Perfect 🔥  
Since you already know Python and basic FastAPI, here’s your **5-day, hour-by-hour roadmap** to become **production-ready with FastAPI** — no wasted time, just focused practical learning.

**⚡️ 5-Day FastAPI Production-Ready Roadmap**

(≈ 4 hours per day)

**🧩 Day 1 — Database Integration (SQLAlchemy + CRUD)**

**Goal:** Build a basic API with a database and CRUD routes.

**Hour 1:**

* Review SQLAlchemy basics: models, sessions, and relationships.
* Setup project structure: main.py, database.py, models.py, schemas.py.

**Hour 2:**

* Create your first model (e.g., User with id, name, email).
* Create matching Pydantic schema for request/response.

**Hour 3:**

* Build CRUD endpoints:
  + POST /users
  + GET /users
  + GET /users/{id}
  + DELETE /users/{id}

**Hour 4:**

* Test using Swagger UI or Postman.
* Add auto table creation (Base.metadata.create\_all(engine)).

🧠 Concepts: ORM, Dependency Injection (Depends), Pydantic models, CRUD.

**🔐 Day 2 — Authentication & Authorization (JWT)**

**Goal:** Add secure login/signup with JWT tokens.

**Hour 1:**

* Learn password hashing using passlib.
* Add signup route → save hashed password.

**Hour 2:**

* Implement login route → verify credentials.
* Generate JWT tokens using python-jose.s

**Hour 3:**

* Protect routes using Depends(get\_current\_user).
* Test access with and without tokens.

**Hour 4:**

* Clean up auth logic into auth.py or routers/auth.py.
* Add environment variable for JWT secret.

🧠 Concepts: JWT, OAuth2PasswordBearer, Hashing, Dependency Injection.

**🧱 Day 3 — Project Structure + Modularity**

**Goal:** Organize your app like a real production backend.

**Hour 1:**

* Create directories:
* app/
* main.py
* routers/
* models/
* schemas/
* core/
* database.py

**Hour 2:**

* Move routes into separate files (routers/users.py, routers/auth.py).
* Import them in main.py with app.include\_router().

**Hour 3:**

* Add custom error handling with HTTPException.
* Add response models for cleaner API responses.

**Hour 4:**

* Add .env file for configs (DB URL, JWT secret).
* Use pydantic-settings or python-dotenv.

🧠 Concepts: Modular structure, routers, dependency injection, clean architecture.

**⚙️ Day 4 — Async I/O + Middleware + Background Tasks**

**Goal:** Improve performance and handle tasks smartly.

**Hour 1:**

* Learn difference between def and async def in endpoints.
* Convert your endpoints to async (use AsyncSession if needed).

**Hour 2:**

* Add middleware for logging or request timing.

**Hour 3:**

* Add a background task (e.g., send email after user registers).
* from fastapi import BackgroundTasks

**Hour 4:**

* Add CORS middleware (from fastapi.middleware.cors import CORSMiddleware).
* Test performance and logs.

🧠 Concepts: Async/Await, Middleware, BackgroundTasks, CORS.

**🚀 Day 5 — Testing + Deployment**

**Goal:** Get your API production-ready & live.

**Hour 1:**

* Write unit tests with pytest and TestClient.
* Test CRUD and auth routes.

**Hour 2:**

* Integrate Alembic for DB migrations.

**Hour 3:**

* Containerize app with Docker (optional but recommended).

**Hour 4:**

* Deploy app on **Render**, **Railway**, or **VPS** (with uvicorn --host 0.0.0.0 --port 8000).
* Verify live API using Swagger docs.

🧠 Concepts: Testing, Migrations, Docker, Deployment, Environment management.

**🧠 After 5 Days — You’ll Be Able To**

✅ Build full FastAPI backends (auth + DB + async + middleware).  
✅ Structure projects for scalability.  
✅ Deploy & maintain production APIs confidently.