**Pseudo Code — FlexTransfer Hub (Multi-file Upload/Download Manager)**

**1) Upload nhiều file (Pause / Resume / Stop)**

CONSTANT CHUNK\_SIZE ← 512KB

FUNCTION StartMultiUpload(file\_list):

FOR file IN file\_list:

SPAWN\_TASK UploadFile(file)

FUNCTION UploadFile(file):

offset ← 0

session ← CreateUploadSession(file.name, file.size)

IF session.error:

RETURN "Create session failed"

offset ← session.resume\_offset OR 0

state ← "ACTIVE"

WHILE offset < file.size:

IF state = "PAUSED":

WAIT UNTIL state = "ACTIVE"

IF state = "STOPPED":

NotifyServerStop(session.id)

RETURN "Stopped by user"

chunk ← READ(file, offset, CHUNK\_SIZE)

SEND\_WS(session.id, "UPLOAD\_CHUNK", {offset, chunk})

ack ← WAIT\_FOR\_ACK(session.id, offset, TIMEOUT=5s)

IF ack.timeout OR ack.invalid:

RETRIES ← 0

WHILE RETRIES < 3 AND (ack.timeout OR ack.invalid):

RESEND\_WS(session.id, "UPLOAD\_CHUNK", {offset, chunk})

ack ← WAIT\_FOR\_ACK(session.id, offset, TIMEOUT=5s)

RETRIES ← RETRIES + 1

IF ack.timeout OR ack.invalid:

SET state ← "PAUSED"

SHOW\_ERROR("Network issue, auto-paused")

CONTINUE

offset ← offset + SIZE(chunk)

UPDATE\_UI\_PROGRESS(file.name, offset / file.size)

SEND\_WS(session.id, "UPLOAD\_COMPLETE", {})

result ← WAIT\_FOR\_SERVER\_CONFIRM(session.id)

IF result.ok:

RETURN "Upload Success"

ELSE:

RETURN "Upload Completed but finalize failed"

PROCEDURE PauseUpload(file):

file.state ← "PAUSED"

PROCEDURE ResumeUpload(file):

file.state ← "ACTIVE"

PROCEDURE StopUpload(file):

file.state ← "STOPPED"

**2) Download nhiều file (hiện chưa hỗ trợ Resume)**

FUNCTION StartMultiDownload(file\_ids):

FOR id IN file\_ids:

SPAWN\_TASK DownloadFile(id)

FUNCTION DownloadFile(file\_id):

meta ← REST\_GET("/files/" + file\_id, token)

IF meta.not\_found:

SHOW\_ERROR("File not found")

RETURN

stream ← HTTP\_GET\_STREAM("/download?file\_id=" + file\_id, token)

local\_path ← SELECT\_SAVE\_PATH(meta.filename)

OPEN local\_path FOR WRITE AS fout

WHILE chunk ← stream.read():

WRITE fout, chunk

UPDATE\_UI\_PROGRESS(meta.filename, stream.received / meta.size)

CLOSE fout

IF stream.error:

SHOW\_ERROR("Download failed")

RETURN

RETURN "Download Success"

**3) Client tự động Resume khi rớt mạng**

EVENT OnWebSocketClose(session):

MARK\_UI\_STATUS(session.file, "DISCONNECTED")

AUTO\_RETRY\_IN(3 seconds):

ok ← WS\_RECONNECT()

IF ok:

SEND\_WS(session.id, "QUERY\_OFFSET", {})

resp ← WAIT\_MSG("OFFSET\_INFO")

IF resp.ok:

ResumeUploadFrom(resp.offset)

ELSE:

SHOW\_ERROR("Resume query failed")

ELSE:

SHOW\_ERROR("Reconnect failed")

**4) Server WebSocket: nhận chunk + ACK + finalize**

ON\_WS\_MESSAGE(session\_id, msg):

SWITCH msg.type:

CASE "CREATE\_SESSION":

file ← msg.filename

size ← msg.filesize

user ← AUTH(msg.token)

IF NOT user:

SEND\_WS("ERROR", "UNAUTHORIZED")

CLOSE

sess ← DB.create\_upload\_session(user.id, file, size)

temp\_path ← PATH("temp\_uploads/", sess.id, ".part")

CREATE\_EMPTY\_FILE\_IF\_NOT\_EXISTS(temp\_path)

offset ← CURRENT\_FILE\_SIZE(temp\_path)

DB.update\_offset(sess.id, offset)

SEND\_WS("SESSION\_CREATED", {session\_id: sess.id, resume\_offset: offset})

CASE "UPLOAD\_CHUNK":

sess ← DB.get\_session(msg.session\_id)

IF NOT sess OR sess.state IN ("STOPPED", "COMPLETED"):

SEND\_WS("ERROR", "INVALID\_STATE")

RETURN

EXPECTED ← sess.offset

IF msg.offset != EXPECTED:

SEND\_WS("ACK\_ERROR", {expected: EXPECTED, got: msg.offset})

RETURN

APPEND\_TO\_FILE(sess.temp\_path, msg.chunk)

NEW\_OFFSET ← EXPECTED + SIZE(msg.chunk)

DB.update\_offset(sess.id, NEW\_OFFSET)

SEND\_WS("ACK", {offset: NEW\_OFFSET})

CASE "UPLOAD\_COMPLETE":

sess ← DB.get\_session(msg.session\_id)

IF sess.offset == sess.total\_size:

FINAL\_PATH ← PATH("remote\_uploads/", sess.user, sess.safe\_filename)

RENAME\_FILE(sess.temp\_path, FINAL\_PATH)

DB.mark\_completed(sess.id, FINAL\_PATH)

SEND\_WS("COMPLETE\_OK", {})

ELSE:

SEND\_WS("COMPLETE\_ERROR", {reason: "Size mismatch"})

CASE "STOP":

DB.mark\_stopped(msg.session\_id)

SEND\_WS("STOP\_OK", {})

DEFAULT:

SEND\_WS("ERROR", "UNKNOWN\_COMMAND")

**5) Quản lý phiên (session token) & xác thực**

FUNCTION Login(username, password):

user ← DB.get\_user(username)

IF NOT user:

RETURN {error: "User not found"}

IF NOT VERIFY\_PBKDF2(password, user.password\_hash, user.salt):

RETURN {error: "Invalid credentials"}

token ← GENERATE\_RANDOM\_TOKEN()

expiry ← NOW + 24h

DB.insert\_session(user.id, token, expiry)

RETURN {token, expiry}

FUNCTION AuthMiddleware(request):

token ← request.headers["Authorization"]

sess ← DB.get\_session\_by\_token(token)

IF NOT sess OR sess.expiry < NOW:

RETURN UNAUTHORIZED

request.user\_id ← sess.user\_id

RETURN OK

**6) Recycle Bin: xóa mềm & khôi phục**

FUNCTION DeleteFile(file\_id, user\_id):

file ← DB.get\_file(file\_id, user\_id)

IF NOT file:

RETURN "Not found"

DB.mark\_in\_recycle(file\_id, deleted\_at=NOW, ttl\_days=30)

MOVE(file.path, RECYCLE\_PATH(file))

LOG("deleted", file\_id)

RETURN "Soft-deleted"

FUNCTION RestoreFile(file\_id, user\_id):

file ← DB.get\_file(file\_id, user\_id)

IF NOT file OR NOT file.in\_recycle:

RETURN "Not in recycle bin"

MOVE(RECYCLE\_PATH(file), ORIGINAL\_PATH(file))

DB.clear\_recycle\_flag(file\_id)

LOG("restored", file\_id)

RETURN "Restored"

CRON DAILY PurgeExpiredRecycle():

expired ← DB.list\_recycle\_expired(NOW)

FOR f IN expired:

DELETE\_FILE(RECYCLE\_PATH(f))

DB.mark\_permanently\_deleted(f.id)

**7) Preview (ảnh, PDF, video, text)**

FUNCTION Preview(file\_id, user\_id):

file ← DB.get\_file(file\_id, user\_id)

IF NOT file:

RETURN 404

mime ← DETECT\_MIME(file.path)

IF mime IN ["image/\*", "application/pdf", "video/\*", "audio/\*", "text/\*"]:

RETURN STREAM(file.path, headers={ "Content-Type": mime })

ELSE:

RETURN 415 // Unsupported preview

**8) Thao tác file: Rename / Move / Create Folder**

FUNCTION CreateFolder(parent\_id, name, user\_id):

SAFE ← SANITIZE\_FOLDER\_NAME(name)

RETURN DB.insert\_folder(user\_id, parent\_id, SAFE)

FUNCTION RenameFile(file\_id, new\_name, user\_id):

file ← DB.get\_file(file\_id, user\_id)

SAFE ← SANITIZE\_FILENAME(new\_name)

NEW\_PATH ← PATH(DIR(file.path), SAFE)

RENAME\_FILE(file.path, NEW\_PATH)

DB.update\_file\_name(file\_id, SAFE, NEW\_PATH)

RETURN "Renamed"

FUNCTION MoveFile(file\_id, target\_folder\_id, user\_id):

file ← DB.get\_file(file\_id, user\_id)

target ← DB.get\_folder(target\_folder\_id, user\_id)

NEW\_PATH ← PATH(FOLDER\_FS\_PATH(target), BASENAME(file.path))

MOVE(file.path, NEW\_PATH)

DB.update\_file\_folder(file\_id, target\_folder\_id, NEW\_PATH)

RETURN "Moved"

**9) Thống kê (Stats)**

FUNCTION GetStats(user\_id):

count\_files ← DB.count\_files(user\_id, status="completed")

total\_size ← DB.sum\_file\_size(user\_id, status="completed")

count\_folders ← DB.count\_folders(user\_id)

in\_recycle ← DB.count\_files(user\_id, in\_recycle=true)

RETURN {

"files": count\_files,

"size": total\_size,

"folders": count\_folders,

"recycle": in\_recycle

}

**10) Xử lý lỗi & Retry (client-side)**

FUNCTION SafeSendChunk(session\_id, chunk, offset):

TRY\_TIMES ← 0

REPEAT:

SEND\_WS(session\_id, "UPLOAD\_CHUNK", {offset, chunk})

ack ← WAIT\_FOR\_ACK(session\_id, offset, TIMEOUT=5s)

IF ack.valid:

RETURN "OK"

TRY\_TIMES ← TRY\_TIMES + 1

UNTIL TRY\_TIMES = 3

AUTO\_PAUSE(session\_id)

SHOW\_ERROR("Network unstable, paused")

RETURN "FAILED"

FUNCTION AutoPause(session\_id):

UPDATE\_UI\_STATUS(session\_id, "PAUSED")

STORE\_LOCAL("resume\_offset", CURRENT\_OFFSET(session\_id))