PRODUCT REQUIREMENTS DOCUMENT (PRD)

Multi-Agent AI Chatbot Platform (Persona-Driven, RAG-Enabled)

---

1. Executive Summary

The Multi-Agent AI Chatbot Platform is an agentic conversational system designed to deliver **accurate, safe, and domain-specific AI assistance** across multiple verticals such as fitness, mental health, personal finance, immigration, and parenting.

Instead of relying on a single monolithic LLM, the platform uses a **persona-based multi-agent architecture**, where each agent has:

- A clearly defined scope
- Domain-specific retrieval sources
- Purpose-built prompts and tools
- Explicit safety and compliance boundaries

This approach mirrors how production AI copilots are designed in modern enterprises, prioritizing **governance, extensibility, and trust**.

---

2. Problem Statement

Market Context

Most conversational AI systems today rely on a single general-purpose LLM. While powerful, this design introduces fundamental weaknesses when applied to real-world, multi-domain use cases.

Key issues include:

- Generic responses that lack domain depth
- High hallucination rates in regulated or sensitive domains
- Prompt complexity that becomes brittle at scale
- Lack of governance and accountability

These issues limit the adoption of AI assistants in production environments where accuracy, safety, and explainability matter.

---

3. Business Problem

Organizations want AI assistants that can:

- Serve **multiple domains** without degrading quality
- Enforce **clear safety and compliance boundaries**
- Scale to new use cases without rewriting the system
- Provide **auditable, explainable behavior**

However, most existing solutions force teams to choose between:

- A single chatbot that does everything poorly
- Multiple siloed bots that are expensive to maintain

The absence of a **modular agent framework** creates both technical and operational inefficiencies.

---

4. Product Vision

The vision is to build a **modular, agent-oriented AI platform** where each agent behaves like a domain expert operating within clearly defined boundaries.

Each agent should:

- Understand only what it is responsible for
- Retrieve knowledge from curated sources
- Use tools selectively and transparently
- Defer or disclaim when outside its authority

This design prioritizes **trust, safety, and long-term scalability** over raw model capability.

---

5. Business Use Cases

Use Case	Description
AI Concierge	Routes users to domain-specific AI experts
Enterprise Copilot	HR, Finance, Policy, or Support assistants
Consumer Education	Fitness, finance, immigration, parenting guidance
Pre-Screening Assistant	Non-diagnostic health or policy education
Internal Knowledge Bot	Department-specific RAG agents

---

6. Target Users

Primary Users

- Consumers seeking reliable, understandable guidance
- Enterprises building internal or customer-facing AI copilots
- Product teams experimenting with AI-driven workflows
- AI engineers designing scalable agent systems

Secondary Stakeholders

- Legal and compliance teams concerned with risk
- Trust & Safety teams responsible for governance
- Operations teams focused on cost and performance

---

7. User Pain Points and Solutions

Pain Point	Platform Solution
Shallow chatbot answers	Persona specialization
Hallucinated responses	Retrieval-Augmented Generation
Unsafe or misleading advice	Guardrails and disclaimers
Difficult scaling	Modular agent design
Poor explainability	Clear agent boundaries

---

8. Functional Requirements

The platform must support:

- Intent detection to classify the user's domain
- Agent routing to select the appropriate persona
- Persona-specific prompts and tools
- Retrieval-Augmented Generation for grounded responses
- Tool execution (search, analysis, document parsing)
- Structured, context-aware response synthesis

---

9. Non-Functional Requirements

Beyond core functionality, the platform is designed to meet several non-functional requirements that are critical for production-grade AI systems.

Performance
The system is designed to deliver responses within an acceptable latency envelope for conversational AI. The target is a P95 end-to-end response time of under three seconds, inclusive of intent classification, retrieval, tool execution, and LLM inference.

Scalability
The architecture must support the addition of new agents with minimal engineering effort. Introducing a new persona should not require changes to existing agents or the orchestration logic beyond registration and configuration.

Safety and Compliance
Each agent must enforce domain-specific guardrails. For sensitive domains such as mental health, finance, and immigration, the system must clearly communicate disclaimers and avoid authoritative or regulated advice.

Observability
The platform should expose sufficient telemetry to evaluate agent behavior, response quality, and latency at each stage of the pipeline. This enables debugging, auditing, and continuous improvement.

Maintainability
The codebase is structured around clear separations of concern so that agent logic, orchestration, retrieval, and UI layers can evolve independently.

---

10. Agent Personas and Governance Model

Each AI agent in the system operates under a **bounded authority model**. Rather than attempting to answer all questions, an agent is explicitly scoped to a specific domain and instructed to defer, disclaim, or redirect when queries fall outside its responsibility.

For example, the Personal Finance Educator can explain financial concepts and terminology but is explicitly restricted from providing investment recommendations. Similarly, the Mental Health Counselor is designed to offer supportive guidance while clearly escalating or disclaiming in crisis-related scenarios.

This governance model is intentional. It reflects how real organizations deploy AI systems responsibly, prioritizing user trust and compliance over maximal coverage.

---

11. Acceptance Criteria

The platform is considered functionally complete when the following conditions are met:

- User queries are routed to the correct agent with high reliability.
- Responses demonstrate clear grounding in retrieved knowledge where applicable.
- Safety disclaimers appear consistently in regulated or sensitive domains.
- Output responses are structured, readable, and contextually relevant.
- New agents can be added without refactoring existing agent logic.

These criteria ensure that success is measured not just by whether the system works, but whether it works **safely and predictably**.

---

12. Key Performance Indicators and Measurement

System success is evaluated using a combination of qualitative and quantitative metrics.

Routing accuracy measures whether user queries are consistently handled by the correct agent. Hallucination rate is tracked through periodic audits of responses, particularly in sensitive domains. Latency is measured end-to-end, with attention paid to bottlenecks introduced by retrieval or tool execution.

User satisfaction is measured through direct feedback and behavioral signals such as follow-up queries or abandonment. Agent usage distribution is monitored to understand demand patterns and inform future optimization.

Latency is measured at multiple stages: intent routing, retrieval, LLM inference, tool execution, and final response delivery. This layered approach enables targeted performance improvements rather than blind optimization.

PROJECT REPORT

Technical and Strategic Deep Dive

---

13. System Design Philosophy

The platform follows an **agent-oriented design philosophy**. Each agent is treated as an autonomous unit with clearly defined responsibilities, while a shared orchestration layer coordinates execution.

This approach prioritizes clarity and safety over raw flexibility. Instead of embedding all logic in a single prompt or model call, responsibilities are decomposed across layers, making the system easier to reason about and evolve.

---

14. Rationale for a Multi-Agent Architecture

A single-LLM chatbot becomes increasingly fragile as the number of supported domains grows. Prompts become longer, edge cases multiply, and governance becomes difficult.

A multi-agent architecture solves this by isolating complexity. Each agent is optimized for its domain, reducing cognitive load on the model and minimizing cross-domain interference. This design enables independent iteration, targeted evaluation, and clearer accountability.

---

15. Role of Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation is a foundational component of the system. Rather than relying solely on model priors, agents retrieve relevant context from curated knowledge sources before generating a response.

This approach reduces hallucinations, enables faster updates to domain knowledge, and improves user trust by grounding responses in real information. Importantly, retrieval is applied **per agent**, allowing each persona to maintain its own knowledge boundaries.

---

16. LLM Strategy and Model Abstraction

The system is intentionally designed to be LLM-agnostic. Models are accessed through abstraction layers, allowing the platform to switch providers or models without architectural changes.

This strategy reduces vendor lock-in, enables cost optimization, and allows experimentation with newer or more specialized models as they become available. Model selection prioritizes reasoning quality, tool-calling support, and latency characteristics rather than parameter count alone.

---

17. Choice of LangChain and Future Use of LangGraph

LangChain was selected as the orchestration framework due to its mature abstractions for agents, tools, memory, and callbacks. It enables rapid development while maintaining architectural clarity.

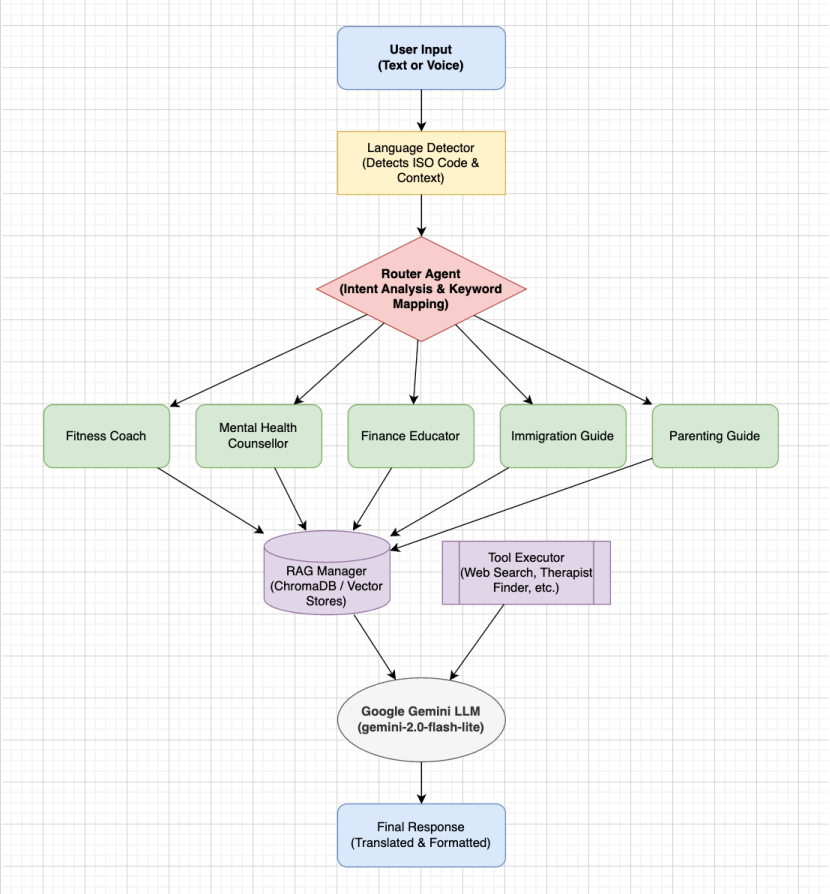
Looking forward, LangGraph represents a natural evolution of the system. It would enable stateful agent workflows, multi-step decision graphs, and long-running processes that extend beyond single conversational turns.

---

18. System Architecture Overview

At a high level, the system consists of a user interface layer, an intent routing layer, an agent orchestration layer, persona-specific agents, retrieval and tool layers, and an LLM reasoning layer.

Each layer has a single responsibility. This separation reduces coupling, simplifies debugging, and makes the system easier to extend as new capabilities are added.



---

19. End-to-End Execution Flow

When a user submits a query, the system first classifies intent to determine the relevant domain. The appropriate agent is then selected and provided with retrieved context and tools as needed.

The agent synthesizes a response using the LLM, applying its domain constraints and safety rules. The final response is returned to the user through the UI. This flow ensures that each response is both relevant and governed.

---

20. Dependencies

The platform is implemented primarily in Python, using Streamlit for the user interface, LangChain for agent orchestration, vector databases for retrieval, and external LLM APIs for reasoning.

Each dependency is chosen for maturity, community support, and alignment with production AI workflows.

---

21. Risks and Mitigations

Hallucinations are mitigated through retrieval grounding and scoped prompts. Compliance risks are addressed through explicit disclaimers and bounded agent authority. Latency risks are mitigated through architectural separation and planned asynchronous execution. Model drift is mitigated by relying on retrieval rather than static model knowledge.

---

22. Future Enhancements

Future iterations of the platform may include stateful agent workflows, persistent memory across sessions, agent performance dashboards, multimodal inputs such as voice or images, and asynchronous execution with caching to further reduce latency.

These enhancements build naturally on the existing architecture without requiring fundamental redesign.

---

23. Final Assessment

This project represents a **production-minded agentic AI system**, not a proof-of-concept chatbot. It demonstrates thoughtful tradeoffs across accuracy, safety, scalability, and maintainability.

The design reflects how modern AI copilots are built in real organizations, emphasizing governance, modularity, and long-term viability.
