Fast API with Python

Module 1: Introduction to FastAPI

- What is FastAPI?
- Key features:
    - High performance (based on Starlette & Pydantic)
    - Async support
    - Auto docs via Swagger & ReDoc
- FastAPI vs Flask/Django
- Installing FastAPI and Uvicorn
- First FastAPI app: “Hello World”
- Project folder structure for APIs

Module 2: Handling Requests

- Path parameters
- Query parameters
- Request bodies (JSON)
- Data validation with **Pydantic** models
- Request forms and files
- Using Depends and Body, Query, Path

Module 3: Handling Responses

- Response models and schemas
- Setting status codes
- Response headers and cookies
- File responses
- Custom response classes
- Using JSONResponse, HTMLResponse, RedirectResponse

Module 4: Path Operations

- Creating GET, POST, PUT, DELETE endpoints
- Handling path, query, and body data together
- Path operation decorators (@app.get(), etc.)
- Tags and grouping endpoints in documentation

Module 5: Dependency Injection

- What are dependencies in FastAPI?
- Using Depends for request lifecycle
- Dependency injection patterns
- Shared dependencies (e.g., DB sessions, auth checks)

Module 6: Pydantic and Data Validation

- Creating Pydantic models for request and response
- Field validation using Field()
- Nested and complex models
- Optional fields and default values
- Custom data types and validators

Module 7: Authentication and Authorization

- Security dependencies
- HTTP Basic Auth
- OAuth2 and JWT authentication (using fastapi.security)
- Role-based access control (RBAC)
- Protecting routes with Depends

Module 8: FastAPI with Databases (CRUD Operations)

- Database connection (SQLAlchemy ORM or Tortoise ORM)
- Creating tables and models
- Creating, reading, updating, deleting records (CRUD)
- Dependency-injected DB sessions
- Async database support with Databases or SQLModel

Module 9: File Handling and Uploads

- Uploading files and images using UploadFile
- Validating file size and type
- Saving and serving static files
- Downloading files with custom responses

Module 10: Testing FastAPI Applications

- Using pytest with FastAPI
- Writing unit tests and integration tests
- Test clients (TestClient)
- Mocking dependencies
- Coverage and test structure

Module 11: Middleware and Background Tasks

- Writing custom middleware
- Using built-in middlewares (CORS, GZip)
- Background tasks with BackgroundTasks
- Use cases: email sending, processing files, long-running tasks

Module 12: Async Programming in FastAPI

- Sync vs async functions in FastAPI
- async def, await, and concurrency

- Using FastAPI with async libraries (e.g., HTTPX, asyncpg)
- Managing connection pooling

Module 13: API Documentation and OpenAPI

- Interactive docs (Swagger UI, ReDoc)
- Customizing OpenAPI schema
- Metadata for tags, descriptions, examples
- Versioning APIs

Module 14: Deployment and Performance

- Using **Uvicorn**, **Gunicorn**, **Docker**
- Environment variables and .env management
- Serving behind Nginx or reverse proxies
- Using HTTPS
- Performance tuning tips

==========

END

==========