Affirm AI Agents Interview Prep

Interview Process (Typical for Affirm)

1. Recruiter screen (30 min)
2. Hiring manager (45-60 min)
3. Technical coding (60 min) - likely Python backend or React frontend
4. System design (60 min) - focus on orchestration, APIs, scalability
5. Behavioral/leadership (45-60 min)
6. Team fit (45-60 min)
7. Final round with leadership

Your Main Story

"I've spent 4 years at Walmart building platforms that orchestrate multiple systems to solve problems - exactly what AI agents do. I'm a top 2% Cursor AI user who's integrated LLMs into production tools. I thrive in ambiguous, newly formed teams. This role combines my system orchestration expertise with my AI journey - it's where I want to go next."

Key Talking Points

1. AI/LLM Experience

What you've done:

- Top 2% Cursor AI user globally
- Integrated LLM APIs (OpenAI, Claude) into internal tooling
- Built AI-powered report generation
- Pioneered AI-assisted development on team (40% productivity boost)
- Mentored engineers on AI workflows and prompt engineering

Why it matters: This role is about building AI agents - you're already an AI adopter.

Be honest about scope:

- "I've integrated LLMs into tools, not built full conversational agents"
- "I understand API calls, prompt engineering, handling responses"
- "I want to learn conversation flow design, context management, fallback handling"

Questions they'll ask:

- "What's your AI/LLM experience?" → Be specific: integrated APIs, used for code generation, built internal tools
- "Have you built conversational AI?" → Honest: "Not conversational agents yet, but I've built orchestration systems that coordinate multiple services based on user intent - similar pattern"
- "How do you handle LLM errors?" → Retry logic, fallbacks, user-friendly error messages, logging for improvement

2. System Orchestration (This Is Your Strength)

What you built:

- Team Portal: orchestrates 10+ systems (data, jobs, secrets, deployments)
- Event Automation: orchestrates multi-region deployments with approvals
- Multi-Signal Scoring: coordinates external APIs and internal services

The parallel to AI agents:

- User states intent → system figures out execution
- Coordinate multiple services to solve problem
- Provide feedback on progress
- Handle failures gracefully

This is THE key connection - you've been building what AI agents need on the backend.

Questions they'll ask:

- "How would you design an AI agent backend?" → Orchestration layer that routes requests, calls services, aggregates responses, handles errors
- "How do you coordinate multiple services?" → Event-driven architecture, REST APIs, async processing, retries, circuit breakers
- "How do you handle service failures?" → Graceful degradation, fallbacks, user-friendly messaging, retry logic

3. Backend at Scale (Python, AWS, Kubernetes)

What you've built:

- Python/Go microservices processing millions of events/second
- AWS infrastructure, Kubernetes deployments
- 99.99% uptime systems with comprehensive monitoring
- Low-latency APIs (sub-50ms p99)

Why it matters: AI agents need reliable backend infrastructure at scale.

Questions they'll ask:

- "Tell me about a system you built at scale" → Pick any Walmart system, emphasize reliability, monitoring, handling traffic spikes
- "How do you ensure API reliability?" → Health checks, circuit breakers, retries, monitoring, alerting, graceful degradation
- "Python vs Go - which do you prefer?" → "Both have places. Python for rapid development, data processing, ML integration. Go for high-performance services, concurrency. I'm comfortable with both."

4. Frontend (React)

What you've built:

- React frontends for Team Portal, Event Automation, ELSA
- Modern web development experience
- User interface design for internal tools

Potential gap:

- Your React experience is for internal tools, not customer-facing
- AI agent interfaces might need more polish

How to position:

- "I've built React apps for internal engineers, not consumer-facing yet"
- "I understand component design, state management, API integration"
- "Customer-facing UI requires more attention to UX - excited to learn"

Questions they'll ask:

- "How would you design an AI chat interface?" → Message history, input field, typing indicators, error states, accessibility
- "What React patterns do you use?" → Hooks, context, component composition, keeping state minimal
- "How do you handle API calls in React?" → Async/await, loading states, error handling, optimistic updates

5. Leadership & Ownership in Ambiguity

What you've done:

- Launched systems with vague requirements (IntelDb, Event Automation)
- Led WCNP Kubernetes migration (ambiguous, evolving)
- Mentor 8 engineers, run design reviews
- Take ownership end-to-end: design, build, monitor, maintain

Why it matters: "Newly formed team taking on big unknowns" - this is your sweet spot.

Questions they'll ask:

- "Tell me about working in ambiguity" → Use IntelDb story: vague requirements, prototyped to clarify, iterated with users, delivered production system
- "How do you take ownership?" → Own from design to production to on-call, care about long-term health, proactive about improvements
- "How do you lead through unknowns?" → Break down problem, prototype to learn, communicate progress, adjust based on feedback

---

Addressing Potential Concerns

Concern: "You haven't built conversational AI agents"

Response: "True - I've integrated LLMs into tools, but not built full conversational agents. But I have built orchestration systems that take user intent, coordinate multiple services, and provide feedback - that's the backend AI agents need. I understand how to call LLM APIs, handle responses, manage errors. What I want to learn is conversation design, context management across turns, and measuring success for conversational interfaces. I'm confident I can ramp up quickly."

Concern: "You're at Staff level, this is Senior"

Response: "I've been doing Staff-level work at Walmart, but I'm applying for Senior because I'm excited about the AI agents space and the newly formed team. I care more about impact and learning than title. If I prove myself quickly and the role grows, we can revisit level. But I'm happy to come in as Senior and earn my way up."

Concern: "You don't know fintech/BNPL"

Response: "True - I've worked in e-commerce and education, not fintech. But I learn domains quickly by talking to users and stakeholders. What I bring is technical expertise in building reliable systems. I'll learn the BNPL domain specifics - understanding customer pain points, regulatory constraints, business metrics. The fundamentals of building good software transfer across domains."

Behavioral Questions

"Tell me about taking ownership in an ambiguous project"

Story: IntelDb Threat Intelligence System

- Situation: Leadership said "we need real-time threat intelligence" but requirements were vague
- My approach: Talked to users (security analysts), understood pain points, prototyped quickly
- Challenges: Requirements kept changing as we learned more
- Ownership: I drove clarification, made technical decisions, delivered production system
- Outcome: Now processes 10M+ events/day, became critical infrastructure
- Lesson: In ambiguity, prototype to learn, stay flexible, drive toward decisions

"Tell me about mentoring/developing others"

Story: AI Adoption at Walmart

- Situation: Team wasn't using AI tools effectively
- My approach: Led by example (became top 2% Cursor user), shared techniques, paired with engineers
- Taught: Prompt engineering, when AI helps vs hinders, how to verify AI suggestions
- Impact: Team productivity increased 40%, AI adoption spread
- Philosophy: Teaching by doing, making it safe to experiment, celebrating successes

"Tell me about handling a production incident"

Story: Pick any Walmart incident

- Detection: Monitoring alerted, on-call responded
- Triage: Checked dashboards, logs, recent changes
- Mitigation: Rolled back deployment / scaled up / fixed bug
- Root cause: Thorough analysis, not just surface fix
- Prevention: Added monitoring, improved testing, updated runbooks
- Communication: Kept stakeholders informed throughout

"Tell me about disagreeing with someone"

Story: Kubernetes Migration Approach

- Situation: Colleague wanted to migrate everything to K8s immediately
- My position: Too risky, some services not ready
- My approach: Presented data on similar migrations, risk analysis, proposed phased approach
- Outcome: Team agreed on incremental migration, avoided major incidents
- Lesson: Disagree with data and alternatives, not just criticism

---

Technical Questions

Coding (Python/React)

Python Backend:

- REST API design
- Async processing with asyncio
- Data processing with pandas
- Error handling and retries
- Testing (pytest)

React Frontend:

- Component design
- State management

- API integration
- Error handling
- Accessibility

Prep: Practice LeetCode medium problems, system design coding

System Design: "Design an AI agent system for customer service"

Clarify requirements:

- Volume: How many conversations simultaneously?
- Channels: Web chat, SMS, phone?
- Vendor: Already using omnichannel vendor or building from scratch?
- Integration: What internal systems need to be accessed?
- Latency: Real-time (<1s response)?

High-level design:

1. **Frontend Layer**
    - Chat interface (React)
    - WebSocket for real-time communication
    - Message history, typing indicators
2. **API Gateway**
    - Routes requests to appropriate services
    - Authentication, rate limiting
    - Load balancing
3. **Orchestration Layer** (Your Strength!)
    - Receives user intent from AI vendor
    - Routes to internal services (account lookup, payment processing, etc.)
    - Aggregates responses
    - Returns structured data to AI vendor
    - Handles retries, timeouts, errors
4. **AI Vendor Integration**
    - Send user message → receive intent + parameters

- Webhook for actions that need Affirm systems
- Return results → AI generates response

5. **Internal Services**
    - Account service
    - Payment service
    - Transaction history
    - Each with REST APIs, authentication
6. **Monitoring/Logging**
    - Conversation logs (for improvement)
    - Latency metrics
    - Error rates
    - Customer satisfaction scores

Deep dives:

- How to handle AI vendor downtime? → Fallback to human agents
- How to handle internal service failures? → Graceful degradation, inform user
- How to ensure security? → Authentication, authorization, PII handling
- How to improve over time? → Log conversations, analyze failures, retrain

Your edge: "This is exactly what I've built at Walmart - orchestration layers that coordinate multiple systems. Let me walk through similarities."

"How would you test an AI agent?"

Layers of testing:

1. **Unit tests** - Individual components
2. **Integration tests** - Service interactions
3. **End-to-end tests** - Full conversation flows
4. **AI testing** (unique challenges):
    - Test prompt variations
    - Test edge cases (typos, ambiguous requests)
    - Test fallback handling

- Measure response quality (human evaluation)

5. **Load testing** - Can it handle traffic?
6. **A/B testing** - Compare agent versions

Monitoring in production:

- Conversation success rate
- Escalation to human rate
- Customer satisfaction
- Response latency
- Error rates by intent type

Questions to Ask Them

For Hiring Manager:

1. "What does success look like for the AI Agents team in 6 months? 1 year?"
2. "What's the biggest unknown the team is tackling right now?"
3. "How do you measure if the AI agent is working well?"
4. "What's the vendor relationship like? How much do we build vs use their platform?"

For Team Members:

1. "What's the most interesting technical challenge you've faced with AI agents?"
2. "How do you handle when the AI doesn't understand a customer?"
3. "What's the deployment process like? How often do you ship?"
4. "What's your favorite part about working on this team?"

For Product:

1. "What customer problems are we solving with AI agents vs human agents?"
2. "How do customers feel about talking to AI vs humans?"
3. "What metrics matter most to the business?"

For Everyone:

1. "What's Affirm's culture like day-to-day?"
2. "How does remote-first work in practice?"

Compensation Negotiation

Affirm Range:

- CA/WA/NY/NJ/CT: $190K - $240K
- Other states: $169K - $219K

Your Target: $210K-$240K (top of higher range)

Strategy:

- Anchor high: "I'm currently at Walmart making [your comp]. I'm targeting the top of your range at $230K-$240K given my experience."
- Emphasize value: "I bring senior/staff-level experience, AI expertise, and platform engineering skills."
- Be flexible: "I'm also interested in equity and excited about Affirm's growth."
- If they push back: "I'm flexible if the opportunity is right. What's the typical offer for someone with my background?"

Remember: You have Netflix options too. Don't undersell yourself.

Final Mindset

Your strengths:

- System orchestration (exactly what AI agents need)
- AI adoption (top 2% user, production LLM integration)
- Leadership in ambiguity (newly formed team needs this)
- Backend expertise (Python, AWS, Kubernetes, APIs)
- Ownership mentality (take it from idea to production)

Your gaps:

- No full conversational AI agents (yet)
- Limited fintech domain knowledge
- React is more internal tools than consumer-facing

The pitch: "I've built the orchestration systems AI agents need on the backend. I'm an AI adopter who's integrated LLMs in production. I thrive in newly formed teams with big unknowns. I want to learn conversational AI design and apply my platform engineering skills to customer-facing agents."

Be excited, be honest, be yourself.

This role is a great fit for where you are and where you want to go. Go get it! 🚀