Upload File

Tutorials

- FastAPI File Upload Download Tutorial

Installation

```
pip install fastapi uvicorn python-multipart
```

Quick Test

```
# curl -X POST "http://127.0.0.1:8000/uploadfile/" -H "accept: application/json" -H "Content-
Type:multipart/form-data" -F "file=@your.pdf"
@app.post("/uploadfile/")
async def create_upload_file(file: UploadFile | None = None):
    if not file:
        return {"message": "No upload file sent"}
    else:
        return {"filename": file.filename}

@app.post("/tempfile/")
async def temp_file(file: UploadFile | None = None):
    with tempfile.NamedTemporaryFile(delete=False) as temp_file:
        content = await file.read()
        temp_file.write(content)
    temp_file.close

    return {"filepath": temp_file.name, "filename": file.filename, "content_type":
file.content_type, "size": file.size}
```

Client Test with curl

```
curl -X POST "http://127.0.0.1:8000/uploadfile/" -H "accept: application/json" -H "Content-
Type:multipart/form-data" -F "file=@/path/to/your.pdf"
```

Upload single file

```
from fastapi import FastAPI, File, UploadFile, HTTPException
from fastapi.responses import JSONResponse
from typing import List
import os
import shutil
from pathlib import Path

# Create upload directory
UPLOAD_DIR = Path("uploads")
UPLOAD_DIR.mkdir(exist_ok=True)

app = FastAPI(title="FastAPI File Upload Service")

@app.post("/upload/single")
async def upload_single_file(file: UploadFile = File(...)):
    """Upload a single file with basic validation"""
    if file.filename == "":
        raise HTTPException(status_code=400, detail="No file selected")

    file_path = UPLOAD_DIR / file.filename

    with open(file_path, "wb") as buffer:
        shutil.copyfileobj(file.file, buffer)

    return {
        "filename": file.filename,
        "content_type": file.content_type,
        "size": file.size,
        "location": str(file_path)
    }
```

With validations

validators.py:

```
from pathlib import Path
from fastapi import UploadFile
```

```
class DocumentValidator:
    def __init__(self, max_size: int = 10 * 1024 * 1024):  # 10MB default
        self.max_size = max_size
        self.allowed_extensions = {'.pdf', '.txt', '.json'}

    async def validate_file(self, file: UploadFile) -> dict:
        """Check if the document file is valid"""
        result = {"valid": True, "errors": []}

        # Check if user selected a file
        if not file.filename or file.filename.strip() == "":
            result["valid"] = False
            result["errors"].append("No file selected")
            return result

        # Check file extension
        file_ext = Path(file.filename).suffix.lower()
        if file_ext not in self.allowed_extensions:
            result["valid"] = False
            result["errors"].append(
                f"File extension '{file_ext}' not allowed. Use: .pdf, .txt, or .json"
            )

        # Read file to check size
        content = await file.read()
        await file.seek(0)  # Reset file pointer for later use

        # Check file size
        file_size = len(content)
        if file_size > self.max_size:
            result["valid"] = False
            result["errors"].append(
                f"File too large ({file_size:,} bytes). Maximum: {self.max_size:,} bytes"
            )

        return result
```

main.py

```python
from fastapi import FastAPI, File, UploadFile, HTTPException
from fastapi.responses import JSONResponse
from typing import List
import os
import shutil
import uuid
from pathlib import Path
from datetime import datetime
from validators import DocumentValidator

# Create upload directory
UPLOAD_DIR = Path("uploads")
UPLOAD_DIR.mkdir(exist_ok=True)

app = FastAPI(title="FastAPI File Upload Service")

# Create validator instance
doc_validator = DocumentValidator(max_size=25 * 1024 * 1024)  # 25MB limit

@app.post("/upload/single")
async def upload_single_file(file: UploadFile = File(...)):
    """Upload a single file with validation"""

    # Validate the file first
    validation = await doc_validator.validate_file(file)

    if not validation["valid"]:
        raise HTTPException(
            status_code=400,
            detail={
                "message": "File validation failed",
                "errors": validation["errors"]
            }
        )

    # Create unique filename to prevent conflicts
    file_ext = Path(file.filename).suffix
    unique_filename = f"{uuid.uuid4()}{file_ext}"
    file_path = UPLOAD_DIR / unique_filename
```

```
    try:
        with open(file_path, "wb") as buffer:
            shutil.copyfileobj(file.file, buffer)
    except Exception as e:
        raise HTTPException(
            status_code=500,
            detail=f"Failed to save file: {str(e)}"
        )

    return {
        "success": True,
        "original_filename": file.filename,
        "stored_filename": unique_filename,
        "content_type": file.content_type,
        "size": file.size,
        "upload_time": datetime.utcnow().isoformat(),
        "location": str(file_path)
    }

@app.get("/")
async def root():
    return {"message": "FastAPI File Upload Service is running"}
```

Revision #6
Created 2025-12-22 20:21:05 CST by A-Lang (Admin)
Updated 2025-12-22 20:34:53 CST by A-Lang (Admin)