Pharma Documents Management System – Developer Guidelines

# 💊 Pharma Documents Management System – Developer Guidelines

## 📘 Table of Contents

1. Project Structure

2. API Design Standards

3. Authentication & Authorization

4. Document Handling

5. Audit Trails & History

6. Validation & Error Handling

7. Security Best Practices

8. Deployment & Environment Setup

9. Testing

10. Versioning

11. Logging & Monitoring

12. Compliance & Retention

## 1. Project Structure

Use a modular Django structure:

pharma\_dms/

├── api/ # REST API logic

│ ├── views.py

│ ├── serializers.py

│ ├── urls.py

│ └── permissions.py

├── documents/ # Document management logic

├── users/ # User management

├── audit/ # Audit trail logic

├── settings/

│ ├── base.py

│ ├── dev.py

│ └── prod.py

## 2. API Design Standards

- Follow \*\*RESTful principles\*\*.

- Use \*\*plural nouns\*\* in URLs: `/documents/`, `/users/`, `/approvals/`.

- Use \*\*nested routes\*\* for relationships:

- `GET /documents/{doc\_id}/history/`

- Prefer \*\*DRF ViewSets\*\* and \*\*Routers\*\*.

- Use consistent \*\*pagination\*\* (`LimitOffsetPagination` or `PageNumberPagination`).

Example Endpoint:

GET /api/documents/

POST /api/documents/

GET /api/documents/{id}/

## 3. Authentication & Authorization

- Use \*\*JWT\*\* or \*\*TokenAuthentication\*\* (via `djangorestframework-simplejwt`).

- Enforce role-based access control (RBAC):

- Initiator

- Reviewer

- Approver

- Admin

Implement custom permissions:

class IsReviewer(permissions.BasePermission):

def has\_permission(self, request, view):

return request.user.role == 'reviewer'

## 4. Document Handling

- Upload documents using `FileField` with secure storage (`MEDIA\_ROOT`).

- Generate unique version IDs and timestamps.

- Store metadata: name, version, status, owner, department, reviewers, expiry, etc.

- Maintain status lifecycle:

- `Draft → Under Review → Approved → Archived`

## 5. Audit Trails & History

- Track:

- Who created/edited/approved

- Timestamps

- Version changes

- Use `django-simple-history` or custom audit models.

- Display history via: `GET /documents/{id}/history/`

## 6. Validation & Error Handling

- Validate:

- Document types

- File size

- Required fields

- Use DRF’s `ValidationError`

raise serializers.ValidationError({"status": "Status is required."})

- Return consistent error format:

{

"detail": "Validation failed.",

"errors": {

"title": ["This field is required."]

}

}

## 7. Security Best Practices

- Enforce HTTPS.

- Limit file upload types (`.pdf`, `.docx`, etc.)

- Use `Content-Disposition` headers for secure downloads.

- Sanitize filenames and avoid direct access to paths.

- Enable CSRF protection where applicable.

- Use `django-environ` to manage secrets.

## 8. Deployment & Environment Setup

- Use `.env` files for config management.

- Production via `gunicorn + nginx` or Docker.

- Set up:

- PostgreSQL

- Redis (if using Celery for tasks)

- Use `whitenoise` for static files.

## 9. Testing

- Unit tests for:

- Models

- Serializers

- Permissions

- API endpoints

- Use `pytest-django` or Django’s `TestCase`.

- Test file uploads and versioning behavior.

## 10. Versioning

- Use \*\*URL versioning\*\*:

- `/api/v1/documents/`

- Document changes in a `CHANGELOG.md`.

## 11. Logging & Monitoring

- Use Python’s `logging` module:

LOGGING = {

"version": 1,

...

}

- Store logs securely with rotation.

- Integrate Sentry or similar for error monitoring.

## 12. Compliance & Retention

- Keep document retention policies in models.

- Track document expiry.

- Implement soft-deletion with `is\_deleted` flag.

- Ensure full traceability for audits.

## 📌 Optional Models Overview

class Document(models.Model):

title = models.CharField(max\_length=255)

version = models.CharField(max\_length=10)

file = models.FileField(upload\_to='documents/')

created\_by = models.ForeignKey(User, on\_delete=models.CASCADE)

department = models.ForeignKey(Department, on\_delete=models.CASCADE)

status = models.CharField(choices=DOC\_STATUS, default='draft')

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

## ✅ Summary Checklist

| Task | Done? |

|-------------------------------|-------|

| JWT Authentication | ✅ |

| RBAC Authorization | ✅ |

| Document Upload & Versioning | ✅ |

| Audit Trail | ✅ |

| RESTful APIs (DRF) | ✅ |

| Unit & Integration Tests | ✅ |

| Security & Validation | ✅ |

| CI/CD (optional) | ⬜ |