Enterprise Customer Onboarding Agent

Solution Design Summary

StackAdapt Case Study | February 2025

A. Solution Architecture

The Enterprise Onboarding Agent is an AI-powered automation system built with **LangGraph** for state machine orchestration and **FastAPI** for the REST interface. It automates the customer journey from closed deal to provisioned account across multiple enterprise systems.

Layer	Components	Purpose
Trigger	Webhooks, API, Cron	Event-driven workflow initiation
Orchestration	LangGraph State Machine	Workflow control with conditional routing
Integration	Salesforce, CLM, NetSuite	Data fetching with error handling
Intelligence	OpenAI GPT-4 + Fallback	Risk analysis and recommendations
Action	Provisioning, Notifications	Autonomous execution with guardrails

Data Flow: Webhook → Initialize State → Fetch (SF/CLM/NS) → Validate → Analyze Risks → Decision (BLOCK/ESCALATE/PROCEED) → Actions → Summary

B. AI Agent Application

The AI agent applies intelligence at multiple stages of the workflow:

Application	Implementation	Fallback
Risk Analysis	LLM assesses business impact, identifies blockers, prioritizes issues	Rule-based scoring
Summary Generation	Human-readable status with context for stakeholders	Template-based output
Recommendations	Prioritized actions with owners and timelines	Predefined action mappings
Error Handling	API errors analyzed and factored into decisions	Direct BLOCK on errors

Decision Logic: The agent evaluates three factors: (1) **API Errors** - system failures → BLOCK, (2) **Violations** - business rule failures → BLOCK, (3) **Warnings** - non-critical issues → ESCALATE. Only when all three are clear does the agent PROCEED with provisioning.

C. Orchestration & Event-Driven Flows

The system supports multiple trigger types for flexibility:

- **Webhooks:** Salesforce fires on Opportunity stage change to 'Closed Won'
- **API Calls:** Manual trigger via REST endpoint for on-demand processing
- **Scheduled Jobs:** Cron-based batch processing of pending onboardings

Error Simulation: Configurable error injection (auth, validation, rate limit, server errors) allows stress testing. Rates are adjustable via API (e.g., auth_rate=1.0 for 100% auth failures).

State Management: LangGraph manages state transitions with full observability via LangSmith tracing. Each run generates structured JSON logs, audit trails, and professional reports.

D. Trade-offs, Assumptions & Considerations

Design Decisions:

Decision	Trade-off	Rationale
API Errors → BLOCK	May delay legitimate onboardings	Data integrity over speed
In-place Error Simulator	More complex implementation	Consistent module references
Sync Processing	Longer response times	Simplicity for demo
Rule-based Fallback	Less intelligent analysis	Availability without LLM
Mock APIs	Not production-ready	Demo without credentials

Scalability: Production deployment would add message queues (SQS/RabbitMQ), distributed state (Redis), horizontal scaling (Kubernetes), and LLM response caching.

Security: OAuth 2.0 authentication, environment-based credential storage, PII masking in logs, immutable audit trails with correlation IDs, permission validation before API calls.

Governance: Every run produces structured logs, LangSmith traces, run reports (MD/HTML), and state snapshots for compliance and debugging.

E. Multi-Agent Collaboration via MCP

The Model Context Protocol (MCP) enables specialized agents to collaborate through shared tool servers:

Agent	Responsibility	MCP Tools
Onboarding Coordinator	Orchestrates workflow, decisions	salesforce.*, provision.*
Contract Agent	Monitors signatures, reminders	clm.get_contract, clm.send_reminder
Finance Agent	Tracks payments, dunning	netsuite.get_invoice, netsuite.send_dunning
Risk Analyzer	Assessments, recommendations	llm.analyze_risks, llm.generate_summary

Architecture: Each MCP Server wraps an external API (Salesforce, NetSuite, CLM, OpenAI) and exposes tools via a standardized interface. The MCP Router directs agent tool calls to the appropriate server. Agents can delegate tasks (Coordinator → Contract Agent) or consult specialists (Coordinator → Risk Analyzer).

Benefits: (1) Separation of concerns - each agent is an expert in its domain, (2) Reusable tool servers - multiple agents share the same integrations, (3) Scalable complexity - add new agents without modifying existing ones, (4) Cross-agent learning - agents can share context and insights.

Production Roadmap

Phase	Timeline	Key Deliverables
Foundation (Current)	Q1 2025	LangGraph orchestration, mock APIs, error simulation, LangSmith
Production Hardening	Q2 2025	Real OAuth, message queues, Redis state, Prometheus metrics
Advanced Features	Q3 2025	Human-in-loop approvals, ML anomaly detection, multi-tenant
Multi-Agent	Q4 2025	MCP servers, specialized agents, coordination protocols