🎯 Goal
Build a Document Management API where:
Users can register & login
Users can upload documents
Admin can approve/reject documents
Only approved documents are publicly accessible
Role-based access control is enforced

📅 Timeline Plan (5 Days)

✅ Day 1 – Project Setup + Authentication
Setup FastAPI project structure (proper modular)
JWT Authentication (access + refresh token)
Password hashing
User registration & login
Role system (admin, user)

✅ Day 2 – Database Design (Relational)
Design tables:
users
documents
document_status_history (optional but recommended)
Use:
SQLAlchemy ORM
Alembic migrations

✅ Day 3 – File Upload + Storage
Upload document (PDF, image only)
Store file locally
Save metadata in DB:
file_path
uploaded_by
status = pending

✅ Day 4 – RBAC (Role Based Access Control)
Rules:
User:
Upload document
View only their documents
Admin:
View all documents
Approve / Reject document
If they implement this properly → they understand dependencies & security.

✅ Day 5 – Background Task
When document is approved:

Trigger background task:
Log approval
Or simulate email sending
Use:
FastAPI BackgroundTasks

✅ Day 6 – Filtering, Pagination, Search
Admin endpoint:
Filter by status
Filter by date range
Pagination
Search by filename

✅ Day 7 – Testing + Documentation
Write basic pytest for testing functions
Add Swagger customization
Proper exception handling
Custom error response format

📌 Detailed Task Requirements

1️⃣ Authentication System

Must implement:
JWT token
Refresh token
Password hashing (bcrypt)
Token dependency injection
Token expiration

2️⃣ Proper Project Structure
They must NOT write everything in main.py
Expected structure:
Copy code

```
app/
 ├── main.py
 ├── models/
 ├── schemas/
 ├── routes/
 ├── services/
 ├── core/
 ├── dependencies/
 └── utils/
```

3️⃣ Role-Based Access Control

Create role field in user model
Create dependency to check role
Protect routes properly
Example:
Copy code
Python
```python
@router.post("/approve")
def approve_document(current_user=Depends(admin_only)):
```

4️⃣ File Validation

Validate file type
Limit file size
Handle invalid upload safely
Beginner will ignore validation.

5️⃣ Advanced Querying

Admin endpoint should support:
?status=approved
?start_date=
?end_date=
?search=
Pagination

6️⃣ Proper Error Handling

Use HTTPException properly
Global exception handler
Structured error responses