Enterprise Customer Onboarding Agent — Solution Design Document

February 2025 | StackAdapt Case Study Submission

1. Architecture Overview

The solution implements an **AI-powered automation agent** that orchestrates enterprise customer onboarding from deal closure through SaaS provisioning. Built with **LangGraph** for state machine orchestration and **FastAPI** for the REST interface, the agent integrates with multiple enterprise systems, validates business rules, assesses risks using LLM intelligence, and takes autonomous actions including tenant provisioning and task management.

1.1 System Integrations

System	Objects & Key Fields
Salesforce CRM	`Account` (Id, Name, IsDeleted, Status, BillingCountry), `Opportunity` (StageName, Amount, CloseDate), `User` (IsActive, Email). [API Ref]
NetSuite ERP	`Invoice` (status, dueDate, amountRemaining, paymentStatus). [API Ref]
CLM System	Contract status, signatories, effective dates, key terms. Mock simulating DocuSign CLM.
Provisioning	Tenant creation + 14-task onboarding checklist with dependencies and due dates.

Field Selection Rationale: `Account.IsDeleted` validates account exists; `Opportunity.StageName="Closed Won"` confirms deal closure; `Invoice.status` identifies payment blockers. Minimizes API calls while capturing decision-critical data.

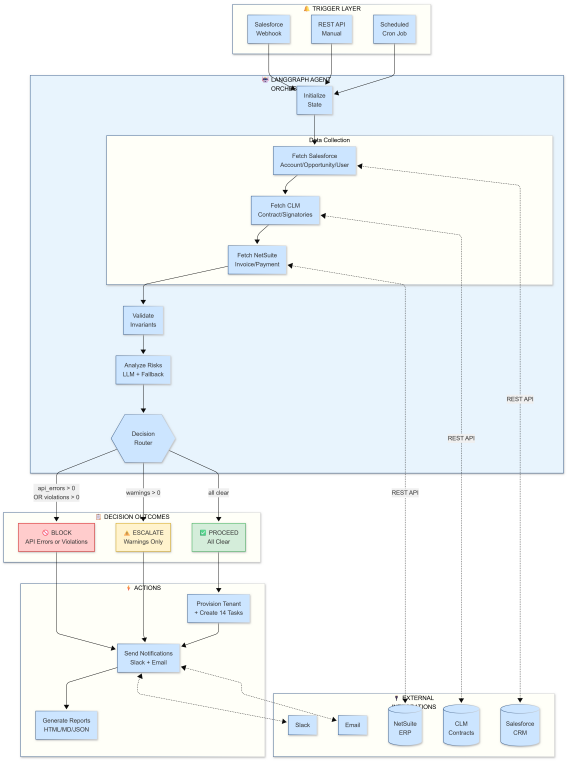

Figure 1: Architecture: triggers, LangGraph orchestration, integrations.

2. AI Agent Application

The agent leverages **OpenAI GPT-4** for intelligent analysis with a deterministic rule-based fallback, ensuring operation even without LLM connectivity.

2.1 LLM-Powered Intelligence

- **Risk Analysis**: Evaluates API errors, violations, warnings → risk levels (low/medium/high/critical) with business impact and resolution time.
- **Summary Generation**: Human-readable reports for CS teams, translating technical states to actionable insights.
- **Action Recommendations**: Prioritized steps with owner assignment (CS, Finance, IT, Legal).
- **Error Interpretation**: Converts technical codes (e.g., `INVALID_SESSION_ID`) to plain English with resolution steps.

2.2 Autonomous Actions

Upon decision: **(1)** Auto-provisions tenant with tier-appropriate config; **(2)** Creates 14-task checklist with dependencies/owners; **(3)** Sends Slack alerts; **(4)** Emails customer welcome; **(5)** Records API failures for audit.

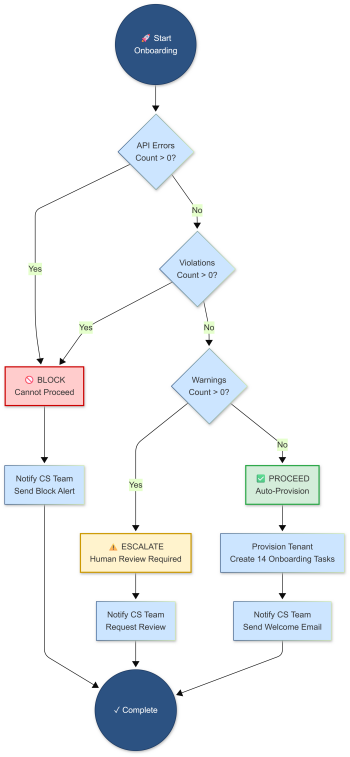

Figure 2: Decision: API Errors/Violations → BLOCK, Warnings → ESCALATE, Clear → PROCEED.

3. Orchestration & Event-Driven Flows

Trigger	Source	Endpoint
Webhook	Salesforce Flow	`POST /webhook/onboarding`
Manual	CS Team	`POST /demo/run/{id}`
Batch	Scheduler	`POST /demo/run-all`

State Machine: Init → Fetch(SF/CLM/NS) → Validate → Analyze(LLM) → Decide → [Provision] → Notify → Complete.

Error Simulation: `/demo/enable-random-errors` injects failures (401 auth, 400 validation, 429 rate limit, 500 server) for chaos testing without modifying business logic.

4. Trade-offs, Assumptions & Considerations

4.1 Key Trade-offs

Decision	Trade-off
API Errors → BLOCK	Integrity over speed
Sync Processing	Simple but slow
Rule-based Fallback	Less smart but reliable
14 Fixed Tasks	Predictable but rigid

4.2 Assumptions

- Salesforce is source of truth for accounts
- CLM status reflects actual signatures
- Invoice payment status is current
- CS monitors Slack for alerts
- Single onboarding per account

4.3 Limitations

- Mock integrations (not real APIs)
- In-memory state (lost on restart)
- Single instance (no scaling)
- No retry for transient failures
- No approval workflow UI

5. Multi-Agent Collaboration via Model Context Protocol (MCP)

The architecture supports multi-agent collaboration through MCP, enabling specialized agents to communicate via standardized tool interfaces.

Agent	Responsibility & MCP Tools
Coordinator	Orchestrates workflow (`salesforce.*`, `provision.*`)
Contract Agent	Monitors signatures (`clm.get_contract`)
Finance Agent	Tracks payments (`netsuite.get_invoice`)
Task Monitor	Detects overdue tasks (`tasks.get_overdue`)

Benefits: Domain expertise • Shared integrations • Scalable complexity • Cross-agent context sharing.

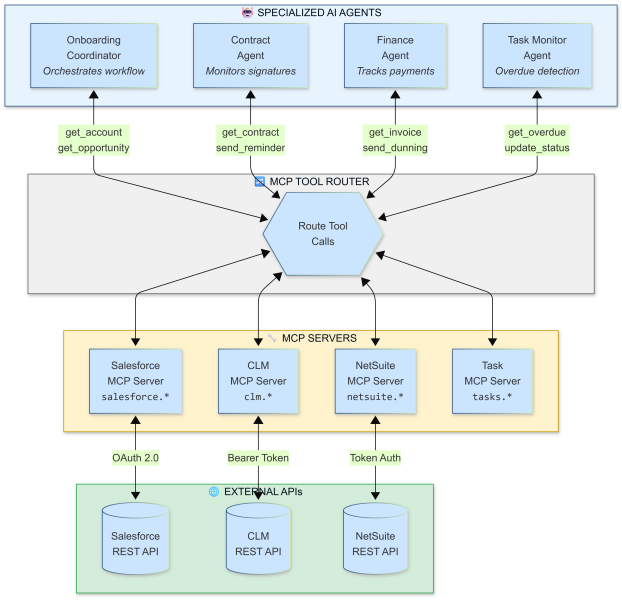

Figure 3: MCP: Agents collaborate via shared servers wrapping APIs.

6. Production Roadmap

Cloud Deployment

Frontend: A React-based operations dashboard providing real-time visibility into onboarding runs, decisions, and system health. Enables Customer Success and Operations teams to review onboarding outcomes, inspect violations or API errors, manually intervene when required, and track provisioning and onboarding task completion.

AWS Infrastructure: The onboarding agent runs as containerized services on ECS/Fargate, allowing horizontal scaling based on inbound webhook volume. LLM inference is handled via **AWS Bedrock** rather than direct OpenAI calls, enabling multi-model flexibility (Claude, Llama, Titan), private networking with traffic contained within the VPC, IAM-based authentication, and enterprise-grade security and compliance. Bedrock also allows cost control and model switching without code changes.

Docker + Kubernetes: Containerization ensures consistent environments across development, staging, and production. Kubernetes orchestration (or ECS equivalents) enables auto-scaling for traffic spikes (e.g., batch onboarding runs), self-healing services, rolling updates, and zero-downtime deployments as the agent logic evolves.

CI/CD & Observability

CI/CD (GitHub Actions / CodePipeline): Automated pipelines validate agent logic on every pull request, including unit tests for invariant checks, risk analysis, and decision routing. Deployments are reproducible and auditable, with fast feedback loops, reduced manual error, and the ability to quickly roll back in case of regressions affecting onboarding decisions.

LangSmith Enterprise: Provides full observability into LLM-powered components of the onboarding agent, including prompt inputs/outputs, token usage, latency, and cost tracking. Supports prompt versioning, regression detection, and A/B testing of risk analysis and summary generation logic without impacting the deterministic rule-based fallback.

Monitoring & Alerting: Prometheus metrics capture agent throughput, error rates, and decision distributions. Distributed tracing via DataDog or Jaeger enables end-to-end visibility across webhook ingestion, integrations, LLM calls, and provisioning. PagerDuty alerts notify on-call teams of critical failures such as sustained API outages or elevated BLOCK rates.