International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Architecting & Engineering Production-Grade Agentic AI for Industry-Specific Needs

Rimjhim (Rémy) Singh

Founder and AI Engineer, Rimjhim AI Kingdom and Rikanverse Ltd., London, United Kingdom

Abstract

Agentic artificial intelligence (AI) systems are increasingly adopted across industries, yet each industry brings its own regulations, users, data and ways of working. A general-purpose chatbot built on a large language model cannot meet these needs on its own; the system around the model has to be designed for the industry it serves. This paper presents a cross-domain practitioner study of four Agentic AI systems that were architected, engineered and deployed by the Author and are presented together in Rimjhim AI Kingdom®, a navigable web ecosystem in which each system appears as a building in a virtual city. The systems serve banking, residential property, hotel review management and e-commerce customer service, and were developed in Python without the use of any no-code or low-code platforms. For each system, the paper traces how the needs of the industry shaped the architecture. Comparing the four systems, it identifies six customisation dimensions along which an AI architect adapts a shared agentic core: interaction mode, knowledge source, reasoning style, compliance layer, integrations and the boundary of automation. The paper argues that these decisions belong to the AI architect and engineer, and it draws design lessons for building industry-specific agentic systems.

Keywords: Agentic AI, AI Architecture, Customisation, Industry Applications, LLM Agents

1. Introduction

Large language model (LLM) agents no longer only generate text. They reason, call tools, read and write records and act on behalf of users. As a result, organisations in very different industries now want agentic systems: banks to answer customers and process requests, property platforms to guide renters, hotels to respond to guest reviews and online retailers to handle customer email. The underlying models are often the same, but the industries are not. A bank must satisfy financial regulation and keep customers' trust; a property platform must understand natural, spoken questions from people of many backgrounds; a hotel must protect its brand voice; and a customer-service team must not lose track of important emails. A single generic AI system cannot meet all of these needs. What makes an agentic system useful in a given industry is the architecture around the model: how users interact with it, where its knowledge comes from, how it reasons, which compliance checks it applies, which systems it connects to and which tasks it is allowed to complete on its own. These are design decisions, and they belong to the AI architect and engineer who builds the system.

This paper examines such decisions in four agentic artificial intelligence (AI) systems built by the Author for four industries. Each system starts from an architecture, with a reasoning engine, a safety layer and tool orchestration, rather than from a single prompt and an application programming interface (API) call.

The research question is: how should AI architects and engineers customise agentic AI systems to the needs of different industries? The paper makes three contributions: (1) an architecture-level description of four industry-specific agentic systems, showing how each industry need led to a design decision; (2) a set of six customisation dimensions along which a shared agentic core is adapted; and (3) design lessons for engineering agentic systems for industry.

2. Related Work

ReAct (Reasoning and Acting) interleaves reasoning traces with actions, allowing a model to gather information and revise its plan during a task (Shunyu Yao, et al., 2023) [1]. Toolformer shows that language models can learn when to call external tools (Timo Schick, et al., 2023) [2], and retrieval-augmented generation grounds model outputs in retrieved documents rather than in model memory alone (Patrick Lewis, et al., 2020) [3]. Surveys of LLM-based autonomous agents organise the field around profiling, memory, planning and action modules (Lei Wang, et al., 2024) [4]. These works describe the building blocks of an agentic core.

Research on automation distinguishes levels of automation, from systems that only suggest actions to systems that act without human approval (Raja Parasuraman, et al., 2000) [5], and guidelines for human-AI interaction recommend making clear what a system can do and supporting efficient correction (Saleema Amershi, et al., 2019) [6]. Work on indirect prompt injection shows that instructions hidden in content an agent processes can redirect its behaviour (Kai Greshake, et al., 2023) [7]. Studies of production machine learning warn that the model is a small part of a deployed system and that most of the engineering lies in the surrounding infrastructure (David Sculley, et al., 2015) [8], and design patterns, in the sense of reusable solutions to recurring problems, have a long history in software engineering (Erich Gamma, et al., 1994) [9]. A companion paper describes the PropTech App in this collection in detail (Rimjhim (Rémy) Singh, 2026) [10].

This paper brings these strands together from an architect's point of view: it shows how the same building blocks are combined differently for different industries. It is an applied practitioner study rather than a new learning method.

3. Materials and Methods

3.1 The Rimjhim AI Kingdom® Ecosystem

Rimjhim AI Kingdom® is a Navigable Virtual City Experience Web App where each building represents one Agentic AI system and opens a panel describing the problem it addresses and the solution built for it. Buildings are grouped into districts by sector, including a Financial District, an Enterprise Zone, a Hospitality Quarter and the Property Towers. A Safety HQ at the centre of the city represents the shared safety core whose methodology is reused across systems, and a Knowledge District links to the author's published technical writing on reliable AI design.

3.2 Study Approach

The study is a comparative analysis of system architectures. For each system, the industry problem, the users served, the architectural decisions made in response and the resulting architecture were documented. A design decision was treated as a customisation dimension when it was made in every system but resolved differently according to the industry. All four systems were developed in Python without the use of any no-code or low-code platforms. Implementation details are withheld because the systems are under commercial development. Table 1 summarises the four systems; Sections 3.3 to 3.6 describe each one.

Table 1: Industry Needs and Customisation Decisions in the Four Systems

System	Industry	Industry Need	Customisation Decisions
Banking Agent	Financial services	Reasoning beyond scripted flows; compliance with financial regulation; dependable handling of risky requests	ReAct orchestration; answers grounded in policy documents; six-gate compliance layer; confidence-based handling of risky actions
Rikanverse® Property App	Residential property	Natural, spoken search; users of many languages; practical renting questions	Voice-first dual pipeline; multilingual conversation; guidance with clarifying questions; matching to Rikanverse listings
Hotel Review Agent	Hospitality	High review volume; consistent brand voice; early warning of damaging reviews	Sentiment analysis; crisis detection; brand-voice drafting; approval dashboard
CRM Email Assistant	E-commerce customer service	Time lost on email triage; important emails buried; replies that need customer context	Urgency and topic classification; thread history and CRM context; few-shot drafting; review queue

3.3 Banking Agent

Industry need: Banks need assistants that can reason about a customer's request rather than follow scripted flows, while every answer and action must comply with the rules of the United Kingdom (UK) Financial Conduct Authority (FCA) and the General Data Protection Regulation (GDPR).

Architecture: The banking agent runs on a multi-agent ReAct orchestration engine. Questions about products and policies are answered from retrieved policy documents, and routine requests such as balance checks and statements are completed automatically. Every action is verified by a six-gate compliance layer against FCA and GDPR rules. For risky actions such as balance transfers, the agent does not act alone: depending on its confidence it asks the customer to confirm or clarify, and a human is involved before the action is carried out; actions can be rolled back if needed.

3.4 Rikanverse® Property Agentic AI App

Industry need: Property search in the UK is largely filter-based, while renters and buyers think in natural language, often speak more than one language, and have practical questions about matters such as Council Tax, tenancy agreements and mortgages.

Architecture: The Rikanverse® Property Agentic AI App is a voice-first, multilingual conversational Agent, built on a ReAct reasoning loop. Voice and text run through separate pipelines with their own latency budgets. The agent reasons across several constraints at once, asks clarifying questions as a human estate agent would, gives practical guidance and matches users to properties listed on Rikanverse. Landlords register listings by speaking instead of completing forms. Session state persists across the conversation, so earlier preferences are not lost.

3.5 Hotel Review Agent

Industry need: Hotels and restaurants receive more reviews than staff can answer, replies must sound like the brand rather than like generic AI text, and a single unanswered crisis review can damage a business.

Architecture: The hotel review agent is a four-layer pipeline. A natural language processing (NLP) layer analyses sentiment, a crisis-detection layer flags reviews that carry compliance or reputational risk, a generation layer drafts replies in the brand's voice, and an approval dashboard lets staff review and approve each reply before it is published.

3.6 CRM Email Assistant

Industry need: Customer-service teams in e-commerce spend hours on email triage, important messages from key customers get buried, and good replies require knowledge of the customer's history. **Architecture:** The customer relationship management (CRM) email assistant classifies each incoming email by urgency and topic, extracts the thread history, retrieves the customer's CRM record and drafts a reply matched to the customer's sentiment using few-shot learning. Drafts enter a review queue where support staff read and approve them before sending, critical issues are escalated to human experts, and every action is logged in an audit trail with data handled in line with GDPR.

4. Results and Discussion

4.1 Customisation Dimensions

Comparing the four architectures shows a shared agentic core, combining an LLM, tools and an orchestration layer, that is adapted along six dimensions (Table 2 and Figure 1). The dimensions are interaction mode, knowledge source, reasoning style, compliance layer, integrations and the boundary of automation, that is, which tasks the system completes on its own and where a customer or a member of staff takes part.

Table 2: How Each System Is Customised Along Six Dimensions

Dimension	Banking	Property	Hotel Review	CRM Email
Interaction mode	Text conversation	Voice-first and text; multilingual	Review feed and dashboard	Email inbox and review queue
Knowledge source	Policy documents (retrieval)	Rikanverse listings; renting guidance	Review text; brand voice examples	Thread history; CRM record; few-shot examples
Reasoning style	Multi-agent ReAct loop	ReAct loop with clarifying questions	Four-layer pipeline	Classification and drafting pipeline
Compliance layer	Six-gate FCA and GDPR verification	Response checks	Crisis and compliance detection	GDPR data handling; audit trail
Integrations	Tool orchestration (LangChain)	Listings database (PostgreSQL); speech interface	Approval dashboard (FastAPI, React)	Email and CRM systems
Automation boundary	Information and routine requests automated; risky actions with confirmation and human involvement	Guidance and listing suggestions automated	Drafts automated; staff approve replies	Triage and drafts automated; staff approve replies

Figure 1: From Industry Needs to Customised Agentic Systems

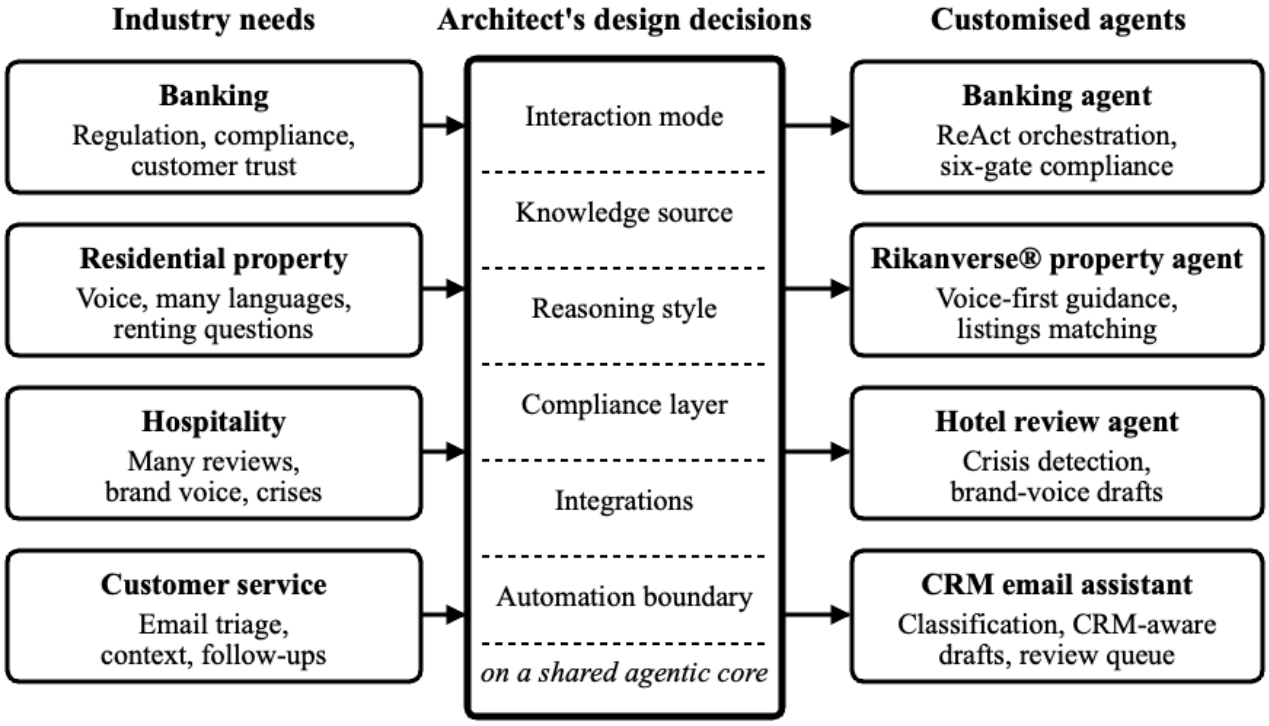


Figure 1 shows the process at a glance: each industry presents its own needs, the architect resolves the six design decisions in response, and the result is a customised agent built on the same agentic core.

4.2 Design Lessons

Four lessons emerge. First, start from the industry, not from the model: in every system, the most important design decisions followed from the needs of the industry, such as regulation in banking, spoken multilingual search in property and brand voice in hospitality. Second, keep a shared core and customise around it: the same reasoning, tool and orchestration components were reused, while the interaction mode, knowledge source, compliance layer, integrations and automation boundary changed from system to system. Third, reuse proven designs across industries: the hotel review agent and the CRM email assistant share the same classify-draft-review design, applied to guest reviews in one case and customer emails in the other, and the gated safety method developed for banking informs the safety core shared across the ecosystem. Fourth, build compliance into the architecture from the start: where an industry is regulated, compliance checks belong in the core of the system rather than being added at the end.

5. Conclusion

This paper compared four agentic AI systems built for banking, residential property, hospitality and e-commerce customer service. It showed that the systems share a common agentic core but differ along six customisation dimensions, each resolved according to the needs of the industry. Deciding how to resolve these dimensions is the central task of the AI architect and engineer. The study is based on four systems built by one engineer, and future work will measure how each customisation decision affects task success and user satisfaction in each industry.

References

1. Shunyu Y., Jeffrey Z., Dian Y., Nan D., Izhak S., Karthik N., Yuan C., "ReAct: Synergizing Reasoning and Acting in Language Models", Proceedings of the International Conference on Learning Representations (ICLR), 2023. https://arxiv.org/abs/2210.03629
2. Timo S., Jane D., Roberto D., Roberta R., Maria L., Luke Z., Nicola C., Thomas S., "Toolformer: Language Models Can Teach Themselves to Use Tools", Advances in Neural Information Processing Systems (NeurIPS), 2023, 36. https://arxiv.org/abs/2302.04761
3. Patrick L., Ethan P., Aleksandra P., Fabio P., Vladimir K., Naman G., Heinrich K., Mike L., Wen-tau Y., Tim R., Sebastian R., Douwe K., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", Advances in Neural Information Processing Systems (NeurIPS), 2020, 33. https://arxiv.org/abs/2005.11401
4. Lei W., Chen M., Xueyang F., Zeyu Z., Hao Y., Jingsen Z., Zhiyuan C., Jiakai T., Xu C., Yankai L., Wayne Z., Zhewei W., Ji-Rong W., "A Survey on Large Language Model Based Autonomous Agents", Frontiers of Computer Science, 2024, 18(6). https://arxiv.org/abs/2308.11432
5. Raja P., Thomas S., Christopher W., "A Model for Types and Levels of Human Interaction with Automation", IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 2000, 30(3), 286–297. https://doi.org/10.1109/3468.844354
6. Saleema A., Dan W., Mihaela V., Adam F., Besmira N., Penny C., Jina S., Shamsi I., Paul B., Kori I., Jaime T., Ruth K., Eric H., "Guidelines for Human-AI Interaction", Proceedings of the CHI Conference on Human Factors in Computing Systems, 2019. https://doi.org/10.1145/3290605.3300233
7. Kai G., Sahar A., Shailesh M., Christoph E., Thorsten H., Mario F., "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection", Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec), 2023. https://arxiv.org/abs/2302.12173
8. David S., Gary H., Daniel G., Eugene D., Todd P., Dietmar E., Vinay C., Michael Y., Jean-François C., Dan D., "Hidden Technical Debt in Machine Learning Systems", Advances in Neural Information Processing Systems (NeurIPS), 2015, 28.
9. Erich G., Richard H., Ralph J., John V., "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley, 1994.
10. Rimjhim (Rémy) S., "Engineering Reliable Agentic AI: Decision-Making in Real-World Deployments", International Journal for Multidisciplinary Research (IJFMR), 2026, 8(6). https://doi.org/10.36948/ijfmr.2026.v08i06.91388