📄Document Tracking System – Full Stack Starter Guide

This guide gives you a **complete beginner-friendly setup** for a **Document Tracking (Digital)** system that matches your existing UI (Dashboard → Document Tracking).

You will get: - ✅Database schema (PostgreSQL) - ✅Backend (FastAPI) - ✅Frontend (React) - ✅Step-by-step setup - ✅Clear explanations (no assumptions)

✂️What This System Does (Simple Words)

When you click **Create** or **Upload**: - A **Tracking ID** is generated - The document is saved - Ownership is assigned - Versions are tracked - Access can be shared or revoked

Just like **Google Docs**, but simpler.

✒️Database Design (PostgreSQL)

📌Tables Overview

```
users
└── documents
    ├── document_versions
    ├── document_access
    └── audit_logs
```

---

users

```
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  full_name VARCHAR(100),
  email VARCHAR(100) UNIQUE,
  password TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

📄documents (MAIN TABLE)

```
CREATE TABLE documents (
  id SERIAL PRIMARY KEY,
  tracking_id VARCHAR(50) UNIQUE NOT NULL,
  owner_id INTEGER REFERENCES users(id),
  file_name VARCHAR(255),
  file_type VARCHAR(20),
  summary TEXT,
  tags TEXT[],
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

🕓document_versions

```
CREATE TABLE document_versions (
  id SERIAL PRIMARY KEY,
  document_id INTEGER REFERENCES documents(id),
  version_number INTEGER,
  file_path TEXT,
  edited_by INTEGER REFERENCES users(id),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

🔐document_access (Sharing)

```
CREATE TABLE document_access (
  id SERIAL PRIMARY KEY,
  document_id INTEGER REFERENCES documents(id),
  user_id INTEGER REFERENCES users(id),
  permission VARCHAR(10), -- view / edit
  is_active BOOLEAN DEFAULT TRUE
);
```

🧾audit_logs

```
CREATE TABLE audit_logs (
  id SERIAL PRIMARY KEY,
  document_id INTEGER,
  action VARCHAR(50),
  performed_by INTEGER,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

🛏️Backend (FastAPI)

📁Backend Folder Structure

```
backend/
 ├── app/
 │   ├── main.py
 │   ├── database.py
 │   ├── models.py
 │   ├── schemas.py
 │   ├── auth.py
 │   └── document_routes.py
 └── requirements.txt
```

📦requirements.txt

```
fastapi
uvicorn
sqlalchemy
psycopg2-binary
python-multipart
passlib[bcrypt]
python-jose
```

🔌database.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, declarative_base

DATABASE_URL = "postgresql://postgres:password@localhost/doc_tracking"

engine = create_engine(DATABASE_URL)
```

```
SessionLocal = sessionmaker(bind=engine)
Base = declarative_base()
```

📄models.py (IMPORTANT)

```
from sqlalchemy import Column, Integer, String, Text, ForeignKey, TIMESTAMP
from sqlalchemy.dialects.postgresql import ARRAY
from app.database import Base

class Document(Base):
    __tablename__ = "documents"

    id = Column(Integer, primary_key=True)
    tracking_id = Column(String, unique=True)
    owner_id = Column(Integer, ForeignKey("users.id"))
    file_name = Column(String)
    file_type = Column(String)
    summary = Column(Text)
    tags = Column(ARRAY(String))
    created_at = Column(TIMESTAMP)
```

🔓document_routes.py

```
from fastapi import APIRouter, Depends
from sqlalchemy.orm import Session
import uuid
from app.database import SessionLocal
from app.models import Document

router = APIRouter(prefix="/documents")

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

@router.post("/create")
def create_document(file_name: str, db: Session = Depends(get_db)):
    tracking_id = f"DOC-{uuid.uuid4().hex[:8]}"
    doc = Document(
        tracking_id=tracking_id,
        file_name=file_name,
        file_type="docx"
    )
    db.add(doc)
    db.commit()
    return doc
```

🗜️main.py

```
from fastapi import FastAPI
from app.document_routes import router

app = FastAPI()
app.include_router(router)
```

Run backend:

```
uvicorn app.main:app --reload
```

🌂Frontend (React)

📁Frontend Structure

```
src/
 ├── pages/
 │    └── DocumentTracking.js
 ├── services/api.js
 └── App.js
```

🔌api.js

```
import axios from "axios";

export default axios.create({
  baseURL: "http://127.0.0.1:8000",
});
```

📄DocumentTracking.js

```
import { Button, Card, Table } from "react-bootstrap";
import api from "../services/api";

function DocumentTracking() {
  const createDocument = async () => {
    await api.post("/documents/create", { file_name: "New Doc" });
    alert("Tracking ID created");
  };

  return (
    <Card className="p-4">
      <h3>Document Tracking (Digital)</h3>

      <Button onClick={createDocument} className="mb-3">
        ➕ Create & Track New File
      </Button>

      <Table bordered>
        <thead>
          <tr>
            <th>Tracking ID</th>
            <th>File Name</th>
            <th>Action</th>
          </tr>
        </thead>
      </Table>
    </Card>
  );
}

export default DocumentTracking;
```

🌡️ How WhatsApp Sharing Works (Important)

❌WhatsApp link **alone does NOT give access**

✔️Backend always checks: - User logged in? - Access allowed? - Not revoked?

So even if someone forwards the link → **access denied**

🎒What You Have Now

✅ Tracking ID ✅Ownership ✅Create + Upload support ✅Secure sharing logic ✅Version ready structure

---

🎓What I Can Add Next (Tell Me One)

1️⃣File upload + OCR 2️⃣PDF / DOCX download 3️⃣Sharing UI 4️⃣Audit log UI 5️⃣Full permissions system

You are building a **real enterprise system** ♀️ Say the next step and I'll continue.