SECTION 1 — Introduction (Tell Me About Yourself)

Final Answer (Improved):
“Hi, I’m Nikhila. I recently completed my Master’s in Computer Science at UNC Charlotte, where my coursework covered algorithms, AI/ML, NLP, databases, software systems design, mobile development, data mining, and networks.
Before my master’s, I spent four years at Accenture working across healthcare and finance clients.
I engineered distributed data pipelines using PySpark, Airflow, and Kafka, migrated Hadoop workloads to AWS EMR to cut ETL runtime by ~40%,
built FastAPI microservices with Docker and Kubernetes, and automated ML workflows with SageMaker and Vertex AI. I also built NLP components using BERT and Elasticsearch.
Recently, I built a Hybrid RAG retrieval system and a multi-agent IVR system with policy-based routing.
These strengthened my debugging, backend development, and system design fundamentals.
I’m strongest in Python and Java and enjoy building scalable backend/data systems.
I’m excited about Intuit because of its engineering culture, customer impact, and opportunities to grow while contributing meaningfully.”

SECTION 2 — STAR Stories (Behavioral Excellence)

■ STAR STORY 1 — Automation Portal (Your BEST story)
Theme: Ownership • Process Improvement • Backend Skills • Customer Impact
■ Expanded, Impressive Version:
“At Accenture, our team validated monthly Cigna claims data using 40–50 TOAD SQL scripts. Every cycle took several hours, was repetitive, and error-prone. Analysts would run scripts one by one, validate outputs manually, and it slowed down the entire pipeline.
My goal was to automate the entire validation workflow.
I designed an internal FastAPI service that wrapped all our SQL checks into callable endpoints, containerized it using Docker, and integrated it with a lightweight internal UI. Instead of manually running scripts, analysts now clicked a single button that triggered the entire suite. I also added structured logs, error surfacing, and priority-based checks so failures were easier to diagnose.
The result was huge: validation time dropped by about 50%, we eliminated manual execution errors, and our downstream ETL tasks started earlier, improving SLA reliability. The portal became the default workflow across our team.
This project taught me how impactful small engineering improvements can be, and it strengthened my confidence in designing end-to-end automation.”
■ STAR STORY 2 — ETL Runtime Optimization (Technical Depth)
Theme: Debugging • Distributed Systems • Data Engineering • Performance
■ Expanded Version:
“One of our key PySpark ETL pipelines at Cigna was running close to SLA deadlines as data volume grew. The job handled multiple regions, large joins, and heavy transformations. Failures started increasing and the pipeline was taking too long.
I took ownership of diagnosing the bottlenecks.
I profiled the PySpark DAG, inspected shuffle stages, and identified expensive wide joins and skewed partitions. I optimized join strategies by broadcasting smaller tables, re-partitioned data based on natural keys, pruned unnecessary transformations, and parallelized region-level ETLs so East, West, Central ran concurrently.
After these fixes, the ETL runtime decreased by about 40%, significantly reducing SLA risk. Failures dropped, and the business team received reports earlier in the day.
This experience built my confidence in debugging distributed systems, thinking about scalability, and optimizing performance under production constraints.”
■ STAR STORY 3 — Upstream Data Issue (Cross-Team Leadership)
Theme: Communication • Ownership • Customer Focus • Problem Solving
■ Expanded Version:
“One month, our ETL pipeline kept failing just after ingestion. The issue wasn’t in our code, but in the raw data — several fields like health codes and provider data were missing for two regions. This blocked the entire workflow.
I took ownership of diagnosing the root cause.
I compared volume trends across months, checked load timestamps, and confirmed that the upstream extract team had changed their incremental load scheduling but hadn’t updated the metadata flags delivered to us.
I reached out, explained the technical and business impact, and worked closely with their team to correct the extract logic. Internally, I added an automated alert: if incoming data volume dropped below expected thresholds, the system would immediately notify us.
This collaborative fix prevented recurring failures, and monthly loads became consistently reliable.
It strengthened my cross-team communication skills and taught me how clarity and initiative can prevent repeated outages.”
■ STAR STORY 4 — Mentoring Juniors (Leadership Without Title)
Theme: Leadership • Teaching • Collaboration • Team Impact
■ Expanded Version:

“As our team grew, several junior analysts joined and were struggling with the complexity of Cigna’s claims logic and the validation workflow. They needed support to become productive quickly, and it was slowing the team down.
I took the initiative to mentor them.
I conducted walkthroughs on claims schema, pipeline flow, and common pitfalls. I paired with them during their first few cycles, reviewed their queries, and documented repeat problems and best practices.
Within a couple of cycles, they were fully independent, confident, and contributing smoothly. Our team velocity increased, and onboarding time reduced significantly.
This experience made me comfortable explaining technical concepts clearly and supporting teammates regardless of age, background, or experience.”
■ STAR STORY 5 — Conflict / Communication Management
Theme: Emotional Intelligence • Adaptability • Ownership • Maturity
■ Expanded Version:
“In monthly delivery cycles, I often had to coordinate with stakeholders who were both much older and much younger than me. Communication styles varied, and misunderstandings sometimes caused delays or friction.
I realized communication needed to adapt depending on the stakeholder.
For senior members, I used structured, data-driven updates with clear timelines and metrics. For junior analysts, I used step-by-step guidance, reassurance, and walkthroughs.
I also set up early reminders, documented expectations, and escalated respectfully only when needed.
This dramatically reduced last-minute issues, improved trust, and ensured cycles ran smoothly.
It taught me that technical work is important, but communication often determines team success.”
■ OPTIONAL EXTRA: STAR STORY 6 — Migration to AWS EMR
(If interviewer probes more on data engineering / cloud)
■ Expanded Version:
“Our Hadoop cluster was static and struggling with scale. The monthly ETL workload hit resource ceilings causing retries and slowdowns. I helped migrate these Spark jobs onto AWS EMR to use autoscaling, spot instances, and S3-based storage. I containerized dependencies, rewrote configs, and tested compatibility.
Once deployed, runtime improved by ~40%, failures decreased, and we gained elasticity.
This experience strengthened my understanding of cloud-based distributed compute.”
■ OPTIONAL EXTRA: STAR STORY 7 — NLP Pipeline with BERT + Elasticsearch
(If interviewer asks about NLP or ML)
■ Expanded Version:
“For a healthcare text processing project, we needed a better way to extract entities and perform semantic search on provider descriptions. I built an NLP pipeline using a BERT-based NER model, generated embeddings, and indexed them into Elasticsearch with custom analyzers.
This enabled fast top-k semantic retrieval and improved accuracy significantly over keyword matching.
I learned how ML, vector stores, and backend systems come together in real-world search applications.”

SECTION 3 — Core Interview Questions (With Improved Answers)

Q2. Why Intuit?
Intuit’s products touch millions, and the engineering culture encourages experimentation, rapid prototyping, and customer-obsessed design. My background in backend systems, AI, and data pipelines aligns well with Intuit’s tech stack, and I’m excited to learn and contribute in an environment with such strong mentorship and impact.

Q3. Strongest Programming Languages?
Python → ML pipelines, RAG, backend APIs, data workflows
Java → Backend, system fundamentals, Android development
Comfortable writing modular, testable code in both.

Q4. Debugging Approach
1. Reproduce reliably
2. Inspect logs, stack traces
3. Narrow the failing area
4. Binary-search through execution path
5. Form hypothesis, run small tests
6. Fix → refactor → regression tests
7. Add guardrails
Used in both RAG chunking fixes & IVR agent handoff debugging.

Q5. How do you use AI Tools?
For boilerplate, brainstorming, generating test cases, validating understanding.
I treat them as assistants — not as authoritative sources.

Q6. How do you verify AI-generated code?
Run personal test cases, review logic line-by-line, check edge cases, verify complexity, write small unit tests, and rewrite suspicious parts.

Q7. Explain Hybrid RAG
* Semantic chunking
* Dense + sparse retrieval (BM25 + embeddings)
* Hybrid scoring
* Metadata filtering
* Reranking
* Grounded LLM response → lower hallucination rates

Q8. Explain Multi-agent IVR System
* Agents for classification, routing, answering, fallback
* Policy-based handoffs
* State tracking for seamless transitions
* Reduced confusion by 40%

Q9. What excites you engineering-wise?
Backend systems, APIs, retrieval pipelines, debugging, performance optimization, and real-world user-facing systems.

Q10. Hardest Technical Challenge

RAG chunking issues → fixed using sentence-based segmentation + overlapping windows → improved retrieval quality significantly.

Q11. What do you want in your next role?

Strong mentorship, backend/fullstack responsibilities, meaningful projects, exposure to large-scale systems.

Q12. Comfortable with next rounds?

Yes — confident in DSA (Python/Java) and SE1-level system design.

SECTION 4 — AI Tooling & Responsible Usage

Easy Questions
* What AI tools do you use?
* What parts of coding do you use AI for?
# **Moderate Questions**
* How do you verify AI output?
* How do you avoid over-dependence?
* Example of improving an AI-generated solution?
# **Difficult Questions (NEW ANSWERS ADDED)**
Q: Tell me about a time AI gave you a wrong answer. How did you catch it?
AI once produced incorrect regex logic for my RAG chunk preprocessing. I caught it because the output chunks didn't align with sentence boundaries. I validated manually, rewrote the logic myself, and added rule-based tests so similar issues wouldn't slip through again.
Q: If AI tools disappeared tomorrow, what would you do?
Go back to fundamentals: documentation, official guides, small prototype scripts, debugging with print/logs, and pair programming. AI accelerates, but it doesn’t replace core engineering thinking.
Q: What’s your philosophy on AI vs human engineers?
AI is a powerful accelerator, but engineers provide context, correctness, architecture decisions, and reliability. AI enhances productivity, but engineering judgment is irreplaceable.

SECTION 5 — Role Fit, Technical Stage & Growth Questions

Easy
* Comfortable with more coding/system rounds? yes.
* Availability? flexible.
# **Moderate**
* What interview type are you strongest in?
Backend + debugging + DSAs + small-scale system design.
Difficult
Q: How do you prepare for technical interviews effectively?
Daily DSA practice, reviewing patterns (two pointers, sliding window, DP), building 1–2 small system design problems, and revising core concepts in APIs, DB design, and concurrency.

SECTION 6 — Attitude, Growth & Work Style

Easy
* Teamwork preference? Collaborative, proactive communicator.
* Handling stress? Break down tasks, prioritize, communicate early.
# **Moderate**
Q: Tell me about a time you disagreed technically.
At Accenture, during ETL optimization, a teammate wanted to expand resources instead of optimizing logic. I proposed DAG and partition tuning first. We tested both approaches — mine resulted in faster performance with fewer costs, and the team adopted it. Transparent data-driven discussions build trust.
Difficult
Q: Describe a failure. What did you learn?
During early Accenture days, I underestimated the complexity of claims logic and delivered incomplete validation rules. I learned to ask clarifying questions, document assumptions, and test with edge cases. It made me a stronger and more careful engineer.
Q: Describe yourself in 3 words.
Curious, reliable, detail-oriented.
(Or: calm, analytical, ownership-driven.)
