Compliance Aware, Agentic Pitch Book Generation For Bankers

A template-aware PowerPoint generation system with agentic data retrieval for investment banking.

A Queen Mary Final Year Dissertation

Andrei Rizea

Student ID: 220300153

Supervised by: Manesha Peiris

Programme of Study:

Digital & Technology Solutions (Software Engineering)

Module: IOT635W

Queen Mary University of London

January 20, 2026

Abstract

Contents

List of Figures

List of Tables

Introduction, Scope and Context

McKinsey found knowledge workers spend a fifth of their time searching for information (McKinsey Global Institute 2012). Investment banking (IB) is worse. By some estimates, document preparation eats up half to three-quarters of a junior banker's week, and pitch books (PBs) account for a big chunk of that (Corporate Finance Institute 2025; Wall Street Oasis 2024). When Goldman Sachs surveyed its first-year analysts in 2021, the average was 98 hours a week; preparing documents was a major driver of the burnout they reported (BBC News 2021). Yet most PBs look nearly identical to one another. Change the company name and update the numbers, and you could reuse last month's deck. This repetitiveness makes them well suited to automation.

Where Analyst Time Goes

IB sells expertise rather than physical goods, so you would expect analyst time to go toward analysis and client relationships. Often it does not.

A typical PB contains an executive summary, company overview, market analysis, valuation work, transaction comparables, and next steps. These sections appear in nearly every deck, along with the same content types: company descriptions, financial tables, market charts, and deal diagrams. The data inside changes, but the shell around it rarely does.

Analyst tasks fall into two buckets. The first holds work that genuinely needs a trained banker: financial modelling, valuation analysis, due diligence, and client advisory. The second holds everything else: hunting down data, reformatting slides, and fixing fonts. Too many hours go into the second bucket.

Banks pay junior analysts well, so time spent on low-skill tasks represents a poor return on that investment. Shifting even some of that document preparation time back toward analytical work would improve the economics. This is not about replacing analysts; it is about freeing them to do analyst work.

Research supports this view. Radhakrishna et al. found that well-designed knowledge management systems boosted productivity by 20 to 25 percent (Radhakrishna et al. 2024). Document production was identified as a prime candidate for automation, provided humans retained control over judgement calls.

Problem Analysis

Markus defined knowledge reuse in 2001, and her framework fits PB production well (Markus 2001). She identified several reuse patterns: drawing on past work, borrowing from colleagues, finding subject matter experts, and mining old documents. PB creation involves all of these, often in the same afternoon.

Three things get in the way.

First, too much time goes to low-value work. Analysts spend hours searching through old decks, extracting useful content, and reformatting slides to fit the current template. None of that requires analytical thinking, yet it crowds out the work that does.

Second, institutional knowledge tends to disappear. Researchers distinguish between tacit knowledge, which stays in people's heads, and explicit knowledge, which gets written down (Dalkir 2011). PBs themselves are explicit since they capture analysis in a shareable format. But the knowledge of how to make a good PB remains tacit: which templates work for which situations, how to structure the narrative, what separates a persuasive pitch from a forgettable one. New analysts pick this up by watching senior colleagues, not by reading a manual.

Third, onboarding takes longer than it should. Each bank has its own templates, preferred data sources, writing conventions, and quality expectations. Junior analysts absorb these through trial and error, which is slow for them and a drain on the senior bankers who review their work.

These three problems reinforce each other. When knowledge is scattered, people waste time searching for it. When standards live in people's heads rather than in documents, output quality varies and review cycles drag on. When training happens informally, experienced staff lose hours to coaching that could have gone toward billable work. A system that tackled even one of these issues would help, but one that addressed all three could meaningfully change how PB production works.

Gap Analysis

Automated presentation tools have existed for years. Early versions converted text to slides using layout algorithms, but although the content was organised, the slides looked bad and nobody wanted to use them (Zheng et al. 2025). Template-based systems came next, where you define the structure upfront and the system fills in the blanks. That fixed the visual problems but made everything rigid.

PPTAgent takes a different approach (Zheng et al. 2025). It analyses reference presentations, extracts their structural patterns, and applies those patterns to new content. Because it learns from real examples rather than following hardcoded rules, it beat older text-to-slide systems on both content quality and design quality in testing.

The catch is that PPTAgent targets general-purpose presentations, and IB PBs are anything but general. They have financial tables with specific column layouts, transaction comparables formatted in particular ways, and market positioning charts that follow visual conventions the industry recognises. Generic tools miss these conventions entirely, and in IB, getting the template wrong kills credibility. You could have the best analysis in the world, but if the deck looks off, nobody will take it seriously.

Commercial tools share this flaw. Beautiful.ai, Tome, and Gamma can all generate slides from a prompt, but the output is generic and every slide needs rework to match firm standards. None of them connect to financial data sources either, so analysts still have to gather numbers and enter them by hand.

Here is the gap. Current systems either produce generic output that needs heavy cleanup, or they require large training datasets to learn industry-specific patterns. Most banks lack those datasets. What nobody seems to have built yet is a system that can take a single reference template, extract its styling and layout rules programmatically, and generate new slides that follow those rules exactly. That kind of template-aware approach would let firms preserve their visual identity while offloading the tedious formatting work.

Agentic AI architectures look promising for this (IBM 2024). Instead of one model doing everything in a single pass, the system breaks tasks into steps and uses the right tool for each one. Wang et al. surveyed LLM-based agents and found that progress has been rapid (Wang et al. 2024). McKinsey estimates generative AI could deliver $340 billion annually in banking (McKinsey & Company 2024). Document-heavy workflows sit right in the sweet spot.

My project addresses this gap. The pipeline extracts layouts, colour palettes, and placeholder positions from a template using python-pptx, uses an LLM to plan content, then assembles slides matching the original style. I am using existing agentic frameworks rather than custom orchestration. RAG against past PBs is a stretch goal.

Aims and Objectives

The aim is to design, build, and test a proof-of-concept (POC) demonstrating how AI-powered document generation can tackle the knowledge reuse problems described above. The system must respect institutional templates throughout; if it generates slides that do not match the house style, the whole thing fails.

On the business side, I want a workflow where an analyst can provide minimal input and get back a solid first draft. That would cut the time spent on formatting grunt work while keeping humans in control of the actual analysis. The target users are junior analysts, the people who currently spend most of their hours on document preparation.

On the learning side, I want to understand how to connect LLM orchestration with programmatic document generation. That means getting into agentic architectures, learning how to extract patterns from templates, and working out how to combine multiple AI capabilities into something that functions end to end. I am also interested in human-in-the-loop (HITL) design, since in a regulated industry like finance, AI tools need to support professional judgement rather than try to replace it.

Five objectives structure the work:

1. Build a template analysis component that takes a reference PowerPoint file and extracts its layout structures, colour palettes, font specifications, and placeholder positions.
2. Build an agentic data retrieval layer that automatically pulls company financials, market data, and regulatory filings from Yahoo Finance and SEC EDGAR.
3. Build an LLM-powered content planning module that produces structured slide specifications adapted to different PB types while keeping the overall narrative coherent.

4. Build a slide assembly component that takes the content plan and template patterns and produces a PowerPoint file meeting formatting standards.
5. Evaluate the system by measuring generation speed, template matching, and content quality.

Scope and Success Criteria

The POC takes user input (company, transaction type, dates, reference template, PB type) and produces a formatted PowerPoint. Target PB types are company overviews, market updates, and transaction summaries. Full valuation presentations are out of scope.

Design choices: established agentic frameworks rather than custom architecture; programmatic template extraction with python-pptx rather than vision models; public APIs only; RAG as a stretch goal. Out of scope: real-time data feeds, production infrastructure, compliance validation. Deal positioning and valuation judgements stay with analysts. Practical limits: one academic term, standard hardware, student API budget.

Success criteria:

- Generate complete PB draft within five minutes
- Match reference template formatting
- Accurate financial data from source APIs
- Content structure following IB conventions, verified against precedent examples

One principle runs through all of this: HITL design. The system produces drafts for human review, not finished products. The goal is a solid starting point rather than a blank slide.

Methodology and Project Plan

This section covers how I will approach development, what the requirements are, which technologies I have chosen and why, the project schedule, and how I am thinking about risks and ethics.

Development Methodology

Software development methodologies range from plan-driven approaches like Waterfall to adaptive approaches like Agile (Sommerville 2016). Choosing the right methodology depends on requirement stability, project complexity, stakeholder availability, and team size. Table 1 compares methodologies against criteria relevant to this project.

Criterion	**Waterfall**	**Scrum**	**Iterative Prototyping**
Requirements stability	Requires fixed requirements upfront	Accommodates changing requirements	Works well when requirements change
Feedback loops	Late feedback after implementation	Sprint reviews every 2-4 weeks	Continuous feedback through prototypes
Risk management	Risks discovered late	Regular risk reassessment	Early risk identification through prototypes
Solo developer suitability	Moderate	Low (designed for teams)	High
Research integration	Poor	Moderate	Excellent

Table 1: Comparison of Development Methodologies

I will use a hybrid methodology combining iterative prototyping with structured research engineering practices. Pure Waterfall cannot accommodate the exploratory nature of this work; full Scrum adds overhead without benefit for a solo developer. Iterative prototyping preserves flexibility while layered engineering practices ensure reproducibility.

Each two-week iteration will produce a working prototype targeting defined requirements. At each cycle's end, I will conduct a retrospective review and document outcomes. Feedback from fortnightly supervisor meetings and, where possible, IB practitioners will shape subsequent cycles.

I will use Git with a simplified GitFlow model (main, develop, feature branches). Continuous integration via GitHub Actions will run tests and linting on every push, enforcing 80% coverage for core modules. I will maintain a decision log using Architecture Decision Records (ADRs) to document technical choices and trade-offs.

Requirements Analysis

Good requirements are needed to measure whether the project succeeded. This section separates business requirements (the value delivered) from functional requirements (what the system does) and non-functional requirements (how well it performs).

Business Requirements

Business requirements describe value the project should deliver:

1. **BR1: Productivity Enhancement**: Reduce time spent on initial PB drafting by generating solid first drafts from minimal input.
2. **BR2: Knowledge Codification**: Capture institutional presentation standards through template analysis, reducing reliance on tacit knowledge.
3. **BR3: Quality Consistency**: Follow corporate visual identity standards consistently, reducing revision cycles caused by formatting issues.
4. **BR4: Data Accuracy**: Financial data in generated PBs should accurately reflect source information.

Functional Requirements

Table 2 lists functional requirements by component using MoSCoW prioritisation (Must have, Should have, Could have, Won't have).

ID	Component	Requirement	Priority
FR1	Template Analyser	Extract slide layouts from reference .pptx files	Must
FR2	Template Analyser	Identify colour palettes and font specifications	Must
FR3	Template Analyser	Map placeholder positions and content types	Must
FR4	Data Retrieval	Fetch company financials from Yahoo Finance API	Must
FR5	Data Retrieval	Retrieve SEC filings via EDGAR API	Should
FR6	Data Retrieval	Perform web searches for company news	Should
FR7	Content Planner	Generate slide-by-slide content specifications	Must
FR8	Content Planner	Adapt content structure to PB type	Must
FR9	Content Planner	Maintain narrative coherence across slides	Should
FR10	Slide Builder	Assemble slides matching template layouts	Must
FR11	Slide Builder	Populate placeholders with generated content	Must
FR12	Slide Builder	Generate charts from financial data	Could
FR13	Orchestration	Coordinate multi-step generation workflow	Must
FR14	Orchestration	Handle API failures gracefully	Should
FR15	RAG Search	Query precedent material repository	Could

Table 2: Functional Requirements Specification

Non-Functional Requirements

Table 3 lists quality requirements with measurable acceptance criteria.

ID	Category	Requirement	Acceptance Criterion
NFR1	Performance	System generates complete PB draft	Within 5 minutes of input submission
NFR2	Performance	API response handling	Timeout after 30 seconds with graceful degradation
NFR3	Reliability	System availability during demonstration	95% uptime during evaluation period
NFR4	Usability	Input specification interface	Requires no technical expertise to operate
NFR5	Maintainability	Codebase documentation	All public functions documented with docstrings
NFR6	Security	API credential management	No credentials in source code; environment variables used
NFR7	Security	Input validation	All user inputs sanitised before processing
NFR8	Compatibility	Output format	Valid .pptx files openable in Microsoft PowerPoint

Table 3: Non-Functional Requirements Specification

Technology Stack

I chose technologies based on whether they fit the task, whether I knew them already, how well-developed their ecosystems were, and how easy they would be to maintain.

Programming Language and Framework

I chose Python as the main language because it dominates AI and ML development and has a large library ecosystem. I am also comfortable with Python from my work experience, which means I can focus on the problem rather than learning a new language. Table 4 compares Python with alternatives.

Criterion	**Python**	**JavaScript/Node.js**	**Java**
AI/ML library support	Excellent (PyTorch, LangChain, OpenAI SDK)	Limited	Moderate
PowerPoint manipulation	python-pptx (mature)	Limited options	Apache POI (verbose)
Async capabilities	asyncio, FastAPI	Native (excellent)	Reactive streams (complex)
Development velocity	High	High	Moderate

Table 4: Programming Language Comparison

FastAPI will handle the backend. It processes requests asynchronously, which matters when making multiple API calls to external data sources at the same time. FastAPI also generates OpenAPI documentation automatically, which will make testing easier.

AI and LLM Integration

The system will use LLMs for content planning and generation. Table 5 compares the options I considered.

Criterion	**GPT-4/GPT-4o**	**Claude 3.5**	**Gemini Pro**
Structured output	Excellent (JSON mode)	Good	Good
Context window	128K tokens	200K tokens	1M tokens
Function calling	Native support	Native support	Native support
Agent frameworks	OpenAI Agents SDK	Claude Agent SDK	LangChain integration

Table 5: Large Language Model Comparison

I chose OpenAI's GPT-4o as the primary LLM. It handles structured output well and has good agent framework support through the OpenAI Agents SDK, which provides tool-use patterns for the agentic data retrieval layer.

Data Sources and APIs

I will use publicly accessible APIs to avoid proprietary data licensing complications. Table 6 summarises the data sources.

Source	Data Provided	Access Method	Rate Limits
Yahoo Finance	Stock prices, financial statements, company profiles	yfinance Python library	Unofficial, fair use
SEC EDGAR	10-K, 10-Q filings, company facts	REST API	10 requests/second
Web Search	Company news, market commentary	SerpAPI or similar	Per subscription

Table 6: External Data Sources

Frontend and Deployment

The frontend will use React with TypeScript, deployed through Vercel. The Python backend will run on Render with managed hosting. Supabase will provide PostgreSQL database services.

Reproducibility

Reproducibility matters for credible research engineering. All Python dependencies will be pinned to exact versions in `requirements.txt`, with a `Dockerfile` for environment replication. LLM API calls will use `temperature=0` for evaluation runs to maximise determinism, and any randomised components will use fixed seeds. Runtime configuration will be separated from code using `config.yaml`, with all settings logged at startup. Each evaluation run will be timestamped and associated with a Git commit hash, with full prompt and response logging for LLM interactions.

Evaluation Methodology

I will assess the system against four metrics:

- **Efficiency**: target under 300 seconds for a 10-slide deck
- **Template fidelity**: programmatic comparison of layout, colour, and font properties
- **Data accuracy**: verification against source APIs
- **Content quality**: rubric-based assessment of narrative coherence and IB conventions

To contextualise performance, I will compare outputs against a naive baseline using generic tools (Gamma, Beautiful.ai) and estimated manual creation time. Ablation studies will test each component's contribution: content generation without template analysis, template analysis without LLM planning, and data retrieval without web search. Evaluation will use fixed test cases covering different company types, sectors, and PB types, documented for replication.

Project Plan

The project will run for thirteen weeks from start to final submission. I have divided it into phases matching the iterative approach. Figure 1 shows the schedule as a Gantt chart.

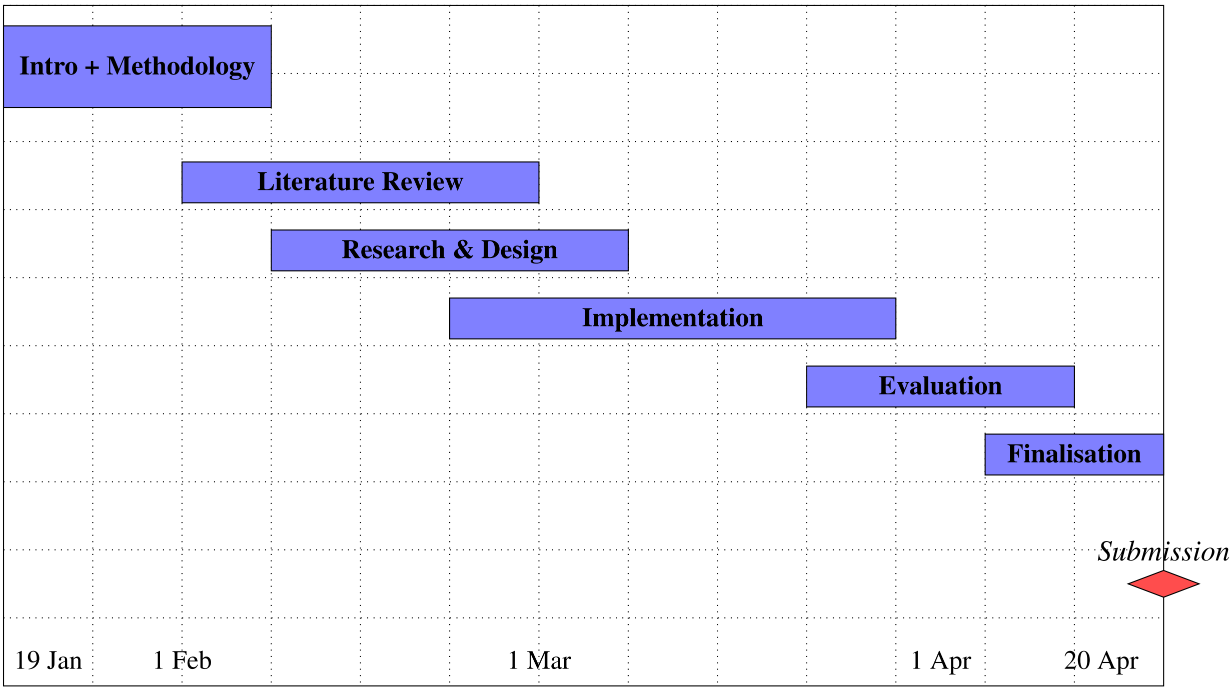

Figure 1: Project Gantt Chart (19 January to 20 April 2026)

Table 7 lists the milestones with their specific deliverables and target dates.

Phase	Date	Milestone	Deliverables
1	19 Jan	Project Initiation	Repository setup, development environment configured
2	31 Jan	Initial Chapters	Introduction, Scope, Methodology chapters complete
3	21 Feb	Literature Review	Literature review chapter complete
4	28 Feb	Design Complete	Architecture diagrams, API specifications, data models
5	14 Mar	Core Components	Template analyser and data retrieval functional
6	28 Mar	MVP Complete	End-to-end generation pipeline operational
7	4 Apr	Testing Complete	Unit and integration tests passing; evaluation data collected
8	11 Apr	Evaluation Complete	Results analysis and evaluation chapter written
9	18 Apr	Document Finalised	All chapters complete, proofread, formatted
10	20 Apr	Submission	Final report submitted via QMPlus

Table 7: Project Milestones and Deliverables

Risk Assessment and Mitigation

Effective risk management requires not only identifying risks and mitigations but also defining triggers that indicate when contingency actions should be activated. Table 8 shows the risk register with likelihood and impact rated on a three-point scale (Low, Medium, High).

ID	Risk Description	Likelihood	Impact	Mitigation Strategy
R1	API rate limits restrict data retrieval during development	Medium	Medium	Implement caching; use mock data for testing; stagger API calls
R2	LLM outputs inconsistent or hallucinated content	High	High	Structured output schemas; validation layers; HITL review
R3	Template analysis fails on complex slide layouts	Medium	High	Scope to common layouts; graceful degradation for unsupported elements
R4	External API deprecation or changes	Low	High	Abstract API interactions; monitor changelogs; maintain fallback sources
R5	Scope creep extends beyond POC boundaries	Medium	Medium	Strict MoSCoW prioritisation; regular scope reviews with supervisor
R6	Technical complexity exceeds available time	Medium	High	Iterative delivery; prioritise core pipeline; defer stretch goals
R7	Data accuracy issues undermine credibility	Medium	High	Verification against manual retrieval; source attribution in outputs
R8	Generated outputs violate compliance requirements	Low	High	Human review mandatory; disclaimer on all outputs; no PII processing

Table 8: Risk Assessment and Mitigation Strategies

For high-impact risks, I have defined contingency triggers: R2 (hallucination) triggers scope reduction if validation failure exceeds 20%; R3 (template failure) triggers narrowed template support if extraction success falls below 70%; R6 (time overrun) triggers dropping "Could have" requirements if milestones slip by more than one week. The risk register will be reviewed at fortnightly supervisor meetings.

Ethics, Data Governance and Compliance

The system keeps humans in control of analytical judgements. Generated content is marked as AI-assisted, and HITL design ensures professional review before distribution. The system prioritises accuracy over completeness, flagging uncertainty rather than presenting unverified information as fact.

For data governance, the system processes only data needed for PB generation, with no retention of user inputs after sessions. All external data includes source attribution. The POC does not process personally identifiable information; company data comes exclusively from public sources.

Regarding compliance, the HITL architecture meets EU AI Act requirements for human oversight (Bank for International Settlements 2024). Professional responsibility for final content remains with the banker.

The system logs generation parameters and data sources to support audit requirements.

Project Tracking

Progress will be tracked using GitHub Issues with a Kanban board (Backlog, In Progress, In Review, Done). Each functional requirement maps to one or more issues, providing traceability from requirements to implementation. Changes to scope or architecture will be documented in the issue history.

Reusability

The architecture separates concerns into independent modules (template analysis, data retrieval, content planning, slide assembly), each with clean interfaces for standalone reuse. All code will include Google-style docstrings, with a detailed README and architecture documentation. The codebase will be released under MIT License to minimise barriers to reuse.

Preliminary Research and Design Documentation

Source	Type	Summary	Link
Knowledge Management & Knowledge Reuse			
Markus (2001). Toward a Theory of Knowledge Reuse	Journal	Foundational theory on knowledge reuse; identifies four types of reuse situations and success factors	Link
Dalkir (2017). Knowledge Management in Theory and Practice	Textbook	Comprehensive KM textbook covering tacit/explicit knowledge and organisational memory systems	Link
Large Language Models			
Zhao et al. (2023). A Survey of Large Language Models	Survey	Comprehensive survey on LLM development; covers GPT, LLaMA, PaLM families	Link
Minaee et al. (2024). Large Language Models: A Survey	Survey	Reviews LLM characteristics, contributions, limitations, and augmentation techniques	Link
Retrieval-Augmented Generation			
Lewis et al. (2020). RAG for Knowledge-Intensive NLP Tasks	Conference	Seminal RAG paper; combines parametric and non-parametric memory for factual generation	Link
Gao et al. (2023). RAG for LLMs: A Survey	Survey	Comprehensive RAG survey covering taxonomy, methods, and applications	Link
Chen et al. (2025). RAG and LLMs for Enterprise KM	Journal	Systematic review of 63 studies; 63.6% use GPT models, 80.5% use FAISS/Elasticsearch	Link
Fan et al. (2024). A Survey on RAG Meeting LLMs	Conference	KDD survey on integrating RAG with LLMs	Link
Agentic AI & Autonomous Agents			
Wang et al. (2024). A Survey on LLM-based Autonomous Agents	Journal	Foundational survey on LLM autonomous agents; proposes unified framework	Link
Guo et al. (2024). LLM Based Multi-agents: A Survey	Conference	IJCAI survey on multi-agent LLM systems for complex problem-solving	Link
Li et al. (2024). A survey on LLM-based multi-agent systems	Journal	Systematic review with five-component architecture framework	Link
LLM Tool Use & Function Calling			
Li (2025). A review of paradigms for LLM-based agents	Conference	Reviews tool use, planning, RAG, and feedback mechanisms	Link
Automated Document Generation			
Zheng et al. (2025). PPTAgent: Generating Presentations	Conference	Two-stage edit-based presentation generation; introduces PPTEval framework	Link

Source	Type	Summary	Link
Liu et al. (2024). We Need Structured Output	Conference	Studies user needs for structured LLM outputs in professional contexts	Link
LLM Structured Output & Prompt Engineering			
Wu et al. (2024). LLM-Driven Structured Output: A Benchmark	Journal	Benchmark for structured outputs; compares fine-tuning, prompting, and RAG	Link
Xu et al. (2025). Structured Data Generation with GPT-4o	Journal	Compares JSON, YAML, CSV prompt styles; JSON best for complex data	Link
LLM Hallucination & Factual Accuracy			
Huang et al. (2024). A Survey on Hallucination in LLMs	Journal	Comprehensive taxonomy of hallucination causes, detection, and benchmarks	Link
Alansari & Luqman (2025). LLM Hallucination Survey	Survey	Review of hallucination causes, detection, and mitigation strategies	Link
Human-in-the-Loop AI			
Wu et al. (2022). A Survey of HITL for ML	Journal	Classifies HITL approaches: data processing, interventional training, system design	Link
Mosqueira-Rey et al. (2024). HITL ML: Role of the user	Journal	Discusses timing, frequency, and workload factors in interactive ML	Link
Rezaeighaleh et al. (2023). Ethical AI Based on HITL	Journal	Examines HITL for ethical AI development	Link
AI in Financial Services			
Alghofaili et al. (2024). AI and ML in Banking Systems	Journal	Examines board role in AI adoption; Saudi Arabian banking sector	Link
Kumari & Tanwar (2023). AI/ML in Financial Services	Journal	Bibliometric analysis of 1,045 BFSI sector articles	Link
AI Regulation & Compliance			
Zetzsche et al. (2024). Regulating AI in investment management	Journal	Discusses EU AI Act, GDPR, MiFID II implications for AI	Link
Knowledge Worker Productivity			
Brynjolfsson et al. (2023). Generative AI at Work	Working Paper	14% productivity increase from AI; greatest impact on novice workers	Link
Noy & Zhang (2023). Productivity effects of generative AI	Journal	Experimental study showing productivity improvements from ChatGPT use	Link
McKinsey (2012). The Social Economy	Report	Knowledge workers spend 20% of time searching for information	Link
Vision-Language Models for Documents			

Source	Type	Summary	Link
Faysse et al. (2024). ColPali: Document Retrieval with VLMs	Research	Uses VLM for direct PDF image retrieval; introduces ViDoRe benchmark	Link
Liao et al. (2024). DocVLM: Efficient Reader	Conference	Integrates OCR into VLMs; improves DocVQA from 56% to 86.6%	Link
Software Engineering Methodology			
Anifa et al. (2024). Systematic Review on Agile Approach	Journal	Systematic review of agile methodology across industries	Link
Sommerville (2016). Software Engineering	Textbook	Standard software engineering reference; covers SDLC and methodologies	Link

Project Implementation and Outcomes

Evaluation and Conclusions

References

Bank for International Settlements (2024). *Regulating AI in the Financial Sector: Recent Developments and Main Challenges*. 63. URL: https://www.bis.org/fsi/publ/insights63.pdf.

BBC News (Mar. 2021). *Goldman Sachs junior bankers rebel over 'inhumane' 100-hour weeks*. URL: https://www.bbc.co.uk/news/business-56452494 (visited on 01/15/2025).

Corporate Finance Institute (2025). *Investment Banking Analyst Job Description, Hours, and Salary*. URL: https://corporatefinanceinstitute.com/resources/career/investment-banking-analyst-job-description-hours-salary/ (visited on 02/24/2025).

Dalkir, Kimiz (2011). *Knowledge Management in Theory and Practice*. 2nd. Cambridge, MA: MIT Press. ISBN: 978-0262015080.

IBM (2024). *What are Agentic Workflows?* URL: https://www.ibm.com/think/topics/agentic-workflows (visited on 11/18/2025).

Markus, M. Lynne (2001). "Toward a Theory of Knowledge Reuse: Types of Knowledge Reuse Situations and Factors in Reuse Success". In: *Journal of Management Information Systems* 18.1, pp. 57–93. DOI: 10.1080/07421222.2001.11045671.

McKinsey & Company (2024). *Capturing the Full Value of Generative AI in Banking*. McKinsey & Company. URL: https://www.mckinsey.com/industries/financial-services/our-insights/capturing-the-full-value-of-generative-ai-in-banking.

McKinsey Global Institute (July 2012). *The Social Economy: Unlocking Value and Productivity Through Social Technologies*. McKinsey & Company. URL: https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/the-social-economy.

Radhakrishna, Vedapradha et al. (2024). "Future of Knowledge Management in Investment Banking: Role of Personal Intelligent Assistants". In: *SAGE Open* 14.4. DOI: 10.1177/20597991241287118.

Sommerville, Ian (2016). *Software Engineering*. 10th. Boston, MA: Pearson Education. ISBN: 978-0133943030.

Wall Street Oasis (2024). *Investment Banking Hours: Typical Schedule, Work-Life Balance, and Expectations*. URL: https://www.wallstreetoasis.com/resources/careers/jobs/investment-banking-hours.

Wang, Lei et al. (2024). "A Survey on Large Language Model based Autonomous Agents". In: *Frontiers of Computer Science* 18.6, p. 186345. DOI: 10.1007/s11704-024-40231-1. arXiv: 2308.11432.

Zheng, Hao et al. (2025). "PPTAgent: Generating and Evaluating Presentations Beyond Text-to-Slides". In: *arXiv preprint*. arXiv: 2501.03936. URL: https://arxiv.org/abs/2501.03936.

Appendices