**MockAttempt**

**TECH ACADEMY • OFFICIAL COURSE SYLLABUS BROCHURE**

MockAttempt Academy Pvt. Ltd. • Live Architect Mentorship, AI Assessments & Client Production

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

**PFSC • FULL STACK DEVELOPMENT** **BEGINNER TO EXPERT • JOB-GUARANTEED***

**100% PLACEMENT GUARANTEED* CAREER PROGRAM (6-MO REFUND*)**

# Python Full Stack Development & Enterprise Generative AI Architecture

Complete Zero-to-Expert Python Foundations to 3.12 Async + OOP + DSA + SQL + HTML/CSS/React 19 + Django 5 + FastAPI + LangGraph GenAI + AWS + DevOps | 100% Placement Guaranteed* (6 Months Fees Refund*)

**Duration:** 32w (480h) • Weekend Only (2h/Day) (508 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: ₹34,999** (Net Tuition ₹29,660 + Included 18% GST ₹5,339) • **Split Option:** ₹5,333/mo × 6 (0% Split) • **Effective ROI:** ~₹69/Guided Hr (~₹407/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) • 32 Weeks (508 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 From Zero to 3.12 Async • OOP, Data Structures & DSA • Django 5 & Async FastAPI • LangGraph GenAI, React 19 & AWS • SQL From Scratch & PostgreSQL 16 • HTML5, CSS3, Tailwind & TypeScript 5 • pgvector & Hybrid RAG Search • Pydantic v2 & SQLAlchemy 2 • Celery & Redis 7 Task Queues • Agentic AI & MCP Tool Calling • WebSockets & SSE Streaming • JWT / OAuth2 & RBAC Security • Pytest Automated Test Suites • Docker & Kubernetes (EKS) • AWS Cloud & Amazon Bedrock • Terraform Infrastructure as Code • GitHub Actions DevSecOps CI/CD • OpenTelemetry & System Design

## 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. Dual-framework Python backend mastery: Django 5 + DRF for enterprise platforms and async FastAPI for high-concurrency/AI APIs.

**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 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

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

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

**Objectives:** ✓ How Programs Run, Python 3.12 Bytecode / PVM Setup, VS Code, Virtual Environments & First Scripts — Purpose: Install Python 3.12 cleanly, isolate project environments from Day 1, and execute your first interactive programs. • ✓ Linux Command Line Navigation, Shell Pipes & Professional Git / GitHub Pull Request Workflows — Purpose: Operate confidently in Linux terminals and manage branches, commits, merges, and code reviews on GitHub. • ✓ Variables, Dynamic vs Strong Typing, Numbers, Strings, Booleans, Operators & Formatted CLI I/O — Purpose: Master Python primitive operations, string slicing, f-strings, and safe user input parsing without type bugs. • ✓ Conditional Branching (if/elif/else, match-case), Loops (for/while) & Step-by-Step Logic Building — Purpose: Develop sharp algorithmic intuition by solving 25+ structured control-flow, number, and pattern exercises.

### 1.1 How Programs Run, Python 3.12 Bytecode / PVM Setup, VS Code, Virtual Environments & First Scripts

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

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for How Programs Run • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply How Programs Run, Python 3.12 Bytecode / PVM Setup, VS Code, Virtual Environments & First Scripts in real Python Full Stack Development & Enterpri...

**What Topic Is Next → Next Topic 1.2: Linux Command Line Navigation, Shell Pipes & Professional Git / GitHub Pull Request Workflows**

### 1.2 Linux Command Line Navigation, Shell Pipes & Professional Git / GitHub Pull Request Workflows

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

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Linux Command Line Navigat... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Linux Command Line Navigation • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Linux Command Line Navigation, Shell Pipes & Professional Git / GitHub Pull Request Workflows in real Python Full Stack Development & Enterprise G...

**What Topic Is Next → Next Topic 1.3: Variables, Dynamic vs Strong Typing, Numbers, Strings, Booleans, Operators & Formatted CLI I/O**

### 1.3 Variables, Dynamic vs Strong Typing, Numbers, Strings, Booleans, Operators & Formatted CLI I/O

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

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Variables • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Variables, Dynamic vs Strong Typing, Numbers, Strings, Booleans, Operators & Formatted CLI I/O in real Python Full Stack Development & Enterprise...

**What Topic Is Next → Next Topic 1.4: Conditional Branching (if/elif/else, match-case), Loops (for/while) & Step-by-Step Logic Building**

### 1.4 Conditional Branching (if/elif/else, match-case), Loops (for/while) & Step-by-Step Logic Building

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

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Conditional Branching (if/... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Conditional Branching (if/elif/else, match-case) • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Conditional Branching (if/elif/else, match-case), Loops (for/while) & Step-by-Step Logic Building in real Python Full Stack Development & Enterpri...

**What Topic Is Next → Module 1 Capstone (omnidoc-m01-python-foundations-cli) → Next Topic 2.1: Lists, Tuples, Sets & Dictionaries: Memory Internals, Hash Lookups, Slicing & Comprehensions**

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

**MODULE 2 OF 22 • STAGE 1 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Python Data Structures, Functions, Modules, File I/O & Exception Handling

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

**Objectives:** ✓ Lists, Tuples, Sets & Dictionaries: Memory Internals, Hash Lookups, Slicing & Comprehensions — Purpose: Choose O(1) dict/set lookups over slow O(N) list scans and write clean, Pythonic data transformations. • ✓ Functions, Positional / Keyword Arguments, *args / **kwargs, LEGB Scope, Closures & Lambda Functions — Purpose: Design modular, reusable functions while avoiding mutable default-argument traps and global-state bugs. • ✓ Packages, Virtual Environments (venv / Poetry / uv), CSV / JSON Processing & Context-Managed File I/O — Purpose: Structure multi-file Python packages cleanly and read/write structured files safely using with context managers. • ✓ Exception Hierarchies, try / except / else / finally, Custom Domain Exceptions & Structured Logging — Purpose: Replace fragile print debugging with explicit error handling and JSON-formatted diagnostic logs.

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

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Lists • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Lists, Tuples, Sets & Dictionaries: Memory Internals, Hash Lookups, Slicing & Comprehensions in real Python Full Stack Development & Enterprise Ge...

**What Topic Is Next → Next Topic 2.2: Functions, Positional / Keyword Arguments, *args / **kwargs, LEGB Scope, Closures & Lambda Functions**

**2.2 Functions, Positional / Keyword Arguments, *args / **kwargs, LEGB Scope, Closures & Lambda Functions** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Functions • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Functions, Positional / Keyword Arguments, *args / **kwargs, LEGB Scope, Closures & Lambda Functions in real Python Full Stack Development & Enter...

**What Topic Is Next → Next Topic 2.3: Packages, Virtual Environments (venv / Poetry / uv), CSV / JSON Processing & Context-Managed File I/O**

**2.3 Packages, Virtual Environments (venv / Poetry / uv), CSV / JSON Processing & Context-Managed File I/O** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Packages • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Packages, Virtual Environments (venv / Poetry / uv), CSV / JSON Processing & Context-Managed File I/O in real Python Full Stack Development & Ente...

**What Topic Is Next → Next Topic 2.4: Exception Hierarchies, try / except / else / finally, Custom Domain Exceptions & Structured Logging**

**2.4 Exception Hierarchies, try / except / else / finally, Custom Domain Exceptions & Structured Logging** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Exception Hierarchies • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Exception Hierarchies, try / except / else / finally, Custom Domain Exceptions & Structured Logging in real Python Full Stack Development & Enterp...

**What Topic Is Next → Module 2 Capstone (omnidoc-m02-python-datastructures-io) → Next Topic 3.1: Classes, Instances, __init__, Instance vs Class Attributes, @classmethod, @staticmethod & @property**

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

**MODULE 3 OF 22 • STAGE 1 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Object-Oriented Python, SOLID Architecture, Type Hints & Pytest TDD

Beginner-to-expert real-repository module covering Object-Oriented Python, SOLID Architecture, Type Hints & Pytest TDD with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ Classes, Instances, __init__, Instance vs Class Attributes, @classmethod, @staticmethod & @property — Purpose: Model clean domain entities with encapsulated invariants and computed properties in modern Python. • ✓ Inheritance, Method Resolution Order (MRO), super(), Dunder Magic Methods & Dataclasses — Purpose: Customize equality, ordering, representation, and immutable value objects using dataclasses and magic hooks. • ✓ Abstract Base Classes (abc), Structural Protocols, Composition vs Inheritance & All 5 SOLID Principles — Purpose: Decouple business rules from database and API adapters using clean Python interfaces and Dependency Inversion. • ✓ Static Type Annotations (typing / mypy / ruff) & Test-Driven Development with Pytest Fixtures & Mocks — Purpose: Catch type mismatches before runtime and verify happy paths and edge cases with parametrized Pytest suites.

**3.1 Classes, Instances, __init__, Instance vs Class Attributes, @classmethod, @staticmethod & @property** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Classes • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets

& atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Classes, Instances, __init__, Instance vs Class Attributes, @classmethod, @staticmethod & @property in real Python Full Stack Development & Enterp...

**What Topic Is Next → Next Topic 3.2: Inheritance, Method Resolution Order (MRO), super(), Dunder Magic Methods & Dataclasses**

### 3.2 Inheritance, Method Resolution Order (MRO), super(), Dunder Magic Methods & Dataclasses [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Inheritance • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Inheritance, Method Resolution Order (MRO), super(), Dunder Magic Methods & Dataclasses in real Python Full Stack Development & Enterprise Generat...

**What Topic Is Next → Next Topic 3.3: Abstract Base Classes (abc), Structural Protocols, Composition vs Inheritance & All 5 SOLID Principles**

### 3.3 Abstract Base Classes (abc), Structural Protocols, Composition vs Inheritance & All 5 SOLID Principles [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Abstract Base Classes (abc... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Abstract Base Classes (abc) • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Abstract Base Classes (abc), Structural Protocols, Composition vs Inheritance & All 5 SOLID Principles in real Python Full Stack Development & Ent...

**What Topic Is Next → Next Topic 3.4: Static Type Annotations (typing / mypy / ruff) & Test-Driven Development with Pytest Fixtures & Mocks**

### 3.4 Static Type Annotations (typing / mypy / ruff) & Test-Driven Development with Pytest Fixtures & Mocks [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Static Type Annotations (t... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Static Type Annotations (typing / mypy / ruff) • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Static Type Annotations (typing / mypy / ruff) & Test-Driven Development with Pytest Fixtures & Mocks in real Python Full Stack Development & Ente...

**What Topic Is Next → Module 3 Capstone (omnidoc-m03-python-oop-solid-pytest) → Next Topic 4.1: CPython Object Memory, Reference Counting, Cyclic Garbage Collection & Profiling (cProfile / memory_profiler)**

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

**MODULE 4 OF 22 • STAGE 1 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Advanced Python Internals, Iterators, Generators, Concurrency & Asyncio

Beginner-to-expert real-repository module covering Advanced Python Internals, Iterators, Generators, Concurrency & Asyncio with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ CPython Object Memory, Reference Counting, Cyclic Garbage Collection & Profiling (cProfile / memory_profiler) — Purpose: Understand Python memory overhead and pinpoint CPU or RAM hotspots on real data pipelines. • ✓ Iterators, Lazy Generators (yield), itertools Pipelines, Parameterized Decorators & Custom Context Managers — Purpose: Stream gigabyte-scale datasets in constant O(1) RAM and build reusable retry, timing, and auth decorators. • ✓ OS Processes vs Threads, GIL Mechanics, ThreadPoolExecutor & ProcessPoolExecutor Parallelism — Purpose: Know exactly when to use thread pools for network I/O versus multiprocessing for CPU-bound transformations. • ✓ Asyncio Event Loop From Scratch: async / await Coroutines, Tasks, Semaphores, Timeouts & httpx Clients — Purpose: Execute hundreds of concurrent HTTP and LLM API requests non-blockingly without starving the event loop.

### 4.1 CPython Object Memory, Reference Counting, Cyclic Garbage Collection & Profiling (cProfile / memory_profiler) [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for CPython Object Memory • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply CPython Object Memory, Reference Counting, Cyclic Garbage Collection & Profiling (cProfile / memory_profiler) in real Python Full Stack Developmen...

**What Topic Is Next → Next Topic 4.2: Iterators, Lazy Generators (yield), itertools Pipelines, Parameterized Decorators & Custom Context Managers**

### 4.2 Iterators, Lazy Generators (yield), itertools Pipelines, Parameterized Decorators & Custom Context Managers [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Iterators • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Iterators, Lazy Generators (yield), itertools Pipelines, Parameterized Decorators & Custom Context Managers in real Python Full Stack Development...

**What Topic Is Next → Next Topic 4.3: OS Processes vs Threads, GIL Mechanics, ThreadPoolExecutor & ProcessPoolExecutor Parallelism**

**4.3 OS Processes vs Threads, GIL Mechanics, ThreadPoolExecutor & ProcessPoolExecutor Parallelism** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for OS Processes vs Threads • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply OS Processes vs Threads, GIL Mechanics, ThreadPoolExecutor & ProcessPoolExecutor Parallelism in real Python Full Stack Development & Enterprise Ge...

**What Topic Is Next → Next Topic 4.4: Asyncio Event Loop From Scratch: async / await Coroutines, Tasks, Semaphores, Timeouts & httpx Clients**

**4.4 Asyncio Event Loop From Scratch: async / await Coroutines, Tasks, Semaphores, Timeouts & httpx Clients** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Asyncio Event Loop From Sc... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Asyncio Event Loop From Scratch • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Asyncio Event Loop From Scratch: async / await Coroutines, Tasks, Semaphores, Timeouts & httpx Clients in real Python Full Stack Development & Ent...

**What Topic Is Next → Module 4 Capstone (omnidoc-m04-python-internals-asyncio) → Next Topic 5.1: Big-O Complexity Analysis, Two Pointers, Sliding Window, Prefix Sums & Binary Search Patterns**

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

**MODULE 5 OF 22 • STAGE 1 • 24 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 Complexity Analysis, Two Pointers, Sliding Window, Prefix Sums & Binary Search Patterns — Purpose: Transform slow nested loops into linear or logarithmic Python algorithms that pass strict online judge limits. • ✓ Linked Lists, Stacks, collections.deque Queues, Counter / defaultdict Hashing & LRU Cache Design — Purpose: Solve high-frequency array, string, stack, and hash-map coding problems using idiomatic Python standard libraries. • ✓ Binary Trees, BSTs, heapq Priority Queues, Prefix Tries & Recursive Backtracking Search — Purpose: Implement tree traversals, top-K streaming, prefix matching, and constraint-satisfaction solvers cleanly. • ✓ Graphs (BFS, DFS, Topological Sort, Dijkstra) & Dynamic Programming (@cache Memoization & Tabulation) — Purpose: Master shortest-path graph algorithms and 1D/2D dynamic programming for product-company SDE interviews.

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

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Big • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Big-O Complexity Analysis • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Big-O Complexity Analysis, Two Pointers, Sliding Window, Prefix Sums & Binary Search Patterns in real Python Full Stack Development & Enterprise G...

**What Topic Is Next → Next Topic 5.2: Linked Lists, Stacks, collections.deque Queues, Counter / defaultdict Hashing & LRU Cache Design**

**5.2 Linked Lists, Stacks, collections.deque Queues, Counter / defaultdict Hashing & LRU Cache Design** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Linked Lists • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Linked Lists, Stacks, collections.deque Queues, Counter / defaultdict Hashing & LRU Cache Design in real Python Full Stack Development & Enterpris...

**What Topic Is Next → Next Topic 5.3: Binary Trees, BSTs, heapq Priority Queues, Prefix Tries & Recursive Backtracking Search**

**5.3 Binary Trees, BSTs, heapq Priority Queues, Prefix Tries & Recursive Backtracking Search** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Binary Trees • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Binary Trees, BSTs, heapq Priority Queues, Prefix Tries & Recursive Backtracking Search in real Python Full Stack Development & Enterprise Generat...

**What Topic Is Next → Next Topic 5.4: Graphs (BFS, DFS, Topological Sort, Dijkstra) & Dynamic Programming (@cache Memoization & Tabulation)**

**5.4 Graphs (BFS, DFS, Topological Sort, Dijkstra) & Dynamic Programming (@cache Memoization & Tabulation)** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Graphs (BFS, DFS, Topologi... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Graphs (BFS, DFS, Topological Sort, Dijkstra) • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Graphs (BFS, DFS, Topological Sort, Dijkstra) & Dynamic Programming (@cache Memoization & Tabulation) in real Python Full Stack Development & Ente...

**What Topic Is Next → Module 5 Capstone (omnidoc-m05-python-dsa-interviews) → Next Topic 6.1: Relational Database Concepts, ER Modeling, 1NF–3NF Normalization & Core SQL (DDL, DML, Constraints)**

**Module 5 Hands-On Capstone Project (100 Marks):** omnidoc-m05-python-dsa-interviews — Architect, implement, test, and push the production GitHub repository 'omnidoc-m05-python-dsa-interviews' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 5: Data...

**MODULE 6 OF 22 • STAGE 2 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

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

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

**Objectives:** ✓ Relational Database Concepts, ER Modeling, 1NF–3NF Normalization & Core SQL (DDL, DML, Constraints) — Purpose: Design normalized transactional schemas from scratch with strict foreign-key and check integrity constraints. • ✓ Multi-Table JOINs, Aggregations, GROUP BY / HAVING, Subqueries & Parameterized Python DB-API Queries — Purpose: Write accurate business analytics queries and eliminate SQL injection vulnerabilities permanently. • ✓ CTEs, Window Functions (ROW_NUMBER, DENSE_RANK, SUM OVER), ACID Transactions & Row-Level Locking — Purpose: Solve complex cohort reporting and prevent lost-update race conditions on concurrent wallet/inventory writes. • ✓ PostgreSQL 16 MVCC, B-Tree & GIN JSONB Indexes, EXPLAIN ANALYZE Query Tuning & pgvector HNSW Indexes — Purpose: Optimize slow production SQL queries scientifically and store dense AI embeddings directly inside PostgreSQL.

**6.1 Relational Database Concepts, ER Modeling, 1NF–3NF Normalization & Core SQL (DDL, DML, Constraints)** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Relational Database Concep... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Relational Database Concepts • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Relational Database Concepts, ER Modeling, 1NF–3NF Normalization & Core SQL (DDL, DML, Constraints) in real Python Full Stack Development & Enterp...

**What Topic Is Next → Next Topic 6.2: Multi-Table JOINs, Aggregations, GROUP BY / HAVING, Subqueries & Parameterized Python DB-API Queries**

**6.2 Multi-Table JOINs, Aggregations, GROUP BY / HAVING, Subqueries & Parameterized Python DB-API Queries** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Multi • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Multi-Table JOINs • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Multi-Table JOINs, Aggregations, GROUP BY / HAVING, Subqueries & Parameterized Python DB-API Queries in real Python Full Stack Development & Enter...

**What Topic Is Next → Next Topic 6.3: CTEs, Window Functions (ROW_NUMBER, DENSE_RANK, SUM OVER), ACID Transactions & Row-Level Locking**

**6.3 CTEs, Window Functions (ROW_NUMBER, DENSE_RANK, SUM OVER), ACID Transactions & Row-Level Locking** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for CTEs • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply CTEs, Window Functions (ROW_NUMBER, DENSE_RANK, SUM OVER), ACID Transactions & Row-Level Locking in real Python Full Stack Development & Enterpris...

**What Topic Is Next → Next Topic 6.4: PostgreSQL 16 MVCC, B-Tree & GIN JSONB Indexes, EXPLAIN ANALYZE Query Tuning & pgvector HNSW Indexes**

**6.4 PostgreSQL 16 MVCC, B-Tree & GIN JSONB Indexes, EXPLAIN ANALYZE Query Tuning & pgvector HNSW Indexes** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for PostgreSQL 16 MVCC • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply PostgreSQL 16 MVCC, B-Tree & GIN JSONB Indexes, EXPLAIN ANALYZE Query Tuning & pgvector HNSW Indexes in real Python Full Stack Development & Enter...

**What Topic Is Next → Module 6 Capstone (omnidoc-m06-sql-postgresql-pgvector) → Next Topic 7.1: Web Architecture (DNS, HTTP/HTTPS, Cookies, CORS), Semantic HTML5 Forms & WCAG Accessibility**

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

**MODULE 7 OF 22 • STAGE 2 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

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

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

**Objectives:** ✓ Web Architecture (DNS, HTTP/HTTPS, Cookies, CORS), Semantic HTML5 Forms & WCAG Accessibility — Purpose: Understand full browser-to-backend request flows and author semantic, accessible HTML5 interfaces. • ✓ CSS3 Box Model, Flexbox, CSS Grid, Responsive Mobile-First Design & Tailwind CSS Component Styling — Purpose: Build crisp, responsive dashboards and landing layouts across mobile and desktop viewports with Tailwind CSS. • ✓ Core JavaScript From Zero: Variables, Functions, Closures, Arrays, Objects, Modules & ES6+ Syntax — Purpose: Write clean modern JavaScript to transform API data, manage UI state, and modularize frontend logic. • ✓ DOM Events,Browser Event Loop, Promises, async/await, Fetch API & Streaming ReadableStream Consumers — Purpose: Wire dynamic frontend views to Python REST and Server-Sent Events (SSE) endpoints smoothly.

### 7.1 Web Architecture (DNS, HTTP/HTTPS, Cookies, CORS), Semantic HTML5 Forms & WCAG Accessibility

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

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Web Architecture (DNS, HTT... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Web Architecture (DNS, HTTP/HTTPS, Cookies, CORS) • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Web Architecture (DNS, HTTP/HTTPS, Cookies, CORS), Semantic HTML5 Forms & WCAG Accessibility in real Python Full Stack Development & Enterprise Ge...

**What Topic Is Next → Next Topic 7.2: CSS3 Box Model, Flexbox, CSS Grid, Responsive Mobile-First Design & Tailwind CSS Component Styling**

### 7.2 CSS3 Box Model, Flexbox, CSS Grid, Responsive Mobile-First Design & Tailwind CSS Component Styling

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

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for CSS3 Box Model • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply CSS3 Box Model, Flexbox, CSS Grid, Responsive Mobile-First Design & Tailwind CSS Component Styling in real Python Full Stack Development & Enterpr...

**What Topic Is Next → Next Topic 7.3: Core JavaScript From Zero: Variables, Functions, Closures, Arrays, Objects, Modules & ES6+ Syntax**

### 7.3 Core JavaScript From Zero: Variables, Functions, Closures, Arrays, Objects, Modules & ES6+ Syntax

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

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Core JavaScript From Zero • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Core JavaScript From Zero: Variables, Functions, Closures, Arrays, Objects, Modules & ES6+ Syntax in real Python Full Stack Development & Enterpri...

**What Topic Is Next → Next Topic 7.4: DOM Events,Browser Event Loop, Promises, async/await, Fetch API & Streaming ReadableStream Consumers**

### 7.4 DOM Events,Browser Event Loop, Promises, async/await, Fetch API & Streaming ReadableStream Consumers

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

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for DOM Events,Browser Event L... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for DOM Events,Browser Event Loop • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply DOM Events,Browser Event Loop, Promises, async/await, Fetch API & Streaming ReadableStream Consumers in real Python Full Stack Development & Enter...

**What Topic Is Next → Module 7 Capstone (omnidoc-m07-web-foundations-js-dom) → Next Topic 8.1: TypeScript 5 Fundamentals, Interfaces, Generics, Discriminated Unions & Zod Runtime Validation**

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

**MODULE 8 OF 22 • STAGE 2 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Modern Full-Stack Frontend Engineering with Strict TypeScript 5 & React 19

Beginner-to-expert real-repository module covering Modern Full-Stack Frontend 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, Interfaces, Generics, Discriminated Unions & Zod Runtime Validation — Purpose: Share rock-solid API contracts between Python Pydantic schemas and TypeScript frontend models. • ✓ React 19 Core: Components, Props, State, useEffect, useRef, Custom Hooks & Controlled Validation Forms — Purpose: Compose reusable, accessible UI components with predictable rendering and clean side-effect lifecycles. • ✓ React Router

Protected RBAC Views, Zustand Client State & TanStack Query Server-State Synchronization — Purpose: Manage multi-tenant navigation, caching, pagination, and optimistic mutations without boilerplate Redux sprawl. • ✓ Live SSE AI Token Streaming Consoles, Citation Side-Drawers, File Upload Progress & Vitest UI Suites — Purpose: Build ChatGPT-grade streaming enterprise workspaces backed by automated component verification tests.

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

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for TypeScript 5 Fundamentals • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply TypeScript 5 Fundamentals, Interfaces, Generics, Discriminated Unions & Zod Runtime Validation in real Python Full Stack Development & Enterprise...

**What Topic Is Next → Next Topic 8.2: React 19 Core: Components, Props, State, useEffect, useRef, Custom Hooks & Controlled Validation Forms**

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

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for React 19 Core • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply React 19 Core: Components, Props, State, useEffect, useRef, Custom Hooks & Controlled Validation Forms in real Python Full Stack Development & Ent...

**What Topic Is Next → Next Topic 8.3: React Router Protected RBAC Views, Zustand Client State & TanStack Query Server-State Synchronization**

**8.3 React Router Protected RBAC Views, Zustand Client 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 PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for React Router Protected RBA... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for React Router Protected RBAC Views • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply React Router Protected RBAC Views, Zustand Client State & TanStack Query Server-State Synchronization in real Python Full Stack Development & Ente...

**What Topic Is Next → Next Topic 8.4: Live SSE AI Token Streaming Consoles, Citation Side-Drawers, File Upload Progress & Vitest UI Suites**

**8.4 Live SSE AI Token Streaming Consoles, Citation Side-Drawers, File Upload Progress & Vitest UI Suites** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Live SSE AI Token Streamin... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Live SSE AI Token Streaming Consoles • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Live SSE AI Token Streaming Consoles, Citation Side-Drawers, File Upload Progress & Vitest UI Suites in real Python Full Stack Development & Enter...

**What Topic Is Next → Module 8 Capstone (omnidoc-m08-react19-typescript-ui) → Next Topic 9.1: Django 5 Project Structure, URL Routing, Request/Response Middleware, Templates & Settings Security**

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

**MODULE 9 OF 22 • STAGE 2 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Django 5 Web Architecture, ORM Internals, Migrations & Admin Customization

Beginner-to-expert real-repository module covering Django 5 Web Architecture, ORM Internals, Migrations & Admin Customization with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ Django 5 Project Structure, URL Routing, Request/Response Middleware, Templates & Settings Security — Purpose: Scaffold clean multi-app Django 5 platforms with environment-driven twelve-factor configuration. • ✓ Django Models, Relationships, Custom QuerySets / Managers & Zero-Downtime Schema Migrations — Purpose: Encapsulate reusable domain queries cleanly and evolve live PostgreSQL tables without breaking deployments. • ✓ Eliminating ORM N+1 Queries with select_related, prefetch_related, F() / Q() Expressions & Transactions — Purpose: Cut per-request SQL counts by 95% and enforce atomic multi-table writes using transaction.atomic(). • ✓ Django Authentication, Custom User Models, Signals vs Explicit Services & Hardened Admin Consoles — Purpose: Build secure operational back-offices and clean service-layer boundaries for enterprise SaaS teams.

**9.1 Django 5 Project Structure, URL Routing, Request/Response Middleware, Templates & Settings Security** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Django 5 Project Structure • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Django 5 Project Structure, URL Routing, Request/Response Middleware, Templates & Settings Security in real Python Full Stack Development & Enterp...

**What Topic Is Next → Next Topic 9.2: Django Models, Relationships, Custom QuerySets / Managers & Zero-Downtime Schema Migrations**

**9.2 Django Models, Relationships, Custom QuerySets / Managers & Zero-Downtime Schema Migrations** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Django Models • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Django Models, Relationships, Custom QuerySets / Managers & Zero-Downtime Schema Migrations in real Python Full Stack Development & Enterprise Gen...

**What Topic Is Next → Next Topic 9.3: Eliminating ORM N+1 Queries with select_related, prefetch_related, F() / Q() Expressions & Transactions**

**9.3 Eliminating ORM N+1 Queries with select_related, prefetch_related, F() / Q() Expressions & Transactions** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Eliminating ORM N+1 Querie... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Eliminating ORM N+1 Queries with select_related • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Eliminating ORM N+1 Queries with select_related, prefetch_related, F() / Q() Expressions & Transactions in real Python Full Stack Development & En...

**What Topic Is Next → Next Topic 9.4: Django Authentication, Custom User Models, Signals vs Explicit Services & Hardened Admin Consoles**

**9.4 Django Authentication, Custom User Models, Signals vs Explicit Services & Hardened Admin Consoles** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Django Authentication • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Django Authentication, Custom User Models, Signals vs Explicit Services & Hardened Admin Consoles in real Python Full Stack Development & Enterpri...

**What Topic Is Next → Module 9 Capstone (omnidoc-m09-django5-orm-platform) → Next Topic 10.1: DRF Serializers, ModelSerializers, Nested Validation, Generic Views, ViewSets & Routers**

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

**MODULE 10 OF 22 • STAGE 2 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Django REST Framework (DRF), Multi-Tenant RBAC, JWT / OAuth2 & API Testing

Beginner-to-expert real-repository module covering Django REST Framework (DRF), Multi-Tenant RBAC, JWT / OAuth2 & API Testing with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ DRF Serializers, ModelSerializers, Nested Validation, Generic Views, ViewSets & Routers — Purpose: Expose clean, self-consistent RESTful JSON resources with strict field and object-level validation. • ✓ Stateless Rotating JWTs (SimpleJWT), OAuth2 Social/SSO Login & Custom Object-Level RBAC Permissions — Purpose: Isolate multi-tenant organization data strictly and block unauthorized cross-tenant resource access. • ✓ Filtering (django-filter), Cursor Pagination, Redis Throttling, ETag Caching & OpenAPI 3 Schema Docs — Purpose: Scale DRF list endpoints to millions of rows and auto-generate interactive Swagger/Redoc API specs. • ✓ Comprehensive API Integration Testing with pytest-django, FactoryBoy & OWASP API Top-10 Hardening — Purpose: Prove 100% permission, validation, and rate-limit correctness automatically on every pull request.

**10.1 DRF Serializers, ModelSerializers, Nested Validation, Generic Views, ViewSets & Routers** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for DRF Serializers • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply DRF Serializers, ModelSerializers, Nested Validation, Generic Views, ViewSets & Routers in real Python Full Stack Development & Enterprise Generat...

**What Topic Is Next → Next Topic 10.2: Stateless Rotating JWTs (SimpleJWT), OAuth2 Social/SSO Login & Custom Object-Level RBAC Permissions**

**10.2 Stateless Rotating JWTs (SimpleJWT), OAuth2 Social/SSO Login & Custom Object-Level RBAC Permissions** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Stateless Rotating JWTs (S... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Stateless Rotating JWTs (SimpleJWT) • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Stateless Rotating JWTs (SimpleJWT), OAuth2 Social/SSO Login & Custom Object-Level RBAC Permissions in real Python Full Stack Development & Enterp...

**What Topic Is Next → Next Topic 10.3: Filtering (django-filter), Cursor Pagination, Redis Throttling, ETag Caching & OpenAPI 3 Schema Docs**

**10.3 Filtering (django-filter), Cursor Pagination, Redis Throttling, ETag Caching & OpenAPI 3 Schema Docs** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Filtering (django • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Filtering (django-filter) • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Filtering (django-filter), Cursor Pagination, Redis Throttling, ETag Caching & OpenAPI 3 Schema Docs in real Python Full Stack Development & Enter...

**What Topic Is Next → Next Topic 10.4: Comprehensive API Integration Testing with pytest-django, FactoryBoy & OWASP API Top-10 Hardening**

**10.4 Comprehensive API Integration Testing with pytest-django, FactoryBoy & OWASP API Top-10 Hardening** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Comprehensive API Integrat... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Comprehensive API Integration Testing with pytest-dj... • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Comprehensive API Integration Testing with pytest-django, FactoryBoy & OWASP API Top-10 Hardening in real Python Full Stack Development & Enterpri...

**What Topic Is Next → Module 10 Capstone (omnidoc-m10-drf-enterprise-apis) → Next Topic 11.1: ASGI vs WSGI Internals, FastAPI Routing, Pydantic v2 Rust-Core Schemas & Dependency Injection**

**Module 10 Hands-On Capstone Project (100 Marks):** omnidoc-m10-drf-enterprise-apis — Architect, implement, test, and push the production GitHub repository 'omnidoc-m10-drf-enterprise-apis' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 10: Djang...

**MODULE 11 OF 22 • STAGE 2 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## High-Concurrency Async FastAPI, Pydantic v2, SQLAlchemy 2.0 & WebSockets

Beginner-to-expert real-repository module covering High-Concurrency Async FastAPI, Pydantic v2, SQLAlchemy 2.0 & WebSockets with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ ASGI vs WSGI Internals, FastAPI Routing, Pydantic v2 Rust-Core Schemas & Dependency Injection — Purpose: Build ultra-fast sub-25ms Python microservices with automatic OpenAPI docs and strict payload validation. • ✓ Async SQLAlchemy 2.0 Sessions, Alembic Migrations, Connection Pooling & Non-Blocking Database Queries — Purpose: Run fully asynchronous PostgreSQL transactions under Uvicorn/Gunicorn workers without blocking I/O threads. • ✓ OAuth2 Scopes, JWT Middleware, BackgroundTasks, Server-Sent Events (SSE) & Bidirectional WebSockets — Purpose: Stream live AI tokens, notifications, and collaborative updates to thousands of connected clients. • ✓ Async Integration Testing with httpx.AsyncClient, Dependency Overrides & Load Benchmarking (Locust/k6) — Purpose: Test async endpoints deterministically and benchmark RPS and p99 tail latency before production rollout.

**11.1 ASGI vs WSGI Internals, FastAPI Routing, Pydantic v2 Rust-Core Schemas & Dependency Injection** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for ASGI vs WSGI Internals • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply ASGI vs WSGI Internals, FastAPI Routing, Pydantic v2 Rust-Core Schemas & Dependency Injection in real Python Full Stack Development & Enterprise G...

**What Topic Is Next → Next Topic 11.2: Async SQLAlchemy 2.0 Sessions, Alembic Migrations, Connection Pooling & Non-Blocking Database Queries**

**11.2 Async SQLAlchemy 2.0 Sessions, Alembic Migrations, Connection Pooling & Non-Blocking Database Queries** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Async SQLAlchemy 2.0 Sessi... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Async SQLAlchemy 2.0 Sessions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Async SQLAlchemy 2.0 Sessions, Alembic Migrations, Connection Pooling & Non-Blocking Database Queries in real Python Full Stack Development & Ente...

**What Topic Is Next → Next Topic 11.3: OAuth2 Scopes, JWT Middleware, BackgroundTasks, Server-Sent Events (SSE) & Bidirectional WebSockets**

**11.3 OAuth2 Scopes, JWT Middleware, BackgroundTasks, Server-Sent Events (SSE) & Bidirectional WebSockets** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for OAuth2 Scopes • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply OAuth2 Scopes, JWT Middleware, BackgroundTasks, Server-Sent Events (SSE) & Bidirectional WebSockets in real Python Full Stack Development & Enterp...

**What Topic Is Next → Next Topic 11.4: Async Integration Testing with httpx.AsyncClient, Dependency Overrides & Load Benchmarking (Locust/k6)**

**11.4 Async Integration Testing with httpx.AsyncClient, Dependency Overrides & Load Benchmarking (Locust/k6)** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Async Integration Testing... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Async Integration Testing with httpx.AsyncClient • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Async Integration Testing with httpx.AsyncClient, Dependency Overrides & Load Benchmarking (Locust/k6) in real Python Full Stack Development & Ent...

**What Topic Is Next → Module 11 Capstone (omnidoc-m11-fastapi-async-gateway) → Next Topic 12.1: Redis 7 Data Structures, Cache-Aside Invalidation, Sliding-Window Lua Rate Limiters & Distributed Locks**

**Module 11 Hands-On Capstone Project (100 Marks):** omnidoc-m11-fastapi-async-gateway — Architect, implement, test, and push the production GitHub repository 'omnidoc-m11-fastapi-async-gateway' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 11: Hig...

**MODULE 12 OF 22 • STAGE 2 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Distributed Caching (Redis 7), Celery Task Queues & Event-Driven Workflows

Beginner-to-expert real-repository module covering Distributed Caching (Redis 7), Celery Task Queues & Event-Driven Workflows with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ Redis 7 Data Structures, Cache-Aside Invalidation, Sliding-Window Lua Rate Limiters & Distributed Locks — Purpose: Protect Python APIs from traffic bursts, cache hot queries in sub-millisecond RAM, and prevent race conditions. • ✓ Celery Distributed Workers, Task Routing, Priority Queues, Canvas Workflows (Chains, Groups, Chords) — Purpose: Offload heavy PDF OCR, email delivery, report exports, and embedding jobs out of HTTP request cycles. • ✓ Idempotent Background Tasks, Exponential Backoff Retries, Dead-Letter Queues & Celery Beat Schedulers — Purpose: Guarantee safe task retries without duplicate billing or double notifications when external APIs flake. • ✓ Flower Queue Monitoring, Webhook Signature Verification (HMAC) & Transactional Outbox Publishing — Purpose: Observe worker health in real time and integrate reliably with Razorpay/Stripe/GitHub webhooks.

**12.1 Redis 7 Data Structures, Cache-Aside Invalidation, Sliding-Window Lua Rate Limiters & Distributed Locks** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Redis 7 Data Structures • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Redis 7 Data Structures, Cache-Aside Invalidation, Sliding-Window Lua Rate Limiters & Distributed Locks in real Python Full Stack Development & En...

**What Topic Is Next → Next Topic 12.2: Celery Distributed Workers, Task Routing, Priority Queues, Canvas Workflows (Chains, Groups, Chords)**

**12.2 Celery Distributed Workers, Task Routing, Priority Queues, Canvas Workflows (Chains, Groups, Chords)** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Celery Distributed Workers • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Celery Distributed Workers, Task Routing, Priority Queues, Canvas Workflows (Chains, Groups, Chords) in real Python Full Stack Development & Enter...

**What Topic Is Next → Next Topic 12.3: Idempotent Background Tasks, Exponential Backoff Retries, Dead-Letter Queues & Celery Beat Schedulers**

**12.3 Idempotent Background Tasks, Exponential Backoff Retries, Dead-Letter Queues & Celery Beat Schedulers** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Idempotent Background Task... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Idempotent Background Tasks • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Idempotent Background Tasks, Exponential Backoff Retries, Dead-Letter Queues & Celery Beat Schedulers in real Python Full Stack Development & Ente...

**What Topic Is Next → Next Topic 12.4: Flower Queue Monitoring, Webhook Signature Verification (HMAC) & Transactional Outbox Publishing**

**12.4 Flower Queue Monitoring, Webhook Signature Verification (HMAC) & Transactional Outbox Publishing** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Flower Queue Monitoring • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Flower Queue Monitoring, Webhook Signature Verification (HMAC) & Transactional Outbox Publishing in real Python Full Stack Development & Enterpris...

**What Topic Is Next → Module 12 Capstone (omnidoc-m12-redis-celery-workers) → Next Topic 13.1: How Transformers Work: Tokenization, Embeddings, Attention Intuition, Context Windows & Sampling Params**

**Module 12 Hands-On Capstone Project (100 Marks):** omnidoc-m12-redis-celery-workers — Architect, implement, test, and push the production GitHub repository 'omnidoc-m12-redis-celery-workers' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 12: Dist...

**MODULE 13 OF 22 • STAGE 3 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## LLM Foundations, Prompt Engineering, Structured Outputs & Multimodal AI APIs

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

**Objectives:** ✓ How Transformers Work: Tokenization, Embeddings, Attention Intuition, Context Windows & Sampling Params — Purpose: Grasp temperature, top-p, token economics, and latency trade-offs clearly without black-box guesswork. • ✓ Systematic Prompt Engineering: System Personas, Few-Shot Grounding, Chain-of-Thought & Delimiter Guards — Purpose: Design deterministic, injection-resistant prompts for classification, extraction, and enterprise reasoning. • ✓ Guaranteed Schema-Constrained JSON & Pydantic Structured Outputs with Retry Validation Loops — Purpose: Turn free-form LLM completions into 100% type-validated Python objects ready for database persistence. • ✓ Multimodal Vision / PDF / Audio Pipelines, Batch Inference & Semantic Response Caching in Redis — Purpose: Extract structured tables from scanned invoices/contracts and cut enterprise token bills by 40–60%.

### 13.1 How Transformers Work: Tokenization, Embeddings, Attention Intuition, Context Windows & Sampling Params [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for How Transformers Work • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply How Transformers Work: Tokenization, Embeddings, Attention Intuition, Context Windows & Sampling Params in real Python Full Stack Development & En...

**What Topic Is Next → Next Topic 13.2: Systematic Prompt Engineering: System Personas, Few-Shot Grounding, Chain-of-Thought & Delimiter Guards**

### 13.2 Systematic Prompt Engineering: System Personas, Few-Shot Grounding, Chain-of-Thought & Delimiter Guards [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Systematic Prompt Engineer... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Systematic Prompt Engineering • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Systematic Prompt Engineering: System Personas, Few-Shot Grounding, Chain-of-Thought & Delimiter Guards in real Python Full Stack Development & En...

**What Topic Is Next → Next Topic 13.3: Guaranteed Schema-Constrained JSON & Pydantic Structured Outputs with Retry Validation Loops**

### 13.3 Guaranteed Schema-Constrained JSON & Pydantic Structured Outputs with Retry Validation Loops [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Guaranteed Schema • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Guaranteed Schema-Constrained JSON • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Guaranteed Schema-Constrained JSON & Pydantic Structured Outputs with Retry Validation Loops in real Python Full Stack Development & Enterprise Ge...

**What Topic Is Next → Next Topic 13.4: Multimodal Vision / PDF / Audio Pipelines, Batch Inference & Semantic Response Caching in Redis**

### 13.4 Multimodal Vision / PDF / Audio Pipelines, Batch Inference & Semantic Response Caching in Redis [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Multimodal Vision / PDF /... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Multimodal Vision / PDF / Audio Pipelines • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Multimodal Vision / PDF / Audio Pipelines, Batch Inference & Semantic Response Caching in Redis in real Python Full Stack Development & Enterprise...

**What Topic Is Next → Module 13 Capstone (omnidoc-m13-llm-structured-pipelines) → Next Topic 14.1: Document Ingestion Pipelines: Layout-Aware PDF/Markdown Parsing, Semantic Chunking & Metadata Tagging**

**Module 13 Hands-On Capstone Project (100 Marks):** omnidoc-m13-llm-structured-pipelines — Architect, implement, test, and push the production GitHub repository 'omnidoc-m13-llm-structured-pipelines' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 13:...

**MODULE 14 OF 22 • STAGE 3 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Enterprise RAG Architecture, Vector Databases, Hybrid Search & RAGAS Evaluation

Beginner-to-expert real-repository module covering Enterprise RAG Architecture, Vector Databases, Hybrid Search & RAGAS Evaluation with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ Document Ingestion Pipelines: Layout-Aware PDF/Markdown Parsing, Semantic Chunking & Metadata Tagging — Purpose: Preserve tables, headings, and access-control metadata so retrieval chunks never lose critical context. • ✓ Dense Embeddings, PostgreSQL pgvector & Qdrant Indexing, Cosine/Dot Similarity & Metadata Pre-Filtering — Purpose: Enforce strict per-tenant RBAC filters during vector search so User A never retrieves User B's

documents. • ✓ Hybrid Search (BM25 Lexical + Dense Vector RRF), Query Rewriting / HyDE & Cross-Encoder Reranking — Purpose: Boost enterprise retrieval recall above 95% on exact SKU codes, legal clauses, and domain jargon. • ✓ Citation Verification, Hallucination Refusal Guards & Automated RAGAS CI Regression Benchmarks — Purpose: Measure faithfulness, context precision, and answer relevancy automatically on every prompt or chunk change.

**14.1 Document Ingestion Pipelines: Layout-Aware PDF/Markdown Parsing, Semantic Chunking & Metadata Tagging** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Document Ingestion Pipelin... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Document Ingestion Pipelines • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Document Ingestion Pipelines: Layout-Aware PDF/Markdown Parsing, Semantic Chunking & Metadata Tagging in real Python Full Stack Development & Ente...

**What Topic Is Next → Next Topic 14.2: Dense Embeddings, PostgreSQL pgvector & Qdrant Indexing, Cosine/Dot Similarity & Metadata Pre-Filtering**

**14.2 Dense Embeddings, PostgreSQL pgvector & Qdrant Indexing, Cosine/Dot Similarity & Metadata Pre-Filtering** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Dense Embeddings • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Dense Embeddings, PostgreSQL pgvector & Qdrant Indexing, Cosine/Dot Similarity & Metadata Pre-Filtering in real Python Full Stack Development & En...

**What Topic Is Next → Next Topic 14.3: Hybrid Search (BM25 Lexical + Dense Vector RRF), Query Rewriting / HyDE & Cross-Encoder Reranking**

**14.3 Hybrid Search (BM25 Lexical + Dense Vector RRF), Query Rewriting / HyDE & Cross-Encoder Reranking** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Hybrid Search (BM25 Lexica... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Hybrid Search (BM25 Lexical + Dense Vector RRF) • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Hybrid Search (BM25 Lexical + Dense Vector RRF), Query Rewriting / HyDE & Cross-Encoder Reranking in real Python Full Stack Development & Enterpri...

**What Topic Is Next → Next Topic 14.4: Citation Verification, Hallucination Refusal Guards & Automated RAGAS CI Regression Benchmarks**

**14.4 Citation Verification, Hallucination Refusal Guards & Automated RAGAS CI Regression Benchmarks** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Citation Verification • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Citation Verification, Hallucination Refusal Guards & Automated RAGAS CI Regression Benchmarks in real Python Full Stack Development & Enterprise...

**What Topic Is Next → Module 14 Capstone (omnidoc-m14-hybrid-rag-evals) → Next Topic 15.1: Tool-Calling Mechanics, ReAct Reasoning Loops & Safe Sandboxed Python Tool Execution**

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

**MODULE 15 OF 22 • STAGE 3 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Agentic AI Systems with LangChain, LangGraph Stateful Workflows & MCP Tools

Beginner-to-expert real-repository module covering Agentic AI Systems with LangChain, LangGraph Stateful Workflows & MCP Tools with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ Tool-Calling Mechanics, ReAct Reasoning Loops & Safe Sandboxed Python Tool Execution — Purpose: Equip AI agents to query SQL databases, search APIs, and calculate figures with strict schema validation. • ✓ LangGraph Stateful Graphs: Nodes, Conditional Edges, Cyclic Reflection Loops & Postgres Checkpointers — Purpose: Build durable, resumable multi-step AI workflows that survive server restarts without losing conversation state. • ✓ Supervisor & Specialist Multi-Agent Teams, Human-in-the-Loop (HITL) Approval Gates & Memory Tiers — Purpose: Pause high-stakes refund or database-write actions automatically until an authorized human clicks approve. • ✓ Model Context Protocol (MCP) Server Authoring, Prompt-Injection Firewalls & Langfuse Trace Telemetry — Purpose: Expose standardized enterprise MCP tools and inspect token cost and latency per agent node in production.

**15.1 Tool-Calling Mechanics, ReAct Reasoning Loops & Safe Sandboxed Python Tool Execution** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Tool • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Tool-Calling Mechanics • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Tool-Calling Mechanics, ReAct Reasoning Loops & Safe Sandboxed Python Tool Execution in real Python Full Stack Development & Enterprise Generative...

**What Topic Is Next → Next Topic 15.2: LangGraph Stateful Graphs: Nodes, Conditional Edges, Cyclic Reflection Loops & Postgres Checkpointers**

**15.2 LangGraph Stateful Graphs: Nodes, Conditional Edges, Cyclic Reflection Loops & Postgres Checkpointers** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for LangGraph Stateful Graphs • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply LangGraph Stateful Graphs: Nodes, Conditional Edges, Cyclic Reflection Loops & Postgres Checkpointers in real Python Full Stack Development & Ente...

**What Topic Is Next → Next Topic 15.3: Supervisor & Specialist Multi-Agent Teams, Human-in-the-Loop (HITL) Approval Gates & Memory Tiers**

**15.3 Supervisor & Specialist Multi-Agent Teams, Human-in-the-Loop (HITL) Approval Gates & Memory Tiers** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Supervisor • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Supervisor & Specialist Multi-Agent Teams, Human-in-the-Loop (HITL) Approval Gates & Memory Tiers in real Python Full Stack Development & Enterpri...

**What Topic Is Next → Next Topic 15.4: Model Context Protocol (MCP) Server Authoring, Prompt-Injection Firewalls & Langfuse Trace Telemetry**

**15.4 Model Context Protocol (MCP) Server Authoring, Prompt-Injection Firewalls & Langfuse Trace Telemetry** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Model Context Protocol (MC... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Model Context Protocol (MCP) Server Authoring • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Model Context Protocol (MCP) Server Authoring, Prompt-Injection Firewalls & Langfuse Trace Telemetry in real Python Full Stack Development & Enter...

**What Topic Is Next → Module 15 Capstone (omnidoc-m15-langgraph-agentic-ai) → Next Topic 16.1: Containers From Scratch: Multi-Stage Python 3.12 Slim Dockerfiles, uv Layer Caching & Non-Root Hardening**

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

**MODULE 16 OF 22 • STAGE 3 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Docker Containerization & Kubernetes Production Orchestration for Python & AI

Beginner-to-expert real-repository module covering Docker Containerization & Kubernetes Production Orchestration for Python & AI with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ Containers From Scratch: Multi-Stage Python 3.12 Slim Dockerfiles, uv Layer Caching & Non-Root Hardening — Purpose: Shrink Python/AI container images by 70% and eliminate build-tool vulnerabilities in production images. • ✓ Docker Compose Multi-Service Stacks: Django, FastAPI, Celery, React, PostgreSQL pgvector & Redis — Purpose: Run and debug the complete full-stack + AI microservice topology locally with hot-reload and healthchecks. • ✓ Kubernetes Core: Pods, Deployments, Services, ConfigMaps, Secrets & Gunicorn/Uvicorn Graceful Shutdown — Purpose: Deploy stateless API pods and background Celery workers onto Kubernetes with zero dropped requests. • ✓ Kubernetes Ingress TLS, Readiness/Liveness Probes, HPA Autoscaling (CPU + Queue Lag) & Helm Packaging — Purpose: Autoscale FastAPI and Celery replicas dynamically when user traffic or document-ingestion queues surge.

**16.1 Containers From Scratch: Multi-Stage Python 3.12 Slim Dockerfiles, uv Layer Caching & Non-Root Hardening** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Containers From Scratch • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Containers From Scratch: Multi-Stage Python 3.12 Slim Dockerfiles, uv Layer Caching & Non-Root Hardening in real Python Full Stack Development & E...

**What Topic Is Next → Next Topic 16.2: Docker Compose Multi-Service Stacks: Django, FastAPI, Celery, React, PostgreSQL pgvector & Redis**

**16.2 Docker Compose Multi-Service Stacks: Django, FastAPI, Celery, React, PostgreSQL pgvector & Redis** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Docker Compose Multi • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Docker Compose Multi-Service Stacks • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Docker Compose Multi-Service Stacks: Django, FastAPI, Celery, React, PostgreSQL pgvector & Redis in real Python Full Stack Development & Enterpris...

**What Topic Is Next → Next Topic 16.3: Kubernetes Core: Pods, Deployments, Services, ConfigMaps, Secrets & Gunicorn/Uvicorn Graceful Shutdown**

**16.3 Kubernetes Core: Pods, Deployments, Services, ConfigMaps, Secrets & Gunicorn/Uvicorn Graceful Shutdown** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Kubernetes Core • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Kubernetes Core: Pods, Deployments, Services, ConfigMaps, Secrets & Gunicorn/Uvicorn Graceful Shutdown in real Python Full Stack Development & Ent...

**What Topic Is Next → Next Topic 16.4: Kubernetes Ingress TLS, Readiness/Liveness Probes, HPA Autoscaling (CPU + Queue Lag) & Helm Packaging**

**16.4 Kubernetes Ingress TLS, Readiness/Liveness Probes, HPA Autoscaling (CPU + Queue Lag) & Helm Packaging** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Kubernetes Ingress TLS • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Kubernetes Ingress TLS, Readiness/Liveness Probes, HPA Autoscaling (CPU + Queue Lag) & Helm Packaging in real Python Full Stack Development & Ente...

**What Topic Is Next → Module 16 Capstone (omnidoc-m16-docker-kubernetes-prod) → Next Topic 17.1: AWS Foundations: Regions/AZs, IAM Roles & Policies, KMS Encryption, Secrets Manager & Billing Guardrails**

**Module 16 Hands-On Capstone Project (100 Marks):** omnidoc-m16-docker-kubernetes-prod — Architect, implement, test, and push the production GitHub repository 'omnidoc-m16-docker-kubernetes-prod' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 16: Do...

**MODULE 17 OF 22 • STAGE 3 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## AWS Cloud Infrastructure, Amazon Bedrock, Terraform IaC & Cloud Security

Beginner-to-expert real-repository module covering AWS Cloud Infrastructure, Amazon Bedrock, Terraform IaC & Cloud Security with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ AWS Foundations: Regions/AZs, IAM Roles & Policies, KMS Encryption, Secrets Manager & Billing Guardrails — Purpose: Operate AWS securely with zero long-lived static credentials and encrypted secrets injection. • ✓ Multi-AZ AWS VPC Networking, EC2, Application Load Balancer, RDS PostgreSQL Multi-AZ & ElastiCache Redis — Purpose: Architect private-subnet database and cache tiers behind HTTPS load balancers and Web Application Firewalls. • ✓ Amazon S3 Presigned Uploads, CloudFront Edge CDN, SQS Event Queues & Amazon Bedrock Enterprise LLMs — Purpose: Process large customer files asynchronously via S3->SQS and invoke VPC-private foundation models on Bedrock. • ✓ Terraform Infrastructure as Code (IaC): Modular HCL, Remote State Locking & Automated Cloud Provisioning — Purpose: Provision repeatable staging and production AWS environments cleanly through peer-reviewed Terraform PRs.

**17.1 AWS Foundations: Regions/AZs, IAM Roles & Policies, KMS Encryption, Secrets Manager & Billing Guardrails** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for AWS Foundations • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply AWS Foundations: Regions/AZs, IAM Roles & Policies, KMS Encryption, Secrets Manager & Billing Guardrails in real Python Full Stack Development & E...

**What Topic Is Next → Next Topic 17.2: Multi-AZ AWS VPC Networking, EC2, Application Load Balancer, RDS PostgreSQL Multi-AZ & ElastiCache Redis**

**17.2 Multi-AZ AWS VPC Networking, EC2, Application Load Balancer, RDS PostgreSQL Multi-AZ & ElastiCache Redis** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Multi • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Multi-AZ AWS VPC Networking • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Multi-AZ AWS VPC Networking, EC2, Application Load Balancer, RDS PostgreSQL Multi-AZ & ElastiCache Redis in real Python Full Stack Development & E...

**What Topic Is Next → Next Topic 17.3: Amazon S3 Presigned Uploads, CloudFront Edge CDN, SQS Event Queues & Amazon Bedrock Enterprise LLMs**

**17.3 Amazon S3 Presigned Uploads, CloudFront Edge CDN, SQS Event Queues & Amazon Bedrock Enterprise LLMs** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Amazon S3 Presigned Upload... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Amazon S3 Presigned Uploads • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Amazon S3 Presigned Uploads, CloudFront Edge CDN, SQS Event Queues & Amazon Bedrock Enterprise LLMs in real Python Full Stack Development & Enterp...

**What Topic Is Next → Next Topic 17.4: Terraform Infrastructure as Code (IaC): Modular HCL, Remote State Locking & Automated Cloud Provisioning**

**17.4 Terraform Infrastructure as Code (IaC): Modular HCL, Remote State Locking & Automated Cloud Provisioning** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Terraform Infrastructure a... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Terraform Infrastructure as Code (IaC) • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Terraform Infrastructure as Code (IaC): Modular HCL, Remote State Locking & Automated Cloud Provisioning in real Python Full Stack Development & E...

**What Topic Is Next → Module 17 Capstone (omnidoc-m17-aws-bedrock-terraform) → Next Topic 18.1: GitHub Actions CI/CD: Ruff Linting, Mypy, Pytest + Postgres Services, Bandit SAST, Trivy & Cloud Deploy**

**Module 17 Hands-On Capstone Project (100 Marks):** omnidoc-m17-aws-bedrock-terraform — Architect, implement, test, and push the production GitHub repository 'omnidoc-m17-aws-bedrock-terraform' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 17: AWS...

**MODULE 18 OF 22 • STAGE 4 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## DevSecOps CI/CD Pipelines, OpenTelemetry Observability & Production Hardening

Beginner-to-expert real-repository module covering DevSecOps CI/CD Pipelines, OpenTelemetry Observability & Production Hardening with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ GitHub Actions CI/CD: Ruff Linting, Mypy, Pytest + Postgres Services, Bandit SAST, Trivy & Cloud Deploy — Purpose: Automate code quality, security scanning, database migration checks, and zero-downtime container releases. • ✓ OpenTelemetry Distributed Tracing Across React, Django, FastAPI, Celery, PostgreSQL & LLM Calls — Purpose: Pinpoint exact latency bottlenecks across web APIs, async workers, and external AI model invocations. • ✓ Prometheus Metrics, Grafana Dashboards, Centralized JSON Logs & Sentry Real-Time Error Triage — Purpose: Track golden signals (RPS, error rate, p95/p99 latency, token burn) with actionable on-call alert thresholds. • ✓ Zero-Downtime Database Rollouts (Expand-Contract), Backup/PITR Drills & FinOps Cloud/Token Cost Control — Purpose: Evolve live schemas safely and keep AWS + LLM inference unit economics profitable at scale.

**18.1 GitHub Actions CI/CD: Ruff Linting, Mypy, Pytest + Postgres Services, Bandit SAST, Trivy & Cloud Deploy** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for GitHub Actions CI/CD • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply GitHub Actions CI/CD: Ruff Linting, Mypy, Pytest + Postgres Services, Bandit SAST, Trivy & Cloud Deploy in real Python Full Stack Development & En...

**What Topic Is Next → Next Topic 18.2: OpenTelemetry Distributed Tracing Across React, Django, FastAPI, Celery, PostgreSQL & LLM Calls**

**18.2 OpenTelemetry Distributed Tracing Across React, Django, FastAPI, Celery, PostgreSQL & LLM Calls** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for OpenTelemetry Distributed... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for OpenTelemetry Distributed Tracing Across React • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply OpenTelemetry Distributed Tracing Across React, Django, FastAPI, Celery, PostgreSQL & LLM Calls in real Python Full Stack Development & Enterprise...

**What Topic Is Next → Next Topic 18.3: Prometheus Metrics, Grafana Dashboards, Centralized JSON Logs & Sentry Real-Time Error Triage**

**18.3 Prometheus Metrics, Grafana Dashboards, Centralized JSON Logs & Sentry Real-Time Error Triage** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Prometheus Metrics • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Prometheus Metrics, Grafana Dashboards, Centralized JSON Logs & Sentry Real-Time Error Triage in real Python Full Stack Development & Enterprise G...

**What Topic Is Next → Next Topic 18.4: Zero-Downtime Database Rollouts (Expand-Contract), Backup/PITR Drills & FinOps Cloud/Token Cost Control**

**18.4 Zero-Downtime Database Rollouts (Expand-Contract), Backup/PITR Drills & FinOps Cloud/Token Cost Control** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Zero • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Zero-Downtime Database Rollouts (Expand-Contract) • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Zero-Downtime Database Rollouts (Expand-Contract), Backup/PITR Drills & FinOps Cloud/Token Cost Control in real Python Full Stack Development & En...

**What Topic Is Next → Module 18 Capstone (omnidoc-m18-devsecops-observability) → Next Topic 19.1: Low-Level Design (LLD) & Machine Coding in Python: Rate Limiter, Job Scheduler, Notification Hub & Splitwise**

**Module 18 Hands-On Capstone Project (100 Marks):** omnidoc-m18-devsecops-observability — Architect, implement, test, and push the production GitHub repository 'omnidoc-m18-devsecops-observability' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 18: D...

**MODULE 19 OF 22 • STAGE 4 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## System Design (LLD + HLD) for Full-Stack Python & Generative AI Interviews

Beginner-to-expert real-repository module covering System Design (LLD + HLD) for Full-Stack Python & Generative AI Interviews with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ Low-Level Design (LLD) & Machine Coding in Python: Rate Limiter, Job Scheduler, Notification Hub & Splitwise — Purpose: Write clean, extensible, thread-safe object-oriented Python systems under 90-minute machine-coding constraints. • ✓ High-Level System Design (HLD) Fundamentals: Load Balancers, Caching, Sharding, Queues & CAP Trade-Offs — Purpose: Design URL shorteners, payment webhooks, real-time chat, and multi-tenant SaaS architectures from scratch. • ✓ GenAI System Design: Enterprise Multi-Tenant RAG Search, Copilot Gateways & Guardrailed Agent Platforms — Purpose: Architect low-latency, hallucination-resistant AI systems handling 50M+ vector chunks and strict RBAC. • ✓ Debugging Production Incidents, Architecture Trade-Off Defense & Mock System Design Whiteboard Rounds — Purpose: Communicate capacity estimates, schema choices, and failure recovery with senior-engineer clarity.

### 19.1 Low-Level Design (LLD) & Machine Coding in Python: Rate Limiter, Job Scheduler, Notification Hub & Splitwise [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Low • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Low-Level Design (LLD) • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Low-Level Design (LLD) & Machine Coding in Python: Rate Limiter, Job Scheduler, Notification Hub & Splitwise in real Python Full Stack Development...

**What Topic Is Next → Next Topic 19.2: High-Level System Design (HLD) Fundamentals: Load Balancers, Caching, Sharding, Queues & CAP Trade-Offs**

### 19.2 High-Level System Design (HLD) Fundamentals: Load Balancers, Caching, Sharding, Queues & CAP Trade-Offs [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for High • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for High-Level System Design (HLD) Fundamentals • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply High-Level System Design (HLD) Fundamentals: Load Balancers, Caching, Sharding, Queues & CAP Trade-Offs in real Python Full Stack Development & En...

**What Topic Is Next → Next Topic 19.3: GenAI System Design: Enterprise Multi-Tenant RAG Search, Copilot Gateways & Guardrailed Agent Platforms**

### 19.3 GenAI System Design: Enterprise Multi-Tenant RAG Search, Copilot Gateways & Guardrailed Agent Platforms [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for GenAI System Design • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply GenAI System Design: Enterprise Multi-Tenant RAG Search, Copilot Gateways & Guardrailed Agent Platforms in real Python Full Stack Development & En...

**What Topic Is Next → Next Topic 19.4: Debugging Production Incidents, Architecture Trade-Off Defense & Mock System Design Whiteboard Rounds**

### 19.4 Debugging Production Incidents, Architecture Trade-Off Defense & Mock System Design Whiteboard Rounds [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Debugging Production Incid... • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Debugging Production Incidents • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Debugging Production Incidents, Architecture Trade-Off Defense & Mock System Design Whiteboard Rounds in real Python Full Stack Development & Ente...

**What Topic Is Next → Module 19 Capstone (omnidoc-m19-python-genai-system-design) → Next Topic 20.1: End-to-End Delivery of OmniDoc Multi-Tenant Django + FastAPI + LangGraph + React 19 SaaS Platform**

**Module 19 Hands-On Capstone Project (100 Marks):** omnidoc-m19-python-genai-system-design — Architect, implement, test, and push the production GitHub repository 'omnidoc-m19-python-genai-system-design' (strictly zero dummy scaffolding) demonstrating end-to-end mastery of Module 19...

**MODULE 20 OF 22 • STAGE 4 • 24 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## OmniDoc Live Client Production Cutover, Portfolio Defense & ≥90% Placement Sprint

Beginner-to-expert real-repository module covering OmniDoc Live Client Production Cutover, Portfolio Defense & ≥90% Placement Sprint with supervised live coding, ≥90% MockAttempt AI Viva Gates, and production GitHub PR verification.

**Objectives:** ✓ End-to-End Delivery of OmniDoc Multi-Tenant Django + FastAPI + LangGraph + React 19 SaaS Platform — Purpose: Unite multi-tenant RBAC, hybrid pgvector RAG, stateful LangGraph agents, and streaming UIs in one flagship system. • ✓ RAGAS Accuracy Gate, OWASP Security Audit, Load Stress Test & Live AWS HTTPS Production Cutover — Purpose: Validate >95% citation faithfulness, sub-second API responsiveness, and production TLS hardening. • ✓ ATS-

Optimized Python + GenAI Resume, 20-Repository GitHub Portfolio & Live Product Demo Pitch — Purpose: Showcase verifiable commits, CI badges, and live cloud deployments that convert recruiter screens into interviews. • ✓ Timed Coding Rounds, Spoken MockAttempt AI Bar-Raiser Vivas & ≥90% Placement Guarantee Activation — Purpose: Clear every ≥90% mastery gate (unlimited free re-attempts included) to unlock 100 curated IT job openings in 6 months.

**20.1 End-to-End Delivery of OmniDoc Multi-Tenant Django + FastAPI + LangGraph + React 19 SaaS Platform** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for End • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for End-to-End Delivery of OmniDoc Multi-Tenant Django +... • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply End-to-End Delivery of OmniDoc Multi-Tenant Django + FastAPI + LangGraph + React 19 SaaS Platform in real Python Full Stack Development & Enterpri...

**What Topic Is Next → Next Topic 20.2: RAGAS Accuracy Gate, OWASP Security Audit, Load Stress Test & Live AWS HTTPS Production Cutover**

**20.2 RAGAS Accuracy Gate, OWASP Security Audit, Load Stress Test & Live AWS HTTPS Production Cutover** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for RAGAS Accuracy Gate • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply RAGAS Accuracy Gate, OWASP Security Audit, Load Stress Test & Live AWS HTTPS Production Cutover in real Python Full Stack Development & Enterprise...

**What Topic Is Next → Next Topic 20.3: ATS-Optimized Python + GenAI Resume, 20-Repository GitHub Portfolio & Live Product Demo Pitch**

**20.3 ATS-Optimized Python + GenAI Resume, 20-Repository GitHub Portfolio & Live Product Demo Pitch** [90 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Input Validation, Edge • Modular Clean • Idiomatic PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for ATS • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for ATS-Optimized Python + GenAI Resume • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply ATS-Optimized Python + GenAI Resume, 20-Repository GitHub Portfolio & Live Product Demo Pitch in real Python Full Stack Development & Enterprise G...

**What Topic Is Next → Next Topic 20.4: Timed Coding Rounds, Spoken MockAttempt AI Bar-Raiser Vivas & ≥90% Placement Guarantee Activation**

**20.4 Timed Coding Rounds, 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 PFSC Implementation • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Timed Coding Rounds • Runnable Code Implementation (PYTHON) • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards • Diagnostics, Profiling & Telemetry: django-debug-toolbar, EXPLAIN ANALYZE & Flower/Prometheus

**Purpose:** To understand and apply Timed Coding Rounds, Spoken MockAttempt AI Bar-Raiser Vivas & ≥90% Placement Guarantee Activation in real Python Full Stack Development & Enterpri...

**What Topic Is Next → Module 20 Capstone (omnidoc-m20-live-client-production-capstone) → Next Topic 21.1: LangGraph Stateful Agent Workflows, Human-in-the-Loop Checkpoints & Tool Sandboxing**

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

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

## Enterprise LangGraph Agents, Hybrid pgvector RAG & Async FastAPI/Django Scale

Self-explanatory list-and-bullet production mastery module covering Enterprise LangGraph Agents, Hybrid pgvector RAG & Async FastAPI/Django Scale in Python Full Stack Development & Enterprise Generative AI Architecture, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

**Objectives:** ✓ LangGraph Stateful Agent Workflows, Human-in-the-Loop Checkpoints & Tool Sandboxing — Purpose: Build controllable, auditable enterprise AI agents instead of fragile single-prompt loops. • ✓ Hybrid Search (BM25 + pgvector), Re-Ranking, Guardrails & Automated RAG Evaluation — Purpose: Verify citation accuracy, block prompt injection, and benchmark retrieval precision in CI. • ✓ High-Throughput Django 5 + Async FastAPI + Celery + Redis Production Integration — Purpose: Combine Django admin/ORM velocity with sub-30ms async FastAPI streaming endpoints. • ✓ Build, verify, and document 'Stateful LangGraph Multi-Agent Workflow + Grounded Citation API Repository' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

**21.1 LangGraph Stateful Agent Workflows, Human-in-the-Loop Checkpoints & Tool Sandboxing** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Security Hygiene & Data Safety Rules for LangGraph Stateful Agent Workflows, Human • Performance Optimization, Complexity & Latency Impact of LangGraph Stateful Agent Workflows, Human • Live Debugging, Stack • Runnable Code Implementation (PYTHON) • Hands • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for LangGraph Stateful Agent Workflows • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply LangGraph Stateful Agent Workflows, Human-in-the-Loop Checkpoints & Tool Sandboxing in real Python Full Stack Development & Enterprise Generative...

**What Topic Is Next → Next Topic 21.2: Hybrid Search (BM25 + pgvector), Re-Ranking, Guardrails & Automated RAG Evaluation**

**21.2 Hybrid Search (BM25 + pgvector), Re-Ranking, Guardrails & Automated RAG Evaluation** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Security Hygiene & Data Safety Rules for Hybrid Search (BM25 + pgvector), Re • Performance Optimization, Complexity & Latency Impact of Hybrid Search (BM25 + pgvector), Re • Live Debugging, Stack • Runnable Code Implementation (PYTHON) • Hands • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Hybrid Search (BM25 + pgvector) • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Hybrid Search (BM25 + pgvector), Re-Ranking, Guardrails & Automated RAG Evaluation in real Python Full Stack Development & Enterprise Generative A...

**What Topic Is Next → Next Topic 21.3: High-Throughput Django 5 + Async FastAPI + Celery + Redis Production Integration**

**21.3 High-Throughput Django 5 + Async FastAPI + Celery + Redis Production Integration** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Security Hygiene & Data Safety Rules for High • Performance Optimization, Complexity & Latency Impact of High • Live Debugging, Stack • Runnable Code Implementation (PYTHON) • Hands • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for High-Throughput Django 5 + Async FastAPI + Celery +... • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply High-Throughput Django 5 + Async FastAPI + Celery + Redis Production Integration in real Python Full Stack Development & Enterprise Generative AI...

**What Topic Is Next → Module 21 Capstone (Stateful LangGraph Multi-Agent Workflow + Grounded Citation API Repository) → Next Topic 22.1: React 19 Streaming UI, Optimistic State & Secure JWT/Session Auth Integration**

**Module 21 Hands-On Capstone Project (100 Marks):** Stateful LangGraph Multi-Agent Workflow + Grounded Citation API Repository — Architect, implement, test, and document 'Stateful LangGraph Multi-Agent Workflow + Grounded Citation API Repository' applying every bullet-wise engineering concept from Module 21 (Enterpris...

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

## AWS Cloud MLOps, Full-Stack React 19 UX & ≥90% Placement Gate Readiness

Self-explanatory list-and-bullet production mastery module covering AWS Cloud MLOps, Full-Stack React 19 UX & ≥90% Placement Gate Readiness in Python Full Stack Development & Enterprise Generative AI Architecture, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

**Objectives:** ✓ React 19 Streaming UI, Optimistic State & Secure JWT/Session Auth Integration — Purpose: Deliver responsive, production-polished full-stack AI and SaaS user interfaces. • ✓ Docker, GitHub Actions CI/CD, AWS Cloud Deployment & LLM Cost/Latency Telemetry — Purpose: Ship containerized Python + React stacks with structured logs, alarms, and token budgets. • ✓ Python Full Stack & GenAI System Design Defense for High-CTC Placement Loops — Purpose: Master Python internals, SQL/ORM tuning, API scaling, and GenAI architecture interviews. • ✓ Build, verify, and document 'Token-Streaming React 19 + FastAPI + AWS Docker Production Stack 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 React 19 Streaming UI, Optimistic State & Secure JWT/Session Auth Integration** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Security Hygiene & Data Safety Rules for React 19 Streaming UI, Optimistic State & Secure JWT/Session Auth Integrat... • Performance Optimization, Complexity & Latency Impact of React 19 Streaming UI, Optimistic State & Secure JWT/Sessi... • Live Debugging, Stack • Runnable Code Implementation (PYTHON) • Hands • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for React 19 Streaming UI • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply React 19 Streaming UI, Optimistic State & Secure JWT/Session Auth Integration in real Python Full Stack Development & Enterprise Generative AI Arc...

**What Topic Is Next → Next Topic 22.2: Docker, GitHub Actions CI/CD, AWS Cloud Deployment & LLM Cost/Latency Telemetry**

**22.2 Docker, GitHub Actions CI/CD, AWS Cloud Deployment & LLM Cost/Latency Telemetry** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Security Hygiene & Data Safety Rules for Docker, GitHub Actions CI/CD, AWS Cloud Deployment & LLM Cost/Latency Tele... • Performance Optimization, Complexity & Latency Impact of Docker, GitHub Actions CI/CD, AWS Cloud Deployment & LLM C... • Live Debugging, Stack • Runnable Code Implementation (PYTHON) • Hands • Idiomatic PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Docker • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Docker, GitHub Actions CI/CD, AWS Cloud Deployment & LLM Cost/Latency Telemetry in real Python Full Stack Development & Enterprise Generative AI A...

**What Topic Is Next → Next Topic 22.3: Python Full Stack & GenAI System Design Defense for High-CTC Placement Loops**

**22.3 Python Full Stack & GenAI System Design Defense for High-CTC Placement Loops** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Security Hygiene & Data Safety Rules for Python Full Stack & GenAI System Design Defense for High • Performance Optimization, Complexity & Latency Impact of Python Full Stack & GenAI System Design Defense for High • Live Debugging, Stack • Runnable Code Implementation (PYTHON) • Hands • Idiomatic

PFSC Implementation: select_related/prefetch_related QuerySets, DRF ViewSets & atomic transactions • Django 5 ORM/ASGI Pipeline, PostgreSQL & Celery Workers Execution & Memory Lifecycle for Python Full Stack • Production Hardening & Security: DRF object-level permissions, SimpleJWT rotation & CSRF/SQLi guards

**Purpose:** To understand and apply Python Full Stack & GenAI System Design Defense for High-CTC Placement Loops in real Python Full Stack Development & Enterprise Generative AI Arch...

**What Topic Is Next → Module 22 Capstone → End-of-Course Live Client Project: OmniDoc Enterprise SaaS Portal, Async FastAPI Gateway & Multi-Agent LangGraph Workspace**

**Module 22 Hands-On Capstone Project (100 Marks):** Token-Streaming React 19 + FastAPI + AWS Docker Production Stack Repository — Architect, implement, test, and document 'Token-Streaming React 19 + FastAPI + AWS Docker Production Stack Repository' applying every bullet-wise engineering concept from Module 22 (AWS Clou...

## 3. LIVE CLIENT PRODUCTION PROJECT & CAREER SCOPE

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

**Project:** OmniDoc Enterprise SaaS Portal, Async FastAPI Gateway & Multi-Agent LangGraph Workspace • **Confirmed Role:** Full Stack Python, Django/FastAPI & GenAI Architect Resident

**Domain & Production Infra:** Enterprise B2B SaaS & Legal/FinTech Intelligence Client (Python 3.12 + Django 5 + FastAPI + LangGraph + pgvector + React 19 + AWS Docker)

- Architected multi-tenant Django 5 + DRF RBAC management APIs alongside sub-30ms async FastAPI inference services.
- Engineered stateful LangGraph multi-agent workflows grounded over PostgreSQL pgvector hybrid search.
- Crafted responsive React 19 + TypeScript dashboards with live SSE token streaming, Celery/Redis jobs, and citation drawers.
- Automated Pytest verification, Docker multi-stage builds, and AWS multi-AZ HTTPS cloud deployment.

**Target Job Roles:** Full Stack Python & GenAI Developer, Backend Software Engineer (Django / FastAPI), Applied AI & RAG Full Stack Engineer, Python Cloud & Platform SDE

**Realistic Annual CTC Scope:** **Placement-Guaranteed Graduate / SDE-1 (0–1 Yr):** ₹6.5 LPA – ₹14.5 LPA **Python Full Stack + GenAI SDE-2 (1–3 Yrs):** ₹15.0 LPA – ₹30.0 LPA **Staff Python & GenAI Systems Architect (3–6+ Yrs):** ₹30.0 LPA – ₹58.0+ LPA

**Optional Worthwhile Global Certifications (100% Exam-Success Mentorship Available):** **AWS Certified Machine Learning Engineer – Associate (MLA-C01):** ₹12.0 LPA – ₹28.0 LPA (+35% Verified AI Premium) • **AWS Certified Solutions Architect – Associate (SAA-C03):** ₹11.0 LPA – ₹25.0 LPA (+30% Cloud Shortlist Rate) • **Certified Kubernetes Application Developer (CKAD):** ₹15.0 LPA – ₹32.0 LPA (Hands-On Lab Credential)

**MockAttempt Academy Pvt. Ltd.** • Email: support@mockattempt.com • Phone/WhatsApp: +91 98765 43210 • Web: https://mockattempt.com/training/python-full-stack-career-program

Try 2 Free Topic-Wise Timed Topic Mock Tests & Spoken MockAttempt AI Interviews daily on every topic above before enrolling.