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: FAPI • **Updated:** 01 Nov 2026
https://mockattempt.com/courses/fastapi-training

FAPI • ASYNC PYTHON APIS & MICROSERVICES | **BEGINNER TO PRINCIPAL FASTAPI ARCHITECT**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

FastAPI Training

Master Python FastAPI from Starlette ASGI Event Loops, Path/Query/Body Schemas & Rust-powered Pydantic v2 Validations to Response Model Secret Filtering, Dependency Injection (`yield`), OAuth2 JWT Security, Middleware, CORS, SQLAlchemy 2.0 & APIRouter Microservices — with runnable Python 3.12 programs, 10×50 MCQ Mock Tests per topic, and Spoken Voice + Gemini AI Interviews at ₹3,999.

Duration: 15–17 Weeks • Live + Hands-On Projects (205 Hrs) • **Modules:** 16 • **Live Topics:** 42 • **Labs & Tasks:** 84 • **Mocks / AI Vivas:** 244 / 244 • **Projects:** 16 Module + 1 Live Client Prod

Fixed Non-Negotiable Tuition Fee: Base Fee **₹3,999** + 18% GST (₹720) = **Total Incl. GST: ₹4,719**

Live Batch Tracks: Weekday Live Online Batch (Mon – Fri) (Monday to Friday (5 Live Interactive Classes / Week) • 1.5 Hours / Day (60 Mins Live Architecture & Coding + 30 Mins Hands-On Lab) • 17–19 Weeks Fast-Track) | Weekend Intensive Live Online Batch (Sat – Sun) (Saturday & Sunday (Designed for Working Professionals & College Students) • 3.0 Hours / Day (2 Hrs Deep-Dive Live Coding + 1 Hr Guided Project Build) • 19–21 Weeks Weekend Track)

Core Technologies & Engineering Stack: Python 3.12 & FastAPI • Starlette ASGI & Uvicorn / uvloop • Pydantic v2 (pydantic-core Rust) • Typing Annotated & Dependency Injection • OAuth2 Bearer JWT & Passlib/Argon2 • SQLAlchemy 2.0 Async & Alembic • APIRouter, Pytest & HTTPX TestClient

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Architect non-blocking ASGI services on Uvicorn/Starlette understanding async def event-loop coroutines versus def worker threadpools
2. Enforce ironclad request/response contracts with Rust-accelerated Pydantic v2 models, Annotated Query/Path/Field constraints, and response_model secret stripping
3. Compose modular, testable microservices using FastAPI Depends() with yield cleanup, OAuth2PasswordBearer JWT security, CORS, and SQLAlchemy 2.0 sessions
4. Clear 200 Timed Mock Exams (10,000 FastAPI MCQs) and 200 Spoken Voice + Gemini AI Python FastAPI Interviews
5. True `async/await` request pipelines using FastAPI routers, lifespan events, and modular dependency injection (`Depends`).
6. Ultra-fast Pydantic v2 Rust-core data validation, custom validators, discriminated unions, and `pydantic-settings`.
7. Async SQLAlchemy 2.0 + `asyncpg` non-blocking database pools, Alembic migrations, and OAuth2/JWT security scopes.
8. Streaming responses (SSE/WebSockets), idempotent background tasks, and auto-generated OpenAPI 3.1 SDK contracts.

Eligibility & Prerequisites: Basic Python 3 syntax, functions, and type hints (`str`, `int`, `list`, `dict`) — covers async/await, ASGI, and Pydantic v2 from scratch • Any laptop/PC running Python 3.10+, `fastapi[standard]`, Uvicorn, and VS Code / PyCharm • No prior FastAPI industry experience required — Module 1 starts from environment setup, core mental models, and plain-English fundamentals. • Basic computer & internet familiarity on a Windows, macOS, or Linux laptop (8 GB+ RAM recommended for smooth live coding labs). • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Commitment to attend live Weekday or Weekend classes, complete daily hands-on lab checklists, and practice topic-wise Mock Tests & AI Interviews.

2. DETAILED MODULE & TOPIC-WISE SYLLABUS (16 MODULES • 42 LIVE TOPICS)

MODULE 1 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 1: ASGI Architecture, Uvicorn Event Loop & Typed Path Parameters

Boot high-concurrency FastAPI applications on Starlette/Uvicorn, generate OpenAPI 3.1 Swagger/ReDoc specs automatically, and parse typed/Enum path parameters.

Objectives: ✓ Distinguish non-blocking async def coroutines from synchronous def threadpool endpoints on the Uvicorn event loop • ✓ Inspect auto-generated OpenAPI 3.1 JSON schemas at /openapi.json, Swagger UI (/docs), and ReDoc (/redoc) • ✓ Declare deterministic route ordering and StrEnum-constrained dynamic path parameters with automatic 422 validation

1.1 FastAPI First Steps — Starlette ASGI Core, Uvicorn Event Loop & OpenAPI 3.1 Auto-Docs [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI First Steps — Starlette ASGI Core, Uvicorn Event Loop & OpenAPI 3.1 Auto-Docs in real FastAPI systems — specifically: How FastAPI unites S...

1.2 FastAPI Path Parameters — Type Coercion, Route Declaration Order, StrEnum & Path Converters [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Path Parameters — Type Coercion, Route Declaration Order, StrEnum & Path Converters in real FastAPI systems — specifically: Extracting typ...

Module 1 Hands-On Capstone Project (100 Marks): OpenAPI-Documented Service Health & Enum-Constrained Catalog Router — Create a FastAPI service exposing async health metadata and typed/Enum resource routes where static paths resolve safely before dynamic {item_id} catch-all segments.

MODULE 2 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 2: Query Parameters & Pydantic v2 Request Body Parsing

Combine required/optional query parameters with Rust-backed Pydantic v2 BaseModel JSON payloads for clean pagination, filtering, and creation endpoints.

Objectives: ✓ Distinguish Path parameters, Query parameters, and Pydantic BaseModel Request Body fields cleanly inside a single endpoint signature • ✓ Parse flexible boolean query flags, required vs default-optional filters, and multi-value query lists • ✓ Leverage Pydantic v2 model_dump() and strict JSON deserialization with zero manual dictionary key checks

2.1 FastAPI Query Parameters — Defaults, Optional Unions (`str | None`), Booleans & Required Flags [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Query Parameters — Defaults, Optional Unions (`str | None`), Booleans & Required Flags in real FastAPI systems — specifically: How functio...

2.2 FastAPI Request Body — Pydantic v2 `BaseModel`, `model_dump()` & Path + Query + Body Mixing [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Request Body — Pydantic v2 `BaseModel`, `model_dump()` & Path + Query + Body Mixing in real FastAPI systems — specifically: Validating inc...

Module 2 Hands-On Capstone Project (100 Marks): Paginated Search & Pydantic v2 Course Enrollment Endpoint — Build an API endpoint suite supporting offset/limit/boolean query filtering alongside a Pydantic v2 BaseModel POST endpoint that computes net discounted pricing.

MODULE 3 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 3: `Annotated` Metadata & Declarative Query / Path Validations

Adopt modern `typing.Annotated` syntax with `Query(...)` and `Path(...)` to enforce string length, regex patterns, multi-value lists, and numeric `ge`/`le`/`gt`/`lt` bounds.

Objectives: ✓ Adopt modern Annotated[T, Query(...)] syntax over legacy default-value = Query(...) signatures • ✓ Validate string length (min_length, max_length), regular expressions (pattern), aliases, and repeated query list values • ✓ Enforce numeric bounds (gt, ge, lt, le) on Path and Query IDs before database queries ever execute

3.1 FastAPI Query Validations — `Annotated[str, Query()]`, `min_length`, `pattern`, Lists & Aliases [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Query Validations — `Annotated[str, Query()]`, `min_length`, `pattern`, Lists & Aliases in real FastAPI systems — specifically: Enforcing...

3.2 FastAPI Path Validations — `Annotated[int, Path(gt=0, le=10000)]` & Numeric Boundary Rules [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Path Validations — `Annotated[int, Path(gt=0, le=10000)]` & Numeric Boundary Rules in real FastAPI systems — specifically: Constraining UR...

Module 3 Hands-On Capstone Project (100 Marks): Hardened Search Filter & Numeric Boundary Guard API — Build an `Annotated` catalog filter endpoint enforcing regex SKU codes, bounded page sizes (`ge=1, le=100`), aliased query keys (`item-tag`), and positive integer resource IDs (`gt=0`).

MODULE 4 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 4: Rich Pydantic v2 `Field()` Constraints & Deeply Nested JSON Models

Model complex enterprise JSON trees using `pydantic.Field()`, `ConfigDict(extra="forbid")`, `HttpUrl`, `set[str]` deduplication, and deeply nested sub-models.

Objectives: ✓ Apply pydantic.Field() constraints, default_factory, schema examples, and ConfigDict(extra="forbid") on request DTOs • ✓ Compose deeply nested BaseModel hierarchies with typed list[SubModel], set[str] unique tags, and HttpUrl validation • ✓ Use @field_validator and @model_validator(mode="after") for cross-field business invariants

4.1 FastAPI Body Fields — `pydantic.Field()`, `default_factory`, `ConfigDict` & OpenAPI Examples [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Body Fields — `pydantic.Field()`, `default_factory`, `ConfigDict` & OpenAPI Examples in real FastAPI systems — specifically: Enforcing att...

4.2 FastAPI Nested Models — Sub-Models, `list[SubModel]`, `set[str]` Deduplication & `HttpUrl` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Nested Models — Sub-Models, `list[SubModel]`, `set[str]` Deduplication & `HttpUrl` in real FastAPI systems — specifically: Modeling arbitr...

Module 4 Hands-On Capstone Project (100 Marks): Nested E-Commerce Checkout Schema with Strict Field & Cross-Field Validation — Design a nested Pydantic v2 Order payload containing CustomerAddress, list[LineItem], unique set[str] promo tags, and Field(gt=0) quantities with extra="forbid" mass-assignment protection.

MODULE 5 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 5: Cookie / Header Extraction & `response_model` Output Filtering

Extract HTTP Cookies and hyphenated Headers cleanly (including Pydantic header/cookie models) and guarantee zero password-hash leaks using `response_model` output schemas.

Objectives: ✓ Bind HTTP Headers (x_request_id -> X-Request-Id automatic underscore conversion) and Cookies with Annotated • ✓ Separate UserCreate (with password) from UserPublic (without password_hash) using strict response_model filtering • ✓ Apply response_model_exclude_unset=True to strip unpopulated optional fields and shrink JSON payload sizes

5.1 FastAPI Cookie & Header Parameters — Underscore-to-Hyphen Conversion, Duplicate Headers & Models [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Cookie & Header Parameters — Underscore-to-Hyphen Conversion, Duplicate Headers & Models in real FastAPI systems — specifically: Extractin...

5.2 FastAPI `response_model` — Secret Stripping, Input/Output DTO Separation & `exclude_unset` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI `response_model` — Secret Stripping, Input/Output DTO Separation & `exclude_unset` in real FastAPI systems — specifically: Using `@app.pos...

Module 5 Hands-On Capstone Project (100 Marks): Header-Traced User Registration API with Zero-Leak `response_model` Separation — Implement a registration endpoint that reads X-Correlation-Id headers and session cookies while guaranteeing hashed_password and internal audit fields never leave the HTTP response.

MODULE 6 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: Semantic HTTP Status Codes, Multipart Forms & Streaming File Uploads

Return RFC-9110 semantic HTTP status codes (`status.HTTP_201_CREATED`, `204_NO_CONTENT`) and process `Form()` fields + spooled `UploadFile` streams safely without RAM exhaustion.

Objectives: ✓ Assign semantic fastapi.status constants (201, 202, 204) and adjust status codes dynamically via injected Response when needed • ✓ Understand why HTML/OAuth2 Form() encoding (application/x-www-form-urlencoded / multipart/form-data) cannot mix with JSON Body on the same request • ✓ Stream multipart uploads via UploadFile (SpooledTemporaryFile) while validating MIME type and max byte size

6.1 FastAPI Response Status Codes — `fastapi.status` Constants, `201 Created`, `204 No Content` & Dynamic Status [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Response Status Codes — `fastapi.status` Constants, `201 Created`, `204 No Content` & Dynamic Status in real FastAPI systems — specificall...

6.2 FastAPI Forms & Files — `Form()`, `UploadFile`, `SpooledTemporaryFile` & Multipart Streaming [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Forms & Files — `Form()`, `UploadFile`, `SpooledTemporaryFile` & Multipart Streaming in real FastAPI systems — specifically: Handling `mul...

Module 6 Hands-On Capstone Project (100 Marks): Chunk-Streamed Resume Uploader with MIME & Size Quota Enforcement — Build a `201 Created` multipart upload endpoint accepting candidate metadata via `Form()` and a PDF via `UploadFile`, rejecting non-PDF MIME types (`415`) or files exceeding quota (`413`) wi...

MODULE 7 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: Structured Error Handling & FastAPI Dependency Injection (`Depends` + `yield`)

Return consistent domain error envelopes via `HTTPException` and `@app.exception_handler` while structuring reusable auth, pagination, and `yield` resource lifecycles with `Depends()`.

Objectives: ✓ Raise HTTPException with custom WWW-Authenticate/rate-limit headers and register centralized @app.exception_handler callbacks • ✓ Build hierarchical sub-dependencies (Depends) with automatic per-request result caching • ✓ Manage database/HTTP client teardown deterministically using generator dependencies with try ... yield ... finally

7.1 FastAPI Error Handling — `HTTPException`, Custom Headers & `@app.exception_handler` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Error Handling — `HTTPException`, Custom Headers & `@app.exception_handler` in real FastAPI systems — specifically: Aborting invalid reque...

7.2 FastAPI Dependencies (`Depends`) — Sub-Dependency DAGs, Per-Request Caching, `yield` & Overrides [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Dependencies (`Depends`) — Sub-Dependency DAGs, Per-Request Caching, `yield` & Overrides in real FastAPI systems — specifically: Powering...

Module 7 Hands-On Capstone Project (100 Marks): Domain Exception Envelope & `Depends` Resource Lifecycle Framework — Create a FastAPI service featuring custom domain exception handlers (returning RFC-style error JSON with correlation IDs) and a `yield`-backed database unit-of-work dependency overridable in...

MODULE 8 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 8: OAuth2 + JWT Security & ASGI HTTP Middleware

Secure endpoints with `OAuth2PasswordBearer`, Argon2/Bcrypt password hashing, short-lived JWT Bearer tokens, RBAC `Security()` scopes, and non-blocking ASGI Middleware.

Objectives: ✓ Wire OAuth2PasswordBearer + PyJWT/Argon2 into a reusable get_current_active_user dependency integrated with Swagger Authorize UI • ✓ Enforce fine-grained OAuth2 SecurityScopes (e.g. items:read, admin:write) via Security() • ✓ Write fast ASGI / @app.middleware("http") layers for X-Process-Time telemetry and security headers without consuming request streams prematurely

8.1 FastAPI Security — `OAuth2PasswordBearer`, JWT Verification, Argon2 Hashing & RBAC Scopes [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Security — `OAuth2PasswordBearer`, JWT Verification, Argon2 Hashing & RBAC Scopes in real FastAPI systems — specifically: First-class Open...

8.2 FastAPI Middleware — `@app.middleware("http")`, Execution Onion Order & Pure ASGI Middleware [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI Middleware — `@app.middleware("http")`, Execution Onion Order & Pure ASGI Middleware in real FastAPI systems — specifically: Intercepting...

Module 8 Hands-On Capstone Project (100 Marks): OAuth2 JWT Bearer Auth Guard & Request Telemetry Middleware — Build an OAuth2 Bearer-protected API that verifies signed token claims, enforces role-based access dependencies, and attaches `X-Process-Time-Ms` + `X-Content-Type-Options: nosniff` headers...

MODULE 9 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 9: Cross-Origin Resource Sharing (`CORSMiddleware`) & SQL Databases (`SQLAlchemy 2.0`)

Configure browser CORS preflight policies safely without credentialed wildcard vulnerabilities and integrate SQLAlchemy 2.0 / SQLModel transactional sessions via `Depends`.

Objectives: ✓ Configure CORSMiddleware for React/Angular/Next.js frontends while respecting the browser rule forbidding allow_origins=["*"] with allow_credentials=True • ✓ Model relational tables with SQLAlchemy 2.0 Mapped[] / SQLModel and Pydantic ConfigDict(from_attributes=True) • ✓ Prevent SQLite/PostgreSQL connection pool leaks and cross-thread hazards using per-request Session dependencies

9.1 FastAPI CORS — `CORSMiddleware`, `OPTIONS` Preflight Handshakes & Credentialed Origin Rules [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI CORS — `CORSMiddleware`, `OPTIONS` Preflight Handshakes & Credentialed Origin Rules in real FastAPI systems — specifically: Managing brows...

9.2 FastAPI SQL Databases — SQLAlchemy 2.0 / SQLModel, `Session` per Request & `from_attributes=True` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI SQL Databases — SQLAlchemy 2.0 / SQLModel, `Session` per Request & `from_attributes=True` in real FastAPI systems — specifically: Connecti...

Module 9 Hands-On Capstone Project (100 Marks): CORS-Hardened SPA Backend with SQLAlchemy 2.0 Transactional Repository — Build a FastAPI CRUD API configured with explicit origin whitelisting on `CORSMiddleware` and backed by an isolated per-request SQLite/SQLAlchemy 2.0 session dependency.

MODULE 10 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 10: Modular Microservice Architecture (`APIRouter`) & `BackgroundTasks`

Scale large FastAPI codebases cleanly across domain packages using `APIRouter`, lifespan context managers, and non-blocking post-response `BackgroundTasks`.

Objectives: ✓ Partition large FastAPI codebases into versioned domain APIRouter modules with shared prefixes, OpenAPI tags, and router-level auth dependencies • ✓ Manage startup/shutdown connection pools cleanly via @asynccontextmanager lifespan(app: FastAPI) • ✓ Offload post-response emails/audit logs with BackgroundTasks while recognizing when durable queues (Celery/ARQ/SQS) are required

10.1 Bigger FastAPI Applications — `APIRouter`, Versioned Prefixes, Router Dependencies & `lifespan` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Bigger FastAPI Applications — `APIRouter`, Versioned Prefixes, Router Dependencies & `lifespan` in real FastAPI systems — specifically: Organizing...

10.2 FastAPI `BackgroundTasks` — Post-Response Execution, Session Safety & Celery / SQS Trade-Offs [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply FastAPI `BackgroundTasks` — Post-Response Execution, Session Safety & Celery / SQS Trade-Offs in real FastAPI systems — specifically: Returning fa...

Module 10 Hands-On Capstone Project (100 Marks): Multi-Router Versioned API (`/api/v1`) with Lifespan Pool & Post-Response Audit Worker — Structure a multi-module FastAPI service mounting `users` and `enrollments` `APIRouter` instances under `/api/v1` with router-level auth dependencies and a `BackgroundTasks` post-checkout au...

MODULE 11 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Async SQLAlchemy 2.0, Connection Pooling, Alembic & Keyset Pagination

Production-grade FastAPI Training engineering module covering Async SQLAlchemy 2.0, Connection Pooling, Alembic & Keyset Pagination with hands-on labs and capstone build.

Objectives: ✓ AsyncEngine, async_sessionmaker & Scoped Per-Request Yield Dependencies — Purpose: Run non-blocking asyncpg queries cleanly without session leaks or blocking the event loop. • ✓ Eager Loading (selectinload) in Async Mode to Prevent MissingGreenlet — Purpose: Avoid implicit lazy-load crashes in async SQLAlchemy while eliminating N+1 queries. • ✓ Alembic Async Migrations & Keyset Cursor Pagination — Purpose: Evolve PostgreSQL schemas safely and paginate millions of rows in constant O(1) index time. • ✓ Transactional Outbox & Atomic SELECT ... FOR UPDATE Locks — Purpose: Guarantee race-free balance updates and reliable downstream event publishing.

11.1 AsyncEngine, async_sessionmaker & Scoped Per-Request Yield Dependencies [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply AsyncEngine, async_sessionmaker & Scoped Per-Request Yield Dependencies in real FastAPI systems — specifically: Run non-blocking asyncpg queries c...

11.2 Eager Loading (selectinload) in Async Mode to Prevent MissingGreenlet [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Eager Loading (selectinload) in Async Mode to Prevent MissingGreenlet in real FastAPI systems — specifically: Avoid implicit lazy-load crashes in...

11.3 Alembic Async Migrations & Keyset Cursor Pagination [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Alembic Async Migrations & Keyset Cursor Pagination in real FastAPI systems — specifically: Evolve PostgreSQL schemas safely and paginate millions...

11.4 Transactional Outbox & Atomic SELECT ... FOR UPDATE Locks [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Transactional Outbox & Atomic SELECT ... FOR UPDATE Locks in real FastAPI systems — specifically: Guarantee race-free balance updates and reliable...

Module 11 Hands-On Capstone Project (100 Marks): Sub-20ms Async PostgreSQL Data Layer for FastAPI — Design, implement, test, and document 'Sub-20ms Async PostgreSQL Data Layer for FastAPI' applying every production concept from Module 11 (Async SQLAlchemy 2.0, Connection Pooling, Alembic &...

MODULE 12 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

OAuth2 Scopes, Rotating JWTs, Redis Rate Limiting & Idempotency Middleware

Production-grade FastAPI Training engineering module covering OAuth2 Scopes, Rotating JWTs, Redis Rate Limiting & Idempotency Middleware with hands-on labs and capstone build.

Objectives: ✓ OAuth2PasswordBearer + Security() Granular Permission Scopes — Purpose: Document and enforce exact read/write scopes right inside OpenAPI & Swagger UI. • ✓ Asymmetric RS256 JWT Verification & Redis Refresh Token Rotation — Purpose: Validate tokens fast across replicas and revoke stolen sessions immediately. • ✓ Sliding-Window Redis Rate Limiter & Idempotency-Key Middleware — Purpose: Prevent brute-force floods and duplicate payment execution under mobile retries. • ✓ Strict Pydantic v2 Validators, CORS Allowlists & TrustedHost Hardening — Purpose: Reject malformed payloads in Rust-powered microseconds before hitting business code.

12.1 OAuth2PasswordBearer + Security() Granular Permission Scopes [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply OAuth2PasswordBearer + Security() Granular Permission Scopes in real FastAPI systems — specifically: Document and enforce exact read/write scopes...

12.2 Asymmetric RS256 JWT Verification & Redis Refresh Token Rotation [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Asymmetric RS256 JWT Verification & Redis Refresh Token Rotation in real FastAPI systems — specifically: Validate tokens fast across replicas and...

12.3 Sliding-Window Redis Rate Limiter & Idempotency-Key Middleware [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Sliding-Window Redis Rate Limiter & Idempotency-Key Middleware in real FastAPI systems — specifically: Prevent brute-force floods and duplicate pa...

12.4 Strict Pydantic v2 Validators, CORS Allowlists & TrustedHost Hardening [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Strict Pydantic v2 Validators, CORS Allowlists & TrustedHost Hardening in real FastAPI systems — specifically: Reject malformed payloads in Rust-p...

Module 12 Hands-On Capstone Project (100 Marks): Zero-Trust FinTech FastAPI Gateway with Redis Guards — Design, implement, test, and document 'Zero-Trust FinTech FastAPI Gateway with Redis Guards' applying every production concept from Module 12 (OAuth2 Scopes, Rotating JWTs, Redis Rate Limiti...

MODULE 13 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

WebSockets, SSE Streaming, ARQ/Celery Queues & Lifespan Resource Pools

Production-grade FastAPI Training engineering module covering WebSockets, SSE Streaming, ARQ/Celery Queues & Lifespan Resource Pools with hands-on labs and capstone build.

Objectives: ✓ FastAPI Lifespan Context Manager for Shared HTTPX, Redis & ML Pools — Purpose: Initialize and tear down connection pools cleanly once per worker process. • ✓ Authenticated WebSockets & Server-Sent Events (StreamingResponse) — Purpose: Stream live notifications, stock ticks, and GenAI tokens with minimal memory overhead. • ✓ BackgroundTasks vs Distributed Redis Worker Queues (ARQ / Celery) — Purpose: Know when lightweight in-process tasks suffice vs durable retryable worker queues. • ✓ Resilient Outbound HTTPX AsyncClient with Timeouts & Circuit Fallbacks — Purpose: Call partner webhooks safely without letting slow external APIs stall Uvicorn.

13.1 FastAPI Lifespan Context Manager for Shared HTTPX, Redis & ML Pools [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply FastAPI Lifespan Context Manager for Shared HTTPX, Redis & ML Pools in real FastAPI systems — specifically: Initialize and tear down connection po...

13.2 Authenticated WebSockets & Server-Sent Events (StreamingResponse) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Authenticated WebSockets & Server-Sent Events (StreamingResponse) in real FastAPI systems — specifically: Stream live notifications, stock ticks,...

13.3 BackgroundTasks vs Distributed Redis Worker Queues (ARQ / Celery) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply BackgroundTasks vs Distributed Redis Worker Queues (ARQ / Celery) in real FastAPI systems — specifically: Know when lightweight in-process tasks s...

13.4 Resilient Outbound HTTPX AsyncClient with Timeouts & Circuit Fallbacks [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Resilient Outbound HTTPX AsyncClient with Timeouts & Circuit Fallbacks in real FastAPI systems — specifically: Call partner webhooks safely withou...

Module 13 Hands-On Capstone Project (100 Marks): Real-Time Streaming & Async Background Processing Service — Design, implement, test, and document 'Real-Time Streaming & Async Background Processing Service' applying every production concept from Module 13 (WebSockets, SSE Streaming, ARQ/Celery Queu...

MODULE 14 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Pytest-Asyncio Integration Suites, OpenTelemetry & Gunicorn/Uvicorn Docker

Production-grade FastAPI Training engineering module covering Pytest-Asyncio Integration Suites, OpenTelemetry & Gunicorn/Uvicorn Docker with hands-on labs and capstone build.

Objectives: ✓ pytest-asyncio + httpx.AsyncClient + Dependency Overrides — Purpose: Test every async endpoint, auth guard, and DB rollback in seconds. • ✓ RFC 9457 Problem-Details Exception Handlers & Correlation Request IDs — Purpose: Return predictable JSON error contracts linked directly to structured JSON logs. • ✓ OpenTelemetry Auto-Instrumentation & Prometheus /metrics Endpoint — Purpose: Measure p50/p95/p99 route latency and async DB query spans in Grafana. • ✓ Multi-Stage uv/pip Dockerfiles, Uvicorn Worker Tuning & Graceful Shutdown — Purpose: Deploy lean non-root FastAPI containers that drain active requests cleanly on deploy.

14.1 pytest-asyncio + httpx.AsyncClient + Dependency Overrides [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply pytest-asyncio + httpx.AsyncClient + Dependency Overrides in real FastAPI systems — specifically: Test every async endpoint, auth guard, and DB ro...

14.2 RFC 9457 Problem-Details Exception Handlers & Correlation Request IDs [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply RFC 9457 Problem-Details Exception Handlers & Correlation Request IDs in real FastAPI systems — specifically: Return predictable JSON error contra...

14.3 OpenTelemetry Auto-Instrumentation & Prometheus /metrics Endpoint [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply OpenTelemetry Auto-Instrumentation & Prometheus /metrics Endpoint in real FastAPI systems — specifically: Measure p50/p95/p99 route latency and as...

14.4 Multi-Stage uv/pip Dockerfiles, Uvicorn Worker Tuning & Graceful Shutdown [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Multi-Stage uv/pip Dockerfiles, Uvicorn Worker Tuning & Graceful Shutdown in real FastAPI systems — specifically: Deploy lean non-root FastAPI con...

Module 14 Hands-On Capstone Project (100 Marks): CI-Tested Containerized FastAPI Production Microservice — Design, implement, test, and document 'CI-Tested Containerized FastAPI Production Microservice' applying every production concept from Module 14 (Pytest-Asyncio Integration Suites, OpenTelem...

MODULE 15 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Async FastAPI Concurrency Patterns, Lifespan Resources & Streaming APIs

Self-explanatory list-and-bullet production mastery module covering Async FastAPI Concurrency Patterns, Lifespan Resources & Streaming APIs in FastAPI Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken Gemini AI Viva.

Objectives: ✓ Event-Loop Safety: async def vs def Threadpool Routing & Lifespan Connection Pools — Purpose: Never block the Uvicorn asyncio loop with synchronous CPU/IO calls under high concurrency. • ✓ Server-Sent Events (SSE), StreamingResponse & WebSocket Real-Time Channels — Purpose: Stream live AI tokens, progress updates, and telemetry feeds cleanly to web and mobile clients. • ✓ Redis Sliding-Window Rate Limiting, Idempotency Middleware & RFC 7807 Errors — Purpose: Standardize validation errors and protect public FastAPI routes from abusive traffic. • ✓ Build, verify, and document 'Token-Streaming & Rate-Limited Async FastAPI Gateway with Shared Lifespan Pools' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + Google Gemini AI Technical Interviews.

15.1 Event-Loop Safety: async def vs def Threadpool Routing & Lifespan Connection Pools [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Event-Loop Safety: async def vs def Threadpool Routing & Lifespan Connection Pools in real FastAPI systems — specifically: Never block the Uvicorn...

15.2 Server-Sent Events (SSE), StreamingResponse & WebSocket Real-Time Channels [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Server-Sent Events (SSE), StreamingResponse & WebSocket Real-Time Channels in real FastAPI systems — specifically: Stream live AI tokens, progress...

15.3 Redis Sliding-Window Rate Limiting, Idempotency Middleware & RFC 7807 Errors [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Redis Sliding-Window Rate Limiting, Idempotency Middleware & RFC 7807 Errors in real FastAPI systems — specifically: Standardize validation errors...

Module 15 Hands-On Capstone Project (100 Marks): Token-Streaming & Rate-Limited Async FastAPI Gateway with Shared Lifespan Pools — Architect, implement, test, and document 'Token-Streaming & Rate-Limited Async FastAPI Gateway with Shared Lifespan Pools' applying every bullet-wise engineering concept from Module 15 (Asyn...

MODULE 16 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Async SQLAlchemy 2.0, pytest-asyncio Suites & Production Uvicorn/Gunicorn

Self-explanatory list-and-bullet production mastery module covering Async SQLAlchemy 2.0, pytest-asyncio Suites & Production Uvicorn/Gunicorn in FastAPI Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken Gemini AI Viva.

Objectives: ✓ Async SQLAlchemy 2.0 Sessions, Alembic Async Migrations & Avoiding MissingGreenlet — Purpose: Write clean non-blocking database repositories with explicit eager loading and transaction scopes. • ✓ Dependency Override Testing with httpx.AsyncClient & pytest-asyncio Fixtures — Purpose: Test auth scopes, validation failures, and DB rollbacks rapidly in isolated async suites. • ✓ Production Uvicorn Workers, OpenTelemetry Tracing & FastAPI Interview Mastery — Purpose: Tune reverse-proxy headers, graceful shutdowns, and Pydantic v2 serialization speed. • ✓ Build, verify, and document 'Containerized Async FastAPI + PostgreSQL Microservice with 95% Route Coverage' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + Google Gemini AI Technical Interviews.

16.1 Async SQLAlchemy 2.0 Sessions, Alembic Async Migrations & Avoiding MissingGreenlet [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Async SQLAlchemy 2.0 Sessions, Alembic Async Migrations & Avoiding MissingGreenlet in real FastAPI systems — specifically: Write clean non-blockin...

16.2 Dependency Override Testing with httpx.AsyncClient & pytest-asyncio Fixtures [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Dependency Override Testing with httpx.AsyncClient & pytest-asyncio Fixtures in real FastAPI systems — specifically: Test auth scopes, validation...

16.3 Production Uvicorn Workers, OpenTelemetry Tracing & FastAPI Interview Mastery [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Production Uvicorn Workers, OpenTelemetry Tracing & FastAPI Interview Mastery in real FastAPI systems — specifically: Tune reverse-proxy headers,...

Module 16 Hands-On Capstone Project (100 Marks): Containerized Async FastAPI + PostgreSQL Microservice with 95% Route Coverage — Architect, implement, test, and document 'Containerized Async FastAPI + PostgreSQL Microservice with 95% Route Coverage' applying every bullet-wise engineering concept from Module 16 (Async...

3. LIVE CLIENT PRODUCTION PROJECT, GLOBAL CERTIFICATIONS & CAREER SCOPE

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

Project: FinPulse Async High-Concurrency Payment & Partner Webhook Microservice • **Confirmed Role:** Python FastAPI Backend Engineering Intern

Domain & Production Infra: Digital Payments & API Platform Client (FastAPI + Pydantic v2 + Async SQLAlchemy + Uvicorn/Gunicorn Docker Production)

- Developed async FastAPI endpoints with strict Pydantic v2 schema validation, dependency injection, and OpenAPI 3.1 docs.
- Implemented OAuth2/JWT security scopes, async database session pooling, and idempotent background task pipelines.
- Wrote comprehensive pytest-asyncio integration suites achieving high endpoint coverage and sub-30ms p95 latency.
- Containerized and deployed the microservice behind reverse-proxy TLS to live client production.

Aligned Global Vendor Certifications: **PCAP-31-03** (PCAP™ – Certified Associate in Python Programming) • **PCPP1-32-101** (PCPP1™ – Certified Professional in Python Programming 1) • **DVA-C02** (AWS Certified Developer – Associate)

Target Job Roles: Python FastAPI Backend Engineer, Async Microservices Developer, AI / LLM Backend API Engineer, Software Engineer — Python APIs

Realistic Annual CTC Scope: **Fresher / Entry-Level (0–1 Yr):** ₹5.5 LPA – ₹10.0 LPA **Mid-Level FastAPI / AI Backend Engineer (1–3 Yrs):** ₹10.5 LPA – ₹21.5 LPA **Senior Python Microservices / GenAI Architect (3–6+ Yrs):** ₹22.0 LPA – ₹42.0+ LPA

MockAttempt Academy Pvt. Ltd. • Email: support@mockattempt.com • Phone/WhatsApp: +91 98765 43210 • Web: https://mockattempt.com/courses/fastapi-training

Try 2 Free Topic-Wise 50-MCQ Mock Tests & Spoken Gemini AI Interviews daily on every topic above before enrolling.