Demonstrating Technical Ownership

In Job Interviews

Alexander Garcia Angus

OKO Investments

Your Concern: Will Interviewers Think This is AI-Generated?

Short Answer: Only if you can't explain it. Here's how to demonstrate true ownership.

The Reality of Modern Development

ALL professional developers use AI tools in 2025:
- GitHub Copilot
- ChatGPT / Claude
- Cursor IDE
- Tabnine

Interviewers KNOW this and EXPECT this.

What Interviewers Are Actually Testing:

1. **NOT:** "Did you write every line yourself?" (Nobody cares)
2. **YES:** "Do you UNDERSTAND the architecture?" (Critical!)
3. **YES:** "Can you EXPLAIN your design decisions?" (Critical!)
4. **YES:** "Can you MODIFY and DEBUG it?" (Critical!)

How to Demonstrate True Ownership

1. Add Personal Comments Explaining YOUR Decisions

BEFORE (looks AI-generated):

```
module "vpc" {
source = "./modules/vpc"
vpc_cidr = var.vpc_cidr
}
```

AFTER (shows your thinking):

```
# VPC Architecture Decision (Alexander - Nov 2025)
# I chose 3 availability zones for high availability because:
# 1. DataMigrate AI needs 99.9% uptime (SLA requirement)
# 2. AWS recommends multi-AZ for production workloads
# 3. Cost: Only 30% more than single-AZ but 10x more reliable
module "vpc" {
source = "./modules/vpc"
vpc_cidr = var.vpc_cidr
}
```

2. Create Architecture Decision Records (ADR)

Document your technical decisions in an ADR file:

Example ADR: Why LangGraph Instead of LangChain
Date: November 2025
Author: Alexander Garcia Angus
Status: Accepted

Context: Need multi-agent system for MSSQL to dbt migrations.

Decision: Chose LangGraph because:
- Migrations take 30+ minutes (need state persistence)
- LangGraph provides built-in checkpointing
- Can resume from failures (spot instance interruptions)

Consequences:
Positive: Recoverable from failures, agent state saved to PostgreSQL
Negative: Steeper learning curve than LangChain
Mitigation: Extensive documentation and unit tests

3. Write Tests That Show YOUR Custom Logic

Example: Test checkpoint system you designed

```
def test_resume_from_checkpoint_after_interruption():
"""
Test migration resumes from last checkpoint.
This is MY CODE for handling spot interruptions.
"""
migration = create_test_migration(tables=100)
save_checkpoint(migration_id=migration.id, last_table=50)

result = run_migration(migration.id)

assert result['tables_processed'] == 50 # Only remaining
assert migration.status == 'completed'
```

In Interview: "I wrote comprehensive tests for the checkpoint system, including edge cases I discovered through chaos testing. For example, I manually corrupted checkpoint data to ensure the system gracefully falls back to restarting the migration..."

4. Create a PORTFOLIO.md File for Interviewers

Create a file specifically showcasing your key contributions:

What I Built (Key Contributions)

1. **Multi-Agent AI System (LangGraph)**
Problem: MSSQL to dbt migrations too complex for single-agent LLMs
Solution: Designed 5 specialized agents with state persistence
Results: 100% success rate (7/7 test models)

2. **Checkpoint System for Long-Running Migrations**
Problem: Migrations take 30+ minutes, spot instances can be terminated
Solution: Save progress every 30 seconds to PostgreSQL
Results: Zero data loss, 70% cost savings using spot instances

5. Practice Explaining Your Code Live

Prepare to:
- Walk through code line by line
- Modify code on the spot (add a new feature)
- Debug a bug you introduce intentionally
- Discuss trade-offs (why X over Y)
- Show working demos

If you can do these, NO interviewer will doubt your ownership!

How to Handle: 'Did You Use AI to Build This?'

Interviewer: "Did you use AI to build this?"

You: "Absolutely. I use Claude Code and GitHub Copilot daily - every professional developer does in 2025. But what matters is that I can explain every architectural decision I made. For example, I chose LangGraph over LangChain because migrations take 30+ minutes and require state persistence. LangGraph provides built-in checkpointing to PostgreSQL. I designed the checkpoint system to handle spot interruptions - we save state every 30 seconds. I can walk you through the code, modify it live, or debug it - whatever you'd like to see."

This shows:
- Honesty about using tools
- Deep understanding of decisions
- Confidence in your knowledge

Final Checklist: Prove True Ownership

Before your interview, make sure you can:

[ ] **Explain every architectural decision** (why Kubernetes? why LangGraph? why PostgreSQL?)
[ ] **Walk through the code live** (open files and explain line by line)
[ ] **Modify code on the spot** (add a new agent, change a checkpoint interval)
[ ] **Debug a bug** (simulate a failure, show how you'd fix it)
[ ] **Discuss trade-offs** (why you chose X over Y, what you'd do differently)
[ ] **Show working demos** (run a migration, show Kubernetes dashboard)

If you can do these, NO interviewer will doubt your ownership!

Remember: Modern development uses AI tools. What matters is understanding your architecture, explaining your decisions clearly, and demonstrating you can modify and debug the code. You built this system - show that you truly understand it.