Admissions & Support: support@mockattempt.com
Phone / WhatsApp: +91 98765 43210
Course Code: GAIF • **Updated:** 03 Nov 2026
https://mockattempt.com/training/genai-full-stack-engineering-program

GAIF • GENERATIVE AI & LLM SYSTEMS **BEGINNER TO EXPERT • JOB-GUARANTEED*** **100% PLACEMENT GUARANTEED* CAREER PROGRAM (6-MO REFUND*)**

Generative AI, Large Language Models (LLM) & Agentic AI Full Stack Engineering

Complete Zero-to-Expert Python, DSA, Math/ML, SQL & Web Foundations to Django/FastAPI, React 19, PyTorch, Hybrid RAG, LangGraph, MCP, QLoRA, vLLM & AWS LLMOps | 100% Placement Guaranteed* (6 Months Fees Refund*)

Duration: 36w (540h) • Weekend Only (2h/Day) (568 Hrs) • **Modules:** 22 • **Live Topics:** 86 • **Subtopics:** 688+ • **Labs & Tasks:** 172 • **Mocks / AI Vivas:** 172 / 172 • **Projects:** 22 Module + 1 Live Client Prod

All-Inclusive Direct Fee (Incl. 18% GST • Zero Hidden Charges): Total Payable: ₹39,999 (Net Tuition ₹33,897 + Included 18% GST ₹6,102) • **Split Option:** ₹6,000/mo × 6 (0% Split) • **Effective ROI:** ~₹70/Guided Hr (~₹465/Live Topic)

Faculty: 100% Active Working IT Professionals • **Weekend-Only Live Batch (7:00 AM – 8:00 PM IST Flexible Slots):** Weekend-Only Career Live Batch (Sat – Sun • 2 Hrs/Day • 7:00 AM – 8:00 PM) (Saturday & Sunday Only (2 Hours Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 2-Hour Weekend Slot Based on Working Professional Trainer Availability) • 36 Weeks (568 Guided Live + Lab Hours))

✓ **Trainer LeetCode & HackerRank Mentorship (Python 3.12, TypeScript 5 & SQL): Live trainer support to crack LeetCode & HackerRank problems, optimize complexity, and handle edge cases.**

✓ **6-Month Placement Guarantee* + Online AI Assessment (OA) Support: Complete 100% assigned repositories & score ≥90% across all Topic Mock Tests & Voice AI Interviews → 100+ verified IT job openings within 6 months with active Senior Trainer coaching to crack employer Online AI Assessments (OA), or 100% net tuition refund*.**

Core Technologies & Engineering Stack: Python, DSA & SQL From Zero • FastAPI, Django 5 & React 19 TS • PyTorch, Hybrid RAG & LangGraph • MCP, QLoRA, vLLM & AWS LLMOps • NumPy, Pandas & Scikit-Learn ML • HTML5, CSS3 & Generative UI • Transformer & Attention Internals • PostgreSQL pgvector & Qdrant • GraphRAG, Reranking & RAGAS Evals • CrewAI & Spoken Voice AI Agents • LoRA / QLoRA & DPO Fine-Tuning • LLM Guardrails & Red-Teaming • Docker, Kubernetes & AWS Bedrock • Terraform, CI/CD & Langfuse Tracing

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Start from Day-1 zero coding experience and master every foundation, framework, database, cloud, and system design layer up to expert level
2. Build and ship 22 real-world production GitHub repositories + 1 Live Client Production Cloud Deployment (zero dummy to-do apps)
3. Achieve ≥90% mastery across every Topic Mock Test and Spoken MockAttempt AI Technical Interview (unlimited free re-attempts allowed below 90%)
4. Earn Dual Barcode-Verifiable Credentials: Course Excellence Certificate + Official Project Internship Confirmation Letter
5. Unlock 6-Month Placement Support guaranteeing 100 curated relevant IT job openings — backed by our 100% Tuition Fee Refund* policy
6. Crack LeetCode and HackerRank problems in Python 3.12, TypeScript 5 & SQL with live trainer mentorship on algorithmic/query patterns, edge cases, and optimal complexity.
7. Receive active Senior Trainer coaching during your 6-month (180-day) Placement Tenure to crack employer Online AI Assessments (OA), timed coding rounds, and automated screening tests.
8. Strong full-stack foundation: Python 3.12, async FastAPI, Django 5, React 19 + TypeScript 5, PostgreSQL 16, and Redis 7.

Eligibility & Prerequisites: Zero prior programming or IT experience required — Module 1 starts from Day-1 setup, logic building, and plain-English fundamentals • Basic computer literacy and commitment to attend Weekend-Only Live Sessions (Saturday & Sunday, 2 Hours/Day between 7:00 AM – 8:00 PM IST taught by active Working IT Professionals) plus guided weekday real-repository sprints • Dedication to complete 100% of assigned tasks and re-attempt assessments until crossing ≥90% mastery • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Dedication to write original code in your GitHub portfolio and complete the mentor-guided Live Client Production Sprint.

2. DETAILED ZERO-TO-EXPERT MODULE, TOPIC & SUBTOPIC SYLLABUS (22 MODULES • 86 LIVE TOPICS • 688+ SUBTOPICS)

Progressive Zero-to-Expert Mastery Stages: Modules 1–5: Stage 1: Day-1 Foundations & Core Mental Models (20 Topics / 160 Subtopics) • Modules 6–12: Stage 2: Applied Engineering, Data & API Workflows (28 Topics / 224 Subtopics) • Modules 13–17: Stage 3: Internals, Concurrency, Security & Performance (20 Topics / 160 Subtopics) • Modules 18–22: Stage 4: Cloud Scale, System Design & ≥90% Placement Gate (18 Topics / 144 Subtopics)

MODULE 1 OF 22 • STAGE 1 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Programming Foundations, Linux CLI, Git & Python 3.12 From Absolute Scratch

Beginner-to-expert real-repository module covering Programming Foundations, Linux CLI, Git & Python 3.12 From Absolute Scratch with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ How Computers & AI Software Execute, Python 3.12 Setup, VS Code, uv / venv & First Verified Programs — Purpose: Start from zero prior coding experience and write clean, well-structured Python programs from Day 1. • ✓ Linux Command Line Navigation, Shell Pipelines, SSH & Professional Git / GitHub PR Workflows — Purpose: Operate Linux GPU/CPU cloud terminals and manage clean Git branches and pull requests like a pro. • ✓ Python Variables, Primitive Types, String Manipulation, Unicode/UTF-8 Text & Numeric Operators — Purpose: Process raw text, numbers, and encodings cleanly — the fundamental building blocks of NLP and AI data. • ✓ Control Flow (if/elif/else, match-case), Loops (for/while), Trace Dry-Runs & Logical Problem Solving — Purpose: Build rock-solid programming logic through 30+ progressive conditional, loop, and pattern challenges.

1.1 How Computers & AI Software Execute, Python 3.12 Setup, VS Code, uv / venv & First Verified Programs [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for How Computers • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply How Computers & AI Software Execute, Python 3.12 Setup, VS Code, uv / venv & First Verified Programs in real Generative AI, Large Language Models...

What Topic Is Next → Next Topic 1.2: Linux Command Line Navigation, Shell Pipelines, SSH & Professional Git / GitHub PR Workflows

1.2 Linux Command Line Navigation, Shell Pipelines, SSH & Professional Git / GitHub PR Workflows [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Linux Command Lin... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Linux Command Line Navigation • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Linux Command Line Navigation, Shell Pipelines, SSH & Professional Git / GitHub PR Workflows in real Generative AI, Large Language Models (LLM) &...

What Topic Is Next → Next Topic 1.3: Python Variables, Primitive Types, String Manipulation, Unicode/UTF-8 Text & Numeric Operators

1.3 Python Variables, Primitive Types, String Manipulation, Unicode/UTF-8 Text & Numeric Operators [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Python Variables • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply Python Variables, Primitive Types, String Manipulation, Unicode/UTF-8 Text & Numeric Operators in real Generative AI, Large Language Models (LLM)...

What Topic Is Next → Next Topic 1.4: Control Flow (if/elif/else, match-case), Loops (for/while), Trace Dry-Runs & Logical Problem Solving

1.4 Control Flow (if/elif/else, match-case), Loops (for/while), Trace Dry-Runs & Logical Problem Solving [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Control Flow (if/... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Control Flow (if/elif/else, match-case) • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Control Flow (if/elif/else, match-case), Loops (for/while), Trace Dry-Runs & Logical Problem Solving in real Generative AI, Large Language Models...

What Topic Is Next → Module 1 Capstone (nexusagent-m01-python-cli-foundations) → Next Topic 2.1: Lists, Tuples, Sets & Dictionaries: Hashing Internals, Slicing, Comprehensions & Memory Complexity

Module 1 Hands-On Capstone Project (100 Marks): nexusagent-m01-python-cli-foundations — Architect, implement, test, and push the production GitHub repository 'nexusagent-m01-python-cli-foundations' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 1:...

MODULE 2 OF 22 • STAGE 1 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Python Data Structures, Functions, File / JSON I/O & Object-Oriented Programming

Beginner-to-expert real-repository module covering Python Data Structures, Functions, File / JSON I/O & Object-Oriented Programming with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Lists, Tuples, Sets & Dictionaries: Hashing Internals, Slicing, Comprehensions & Memory Complexity — Purpose: Manipulate nested JSON records, token vocabularies, and metadata maps with O(1) dictionary efficiency. • ✓ Modular Functions, *args / **kwargs, Closures, Decorators, Modules &CSV / JSON / JSONL Streaming I/O — Purpose: Read and transform million-line `.jsonl` AI training and evaluation datasets without RAM crashes. • ✓ Complete Object-Oriented Python: Classes, __init__, Properties, Inheritance, Dunder Methods & Dataclasses — Purpose: Design clean model wrappers, document loaders, and vector store clients using modular OOP blueprints. • ✓ Exception Handling, Context Managers (with), SOLID Design Principles & Structured Diagnostic Logging — Purpose: Ensure long-running AI ingestion and agent pipelines handle malformed inputs gracefully without crashing.

2.1 Lists, Tuples, Sets & Dictionaries: Hashing Internals, Slicing, Comprehensions & Memory Complexity [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Lists • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply Lists, Tuples, Sets & Dictionaries: Hashing Internals, Slicing, Comprehensions & Memory Complexity in real Generative AI, Large Language Models (L...

What Topic Is Next → Next Topic 2.2: Modular Functions, *args / **kwargs, Closures, Decorators, Modules &CSV / JSON / JSONL Streaming I/O

2.2 Modular Functions, *args / **kwargs, Closures, Decorators, Modules &CSV / JSON / JSONL Streaming I/O [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Modular Functions • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply Modular Functions, *args / **kwargs, Closures, Decorators, Modules &CSV / JSON / JSONL Streaming I/O in real Generative AI, Large Language Models...

What Topic Is Next → Next Topic 2.3: Complete Object-Oriented Python: Classes, __init__, Properties, Inheritance, Dunder Methods & Dataclasses

2.3 Complete Object-Oriented Python: Classes, __init__, Properties, Inheritance, Dunder Methods & Dataclasses [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Complete Object • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Complete Object-Oriented Python • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Complete Object-Oriented Python: Classes, __init__, Properties, Inheritance, Dunder Methods & Dataclasses in real Generative AI, Large Language Mo...

What Topic Is Next → Next Topic 2.4: Exception Handling, Context Managers (with), SOLID Design Principles & Structured Diagnostic Logging

2.4 Exception Handling, Context Managers (with), SOLID Design Principles & Structured Diagnostic Logging [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Exception Handlin... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Exception Handling • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Exception Handling, Context Managers (with), SOLID Design Principles & Structured Diagnostic Logging in real Generative AI, Large Language Models...

What Topic Is Next → Module 2 Capstone (nexusagent-m02-python-oop-datastructures) → Next Topic 3.1: Iterators, Lazy Generators (yield), itertools & Memory-Bounded Streaming Data Pipelines

Module 2 Hands-On Capstone Project (100 Marks): nexusagent-m02-python-oop-datastructures — Architect, implement, test, and push the production GitHub repository 'nexusagent-m02-python-oop-datastructures' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module...

MODULE 3 OF 22 • STAGE 1 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Advanced Python, Static Typing, Pydantic v2, Asyncio Concurrency & Pytest TDD

Beginner-to-expert real-repository module covering Advanced Python, Static Typing, Pydantic v2, Asyncio Concurrency & Pytest TDD with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Iterators, Lazy Generators (yield), itertools & Memory-Bounded Streaming Data Pipelines — Purpose: Stream large document corpora and live LLM token chunks in constant memory. • ✓ Static Type Annotations (typing / Generics / Protocols) & Rust-Powered Pydantic v2 Validation Schemas — Purpose: Master Pydantic v2 — the backbone of FastAPI, LangChain, LangGraph, MCP, and structured LLM outputs. • ✓ Python Concurrency Deep Dive: Threads vs Multiprocessing vs Asyncio Event Loop, Tasks & Semaphores — Purpose: Fan out 100+ parallel embedding and LLM API calls safely with bounded rate-limit semaphores and timeouts. • ✓ Test-Driven Engineering with Pytest: Fixtures, Parametrization, Asyncio Tests & Deterministic API Mocks — Purpose: Test complex AI and backend pipelines deterministically in CI without burning live LLM API credits.

3.1 Iterators, Lazy Generators (yield), itertools & Memory-Bounded Streaming Data Pipelines [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Iterators • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed

Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply Iterators, Lazy Generators (yield), itertools & Memory-Bounded Streaming Data Pipelines in real Generative AI, Large Language Models (LLM) & Agent...

What Topic Is Next → Next Topic 3.2: Static Type Annotations (typing / Generics / Protocols) & Rust-Powered Pydantic v2 Validation Schemas

3.2 Static Type Annotations (typing / Generics / Protocols) & Rust-Powered Pydantic v2 Validation Schemas [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Static Type Annot... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Static Type Annotations (typing / Generics / Protoco... • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Static Type Annotations (typing / Generics / Protocols) & Rust-Powered Pydantic v2 Validation Schemas in real Generative AI, Large Language Models...

What Topic Is Next → Next Topic 3.3: Python Concurrency Deep Dive: Threads vs Multiprocessing vs Asyncio Event Loop, Tasks & Semaphores

3.3 Python Concurrency Deep Dive: Threads vs Multiprocessing vs Asyncio Event Loop, Tasks & Semaphores [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Python Concurrenc... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Python Concurrency Deep Dive • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Python Concurrency Deep Dive: Threads vs Multiprocessing vs Asyncio Event Loop, Tasks & Semaphores in real Generative AI, Large Language Models (L...

What Topic Is Next → Next Topic 3.4: Test-Driven Engineering with Pytest: Fixtures, Parametrization, Asyncio Tests & Deterministic API Mocks

3.4 Test-Driven Engineering with Pytest: Fixtures, Parametrization, Asyncio Tests & Deterministic API Mocks [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Test • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Test-Driven Engineering with Pytest • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Test-Driven Engineering with Pytest: Fixtures, Parametrization, Asyncio Tests & Deterministic API Mocks in real Generative AI, Large Language Mode...

What Topic Is Next → Module 3 Capstone (nexusagent-m03-pydantic-asyncio-pytest) → Next Topic 4.1: Big-O Time & Space Complexity, Two Pointers, Sliding Window, Prefix Sums & Binary Search

Module 3 Hands-On Capstone Project (100 Marks): nexusagent-m03-pydantic-asyncio-pytest — Architect, implement, test, and push the production GitHub repository 'nexusagent-m03-pydantic-asyncio-pytest' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 3:...

MODULE 4 OF 22 • STAGE 1 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Data Structures, Algorithms (DSA) & Coding Round Mastery in Python

Beginner-to-expert real-repository module covering Data Structures, Algorithms (DSA) & Coding Round Mastery in Python with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Big-O Time & Space Complexity, Two Pointers, Sliding Window, Prefix Sums & Binary Search — Purpose: Clear Round-1 algorithmic coding screens at AI startups, unicorns, and global capability centers. • ✓ Linked Lists, Stacks, Deques, Hash Maps, Frequency Counters & Thread-Safe LRU / TTL Cache Design — Purpose: Implement custom token-window buffers and low-latency in-memory caches from scratch. • ✓ Trees, Binary Search Trees, Heaps (Top-K Beam/Vector Retrieval), Prefix Tries & Backtracking — Purpose: Understand the exact heap and trie data structures powering tokenizers, autocomplete, and top-K search. • ✓ Graph Algorithms (BFS, DFS, Topological Sort, Shortest Paths) & Dynamic Programming in Python — Purpose: Master DAG execution ordering (used directly in LangGraph!) and dynamic programming interview problems.

4.1 Big-O Time & Space Complexity, Two Pointers, Sliding Window, Prefix Sums & Binary Search [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Big • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Big-O Time • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Big-O Time & Space Complexity, Two Pointers, Sliding Window, Prefix Sums & Binary Search in real Generative AI, Large Language Models (LLM) & Agen...

What Topic Is Next → Next Topic 4.2: Linked Lists, Stacks, Deques, Hash Maps, Frequency Counters & Thread-Safe LRU / TTL Cache Design

4.2 Linked Lists, Stacks, Deques, Hash Maps, Frequency Counters & Thread-Safe LRU / TTL Cache Design [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Linked Lists • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply Linked Lists, Stacks, Deques, Hash Maps, Frequency Counters & Thread-Safe LRU / TTL Cache Design in real Generative AI, Large Language Models (LLM...

What Topic Is Next → Next Topic 4.3: Trees, Binary Search Trees, Heaps (Top-K Beam/Vector Retrieval), Prefix Tries & Backtracking

4.3 Trees, Binary Search Trees, Heaps (Top-K Beam/Vector Retrieval), Prefix Tries & Backtracking [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Trees • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply Trees, Binary Search Trees, Heaps (Top-K Beam/Vector Retrieval), Prefix Tries & Backtracking in real Generative AI, Large Language Models (LLM) &...

What Topic Is Next → Next Topic 4.4: Graph Algorithms (BFS, DFS, Topological Sort, Shortest Paths) & Dynamic Programming in Python

4.4 Graph Algorithms (BFS, DFS, Topological Sort, Shortest Paths) & Dynamic Programming in Python [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Graph Algorithms... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Graph Algorithms (BFS, DFS, Topological Sort, Shorte... • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Graph Algorithms (BFS, DFS, Topological Sort, Shortest Paths) & Dynamic Programming in Python in real Generative AI, Large Language Models (LLM) &...

What Topic Is Next → Module 4 Capstone (nexusagent-m04-python-dsa-interviews) → Next Topic 5.1: Linear Algebra, Vectors, Dot Products, Cosine Similarity, Probability & Gradient Descent Visualized

Module 4 Hands-On Capstone Project (100 Marks): nexusagent-m04-python-dsa-interviews — Architect, implement, test, and push the production GitHub repository 'nexusagent-m04-python-dsa-interviews' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 4: D...

MODULE 5 OF 22 • STAGE 1 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Applied Math Intuition, NumPy Vectorization, Pandas / Polars & Classical ML

Beginner-to-expert real-repository module covering Applied Math Intuition, NumPy Vectorization, Pandas / Polars & Classical ML with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Linear Algebra, Vectors, Dot Products, Cosine Similarity, Probability & Gradient Descent Visualized — Purpose: Gain intuitive visual mastery of vectors, matrices, embeddings, and loss optimization without dry proofs. • ✓ High-Speed Numerical Computing with NumPy: Broadcasting, Matrix Multiplication & Vectorized Top-K Search — Purpose: Build a 50,000-vector cosine-similarity search engine in pure vectorized NumPy with zero Python loops. • ✓ Data Wrangling & Exploratory Analytics with Pandas & Polars: Joins, GroupBy, Time-Series & Cleaning — Purpose: Clean messy enterprise CSV, Parquet, and SQL datasets into leak-free analytical and AI datasets. • ✓ Classical Machine Learning with Scikit-Learn: Feature Pipelines, Regression, Trees/XGBoost, Clustering & Metrics — Purpose: Know when a 2ms XGBoost/classifier beats a 2-second LLM call — and evaluate precision, recall, ROC-AUC, and F1.

5.1 Linear Algebra, Vectors, Dot Products, Cosine Similarity, Probability & Gradient Descent Visualized [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Linear Algebra • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply Linear Algebra, Vectors, Dot Products, Cosine Similarity, Probability & Gradient Descent Visualized in real Generative AI, Large Language Models (...

What Topic Is Next → Next Topic 5.2: High-Speed Numerical Computing with NumPy: Broadcasting, Matrix Multiplication & Vectorized Top-K Search

5.2 High-Speed Numerical Computing with NumPy: Broadcasting, Matrix Multiplication & Vectorized Top-K Search [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for High • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for High-Speed Numerical Computing with NumPy • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply High-Speed Numerical Computing with NumPy: Broadcasting, Matrix Multiplication & Vectorized Top-K Search in real Generative AI, Large Language Mod...

What Topic Is Next → Next Topic 5.3: Data Wrangling & Exploratory Analytics with Pandas & Polars: Joins, GroupBy, Time-Series & Cleaning

5.3 Data Wrangling & Exploratory Analytics with Pandas & Polars: Joins, GroupBy, Time-Series & Cleaning [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Data Wrangling • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply Data Wrangling & Exploratory Analytics with Pandas & Polars: Joins, GroupBy, Time-Series & Cleaning in real Generative AI, Large Language Models (...

What Topic Is Next → Next Topic 5.4: Classical Machine Learning with Scikit-Learn: Feature Pipelines, Regression, Trees/XGBoost, Clustering & Metrics

5.4 Classical Machine Learning with Scikit-Learn: Feature Pipelines, Regression, Trees/XGBoost, Clustering & Metrics [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Classical Machine... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Classical Machine Learning with Scikit-Learn • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Classical Machine Learning with Scikit-Learn: Feature Pipelines, Regression, Trees/XGBoost, Clustering & Metrics in real Generative AI, Large Lang...

What Topic Is Next → Module 5 Capstone (nexusagent-m05-numpy-pandas-ml-core) → Next Topic 6.1: Relational Modeling From Scratch, Normalization & Core SQL (DDL, DML, Constraints, INNER/LEFT JOINs)

Module 5 Hands-On Capstone Project (100 Marks): nexusagent-m05-numpy-pandas-ml-core — Architect, implement, test, and push the production GitHub repository 'nexusagent-m05-numpy-pandas-ml-core' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 5: Ap...

MODULE 6 OF 22 • STAGE 2 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Relational Databases & SQL From Scratch to PostgreSQL 16 & pgvector Search

Beginner-to-expert real-repository module covering Relational Databases & SQL From Scratch to PostgreSQL 16 & pgvector Search with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Relational Modeling From Scratch, Normalization & Core SQL (DDL, DML, Constraints, INNER/LEFT JOINs) — Purpose: Master SQL fundamentals from zero so you can query any corporate data warehouse or transactional DB. • ✓ Advanced SQL: Aggregations, CTEs, Window Functions, JSONB Queries & ACID Transaction Isolation — Purpose: Combine structured SQL analytics with semi-structured JSONB agent metadata inside one database. • ✓ PostgreSQL 16 Internals, B-Tree / GIN Indexing, EXPLAIN ANALYZE Tuning & Async Connection Pooling — Purpose: Optimize slow database queries and prevent async Python microservices from exhausting DB connections. • ✓ PostgreSQL pgvector Mastery: Vector Columns, Cosine/L2/Inner-Product Operators & HNSW / IVFFlat Indexes — Purpose: Turn PostgreSQL 16 into an ACID-compliant enterprise vector database unifying SQL RBAC + AI search.

6.1 Relational Modeling From Scratch, Normalization & Core SQL (DDL, DML, Constraints, INNER/LEFT JOINs) [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Relational Modeli... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Relational Modeling From Scratch • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Relational Modeling From Scratch, Normalization & Core SQL (DDL, DML, Constraints, INNER/LEFT JOINs) in real Generative AI, Large Language Models...

What Topic Is Next → Next Topic 6.2: Advanced SQL: Aggregations, CTEs, Window Functions, JSONB Queries & ACID Transaction Isolation

6.2 Advanced SQL: Aggregations, CTEs, Window Functions, JSONB Queries & ACID Transaction Isolation [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Advanced SQL • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply Advanced SQL: Aggregations, CTEs, Window Functions, JSONB Queries & ACID Transaction Isolation in real Generative AI, Large Language Models (LLM)...

What Topic Is Next → Next Topic 6.3: PostgreSQL 16 Internals, B-Tree / GIN Indexing, EXPLAIN ANALYZE Tuning & Async Connection Pooling

6.3 PostgreSQL 16 Internals, B-Tree / GIN Indexing, EXPLAIN ANALYZE Tuning & Async Connection Pooling [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for PostgreSQL 16 Int... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for PostgreSQL 16 Internals • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply PostgreSQL 16 Internals, B-Tree / GIN Indexing, EXPLAIN ANALYZE Tuning & Async Connection Pooling in real Generative AI, Large Language Models (LL...

What Topic Is Next → Next Topic 6.4: PostgreSQL pgvector Mastery: Vector Columns, Cosine/L2/Inner-Product Operators & HNSW / IVFFlat Indexes

6.4 PostgreSQL pgvector Mastery: Vector Columns, Cosine/L2/Inner-Product Operators & HNSW / IVFFlat Indexes [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for PostgreSQL pgvect... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for PostgreSQL pgvector Mastery • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply PostgreSQL pgvector Mastery: Vector Columns, Cosine/L2/Inner-Product Operators & HNSW / IVFFlat Indexes in real Generative AI, Large Language Mode...

What Topic Is Next → Module 6 Capstone (nexusagent-m06-sql-postgresql-pgvector) → Next Topic 7.1: Web Protocols (HTTP/HTTPS, REST, SSE, WebSockets, CORS), Semantic HTML5 & Accessible UI Landmarks

Module 6 Hands-On Capstone Project (100 Marks): nexusagent-m06-sql-postgresql-pgvector — Architect, implement, test, and push the production GitHub repository 'nexusagent-m06-sql-postgresql-pgvector' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 6:...

MODULE 7 OF 22 • STAGE 2 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Web Frontend Foundations From Scratch: HTML5, CSS3, Tailwind CSS & Core JS (ES6+)

Beginner-to-expert real-repository module covering Web Frontend Foundations From Scratch: HTML5, CSS3, Tailwind CSS & Core JS (ES6+) with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Web Protocols (HTTP/HTTPS, REST, SSE, WebSockets, CORS), Semantic HTML5 & Accessible UI Landmarks — Purpose: Understand browser-to-AI-gateway network streaming protocols and semantic document structure. • ✓ Modern CSS3 Box Model, Flexbox, CSS Grid, Dark/Light Themes & Tailwind CSS Responsive Design — Purpose: Style sleek, executive-grade AI chat, document-copilot, and analytics interfaces effortlessly. • ✓ Core JavaScript From Zero: Variables, Closures, Arrays, Objects, ES Modules & Async Promises / Fetch — Purpose: Gain complete JavaScript fluency to build interactive browser logic and consume streaming AI APIs. • ✓ DOM Events, ReadableStream SSE Parsing, Markdown/Code Syntax Rendering & Responsive Split-Pane Layouts — Purpose: Render token-by-token streaming markdown, tables, and cited code blocks smoothly at 60fps.

7.1 Web Protocols (HTTP/HTTPS, REST, SSE, WebSockets, CORS), Semantic HTML5 & Accessible UI Landmarks

[90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Web Protocols (HT... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Web Protocols (HTTP/HTTPS, REST, SSE, WebSockets, CO... • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Web Protocols (HTTP/HTTPS, REST, SSE, WebSockets, CORS), Semantic HTML5 & Accessible UI Landmarks in real Generative AI, Large Language Models (LL...

What Topic Is Next → Next Topic 7.2: Modern CSS3 Box Model, Flexbox, CSS Grid, Dark/Light Themes & Tailwind CSS Responsive Design

7.2 Modern CSS3 Box Model, Flexbox, CSS Grid, Dark/Light Themes & Tailwind CSS Responsive Design

[90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Modern CSS3 Box M... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Modern CSS3 Box Model • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Modern CSS3 Box Model, Flexbox, CSS Grid, Dark/Light Themes & Tailwind CSS Responsive Design in real Generative AI, Large Language Models (LLM) &...

What Topic Is Next → Next Topic 7.3: Core JavaScript From Zero: Variables, Closures, Arrays, Objects, ES Modules & Async Promises / Fetch

7.3 Core JavaScript From Zero: Variables, Closures, Arrays, Objects, ES Modules & Async Promises / Fetch

[90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Core JavaScript F... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Core JavaScript From Zero • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Core JavaScript From Zero: Variables, Closures, Arrays, Objects, ES Modules & Async Promises / Fetch in real Generative AI, Large Language Models...

What Topic Is Next → Next Topic 7.4: DOM Events, ReadableStream SSE Parsing, Markdown/Code Syntax Rendering & Responsive Split-Pane Layouts

7.4 DOM Events, ReadableStream SSE Parsing, Markdown/Code Syntax Rendering & Responsive Split-Pane Layouts

[90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for DOM Events • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply DOM Events, ReadableStream SSE Parsing, Markdown/Code Syntax Rendering & Responsive Split-Pane Layouts in real Generative AI, Large Language Model...

What Topic Is Next → Module 7 Capstone (nexusagent-m07-web-foundations-js) → Next Topic 8.1: TypeScript 5 Fundamentals, Generics, Discriminated Unions (Message/Tool States) & Zod Validation

Module 7 Hands-On Capstone Project (100 Marks): nexusagent-m07-web-foundations-js — Architect, implement, test, and push the production GitHub repository 'nexusagent-m07-web-foundations-js' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 7: Web...

MODULE 8 OF 22 • STAGE 2 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Modern Generative UI Engineering with Strict TypeScript 5 & React 19

Beginner-to-expert real-repository module covering Modern Generative UI Engineering with Strict TypeScript 5 & React 19 with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ TypeScript 5 Fundamentals, Generics, Discriminated Unions (Message/Tool States) & Zod Validation — Purpose: Model multi-turn chat, tool-invocation cards, and citation states with 100% compile-time type safety. • ✓ React 19 Core: Components, Hooks (useState, useEffect, useRef, useOptimistic), Forms & Custom Hooks — Purpose: Build modular React 19 workspaces with clean abortable streaming hooks and zero render flicker. • ✓ Protected RBAC

Routing, Zustand Workspace State & TanStack Query Server-State Synchronization — Purpose: Manage conversation histories, knowledge-base uploads, and multi-tenant admin dashboards cleanly. • ✓ Generative UI Components: Interactive Tool Approval Cards, PDF Citation Highlighters & Vitest Suites — Purpose: Move beyond plain chat boxes by rendering interactive charts, diffs, and HITL approval widgets in React.

8.1 TypeScript 5 Fundamentals, Generics, Discriminated Unions (Message/Tool States) & Zod Validation [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for TypeScript 5 Fund... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for TypeScript 5 Fundamentals • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply TypeScript 5 Fundamentals, Generics, Discriminated Unions (Message/Tool States) & Zod Validation in real Generative AI, Large Language Models (LLM...

What Topic Is Next → Next Topic 8.2: React 19 Core: Components, Hooks (useState, useEffect, useRef, useOptimistic), Forms & Custom Hooks

8.2 React 19 Core: Components, Hooks (useState, useEffect, useRef, useOptimistic), Forms & Custom Hooks [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for React 19 Core • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply React 19 Core: Components, Hooks (useState, useEffect, useRef, useOptimistic), Forms & Custom Hooks in real Generative AI, Large Language Models (...

What Topic Is Next → Next Topic 8.3: Protected RBAC Routing, Zustand Workspace State & TanStack Query Server-State Synchronization

8.3 Protected RBAC Routing, Zustand Workspace State & TanStack Query Server-State Synchronization [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Protected RBAC Ro... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Protected RBAC Routing • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Protected RBAC Routing, Zustand Workspace State & TanStack Query Server-State Synchronization in real Generative AI, Large Language Models (LLM) &...

What Topic Is Next → Next Topic 8.4: Generative UI Components: Interactive Tool Approval Cards, PDF Citation Highlighters & Vitest Suites

8.4 Generative UI Components: Interactive Tool Approval Cards, PDF Citation Highlighters & Vitest Suites [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Generative UI Com... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Generative UI Components • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Generative UI Components: Interactive Tool Approval Cards, PDF Citation Highlighters & Vitest Suites in real Generative AI, Large Language Models...

What Topic Is Next → Module 8 Capstone (nexusagent-m08-react19-typescript-genui) → Next Topic 9.1: Django 5 Architecture, ORM Models, Zero-Downtime Migrations, select_related / prefetch_related & Admin

Module 8 Hands-On Capstone Project (100 Marks): nexusagent-m08-react19-typescript-genui — Architect, implement, test, and push the production GitHub repository 'nexusagent-m08-react19-typescript-genui' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 8...

MODULE 9 OF 22 • STAGE 2 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Enterprise SaaS Backend Engineering with Django 5, DRF, Celery & Redis 7

Beginner-to-expert real-repository module covering Enterprise SaaS Backend Engineering with Django 5, DRF, Celery & Redis 7 with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Django 5 Architecture, ORM Models, Zero-Downtime Migrations, select_related / prefetch_related & Admin — Purpose: Build the rock-solid multi-tenant SaaS user, billing, and workspace management backbone in Django 5. • ✓ Django REST Framework (DRF), Serializers, ViewSets, Rotating JWT / OAuth2 SSO & Multi-Tenant RBAC — Purpose: Enforce organization roles, API keys, and per-seat token quotas across enterprise customers. • ✓ Redis 7 Caching, Sliding-Window Rate Limiters & Distributed Celery Background Ingestion Pipelines — Purpose: Process 500-page PDF uploads, OCR, chunking, and embedding generation asynchronously via Celery + Redis. • ✓ Transactional Webhooks (Razorpay/Stripe Usage Billing), Audit Logs & pytest-django API Verification — Purpose: Meter token/credit consumption accurately and verify every SaaS permission boundary with automated tests.

9.1 Django 5 Architecture, ORM Models, Zero-Downtime Migrations, select_related / prefetch_related & Admin [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Django 5 Architec... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Django 5 Architecture • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Django 5 Architecture, ORM Models, Zero-Downtime Migrations, select_related / prefetch_related & Admin in real Generative AI, Large Language Model...

What Topic Is Next → Next Topic 9.2: Django REST Framework (DRF), Serializers, ViewSets, Rotating JWT / OAuth2 SSO & Multi-Tenant RBAC

9.2 Django REST Framework (DRF), Serializers, ViewSets, Rotating JWT / OAuth2 SSO & Multi-Tenant RBAC [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Django REST Frame... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Django REST Framework (DRF) • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Django REST Framework (DRF), Serializers, ViewSets, Rotating JWT / OAuth2 SSO & Multi-Tenant RBAC in real Generative AI, Large Language Models (LL...

What Topic Is Next → Next Topic 9.3: Redis 7 Caching, Sliding-Window Rate Limiters & Distributed Celery Background Ingestion Pipelines

9.3 Redis 7 Caching, Sliding-Window Rate Limiters & Distributed Celery Background Ingestion Pipelines [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Redis 7 Caching • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply Redis 7 Caching, Sliding-Window Rate Limiters & Distributed Celery Background Ingestion Pipelines in real Generative AI, Large Language Models (LL...

What Topic Is Next → Next Topic 9.4: Transactional Webhooks (Razorpay/Stripe Usage Billing), Audit Logs & pytest-django API Verification

9.4 Transactional Webhooks (Razorpay/Stripe Usage Billing), Audit Logs & pytest-django API Verification [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Transactional Web... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Transactional Webhooks (Razorpay/Stripe Usage Billin... • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Transactional Webhooks (Razorpay/Stripe Usage Billing), Audit Logs & pytest-django API Verification in real Generative AI, Large Language Models (...

What Topic Is Next → Module 9 Capstone (nexusagent-m09-django5-drf-celery-saas) → Next Topic 10.1: FastAPI Internals, ASGI Event Loop, Pydantic v2 Contracts, Dependency Injection & Lifespan Warmup

Module 9 Hands-On Capstone Project (100 Marks): nexusagent-m09-django5-drf-celery-saas — Architect, implement, test, and push the production GitHub repository 'nexusagent-m09-django5-drf-celery-saas' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 9:...

MODULE 10 OF 22 • STAGE 2 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

High-Concurrency Async AI Gateways with FastAPI, SQLAlchemy 2, SSE & WebSockets

Beginner-to-expert real-repository module covering High-Concurrency Async AI Gateways with FastAPI, SQLAlchemy 2, SSE & WebSockets with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ FastAPI Internals, ASGI Event Loop, Pydantic v2 Contracts, Dependency Injection & Lifespan Warmup — Purpose: Serve ultra-low-latency async Python inference and RAG gateways capable of thousands of concurrent streams. • ✓ Async SQLAlchemy 2.0, Alembic Migrations, Non-Blocking pgvector Queries & Connection Pool Sizing — Purpose: Execute async relational + vector queries without ever blocking the Uvicorn event loop. • ✓ Server-Sent Events (SSE) Token Streaming, Bidirectional WebSockets, Client Disconnect Cancellation & Backpressure — Purpose: Stop upstream LLM token generation immediately when a user clicks "Stop Generating" to save 40%+ compute. • ✓ API Key / JWT Auth Scopes, Per-Tenant Token Rate Limiting & Async Load Testing (httpx + Locust/k6) — Purpose: Harden your AI gateway against abuse and benchmark concurrent streaming throughput scientifically.

10.1 FastAPI Internals, ASGI Event Loop, Pydantic v2 Contracts, Dependency Injection & Lifespan Warmup [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for FastAPI Internals • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply FastAPI Internals, ASGI Event Loop, Pydantic v2 Contracts, Dependency Injection & Lifespan Warmup in real Generative AI, Large Language Models (LL...

What Topic Is Next → Next Topic 10.2: Async SQLAlchemy 2.0, Alembic Migrations, Non-Blocking pgvector Queries & Connection Pool Sizing

10.2 Async SQLAlchemy 2.0, Alembic Migrations, Non-Blocking pgvector Queries & Connection Pool Sizing [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Async SQLAlchemy... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Async SQLAlchemy 2.0 • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Async SQLAlchemy 2.0, Alembic Migrations, Non-Blocking pgvector Queries & Connection Pool Sizing in real Generative AI, Large Language Models (LLM...

What Topic Is Next → Next Topic 10.3: Server-Sent Events (SSE) Token Streaming, Bidirectional WebSockets, Client Disconnect Cancellation & Backpressure

10.3 Server-Sent Events (SSE) Token Streaming, Bidirectional WebSockets, Client Disconnect Cancellation & Backpressure [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Server • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Server-Sent Events (SSE) Token Streaming • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Server-Sent Events (SSE) Token Streaming, Bidirectional WebSockets, Client Disconnect Cancellation & Backpressure in real Generative AI, Large Lan...

What Topic Is Next → Next Topic 10.4: API Key / JWT Auth Scopes, Per-Tenant Token Rate Limiting & Async Load Testing (httpx + Locust/k6)

10.4 API Key / JWT Auth Scopes, Per-Tenant Token Rate Limiting & Async Load Testing (httpx + Locust/k6) [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for API Key / JWT Aut... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for API Key / JWT Auth Scopes • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply API Key / JWT Auth Scopes, Per-Tenant Token Rate Limiting & Async Load Testing (httpx + Locust/k6) in real Generative AI, Large Language Models (L...

What Topic Is Next → Module 10 Capstone (nexusagent-m10-fastapi-streaming-gateway) → Next Topic 11.1: Neural Networks From Scratch: Perceptrons, Activations (ReLU, GELU, SiLU), Loss Functions & Backpropagation

Module 10 Hands-On Capstone Project (100 Marks): nexusagent-m10-fastapi-streaming-gateway — Architect, implement, test, and push the production GitHub repository 'nexusagent-m10-fastapi-streaming-gateway' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module...

MODULE 11 OF 22 • STAGE 2 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Deep Learning From Perceptrons to PyTorch & Transformer Attention Internals

Beginner-to-expert real-repository module covering Deep Learning From Perceptrons to PyTorch & Transformer Attention Internals with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Neural Networks From Scratch: Perceptrons, Activations (ReLU, GELU, SiLU), Loss Functions & Backpropagation — Purpose: Understand forward passes, gradients, overfitting, and regularization from first principles. • ✓ PyTorch Mastery: Tensors, GPU/CUDA Acceleration, Autograd, nn.Module, DataLoaders & Mixed-Precision Training — Purpose: Write clean PyTorch training and evaluation loops on CPU and GPU hardware. • ✓ Tokenization (BPE / Tiktoken), Learned Embeddings, Positional Encoding (RoPE) & Self-Attention Math — Purpose: Implement Multi-Head Causal Self-Attention in PyTorch so LLMs are never a mysterious black box to you. • ✓ Decoder-Only vs Encoder Architectures, KV-Cache Memory Math, Context Windows, FlashAttention & MoE Models — Purpose: Calculate exact GPU VRAM requirements and understand why KV-cache governs long-context LLM latency.

11.1 Neural Networks From Scratch: Perceptrons, Activations (ReLU, GELU, SiLU), Loss Functions & Backpropagation [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Neural Networks F... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Neural Networks From Scratch • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Neural Networks From Scratch: Perceptrons, Activations (ReLU, GELU, SiLU), Loss Functions & Backpropagation in real Generative AI, Large Language...

What Topic Is Next → Next Topic 11.2: PyTorch Mastery: Tensors, GPU/CUDA Acceleration, Autograd, nn.Module, DataLoaders & Mixed-Precision Training

11.2 PyTorch Mastery: Tensors, GPU/CUDA Acceleration, Autograd, nn.Module, DataLoaders & Mixed-Precision Training [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for PyTorch Mastery • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply PyTorch Mastery: Tensors, GPU/CUDA Acceleration, Autograd, nn.Module, DataLoaders & Mixed-Precision Training in real Generative AI, Large Language...

What Topic Is Next → Next Topic 11.3: Tokenization (BPE / Tiktoken), Learned Embeddings, Positional Encoding (RoPE) & Self-Attention Math

11.3 Tokenization (BPE / Tiktoken), Learned Embeddings, Positional Encoding (RoPE) & Self-Attention Math [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Tokenization (BPE... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Tokenization (BPE / Tiktoken) • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Tokenization (BPE / Tiktoken), Learned Embeddings, Positional Encoding (RoPE) & Self-Attention Math in real Generative AI, Large Language Models (...

What Topic Is Next → Next Topic 11.4: Decoder-Only vs Encoder Architectures, KV-Cache Memory Math, Context Windows, FlashAttention & MoE Models

11.4 Decoder-Only vs Encoder Architectures, KV-Cache Memory Math, Context Windows, FlashAttention & MoE Models [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Decoder • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Decoder-Only vs Encoder Architectures • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Decoder-Only vs Encoder Architectures, KV-Cache Memory Math, Context Windows, FlashAttention & MoE Models in real Generative AI, Large Language Mo...

What Topic Is Next → Module 11 Capstone (nexusagent-m11-pytorch-transformer-internals) → Next Topic 12.1: Decoding Strategies (Greedy, Temperature, Top-K, Top-P), Logprobs, System Prompts & Context Caching

Module 11 Hands-On Capstone Project (100 Marks): nexusagent-m11-pytorch-transformer-internals — Architect, implement, test, and push the production GitHub repository 'nexusagent-m11-pytorch-transformer-internals' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Mod...

MODULE 12 OF 22 • STAGE 2 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

LLM Mechanics, Prompt Engineering, Guaranteed Structured Outputs & Multimodal AI

Beginner-to-expert real-repository module covering LLM Mechanics, Prompt Engineering, Guaranteed Structured Outputs & Multimodal AI with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Decoding Strategies (Greedy, Temperature, Top-K, Top-P), Logprobs, System Prompts & Context Caching — Purpose: Control determinism vs creativity precisely and slash repeated system-prompt costs by up to 75%. • ✓ Production Prompt Engineering: Few-Shot Exemplars, Chain-of-Thought, Decomposition & XML/Delimiter Guards — Purpose: Author versioned, testable enterprise prompt templates resilient against ambiguous user inputs. • ✓ Constrained Decoding & Guaranteed Structured Outputs (JSON Schema / Pydantic / Instructor) for Extraction — Purpose: Extract 100% schema-valid invoices, resumes, medical entities, and SQL filters from unstructured text. • ✓ Multimodal Vision, Document Layout OCR & Speech-to-Text / Text-to-Speech Pipelines in Production — Purpose: Build unified pipelines that reason over charts, scanned PDFs, screenshots, and spoken audio.

12.1 Decoding Strategies (Greedy, Temperature, Top-K, Top-P), Logprobs, System Prompts & Context Caching [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Decoding Strategi... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Decoding Strategies (Greedy, Temperature, Top-K, Top... • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Decoding Strategies (Greedy, Temperature, Top-K, Top-P), Logprobs, System Prompts & Context Caching in real Generative AI, Large Language Models (...

What Topic Is Next → Next Topic 12.2: Production Prompt Engineering: Few-Shot Exemplars, Chain-of-Thought, Decomposition & XML/Delimiter Guards

12.2 Production Prompt Engineering: Few-Shot Exemplars, Chain-of-Thought, Decomposition & XML/Delimiter Guards [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Production Prompt... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Production Prompt Engineering • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Production Prompt Engineering: Few-Shot Exemplars, Chain-of-Thought, Decomposition & XML/Delimiter Guards in real Generative AI, Large Language Mo...

What Topic Is Next → Next Topic 12.3: Constrained Decoding & Guaranteed Structured Outputs (JSON Schema / Pydantic / Instructor) for Extraction

12.3 Constrained Decoding & Guaranteed Structured Outputs (JSON Schema / Pydantic / Instructor) for Extraction [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Constrained Decod... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Constrained Decoding • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Constrained Decoding & Guaranteed Structured Outputs (JSON Schema / Pydantic / Instructor) for Extraction in real Generative AI, Large Language Mo...

What Topic Is Next → Next Topic 12.4: Multimodal Vision, Document Layout OCR & Speech-to-Text / Text-to-Speech Pipelines in Production

12.4 Multimodal Vision, Document Layout OCR & Speech-to-Text / Text-to-Speech Pipelines in Production [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Multimodal Vision • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply Multimodal Vision, Document Layout OCR & Speech-to-Text / Text-to-Speech Pipelines in Production in real Generative AI, Large Language Models (LLM...

What Topic Is Next → Module 12 Capstone (nexusagent-m12-llm-structured-multimodal) → Next Topic 13.1: Dense vs Sparse Embedding Models, Dimensionality / Matryoshka Trade-Offs & Domain Embedding Selection

Module 12 Hands-On Capstone Project (100 Marks): nexusagent-m12-llm-structured-multimodal — Architect, implement, test, and push the production GitHub repository 'nexusagent-m12-llm-structured-multimodal' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module...

MODULE 13 OF 22 • STAGE 3 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Embeddings, Semantic Chunking, Vector Databases (pgvector & Qdrant) & Core RAG

Beginner-to-expert real-repository module covering Embeddings, Semantic Chunking, Vector Databases (pgvector & Qdrant) & Core RAG with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Dense vs Sparse Embedding Models, Dimensionality / Matryoshka Trade-Offs & Domain Embedding Selection — Purpose: Benchmark open-weight vs commercial embedding models on recall, latency, and storage footprint. • ✓ Enterprise Ingestion: PDF/HTML/Code Parsing, Fixed vs Recursive vs Semantic vs Parent-Child Chunking — Purpose: Eliminate broken mid-sentence/mid-table chunks — the #1 silent cause of poor RAG answer quality. • ✓ Vector Databases in Depth: PostgreSQL pgvector & Qdrant Collections, HNSW Tuning, Payload Indexes & Quantization — Purpose: Tune HNSW ef_search/m parameters and scalar/binary quantization for sub-15ms million-vector queries. • ✓ End-to-End Grounded RAG Pipeline: Metadata RBAC Pre-Filtering, Inline Source Citations & "I Don’t Know" Refusals — Purpose: Guarantee every generated claim cites an authorized source chunk or cleanly abstains.

13.1 Dense vs Sparse Embedding Models, Dimensionality / Matryoshka Trade-Offs & Domain Embedding Selection [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Dense vs Sparse E... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Dense vs Sparse Embedding Models • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Dense vs Sparse Embedding Models, Dimensionality / Matryoshka Trade-Offs & Domain Embedding Selection in real Generative AI, Large Language Models...

What Topic Is Next → Next Topic 13.2: Enterprise Ingestion: PDF/HTML/Code Parsing, Fixed vs Recursive vs Semantic vs Parent-Child Chunking

13.2 Enterprise Ingestion: PDF/HTML/Code Parsing, Fixed vs Recursive vs Semantic vs Parent-Child Chunking [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Enterprise Ingest... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Enterprise Ingestion • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Enterprise Ingestion: PDF/HTML/Code Parsing, Fixed vs Recursive vs Semantic vs Parent-Child Chunking in real Generative AI, Large Language Models...

What Topic Is Next → Next Topic 13.3: Vector Databases in Depth: PostgreSQL pgvector & Qdrant Collections, HNSW Tuning, Payload Indexes & Quantization

13.3 Vector Databases in Depth: PostgreSQL pgvector & Qdrant Collections, HNSW Tuning, Payload Indexes & Quantization [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Vector Databases... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Vector Databases in Depth • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Vector Databases in Depth: PostgreSQL pgvector & Qdrant Collections, HNSW Tuning, Payload Indexes & Quantization in real Generative AI, Large Lang...

What Topic Is Next → Next Topic 13.4: End-to-End Grounded RAG Pipeline: Metadata RBAC Pre-Filtering, Inline Source Citations & "I Don’t Know" Refusals

13.4 End-to-End Grounded RAG Pipeline: Metadata RBAC Pre-Filtering, Inline Source Citations & "I Don’t Know" Refusals [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for End • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for End-to-End Grounded RAG Pipeline • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply End-to-End Grounded RAG Pipeline: Metadata RBAC Pre-Filtering, Inline Source Citations & "I Don’t Know" Refusals in real Generative AI, Large Lang...

What Topic Is Next → Module 13 Capstone (nexusagent-m13-embeddings-vectordb-rag) → Next Topic 14.1: Hybrid Retrieval Architecture: BM25 / SPLADE Lexical Search + Dense Vectors with Reciprocal Rank Fusion (RRF)

Module 13 Hands-On Capstone Project (100 Marks): nexusagent-m13-embeddings-vectordb-rag — Architect, implement, test, and push the production GitHub repository 'nexusagent-m13-embeddings-vectordb-rag' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 13...

MODULE 14 OF 22 • STAGE 3 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Advanced Enterprise RAG: Hybrid Search, Reranking, GraphRAG & RAGAS CI Evals

Beginner-to-expert real-repository module covering Advanced Enterprise RAG: Hybrid Search, Reranking, GraphRAG & RAGAS CI Evals with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Hybrid Retrieval Architecture: BM25 / SPLADE Lexical Search + Dense Vectors with Reciprocal Rank Fusion (RRF) — Purpose: Combine exact part-number/code matching with semantic meaning for superior enterprise accuracy. • ✓ Query Transformation (Multi-Query, Step-Back, HyDE), Text-to-SQL Routing & Cross-Encoder Rerankers — Purpose: Re-score top-50 candidates down to the golden top-5 chunks using Cohere / BGE cross-encoders. • ✓ Corrective RAG (CRAG), Self-RAG & Knowledge-Graph Enhanced Retrieval (GraphRAG) for Multi-Hop Questions — Purpose: Answer complex cross-document synthesis questions that single-chunk vector search misses. • ✓ Golden Evaluation Datasets, LLM-as-a-Judge & Automated RAGAS CI Gates (Faithfulness, Recall, Precision) — Purpose: Fail GitHub pull requests automatically if a prompt or chunker change drops RAG faithfulness below 92%.

14.1 Hybrid Retrieval Architecture: BM25 / SPLADE Lexical Search + Dense Vectors with Reciprocal Rank Fusion (RRF) [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Hybrid Retrieval... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Hybrid Retrieval Architecture • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Hybrid Retrieval Architecture: BM25 / SPLADE Lexical Search + Dense Vectors with Reciprocal Rank Fusion (RRF) in real Generative AI, Large Languag...

What Topic Is Next → Next Topic 14.2: Query Transformation (Multi-Query, Step-Back, HyDE), Text-to-SQL Routing & Cross-Encoder Rerankers

14.2 Query Transformation (Multi-Query, Step-Back, HyDE), Text-to-SQL Routing & Cross-Encoder Rerankers [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Query Transformat... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Query Transformation (Multi-Query, Step-Back, HyDE) • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Query Transformation (Multi-Query, Step-Back, HyDE), Text-to-SQL Routing & Cross-Encoder Rerankers in real Generative AI, Large Language Models (L...

What Topic Is Next → Next Topic 14.3: Corrective RAG (CRAG), Self-RAG & Knowledge-Graph Enhanced Retrieval (GraphRAG) for Multi-Hop Questions

14.3 Corrective RAG (CRAG), Self-RAG & Knowledge-Graph Enhanced Retrieval (GraphRAG) for Multi-Hop Questions [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Corrective RAG (C... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Corrective RAG (CRAG) • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Corrective RAG (CRAG), Self-RAG & Knowledge-Graph Enhanced Retrieval (GraphRAG) for Multi-Hop Questions in real Generative AI, Large Language Mode...

What Topic Is Next → Next Topic 14.4: Golden Evaluation Datasets, LLM-as-a-Judge & Automated RAGAS CI Gates (Faithfulness, Recall, Precision)

14.4 Golden Evaluation Datasets, LLM-as-a-Judge & Automated RAGAS CI Gates (Faithfulness, Recall, Precision) [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Golden Evaluation... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Golden Evaluation Datasets • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Golden Evaluation Datasets, LLM-as-a-Judge & Automated RAGAS CI Gates (Faithfulness, Recall, Precision) in real Generative AI, Large Language Mode...

What Topic Is Next → Module 14 Capstone (nexusagent-m14-hybrid-graphrag-ragas) → Next Topic 15.1: Agentic Design Patterns From Scratch: ReAct, Plan-and-Execute, Reflection, Routing & Parallel Fan-Out

Module 14 Hands-On Capstone Project (100 Marks): nexusagent-m14-hybrid-graphrag-ragas — Architect, implement, test, and push the production GitHub repository 'nexusagent-m14-hybrid-graphrag-ragas' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 14:...

MODULE 15 OF 22 • STAGE 3 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Agentic AI Engineering: Tool Calling, LangChain, LangGraph StateGraphs & CrewAI

Beginner-to-expert real-repository module covering Agentic AI Engineering: Tool Calling, LangChain, LangGraph StateGraphs & CrewAI with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Agentic Design Patterns From Scratch: ReAct, Plan-and-Execute, Reflection, Routing & Parallel Fan-Out — Purpose: Know when a deterministic chain suffices versus when an autonomous tool-using agent loop is warranted. • ✓ Safe Tool / Function Calling: Strict Pydantic Tool Schemas, Sandboxed Code Execution & Error Self-Correction — Purpose: Let agents query SQL/APIs, recover from tool errors, and enforce max-iteration and dollar-budget caps. • ✓ LangGraph Mastery: Typed StateGraphs, Conditional Edges, Subgraphs, Streaming Events & Postgres Checkpointers — Purpose: Build production-grade cyclic, resumable, time-travel-debuggable agent workflows in LangGraph. • ✓ Multi-Agent Orchestration (Supervisor, Hierarchical & CrewAI Role Teams) + Long-Term Episodic/Semantic Memory — Purpose: Coordinate specialized Researcher, Coder, Analyst, and Reviewer agents with persistent user memory.

15.1 Agentic Design Patterns From Scratch: ReAct, Plan-and-Execute, Reflection, Routing & Parallel Fan-Out [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Agentic Design Pa... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking

endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Agentic Design Patterns From Scratch • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Agentic Design Patterns From Scratch: ReAct, Plan-and-Execute, Reflection, Routing & Parallel Fan-Out in real Generative AI, Large Language Models...

What Topic Is Next → Next Topic 15.2: Safe Tool / Function Calling: Strict Pydantic Tool Schemas, Sandboxed Code Execution & Error Self-Correction

15.2 Safe Tool / Function Calling: Strict Pydantic Tool Schemas, Sandboxed Code Execution & Error Self-Correction [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Safe Tool / Funct... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Safe Tool / Function Calling • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Safe Tool / Function Calling: Strict Pydantic Tool Schemas, Sandboxed Code Execution & Error Self-Correction in real Generative AI, Large Language...

What Topic Is Next → Next Topic 15.3: LangGraph Mastery: Typed StateGraphs, Conditional Edges, Subgraphs, Streaming Events & Postgres Checkpointers

15.3 LangGraph Mastery: Typed StateGraphs, Conditional Edges, Subgraphs, Streaming Events & Postgres Checkpointers [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for LangGraph Mastery • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply LangGraph Mastery: Typed StateGraphs, Conditional Edges, Subgraphs, Streaming Events & Postgres Checkpointers in real Generative AI, Large Languag...

What Topic Is Next → Next Topic 15.4: Multi-Agent Orchestration (Supervisor, Hierarchical & CrewAI Role Teams) + Long-Term Episodic/Semantic Memory

15.4 Multi-Agent Orchestration (Supervisor, Hierarchical & CrewAI Role Teams) + Long-Term Episodic/Semantic Memory [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Multi • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Multi-Agent Orchestration (Supervisor, Hierarchical... • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Multi-Agent Orchestration (Supervisor, Hierarchical & CrewAI Role Teams) + Long-Term Episodic/Semantic Memory in real Generative AI, Large Languag...

What Topic Is Next → Module 15 Capstone (nexusagent-m15-langgraph-multiagent-systems) → Next Topic 16.1: Model Context Protocol (MCP) Architecture: Hosts, Clients, Servers, Tools, Resources & Stdio/SSE Transports

Module 15 Hands-On Capstone Project (100 Marks): nexusagent-m15-langgraph-multiagent-systems — Architect, implement, test, and push the production GitHub repository 'nexusagent-m15-langgraph-multiagent-systems' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Modu...

MODULE 16 OF 22 • STAGE 3 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Model Context Protocol (MCP) Servers, Human-in-the-Loop Guards & Voice AI Agents

Beginner-to-expert real-repository module covering Model Context Protocol (MCP) Servers, Human-in-the-Loop Guards & Voice AI Agents with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Model Context Protocol (MCP) Architecture: Hosts, Clients, Servers, Tools, Resources & Stdio/SSE Transports — Purpose: Build standardized MCP servers in Python/TypeScript that plug cleanly into enterprise AI copilots. • ✓ Enterprise MCP Connectors for PostgreSQL, GitHub, Slack & Internal REST APIs with OAuth2 / RBAC Scopes — Purpose: Expose company tools to AI agents safely with per-user authorization and full audit logging. • ✓ Human-in-the-Loop (HITL) Interrupts: LangGraph interrupt(), Diff Preview Cards, State Editing & Resume Flows — Purpose: Require explicit human sign-off before any agent executes refunds, emails, or production writes. • ✓ Low-Latency Spoken Voice AI Agents: VAD, Streaming STT -> Agent -> Streaming TTS & Barge-In Interruption — Purpose: Engineer natural sub-600ms conversational voice agents for customer support and technical interviews.

16.1 Model Context Protocol (MCP) Architecture: Hosts, Clients, Servers, Tools, Resources & Stdio/SSE Transports [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Model Context Pro... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Model Context Protocol (MCP) Architecture • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Model Context Protocol (MCP) Architecture: Hosts, Clients, Servers, Tools, Resources & Stdio/SSE Transports in real Generative AI, Large Language...

What Topic Is Next → Next Topic 16.2: Enterprise MCP Connectors for PostgreSQL, GitHub, Slack & Internal REST APIs with OAuth2 / RBAC Scopes

16.2 Enterprise MCP Connectors for PostgreSQL, GitHub, Slack & Internal REST APIs with OAuth2 / RBAC Scopes [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Enterprise MCP Co... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking

endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Enterprise MCP Connectors for PostgreSQL • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Enterprise MCP Connectors for PostgreSQL, GitHub, Slack & Internal REST APIs with OAuth2 / RBAC Scopes in real Generative AI, Large Language Model...

What Topic Is Next → Next Topic 16.3: Human-in-the-Loop (HITL) Interrupts: LangGraph interrupt(), Diff Preview Cards, State Editing & Resume Flows

16.3 Human-in-the-Loop (HITL) Interrupts: LangGraph interrupt(), Diff Preview Cards, State Editing & Resume Flows [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Human • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Human-in-the-Loop (HITL) Interrupts • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Human-in-the-Loop (HITL) Interrupts: LangGraph interrupt(), Diff Preview Cards, State Editing & Resume Flows in real Generative AI, Large Language...

What Topic Is Next → Next Topic 16.4: Low-Latency Spoken Voice AI Agents: VAD, Streaming STT -> Agent -> Streaming TTS & Barge-In Interruption

16.4 Low-Latency Spoken Voice AI Agents: VAD, Streaming STT -> Agent -> Streaming TTS & Barge-In Interruption [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Low • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Low-Latency Spoken Voice AI Agents • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Low-Latency Spoken Voice AI Agents: VAD, Streaming STT -> Agent -> Streaming TTS & Barge-In Interruption in real Generative AI, Large Language Mod...

What Topic Is Next → Module 16 Capstone (nexusagent-m16-mcp-hitl-voice-agents) → Next Topic 17.1: RAG vs Prompting vs Fine-Tuning Decision Matrix & Curating High-Signal Instruction / ShareGPT Datasets

Module 16 Hands-On Capstone Project (100 Marks): nexusagent-m16-mcp-hitl-voice-agents — Architect, implement, test, and push the production GitHub repository 'nexusagent-m16-mcp-hitl-voice-agents' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 16:...

MODULE 17 OF 22 • STAGE 3 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Domain Adaptation: Synthetic Datasets, LoRA / QLoRA Fine-Tuning, DPO & Quantization

Beginner-to-expert real-repository module covering Domain Adaptation: Synthetic Datasets, LoRA / QLoRA Fine-Tuning, DPO & Quantization with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ RAG vs Prompting vs Fine-Tuning Decision Matrix & Curating High-Signal Instruction / ShareGPT Datasets — Purpose: Avoid wasting weeks fine-tuning for facts (use RAG) — fine-tune for domain style, format, and latency!. • ✓ Synthetic Data Generation, Decontamination, Quality Filtering & Formatting Chat Templates Correctly — Purpose: Distill compact 3B–8B specialist models from frontier teacher outputs cleanly without test leakage. • ✓ Parameter-Efficient Fine-Tuning (PEFT): LoRA & 4-Bit QLoRA with HuggingFace, TRL, Unsloth & Weights/Trackers — Purpose: Fine-tune open-weight Llama/Qwen/Mistral models on a single affordable GPU with 70% less VRAM. • ✓ Preference Alignment (DPO), Adapter Merging & Post-Training Quantization (AWQ, GPTQ, GGUF FP8/INT4) — Purpose: Align model tone/safety and compress weights for 3x faster, cheaper production inference.

17.1 RAG vs Prompting vs Fine-Tuning Decision Matrix & Curating High-Signal Instruction / ShareGPT Datasets [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for RAG vs Prompting... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for RAG vs Prompting vs Fine-Tuning Decision Matrix • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply RAG vs Prompting vs Fine-Tuning Decision Matrix & Curating High-Signal Instruction / ShareGPT Datasets in real Generative AI, Large Language Model...

What Topic Is Next → Next Topic 17.2: Synthetic Data Generation, Decontamination, Quality Filtering & Formatting Chat Templates Correctly

17.2 Synthetic Data Generation, Decontamination, Quality Filtering & Formatting Chat Templates Correctly [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Synthetic Data Ge... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Synthetic Data Generation • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Synthetic Data Generation, Decontamination, Quality Filtering & Formatting Chat Templates Correctly in real Generative AI, Large Language Models (...

What Topic Is Next → Next Topic 17.3: Parameter-Efficient Fine-Tuning (PEFT): LoRA & 4-Bit QLoRA with HuggingFace, TRL, Unsloth & Weights/Trackers

17.3 Parameter-Efficient Fine-Tuning (PEFT): LoRA & 4-Bit QLoRA with HuggingFace, TRL, Unsloth & Weights/Trackers [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Parameter • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Parameter-Efficient Fine-

Tuning (PEFT) • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Parameter-Efficient Fine-Tuning (PEFT): LoRA & 4-Bit QLoRA with HuggingFace, TRL, Unsloth & Weights/Trackers in real Generative AI, Large Language...

What Topic Is Next → Next Topic 17.4: Preference Alignment (DPO), Adapter Merging & Post-Training Quantization (AWQ, GPTQ, GGUF FP8/INT4)

17.4 Preference Alignment (DPO), Adapter Merging & Post-Training Quantization (AWQ, GPTQ, GGUF FP8/INT4) [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Preference Alignm... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Preference Alignment (DPO) • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Preference Alignment (DPO), Adapter Merging & Post-Training Quantization (AWQ, GPTQ, GGUF FP8/INT4) in real Generative AI, Large Language Models (...

What Topic Is Next → Module 17 Capstone (nexusagent-m17-qlora-finetuning-dpo) → Next Topic 18.1: Production Model Serving Internals: Continuous Batching, PagedAttention & Deploying vLLM / TGI OpenAI-Compatible APIs

Module 17 Hands-On Capstone Project (100 Marks): nexusagent-m17-qlora-finetuning-dpo — Architect, implement, test, and push the production GitHub repository 'nexusagent-m17-qlora-finetuning-dpo' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 17: D...

MODULE 18 OF 22 • STAGE 4 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

High-Throughput LLM Serving (vLLM), Semantic Caching, Guardrails & Red-Teaming

Beginner-to-expert real-repository module covering High-Throughput LLM Serving (vLLM), Semantic Caching, Guardrails & Red-Teaming with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Production Model Serving Internals: Continuous Batching, PagedAttention & Deploying vLLM / TGI OpenAI-Compatible APIs — Purpose: Achieve 5x–10x higher token throughput per GPU compared to naive HuggingFace generate() loops. • ✓ Multi-Model Routing Gateways, Fallback Chains, Redis Vector Semantic Caching & Token Unit-Economics FinOps — Purpose: Route simple intents to fast small models, cache frequent answers, and guarantee 99.9% AI uptime. • ✓ Input / Output Guardrails: PII Redaction (PAN/Aadhaar/Cards), Toxicity Filters, Topic Rails & Schema Validators — Purpose: Ensure enterprise AI responses never leak PII, discuss competitors, or violate compliance rules. • ✓ OWASP Top-10 for LLM Applications: Indirect Prompt Injection Defense, Tool Privilege Escalation & Automated Red-Teaming — Purpose: Harden RAG and agentic tool pipelines against adversarial jailbreaks and poisoned documents.

18.1 Production Model Serving Internals: Continuous Batching, PagedAttention & Deploying vLLM / TGI OpenAI-Compatible APIs [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Production Model... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Production Model Serving Internals • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Production Model Serving Internals: Continuous Batching, PagedAttention & Deploying vLLM / TGI OpenAI-Compatible APIs in real Generative AI, Large...

What Topic Is Next → Next Topic 18.2: Multi-Model Routing Gateways, Fallback Chains, Redis Vector Semantic Caching & Token Unit-Economics FinOps

18.2 Multi-Model Routing Gateways, Fallback Chains, Redis Vector Semantic Caching & Token Unit-Economics FinOps [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Multi • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Multi-Model Routing Gateways • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Multi-Model Routing Gateways, Fallback Chains, Redis Vector Semantic Caching & Token Unit-Economics FinOps in real Generative AI, Large Language M...

What Topic Is Next → Next Topic 18.3: Input / Output Guardrails: PII Redaction (PAN/Aadhaar/Cards), Toxicity Filters, Topic Rails & Schema Validators

18.3 Input / Output Guardrails: PII Redaction (PAN/Aadhaar/Cards), Toxicity Filters, Topic Rails & Schema Validators [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Input / Output Gu... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Input / Output Guardrails • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Input / Output Guardrails: PII Redaction (PAN/Aadhaar/Cards), Toxicity Filters, Topic Rails & Schema Validators in real Generative AI, Large Langu...

What Topic Is Next → Next Topic 18.4: OWASP Top-10 for LLM Applications: Indirect Prompt Injection Defense, Tool Privilege Escalation & Automated Red-Teaming

18.4 OWASP Top-10 for LLM Applications: Indirect Prompt Injection Defense, Tool Privilege Escalation & Automated Red-Teaming [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for OWASP Top • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for OWASP Top-10 for LLM Applications • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply OWASP Top-10 for LLM Applications: Indirect Prompt Injection Defense, Tool Privilege Escalation & Automated Red-Teaming in real Generative AI, Lar...

What Topic Is Next → Module 18 Capstone (nexusagent-m18-vllm-guardrails-security) → Next Topic 19.1: Containerizing Full-Stack AI Platforms: Multi-Stage Python/React Dockerfiles, GPU Runtime Containers & Compose

Module 18 Hands-On Capstone Project (100 Marks): nexusagent-m18-vllm-guardrails-security — Architect, implement, test, and push the production GitHub repository 'nexusagent-m18-vllm-guardrails-security' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 1...

MODULE 19 OF 22 • STAGE 4 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Cloud LLMOps: Docker, Kubernetes, AWS Bedrock, Terraform IaC & Langfuse Tracing

Beginner-to-expert real-repository module covering Cloud LLMOps: Docker, Kubernetes, AWS Bedrock, Terraform IaC & Langfuse Tracing with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ Containerizing Full-Stack AI Platforms: Multi-Stage Python/React Dockerfiles, GPU Runtime Containers & Compose — Purpose: Package FastAPI, Django, Celery, React, pgvector, and Qdrant cleanly for cloud deployment. • ✓ Kubernetes Orchestration for AI: Deployments, Ingress TLS, Secrets, Queue-Driven HPA & GPU Node Scheduling — Purpose: Autoscale stateless AI gateways and background ingestion workers smoothly on Kubernetes (EKS). • ✓ AWS Cloud & Enterprise AI Infrastructure: Multi-AZ VPC, RDS pgvector, S3, SQS, AWS Bedrock & Terraform IaC — Purpose: Provision compliant, VPC-isolated AWS cloud + Bedrock enterprise AI stacks via Terraform HCL. • ✓ GitHub Actions CI/CD + AI Eval Gates & Full-Stack LLMOps Observability (Langfuse, OpenTelemetry & Prometheus) — Purpose: Trace every prompt version, tool span, token cost, user thumbs-up/down score, and latency regression live.

19.1 Containerizing Full-Stack AI Platforms: Multi-Stage Python/React Dockerfiles, GPU Runtime Containers & Compose [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Containerizing Fu... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Containerizing Full-Stack AI Platforms • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Containerizing Full-Stack AI Platforms: Multi-Stage Python/React Dockerfiles, GPU Runtime Containers & Compose in real Generative AI, Large Langua...

What Topic Is Next → Next Topic 19.2: Kubernetes Orchestration for AI: Deployments, Ingress TLS, Secrets, Queue-Driven HPA & GPU Node Scheduling

19.2 Kubernetes Orchestration for AI: Deployments, Ingress TLS, Secrets, Queue-Driven HPA & GPU Node Scheduling [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Kubernetes Orches... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Kubernetes Orchestration for AI • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Kubernetes Orchestration for AI: Deployments, Ingress TLS, Secrets, Queue-Driven HPA & GPU Node Scheduling in real Generative AI, Large Language M...

What Topic Is Next → Next Topic 19.3: AWS Cloud & Enterprise AI Infrastructure: Multi-AZ VPC, RDS pgvector, S3, SQS, AWS Bedrock & Terraform IaC

19.3 AWS Cloud & Enterprise AI Infrastructure: Multi-AZ VPC, RDS pgvector, S3, SQS, AWS Bedrock & Terraform IaC [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for AWS Cloud • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply AWS Cloud & Enterprise AI Infrastructure: Multi-AZ VPC, RDS pgvector, S3, SQS, AWS Bedrock & Terraform IaC in real Generative AI, Large Language M...

What Topic Is Next → Next Topic 19.4: GitHub Actions CI/CD + AI Eval Gates & Full-Stack LLMOps Observability (Langfuse, OpenTelemetry & Prometheus)

19.4 GitHub Actions CI/CD + AI Eval Gates & Full-Stack LLMOps Observability (Langfuse, OpenTelemetry & Prometheus) [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for GitHub Actions CI... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for GitHub Actions CI/CD + AI Eval Gates • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply GitHub Actions CI/CD + AI Eval Gates & Full-Stack LLMOps Observability (Langfuse, OpenTelemetry & Prometheus) in real Generative AI, Large Languag...

What Topic Is Next → Module 19 Capstone (nexusagent-m19-cloud-llmops-kubernetes) → Next Topic 20.1: GenAI & Full-Stack System Design Interviews: Designing Perplexity Search, GitHub Copilot, Voice Agents & Multi-Tenant RAG

Module 19 Hands-On Capstone Project (100 Marks): nexusagent-m19-cloud-llmops-kubernetes — Architect, implement, test, and push the production GitHub repository 'nexusagent-m19-cloud-llmops-kubernetes' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 19...

MODULE 20 OF 22 • STAGE 4 • 27 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

GenAI System Design & NexusAgent Live Client Production Cutover (≥90% Placement Sprint)

Beginner-to-expert real-repository module covering GenAI System Design & NexusAgent Live Client Production Cutover (≥90% Placement Sprint) with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

Objectives: ✓ GenAI & Full-Stack System Design Interviews: Designing Perplexity Search, GitHub Copilot, Voice Agents & Multi-Tenant RAG — Purpose: Ace frontier AI Engineer and Senior Full-Stack system design rounds with concrete latency/cost math. • ✓ End-to-End Delivery of NexusAgent Multi-Agent LangGraph + MCP + Hybrid RAG + Voice + React 19 SaaS Cloud — Purpose: Ship our flagship full-stack Agentic AI platform with HITL approvals, citations, and LLMOps tracing. • ✓ Red-Team Security Audit, RAGAS >95% Accuracy Gate, Live AWS HTTPS Cutover & 20-Repo GitHub Portfolio — Purpose: Deploy to live cloud production, polish your ATS GenAI resume, and earn your Verified Internship Letter. • ✓ Timed Python DSA + GenAI Mocks, Spoken MockAttempt AI Bar-Raiser Vivas & ≥90% Placement Guarantee Activation — Purpose: Clear all ≥90% topic mocks and AI vivas (unlimited free re-attempts) to unlock 100 job openings in 6 months.

20.1 GenAI & Full-Stack System Design Interviews: Designing Perplexity Search, GitHub Copilot, Voice Agents & Multi-Tenant RAG [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for GenAI • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas • Diagnostics, Profiling & Telemetry: OpenTelemetry traces, latency histograms & vector recall benchmarks

Purpose: To understand and apply GenAI & Full-Stack System Design Interviews: Designing Perplexity Search, GitHub Copilot, Voice Agents & Multi-Tenant RAG in real Generative AI, L...

What Topic Is Next → Next Topic 20.2: End-to-End Delivery of NexusAgent Multi-Agent LangGraph + MCP + Hybrid RAG + Voice + React 19 SaaS Cloud

20.2 End-to-End Delivery of NexusAgent Multi-Agent LangGraph + MCP + Hybrid RAG + Voice + React 19 SaaS Cloud [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for End • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for End-to-End Delivery of NexusAgent Multi-Agent LangGr... • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply End-to-End Delivery of NexusAgent Multi-Agent LangGraph + MCP + Hybrid RAG + Voice + React 19 SaaS Cloud in real Generative AI, Large Language Mod...

What Topic Is Next → Next Topic 20.3: Red-Team Security Audit, RAGAS >95% Accuracy Gate, Live AWS HTTPS Cutover & 20-Repo GitHub Portfolio

20.3 Red-Team Security Audit, RAGAS >95% Accuracy Gate, Live AWS HTTPS Cutover & 20-Repo GitHub Portfolio [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Red • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Red-Team Security Audit • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Red-Team Security Audit, RAGAS >95% Accuracy Gate, Live AWS HTTPS Cutover & 20-Repo GitHub Portfolio in real Generative AI, Large Language Models...

What Topic Is Next → Next Topic 20.4: Timed Python DSA + GenAI Mocks, Spoken MockAttempt AI Bar-Raiser Vivas & ≥90% Placement Guarantee Activation

20.4 Timed Python DSA + GenAI Mocks, Spoken MockAttempt AI Bar-Raiser Vivas & ≥90% Placement Guarantee Activation [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Input Validation, Edge • Modular Clean • Idiomatic GAIF Implementation • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Timed Python DSA... • Runnable Code Implementation (PYTHON) • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Timed Python DSA + GenAI Mocks • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Timed Python DSA + GenAI Mocks, Spoken MockAttempt AI Bar-Raiser Vivas & ≥90% Placement Guarantee Activation in real Generative AI, Large Language...

What Topic Is Next → Module 20 Capstone (nexusagent-m20-live-client-production-capstone) → Next Topic 21.1: Agentic Design Patterns: Planner-Executor, Reflection, Tool Sandboxing & State Persistence

Module 20 Hands-On Capstone Project (100 Marks): nexusagent-m20-live-client-production-capstone — Architect, implement, test, and push the production GitHub repository 'nexusagent-m20-live-client-production-capstone' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of M...

MODULE 21 OF 22 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Multi-Agent Orchestration, Tool-Use Guardrails & Production RAG Evaluation

Self-explanatory list-and-bullet production mastery module covering Multi-Agent Orchestration, Tool-Use Guardrails & Production RAG Evaluation in Generative AI, Large Language Models (LLM) & Agentic AI Full Stack Engineering, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ Agentic Design Patterns: Planner-Executor, Reflection, Tool Sandboxing & State Persistence — Purpose: Engineer reliable multi-step AI agents with bounded retries, timeouts, and human approval gates. • ✓ Advanced Hybrid Multimodal RAG, Re-Ranking & Automated Faithfulness/Safety Evals — Purpose: Eliminate hallucinations on complex PDFs, tables, and enterprise knowledge bases. • ✓ Prompt-Injection Defense, PII Redaction, Output Schema Enforcement & Audit Logging — Purpose: Meet enterprise security and compliance standards before exposing LLM endpoints to users. • ✓ Build, verify, and document 'Evaluated Multi-Agent Research & Action Platform with LangGraph, MCP Tools & Guardrails' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

21.1 Agentic Design Patterns: Planner-Executor, Reflection, Tool Sandboxing & State Persistence [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Agentic Design Patterns • Performance Optimization, Complexity & Latency Impact of Agentic Design Patterns • Live Debugging, Stack • Runnable Code Implementation (PYTHON) • Hands • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Agentic Design Patterns • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Agentic Design Patterns: Planner-Executor, Reflection, Tool Sandboxing & State Persistence in real Generative AI, Large Language Models (LLM) & Ag...

What Topic Is Next → Next Topic 21.2: Advanced Hybrid Multimodal RAG, Re-Ranking & Automated Faithfulness/Safety Evals

21.2 Advanced Hybrid Multimodal RAG, Re-Ranking & Automated Faithfulness/Safety Evals [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Advanced Hybrid Multimodal RAG, Re • Performance Optimization, Complexity & Latency Impact of Advanced Hybrid Multimodal RAG, Re • Live Debugging, Stack • Runnable Code Implementation (PYTHON) • Hands • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Advanced Hybrid Multimodal RAG • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Advanced Hybrid Multimodal RAG, Re-Ranking & Automated Faithfulness/Safety Evals in real Generative AI, Large Language Models (LLM) & Agentic AI F...

What Topic Is Next → Next Topic 21.3: Prompt-Injection Defense, PII Redaction, Output Schema Enforcement & Audit Logging

21.3 Prompt-Injection Defense, PII Redaction, Output Schema Enforcement & Audit Logging [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Prompt • Performance Optimization, Complexity & Latency Impact of Prompt • Live Debugging, Stack • Runnable Code Implementation (PYTHON) • Hands • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Prompt-Injection Defense • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Prompt-Injection Defense, PII Redaction, Output Schema Enforcement & Audit Logging in real Generative AI, Large Language Models (LLM) & Agentic AI...

What Topic Is Next → Module 21 Capstone (Evaluated Multi-Agent Research & Action Platform with LangGraph, MCP Tools & Guardrails) → Next Topic 22.1: Semantic Caching, Model Routing, Quantization/LoRA Trade-Offs & Token FinOps

Module 21 Hands-On Capstone Project (100 Marks): Evaluated Multi-Agent Research & Action Platform with LangGraph, MCP Tools & Guardrails — Architect, implement, test, and document 'Evaluated Multi-Agent Research & Action Platform with LangGraph, MCP Tools & Guardrails' applying every bullet-wise engineering concept from Module...

MODULE 22 OF 22 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

LLM Inference Optimization, Full-Stack Streaming UX & GenAI Placement Defense

Self-explanatory list-and-bullet production mastery module covering LLM Inference Optimization, Full-Stack Streaming UX & GenAI Placement Defense in Generative AI, Large Language Models (LLM) & Agentic AI Full Stack Engineering, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ Semantic Caching, Model Routing, Quantization/LoRA Trade-Offs & Token FinOps — Purpose: Deliver sub-second time-to-first-token while cutting production LLM inference costs. • ✓ Full-Stack React 19 + TypeScript Streaming Interfaces, Citations & Feedback Loops — Purpose: Build intuitive AI product UX with live tool-status indicators and thumbs-up/down eval capture. • ✓ End-to-End GenAI & LLM System Design Defense for Top AI Engineering Roles — Purpose: Architect enterprise copilots, agent platforms, and evaluation pipelines in final interviews. • ✓ Build, verify, and document 'vLLM/FastAPI Streaming Gateway + React 19 Copilot UI + AWS Observability Repository' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

22.1 Semantic Caching, Model Routing, Quantization/LoRA Trade-Offs & Token FinOps [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Semantic Caching, Model Routing, Quantization/LoRA Trade • Performance Optimization, Complexity & Latency Impact of Semantic Caching, Model Routing, Quantization/LoRA Trade • Live Debugging, Stack • Runnable Code Implementation (PYTHON) • Hands • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Semantic Caching • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Semantic Caching, Model Routing, Quantization/LoRA Trade-Offs & Token FinOps in real Generative AI, Large Language Models (LLM) & Agentic AI Full...

What Topic Is Next → Next Topic 22.2: Full-Stack React 19 + TypeScript Streaming Interfaces, Citations & Feedback Loops

22.2 Full-Stack React 19 + TypeScript Streaming Interfaces, Citations & Feedback Loops [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Full • Performance Optimization, Complexity & Latency Impact of Full • Live Debugging, Stack • Runnable Code Implementation (PYTHON) • Hands • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for Full-Stack React 19 + TypeScript Streaming Interface... • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply Full-Stack React 19 + TypeScript Streaming Interfaces, Citations & Feedback Loops in real Generative AI, Large Language Models (LLM) & Agentic AI...

What Topic Is Next → Next Topic 22.3: End-to-End GenAI & LLM System Design Defense for Top AI Engineering Roles

22.3 End-to-End GenAI & LLM System Design Defense for Top AI Engineering Roles [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for End • Performance Optimization, Complexity & Latency Impact of End • Live Debugging, Stack • Runnable Code Implementation (PYTHON) • Hands • Idiomatic GAIF Implementation: async/await non-blocking endpoints, typed Pydantic contracts & grounded RAG • Asyncio Event Loop, FastAPI ASGI, Pydantic v2 & pgvector/PyTorch Execution & Memory Lifecycle for End-to-End GenAI • Production Hardening & Security: OAuth2 scopes, prompt-injection guardrails & token-rate quotas

Purpose: To understand and apply End-to-End GenAI & LLM System Design Defense for Top AI Engineering Roles in real Generative AI, Large Language Models (LLM) & Agentic AI Full Sta...

What Topic Is Next → Module 22 Capstone → End-of-Course Live Client Project: NexusAgent Enterprise Multi-Agent RAG Copilot, Voice/MCP Gateway & Full-Stack LLMOps Cloud

Module 22 Hands-On Capstone Project (100 Marks): vLLM/FastAPI Streaming Gateway + React 19 Copilot UI + AWS Observability Repository — Architect, implement, test, and document 'vLLM/FastAPI Streaming Gateway + React 19 Copilot UI + AWS Observability Repository' applying every bullet-wise engineering concept from Module 22 (...

3. LIVE CLIENT PRODUCTION PROJECT & CAREER SCOPE

End-of-Course Live Client Production Project + Official Internship Confirmation Letter

Project: NexusAgent Enterprise Multi-Agent RAG Copilot, Voice/MCP Gateway & Full-Stack LLMOps Cloud • **Confirmed Role:** Generative AI, Agentic LLM & Full-Stack Cloud Engineering Resident

Domain & Production Infra: Enterprise AI Copilot, FinTech & Autonomous Support Client (Python 3.12 + FastAPI + Django 5 + LangGraph + MCP + pgvector + React 19 TS + AWS)

- Engineered hybrid BM25 + pgvector semantic retrieval, cross-encoder reranking, and hallucination-free citation pipelines.
- Built autonomous LangGraph multi-agent orchestrators and custom MCP tool servers with human-in-the-loop approvals.
- Fine-tuned domain adapters with QLoRA, enforced prompt-injection guardrails, and automated RAGAS regression evals in CI.
- Shipped React 19 TypeScript streaming workspaces and async FastAPI/Django backends onto Docker, Kubernetes, and AWS.

Target Job Roles: Generative AI & Agentic LLM Full Stack Engineer, Applied AI / LLM RAG & Multi-Agent Engineer, Full Stack AI Product Engineer (Python + React + LLMOps), LLMOps & Enterprise AI Platform Engineer

Realistic Annual CTC Scope: **Placement-Guaranteed GenAI Full Stack Engineer (0–1 Yr):** ₹7.5 LPA – ₹16.5 LPA **Senior GenAI / Agentic AI Systems SDE-2 (1–3 Yrs):** ₹17.0 LPA – ₹34.0 LPA **Principal AI Architect / Staff LLM Engineer (3–6+ Yrs):** ₹35.0 LPA – ₹70.0+ LPA

Optional Worthwhile Global Certifications (100% Exam-Success Mentorship Available): **AWS Certified Machine Learning Engineer – Associate (MLA-C01):** ₹14.0 LPA – ₹32.0 LPA (+35–45% AI Salary Premium) • **Databricks Certified Generative AI Engineer Associate (GENAI-ASSOC):** ₹15.0 LPA – ₹35.0 LPA (High-Demand GCC Credential) • **NVIDIA Generative AI LLMs & CNCF Kubernetes (LLMOps) (NCA-GENL / CKAD):** ₹18.0 LPA – ₹44.0 LPA (Frontier AI Infrastructure)