CS310 FINAL REPORT

Ruiyang Peng, Jingze Cheng, Xiaoyuan Zhang, Nan Zhou

December 2023

1 Introduction

Our project is focused on developing an application specifically designed for image dehazing. To access this advanced feature, users are required to create an account. For those who are new to our application, we provide a straightforward registration process. Once registered, users can effortlessly log in and begin utilizing the application's core functionality.

After logging in, users have the capability to upload images directly from their local storage to the application. Furthermore, the application maintains a history of the users' jobs, enabling them to effortlessly track their past activities. Most importantly, users can retrieve the results of their image processing tasks by simply using the corresponding job ID, ensuring a seamless and efficient experience. Besides, users can also delete all the history if they want.

2 Backend Features

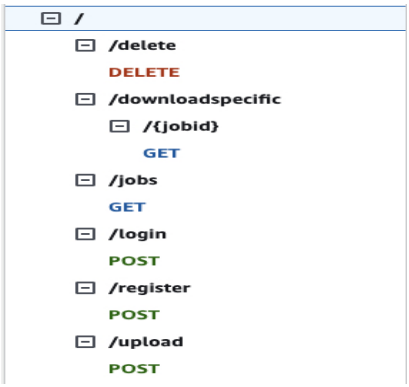

Figure 1: API-gateway

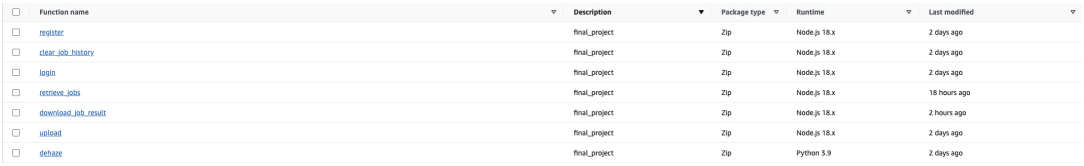

Figure 2: Lambda Functions

Function	register	login	clear_job_history	retrieve_jobs	download_job_result	dehaze	upload
Trigger method	/register	/login	/delete	/jobs	/download	triggered by s3	/upload

Table 1: Functions and trigger method

Our application architecture leverages AWS Lambda functions for various operations, with most functions being invoked via the API Gateway. However, the dehaze function is distinctively designed to be triggered by S3. This function springs into action automatically whenever .jpg images are uploaded to a designated S3 bucket using the application's upload feature. Upon activation, the dehaze function processes these images and seamlessly outputs the results back into the same S3 bucket. This setup ensures an efficient, event-driven workflow, where image dehazing occurs in real-time, aligning perfectly with the dynamic and responsive nature of our application.

userid	email	username	password	bucketfolder

Table 2: Table user_info columns

jobid	userid	status	originaldatafile	datafilekey	resultsfilekey

Table 3: Table jobs columns

In our project, the database is structured with two primary tables: the user_info table and the jobs table.The user_info table is meticulously designed to record essential user details, including userid, email, username, password, and bucketfolder. When users register through the application, their information is securely stored in this table. Subsequently, during the login process, the application retrieves and verifies the accuracy of this stored information, ensuring that users' credentials match our records.

Regarding the jobs table, it comes into play when users upload images. Each image upload initiates the creation of a new job entry in this table. As the dehaze function processes an image, it dynamically updates the corresponding entry in the Jobs table. This update involves changing the job's status from 'pending' to either 'completed' or 'error', depending on the outcome of the process, and also includes the path to the result file. This robust database design facilitates efficient tracking and management of user interactions and image processing tasks within our application.

3 Frontend Features

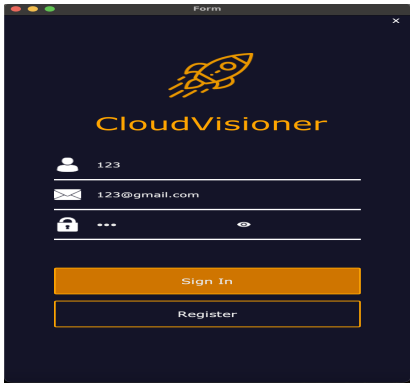

Figure 3: Login and Register

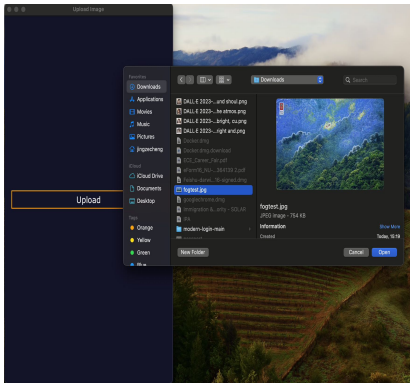

Figure 4: Upload image

Figure 5: Joblist and download

- **User Authentication:** Implemented using robust PyQt5 GUI elements, allowing users to securely register and log in with credentials like username, email, and password.
- **Image Uploading:** Once login successfully, it will automatically jump to the upload interface in Figure 4. Users can upload images in formats like PNG, JPEG, and JPG. Uploaded images are then processed in the backend.
- **Job List Management:** A dynamic display of tasks, represented as jobs, is available. This feature utilizes PyQt5's list widgets to present each job's ID, status, and original file name.
- **Downloading Job Results:** Users can download the results of specific jobs locally by entering the job ID. PyQt5's interface elements are used to handle user input.
- **Clearing Job List:** Users can clear their task history, a feature that enhances the application's usability and user control.

4 Conclusion

The client-side application is built using PyQt5, offering a rich set of GUI components that enhance user experience and interaction. The backend API, hosted on AWS Lambda, communicates with the client over RESTful endpoints, ensuring efficient processing of image data and task management. This PyQt5-based client application combines a user-friendly graphical interface with powerful backend services. It provides a seamless and efficient experience for users looking to manage and retrieve processed image data.