API Pseudocode and Detailed Flow Description

# Common Step for All APIs: JWT Authentication

1. Receive Authorization Header: Extract JWT token from request header `Authorization: Bearer <token>`.

2. Validate Token: Decode JWT using shared secret or public key. If token is invalid or expired, return 401 Unauthorized.

3. Extract User Info: Extract `user\_id`, roles, and permissions from token claims.

4. Log Authentication Details: Log authenticated user\_id and access time.

# Create Memo API

## Endpoint

POST /create\_new\_memo

## Receive Request

Accept JSON with `user\_id`, `client\_name`, and `content\_inputs`.

## Logging

Log request with timestamp and `user\_id`.

## Input Validation

Check presence and types of `user\_id`, `client\_name`, `content\_inputs`. Return 400 if validation fails.

## Insert Placeholder into PostgreSQL

Insert `user\_id`, `client\_name`, `status='Processing'`, `created\_at=NOW()` into `memos`. Log the generated `memo\_id`.

## Trigger Async Content Generation

Send data to background job/worker. Log generator invocation.

## On Completion - Update Memo

Update memo record with generated content and `status='Completed'`. Log update result.

## Return Response

JSON: `{ memo\_id, client\_name, status }`

## Error Handling

400: Invalid request. 500: Log and return internal error message.

# Check Processing Status API

## Endpoint

GET /check\_processing\_status/{user\_id}/{memo\_ids}

## Receive Request

Path params: `user\_id`, `memo\_ids` (comma-separated).

## Logging

Log `user\_id` and `memo\_ids`.

## Validation

Validate user\_id and memo\_ids are correct and belong to user.

## Query PostgreSQL

SELECT memo\_id, processing\_status FROM memos WHERE user\_id = ? AND memo\_id IN (...)

## Return Response

JSON: `{ memo\_id: processing\_status, ... }`

## Error Handling

400: Bad input. 500: DB error logged.

# Get Memo Data API

## Endpoint

GET /get\_memo\_content/{memo\_id}

## Receive Request

Path param: `memo\_id`

## Logging

Log `memo\_id` and `user\_id`

## Validation

Check if `memo\_id` belongs to authenticated `user\_id`

## Query PostgreSQL

SELECT \* FROM memos WHERE memo\_id = ?

## Update Access Timestamp

UPDATE memos SET last\_opened\_at = NOW() WHERE memo\_id = ?

## Return Response

Full memo content

## Error Handling

404: Memo not found or access denied. 500: DB errors logged.

# List Memos API

## Endpoint

GET /list\_memos/{user\_id}

## Receive Request

Path param: `user\_id`

## Logging

Log memo list request.

## Validation

Confirm authenticated `user\_id` matches path param.

## Query PostgreSQL

SELECT memo\_id, client\_name, status, updated\_at FROM memos WHERE user\_id = ?

## Return Response

List of memos

## Error Handling

500: DB query failure.

# List Recent Memos API

## Endpoint

GET /list\_recent\_memos/{user\_id}

## Receive Request

Path param: `user\_id`

## Logging

Log recent memos request.

## Validation

Confirm ownership of memos.

## Query PostgreSQL

SELECT \* FROM memos WHERE user\_id = ? ORDER BY updated\_at DESC LIMIT 2

## Return Response

JSON list of last 2 memos

## Error Handling

500: DB error.

# Get Client Names API

## Endpoint

GET /get\_client\_names

## Logging

Log request.

## Query Snowflake

SELECT DISTINCT client\_name FROM client\_attributes\_view

## Return Response

JSON list of client names

## Error Handling

500: Snowflake failure.

# Get Client Contacts API

## Endpoint

GET /get\_client\_contacts?client\_name=XYZ

## Receive Request

Query param: `client\_name`

## Logging

Log query.

## Validation

Confirm `client\_name` is present

## Query Snowflake

SELECT \* FROM client\_contacts\_view WHERE client\_name = ? AND type IN ('External', 'External Consultant')

## Return Response

Contact list JSON

## Error Handling

400: Missing input. 500: Snowflake error.

# Update Memo Content API

## Endpoint

POST /update\_memo\_content

## Receive Request

JSON: `user\_id`, `memo\_id`, `section\_name`, `updated\_content`

## Logging

Log section update request

## Validation

Confirm section is editable. Ensure memo\_id belongs to user.

## Update PostgreSQL

UPDATE memos SET section\_name = updated\_content WHERE memo\_id = ?

## Insert Version Record

INSERT INTO versions (memo\_id, section, content, updated\_at)

## Return Response

JSON: `{ status: 'Success', memo\_id, updated\_section }`

## Error Handling

400: Locked section. 500: Log DB failure

## Logging

Log success/failure and timestamps