FastAPI – Day 1 Full Guide

1. What is FastAPI
2. Modern Python framework for building APIs
3. Built on Starlette (web toolkit) and Pydantic (data validation)
4. Features:
5. High performance
6. Type hints support
7. Async requests
8. Automatic Swagger + ReDoc docs

Real-world example: FastAPI is like a smart restaurant kitchen: - Customers = Flutter app requests - Chefs = FastAPI endpoints - Orders = API requests (GET, POST) - Food = JSON response - Menu = Swagger docs

1. Why FastAPI
2. Fast (async + Starlette)
3. Less code, more features
4. Automatic validation (Pydantic)
5. Auto interactive documentation
6. Asynchronous support for multiple simultaneous requests
7. Companies using it: Netflix, Uber, Microsoft

Real-world example: Multiple orphanages submit data at the same time → FastAPI handles all requests simultaneously.

1. How FastAPI Works
2. ASGI Server (Uvicorn) – handles multiple requests concurrently
3. Routes / Path Operations – URLs mapped to Python functions
4. Type hints + Pydantic – automatic validation of incoming data
5. Dependency Injection – modular, reusable components
6. Response Serialization – converts Python dicts into JSON automatically
7. Automatic Documentation – Swagger UI & ReDoc

Real-world example: Flutter app sends orphanage data → FastAPI validates → saves in DB → responds with JSON → Flutter displays confirmation.

1. Type Hints
2. Define expected types for variables, function parameters, and return values
3. FastAPI uses type hints to validate incoming data automatically

Example:

```python
def add(a: int, b: int) -> int:
    return a + b
```

Real-world example: Flutter sends phone number as a string instead of integer → FastAPI automatically rejects it.

1. Dependency Injection (DI)

2. Inject external resources into endpoints (DB connections, authentication)
3. Keeps code modular, reusable, and testable

Example:

```
def get_db():
    db = "database connection"
    return db

@app.get("/orphanages")
def get_orphanages(db=Depends(get_db)):
    return db.get_all()
```

Real-world example: All endpoints share a single DB connection → no repetition → easy maintenance.

1. Real-Time Example – Orphanage Registration
2. Flutter app sends POST request:

   ```
   {
     "name": "Hope Orphanage",
     "address": "123 MG Road",
     "phone": 9876543210
   }
   ```

3. FastAPI endpoint validates data via Pydantic
4. Dependency Injection provides DB connection
5. Data stored in MongoDB
6. Flutter fetches updated data via GET endpoint

Flow Analogy: Flutter = Customer ordering, FastAPI = Waiter, Pydantic = Kitchen manager checking ingredients, Database = Storage, JSON Response = Served dish

1. FastAPI vs Flask vs Django vs Node.js vs Go

| Feature | FastAPI | Flask | Django | Node.js | Go |
|---------|---------|-------|--------|---------|----|
| Language | Python | Python | Python | JS | Go |
| Performance | ⚡Very High | Moderate | Moderate | High | Extremely High |
| Async Support | 🦜Native | 🦡Limited | 🦡Limited | 🦜Event-driven | 🦜Built-in |
| Automatic Validation | 🦜Pydantic | 🦡Manual | 🦜Forms/Models | 🦡Manual | 🦡Manual |
| Documentation | 🦜Auto | 🦡Manual | 🦡Manual | 🦡Manual | 🦡Manual |
| Scalability | 🦜High | 🦡Moderate | 🦜Moderate | 🦜High | 🦜Very High |
| Flutter Integration | 🦜Excellent | 🦜Good | 🦜Good | 🦜Excellent | 🦜Excellent |

Real-world analogy: FastAPI → Modern smart kitchen, Flask → Small kitchen, Django → 5-star hotel kitchen, Node.js → Busy street food stall, Go → High-efficiency factory kitchen

1. Performance & Speed
2. ASGI + Starlette + Pydantic + Type hints
3. Handles thousands of requests per second
4. Async support ensures multiple requests processed simultaneously

Analogy: Django/Flask → One waiter per table → slower, FastAPI/Node.js → Waiters handle multiple tables → faster, Go → Super-efficient factory → fastest

Day 1 Summary - Understand FastAPI fundamentals - Know why it's fast, modern, and suitable for Flutter backend - Learned type hints, DI, real-time examples, and comparisons - Ready for Day 2: Installation + POST endpoints + Database integration