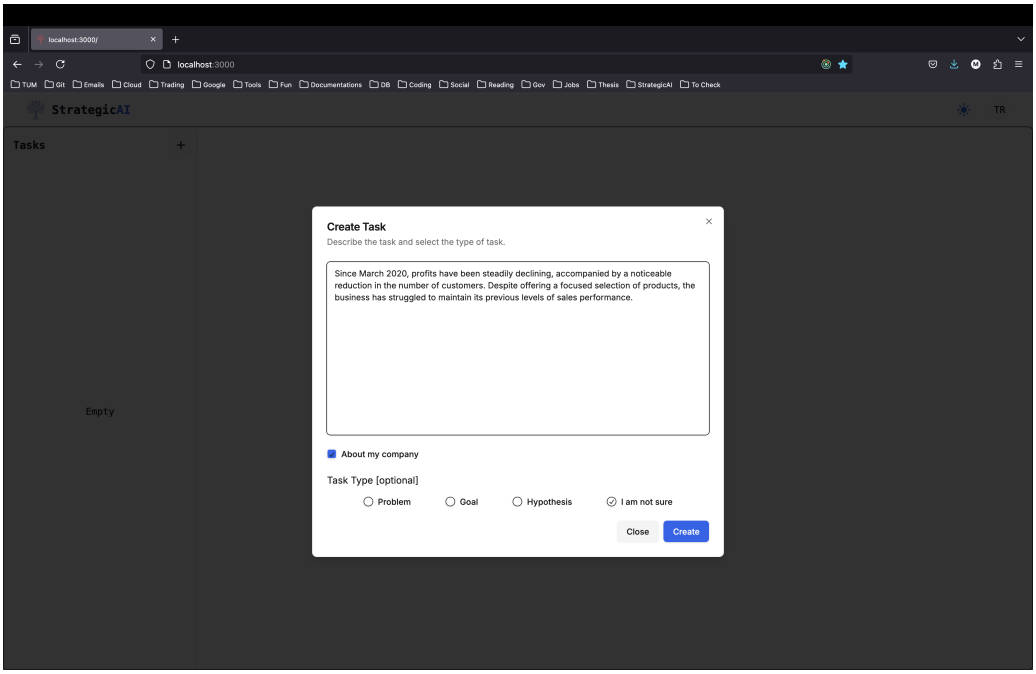

Figure 19 Task Creation Dialog

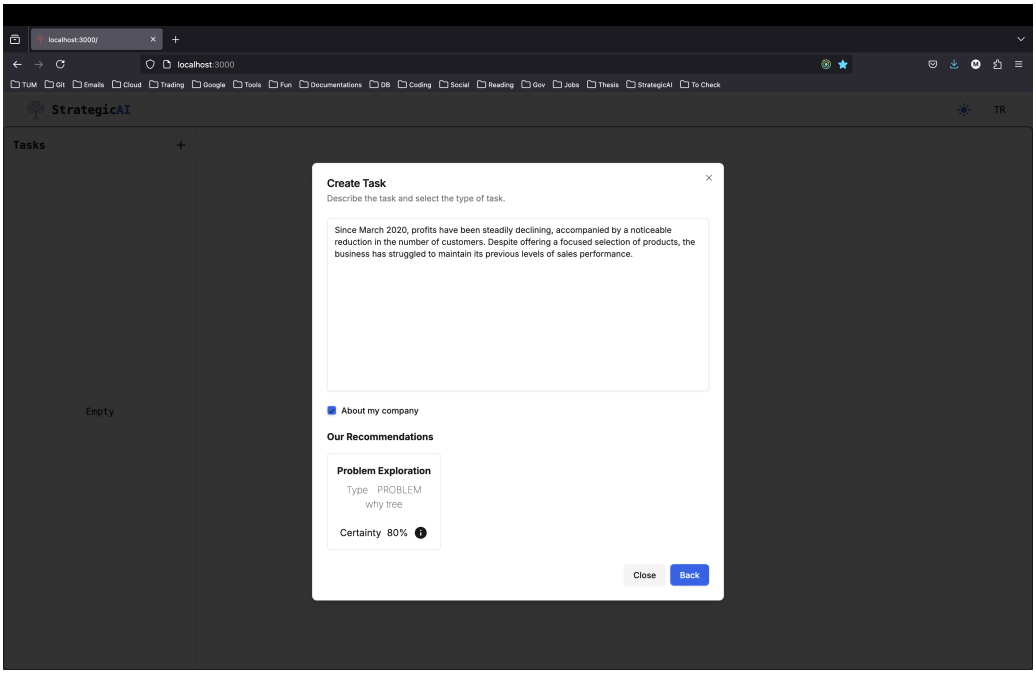

Figure 20 Task Creation Recommendation Dialog

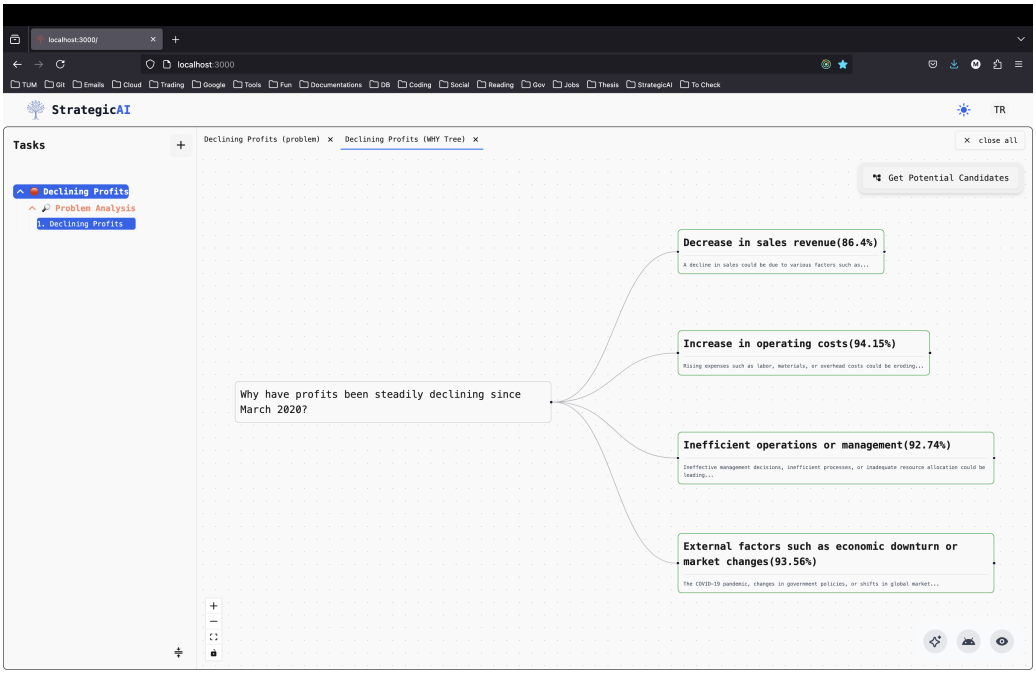

Figure 21 Initial Why Tree with the first layer

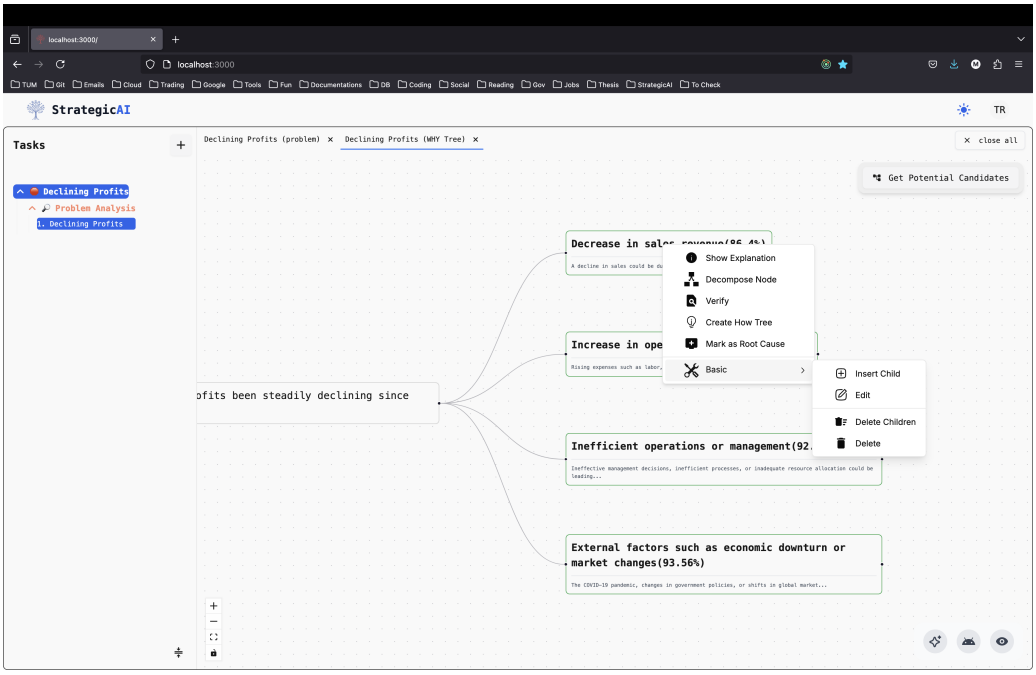

Figure 22 Node options for a node in the why tree

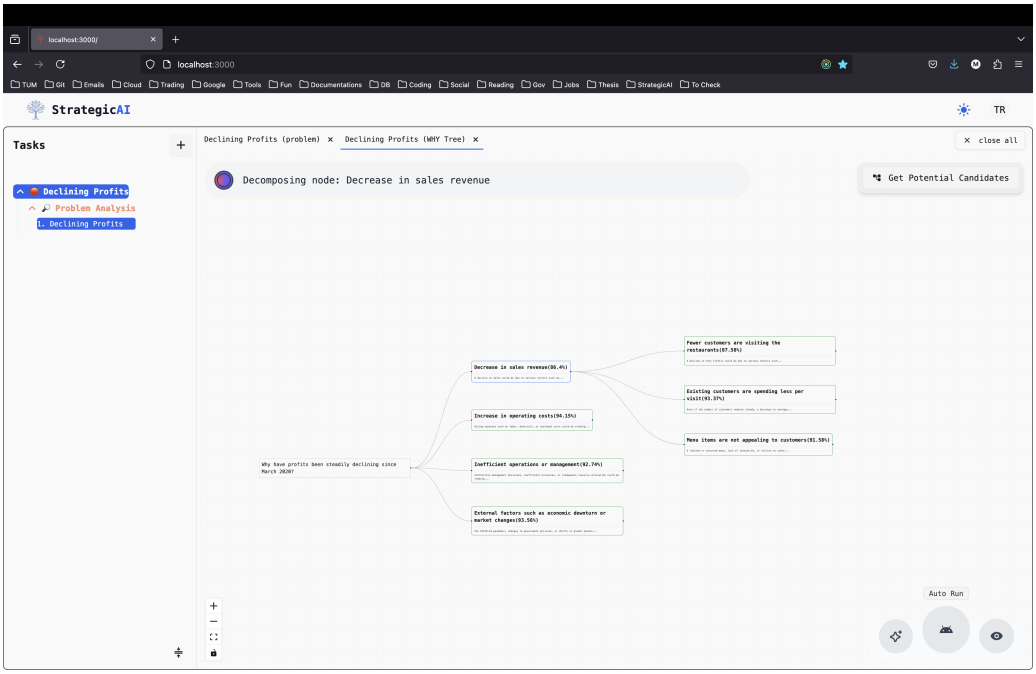

Figure 23 AutoRun in progress with the notification bar

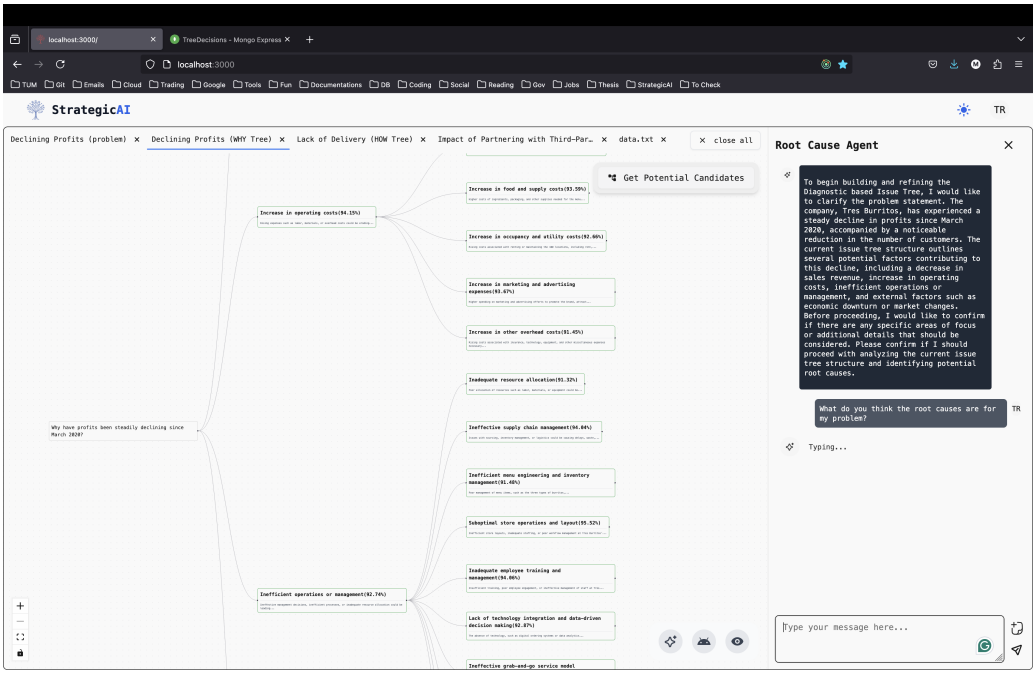

Figure 24 Root Cause Agent Chat in why tree

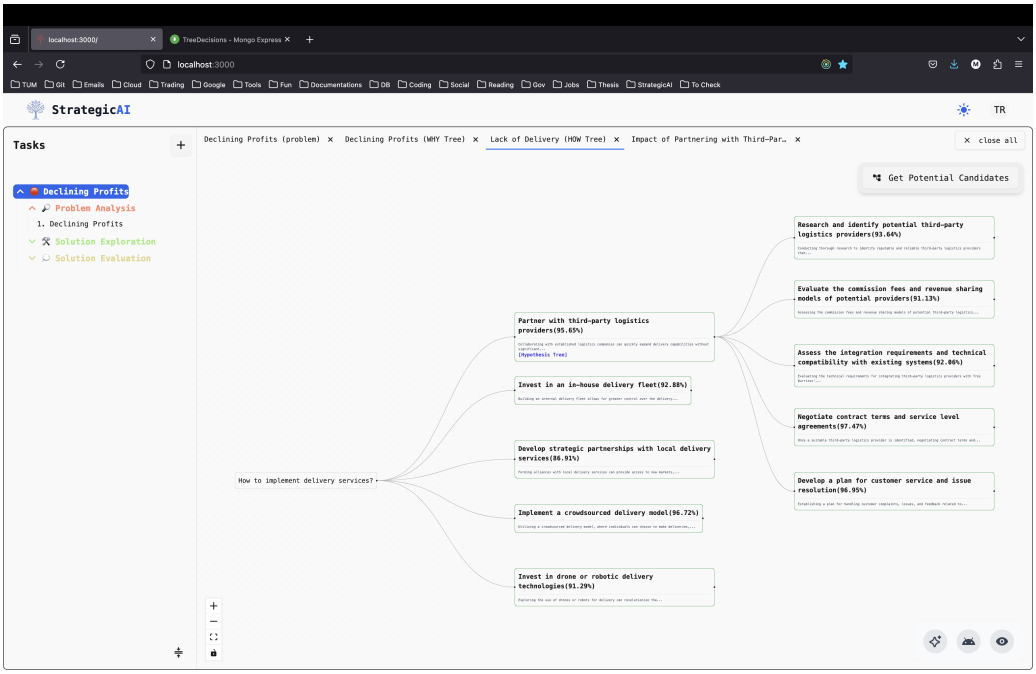

Figure 25 How tree for a root cause

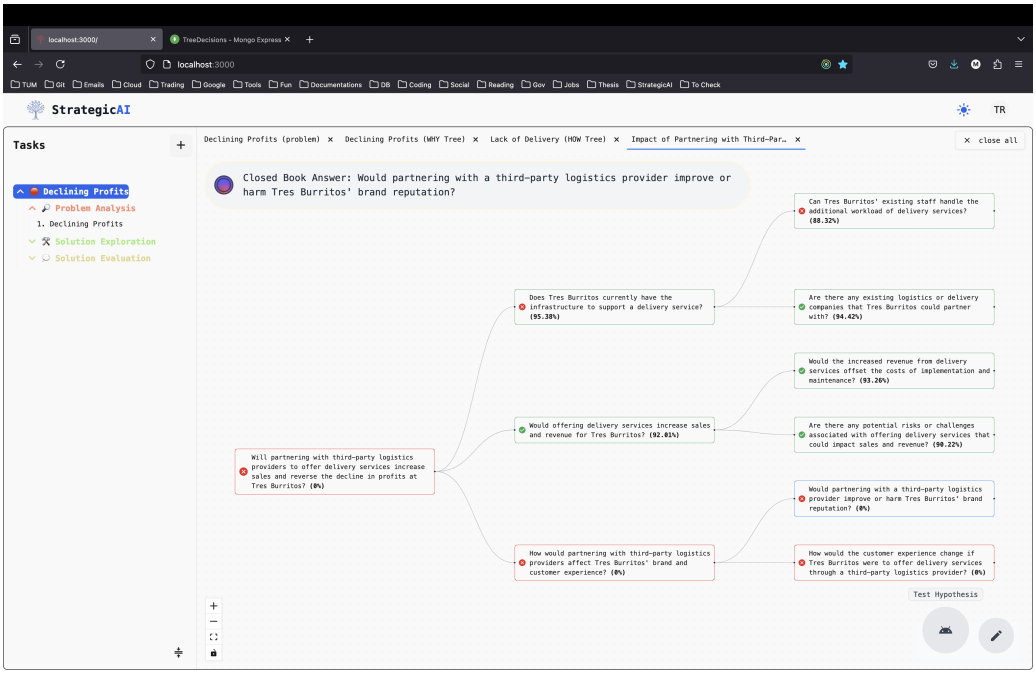

Figure 26 Hypothesis tree for a solution from a how tree in progress while hypothesis testing is running

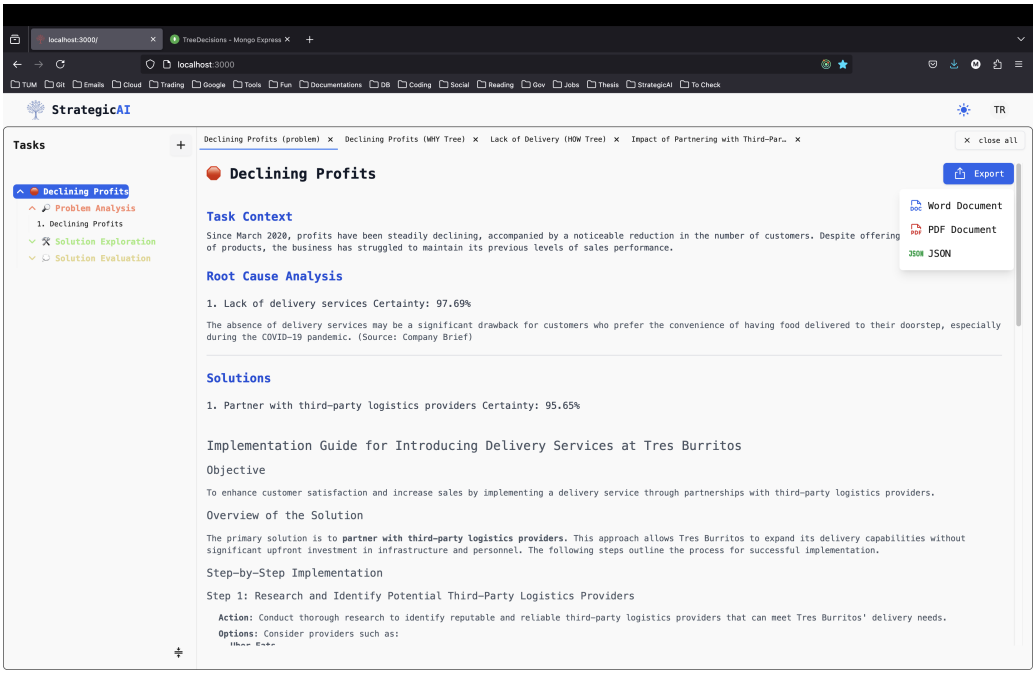

Figure 27 Task details page