Activity 6 lab

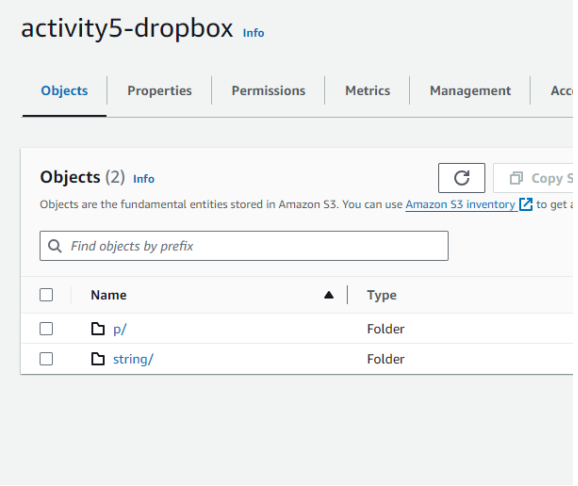
☑ Is study	☐
↗ Project	☁️ Cloud comp

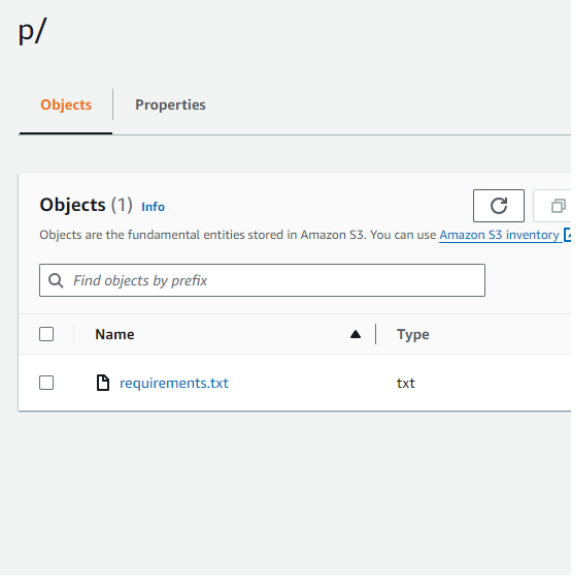
S3 design

> 💡 How your many users will store their files in the same S3 bucket
> • How you will know which file belongs to which user
> • How you will ensure that if user A uploads a file, and user B uploads another file with the same name, that they are NOT THE SAME object in your bucket
> • etc.
> • Draw/write up your design as 1-2 slides and include them with your homework submission.

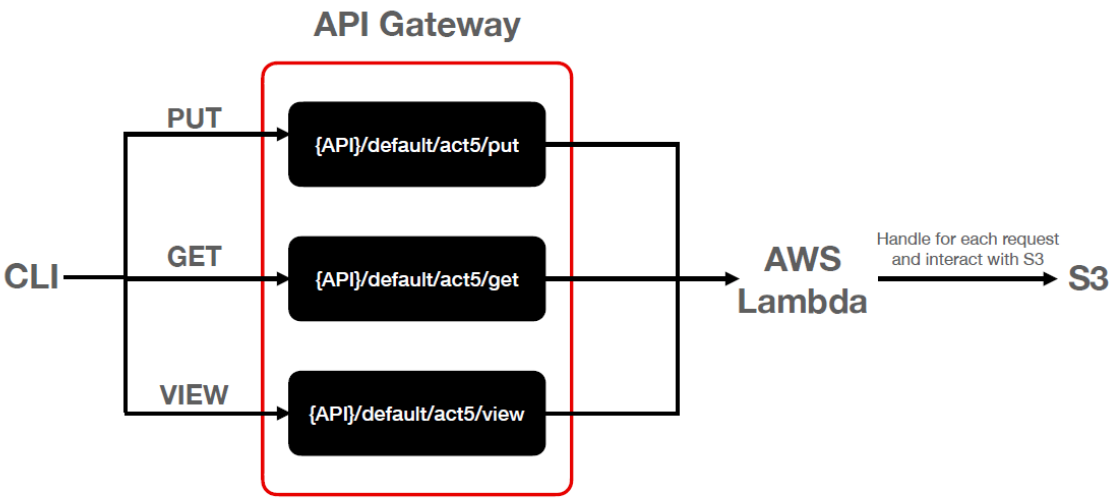
ใช้วิธีการจัดการเมื่อมี User หลายคนโดยการสร้าง Folder แยกสำหรับแต่ละ user ทำให้เมื่อมีชื่อไฟล์เดียวกันก็จะไม่ใช้ object เดียวกันเพราะ key เป็นอันเดียวกัน

- ซึ่ง user ถูกแบ่งให้ distinct ด้วย email ในอนาคตก็จะสร้างชื่อ folder ตาม user email





Overview system

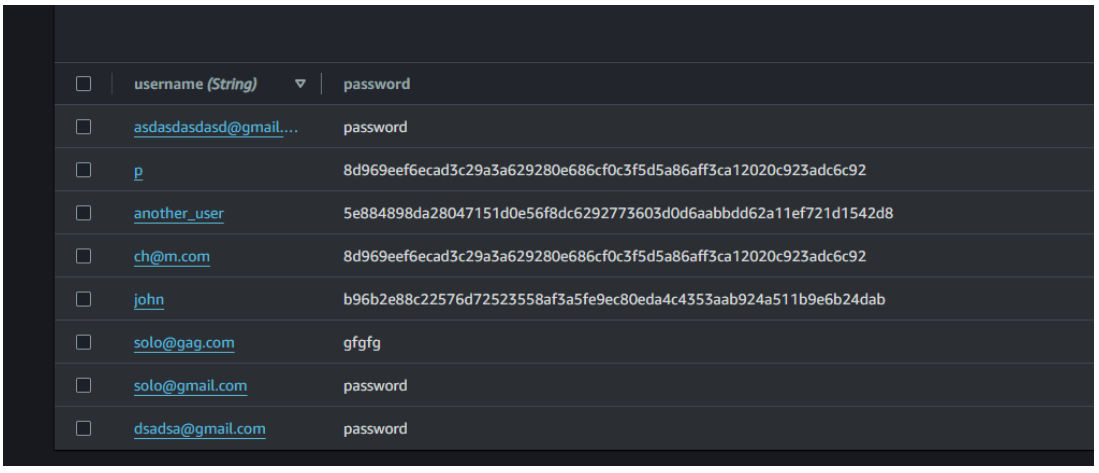


What you will need to set up in DynamoDB

Table Design:

We'll create two separate tables for user information and file metadata:

1. **"myDropboxUsers" Table:**
   - **Primary Key:** `username` (string)
   - **Attributes:**
     - `password` (string, securely hashed and salted)



username (String)	password
asdasdasdasd@gmail....	password
p	8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92
another_user	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8
ch@m.com	8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92
john	b96b2e88c22576d72523558af3a5fe9ec80eda4c4353aab924a511b9e6b24dab
solo@gag.com	gfgfg
solo@gmail.com	password
dsadsa@gmail.com	password

2. **"myDropboxShares" Table:**
   - **Primary Key:** username (string)
   - **Attributes:**
     - `owner` (string, username referring to "users" table)
     - `filename` (string)
     - `shared_with` (list of usernames from "users" table, optional)

	username *(String)*	filename *(String)*	shareTo
	a	b.txt	b
	dsadsa@gmail.com	new.txt	solo@gag.com
	ch@m.com	requirements.txt	p

Readme for myDropboxClient:

Description:

myDropbox is a Python-based command-line application that allows you to securely manage your files through an API. It provides functionalities for:

- Creating new user accounts
- Logging in to existing accounts
- Uploading and downloading files
- Viewing a list of your files

Installation:

1. Install required dependencies:

```
pip install -r requirements.txt
```

2. Create a `.env` file in the root directory of the project and add the following line:

```
API_GATEWAY = "https://2rgp0s50l9.execute-api.us-west-2.
amazonaws.com/default"
```

Usage:

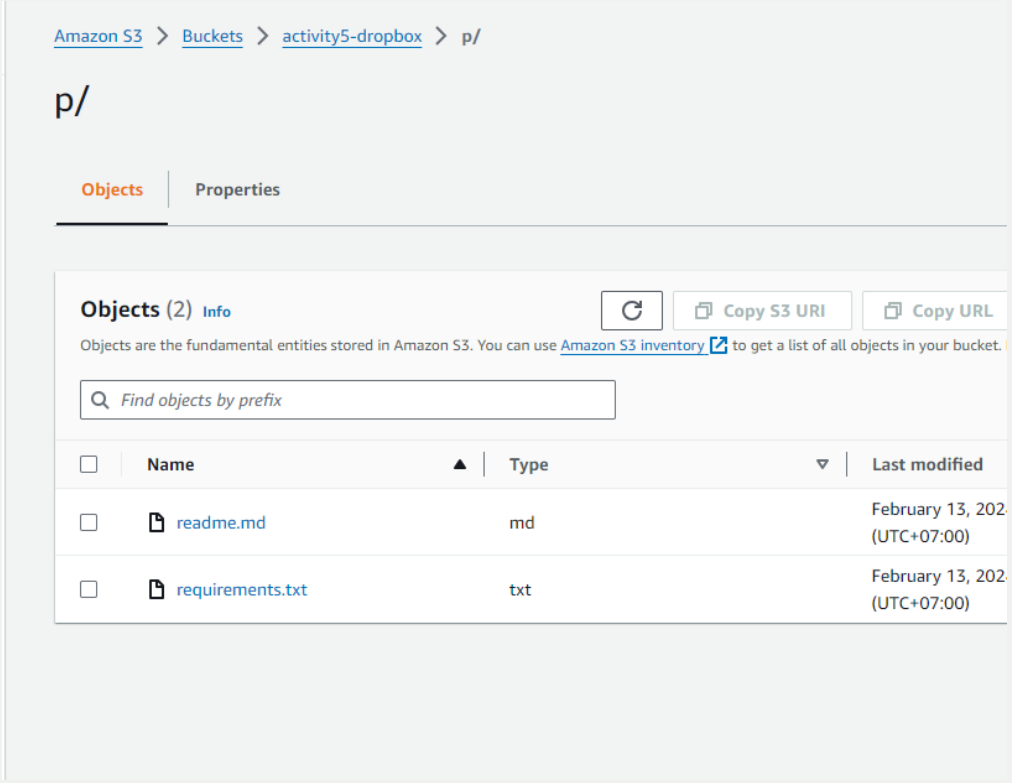
1. Start the application:

   ```
   python myDropboxClient_6330078421.py
   ```

2. Follow the on-screen instructions to create a new user account, log in, or use available commands:

   - `newuser username password password`: Create a new user
   - `login username password`: Log in to existing account
   - `put filename`: Upload a file
   - `get filename username`: Download a file
   - `view`: List your files
   - `quit`: Exit the application

💡 This is my sample data in S3



- you can try command to add data into this bucket username = `p`
  - `view` (still can view only username = `p`)
  - `get readme.md p`
  - `put <filename.txt>` to push data into username = `p`

myDropbox Application

This is a command-line interface for a file sharing application similar to Dropbox. It uses the `requests` library to make HTTP requests to an API, and `boto3` to interact with AWS services. The script also uses `hashlib` for password hashing and `base64` for file encoding.

Installation

1. Clone this repository to your local machine.
2. Install the required Python packages using pip:

```
pip install -r requirements.txt
```

Usage

Run the script using Python:

```
python myDropboxClient_6330078421.py
```

The script will print out the available commands:

- `newuser username password password`: Create a new user
- `login username password`: Login to your account
- `logout`: Logout from your account
- `put filename`: Upload a file
- `view`: List your files that you have access to
- `get filename username`: Download a file from specific user owner
- `share filename recipientUsername`: Share a file with another user
- `quit`: Exit the program

Enter a command at the `>>` prompt to execute it.

Environment Variables

The script uses the following environment variables:

- `API_GATEWAY`: The URL of the API gateway.

These variables should be set in a `.env` file in the same directory as the script. The script uses the `dotenv` package to load these variables.

API Documentation

Register User

Endpoint: `/act5/api/v1/register`

Method: `POST`

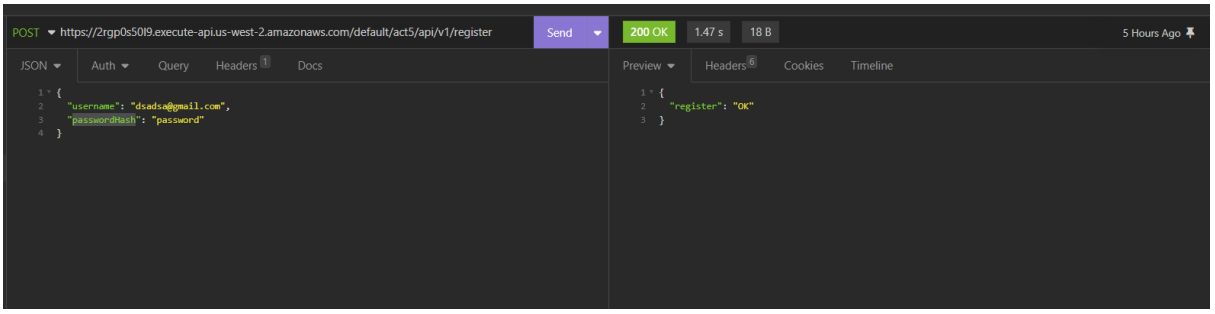
Body:

```
{
    "username": "<username>",
    "passwordHash": "<passwordHash>"
}
```

Response:

```
{
"register": "OK"
}
```

Example:



Login User

Endpoint: `/act5/api/v1/login`

Method: `POST`

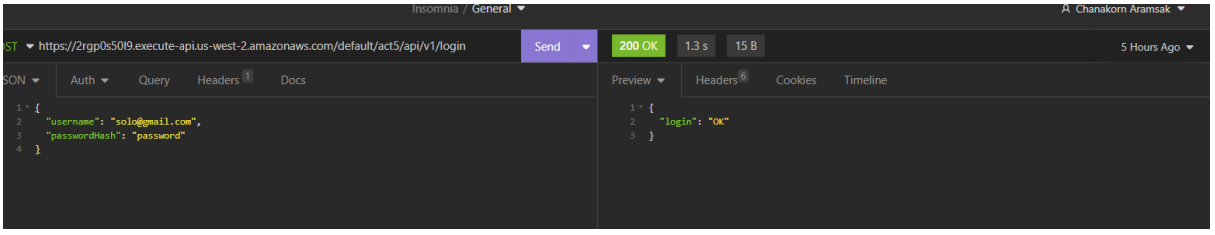
Body:

```
{

"username": "<username>",
"passwordHash": "<passwordHash>"
}
```

Response:

```
{

"login": "OK"

}
```

Example:



Upload File

Endpoint: `/act5/api/v1/put`

Method: `POST`

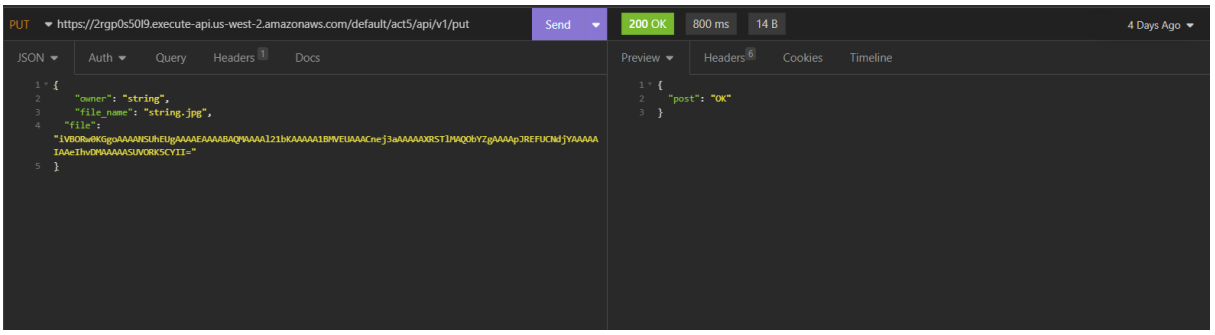
Body:

```
{

"owner": "<owner>",

"file_name": "<file_name>",

"file": "<base64_encoded_file_content>"

}
```

Response:

```
{

"post": "OK"

}
```

Example:



Get File URL

Endpoint: `/act5/api/v1/get`

Method: `GET`

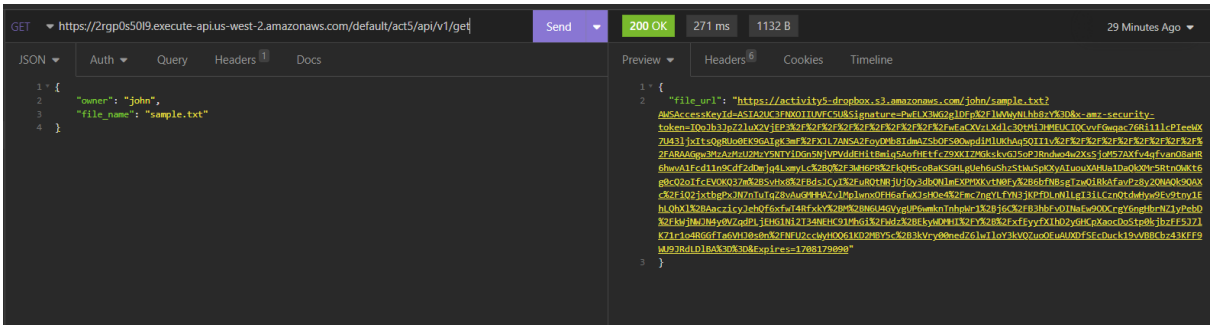
Body:

```
{

"owner": "<owner>",

"file_name": "<file_name>"

}
```

Response:

```
{

"file_url": "<presigned_s3_url>"

}
```

Example:



List Files

Endpoint: `/act5/api/v1/view`

Method: `GET`

Body:

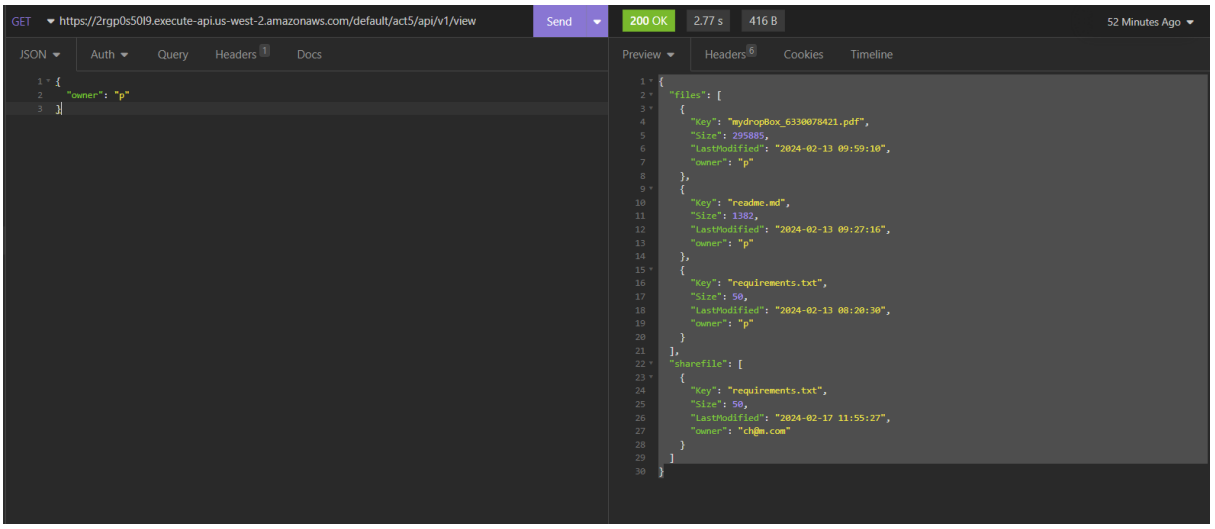
```
{

"owner": "<owner>"

}
```

Response:

```
{

"files": [

{

"Key": "<file_name>",

"Size": "<file_size>",

"LastModified": "<last_modified_date>",

"owner": "<owner>"

}

],
```

```
"sharefile": [

{

"Key": "<file_name>",

"Size": "<file_size>",

"LastModified": "<last_modified_date>",

"owner": "<owner>"

}

]

}
```

Example:



Share File

Endpoint: `/act5/api/v1/share`

Method: `POST`

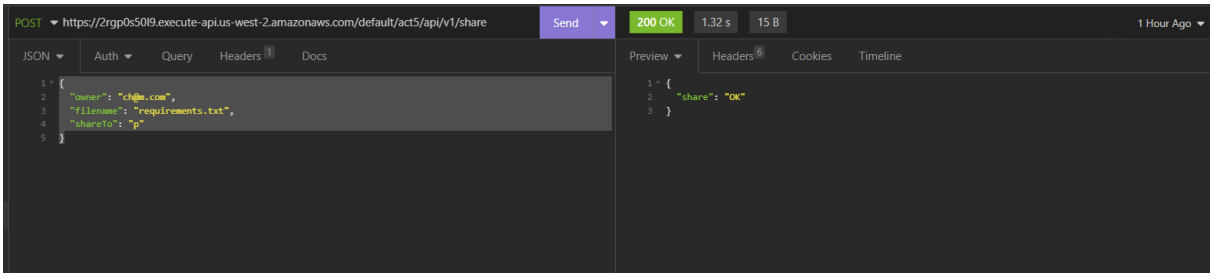
Body:

```
{

"owner": "<owner>",

"filename": "<filename>",

"shareTo": "<shareTo>"

}
```

Response:

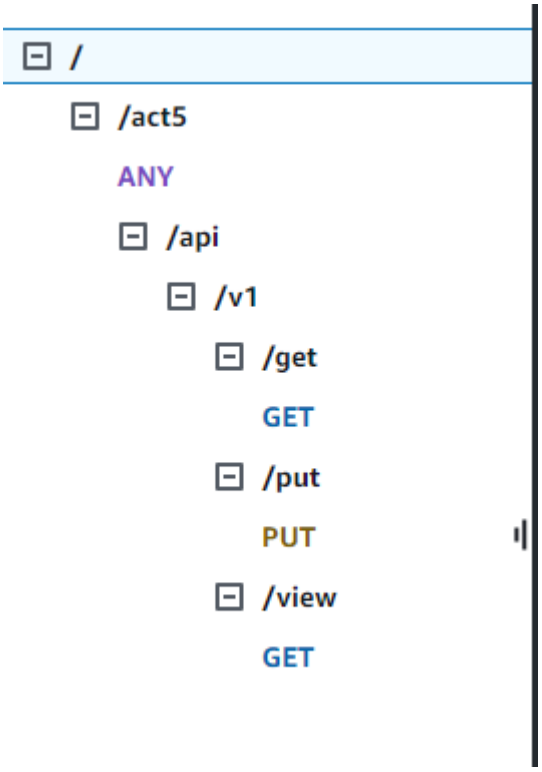
```
{

"share": "OK"

}
```

Example:



Prerequisites

- AWS Account
- Amazon S3 bucket
- Amazon Lamda function configured with API Gateway \


/
/act5
ANY
/api
/v1
/get
GET
/put
PUT
/view
GET