Architecture Refactoring Plan

Current State

Currently, `scheduler_service.py` acts as a "God Object," containing:

1. Database configuration
2. SQLAlchemy ORM Models
3. Pydantic API Schemas
4. Business Logic (Gurobi optimization)
5. API Routes (FastAPI)

Proposed Modular Structure

To modularize the application, we should split the single file into a standard Python package structure using the **Service-Repository Pattern**.

Recommended Directory Tree

```
hospital_scheduler/
├── app/
│   ├── __init__.py
│   ├── main.py              # Entry point (FastAPI app initialization)
│   ├── config.py            # Environment variables & logging config
│   ├── database.py          # DB connection & session handling
│   ├── models.py            # SQLAlchemy Database Models (Employee, Shift, etc.)
│   ├── schemas.py           # Pydantic Models (Data Validation for API)
│   │
│   ├── api/                 # API Route Controllers
│   │   ├── __init__.py
│   │   ├── endpoints.py     # The GET/POST endpoints
│   │   └── dependencies.py  # Dependency injection (get_db, user_auth)
│   │
│   └── services/            # Business Logic
│       ├── __init__.py
│       └── solver.py        # The GurobiScheduler class (Pure logic)
│
├── tests/                   # Unit and Integration Tests
├── requirements.txt
└── run.py                   # Script to run uvicorn
```

Step-by-Step Migration Guide

1. Extract Models (`app/models.py`)

Move all classes inheriting from `Base` (`Employee` , `Shift` , `Demand` , `ScheduleRun` , `Assignment`) into this file.

- **Benefit**: Separates data structure from logic.

2. Extract Schemas (`app/schemas.py` **)**

Move all classes inheriting from `BaseModel` (`SolveRequest` , `RunResponse` , etc.) here.

- **Benefit**: Keeps API contracts clean and readable.

3. Isolate the Core Logic (`app/services/solver.py` **)**

Move the `GurobiScheduler` class here.

- **Refactor**: Remove direct API dependencies. The solver should just take a DB session and parameters, run the math, and save results. It shouldn't know about FastAPI or HTTP.

4. Setup Database (`app/database.py` **)**

Move `DATABASE_URL` , `engine` , `SessionLocal` , `Base` , and `get_db` here.

- **Benefit**: Makes it easier to swap SQLite for PostgreSQL later without touching the rest of the code.

5. Clean up Routes (`app/api/endpoints.py` **)**

Move the `@app.post` and `@app.get` functions here.

- **Import**: Import the necessary models and services from the files created above.

6. Main Entry Point (`app/main.py` **)**

This file becomes very small. It simply creates the `app = FastAPI()` instance and includes the router from `endpoints.py` .

Benefits of Modularization

1. **Teamwork**: One developer can improve the Gurobi logic in `solver.py` while another builds a frontend for the API in `endpoints.py` without merge conflicts.
2. **Testing**: You can write unit tests for `solver.py` without needing to start the web server.
3. **Readability**: New developers know exactly where to look. "I need to add a column to the Employee table" -> Go to `models.py` .