
Automatic Shopping Guide

Benli Wang

Yu Sun

December 10, 2020



Introduction

Since Machine Learning technology is becoming more reliable and accurate, there is more information that can be detected from the image and videos. Meanwhile, online shopping is also becoming more and more popular nowadays, especially during the COVID-19 period. Thus we now have an idea to create an application that can online detect the items in the given images and generate some shopping links for our users.

Some relative applications

Currently, if customers want to purchase items on Amazon, they have to type the keywords such as the brand name, item category and even the specific model number. However, most people don't know the full information they are looking for. For example, when I see a nice chair in the library and want to purchase a similar chair like that, the only keyword I have is the simple word "chair". It definitely will take me a lot of time to pick online. Now, Google has the functionality to read the information from an image and generate some relative links for the clients. It might be the only solution that can contribute people to buy products online. However, the biggest online shopping website in China, www.taobao.com has the functionality that allows clients to upload images to find products, it is a more efficient way for shopping.

Technical functionality for our application

- allow users to register accounts
- user login by username and password, recover password by email
- user can change password
- store user infomation on AWS dynamoDB
- allow user to upload local images
- use AWS Rekognition to detect the products
- generate products' links for user
- keep uploaded images and generated links on AWS S3
- allow user to delete uploaded images and links
- image saved on AWS S3 only
- monitor number of accounts and number of image uploaded
- send database detail to administrator by email

Application user instruction

Website link: <https://4k4f1wahxg.execute-api.us-east-1.amazonaws.com/dev>



The sign-in page features a title "Please Sign In" at the top. Below it are two input fields: "Username" and "Password", each with a placeholder text inside. A horizontal progress bar consisting of five colored segments (green, yellow, orange, red, blue) is positioned below the password field. At the bottom are two buttons: a green "Sign In" button on the left and a blue "Register new Account" button on the right. A small blue link "Forgot Password?" is located to the right of the "Sign In" button.

Enter

the username and password and click the “Sign in” to login to index page

Click “Register new Account” to create account

Click “Forgot password” to reset password



The register page features a title "Please Register" at the top. Below it are three input fields: "Username", "Password", and "Email", each with a placeholder text inside. A horizontal progress bar consisting of five colored segments (green, yellow, orange, red, blue) is positioned below the email field. At the bottom are two buttons: a green "Confirm Register" button on the left and a blue "Back to Sign in Page" button on the right.

Fill up the username, password and email and click “Confirm Register” to create account

Click “Register new Account” can switch back to login page

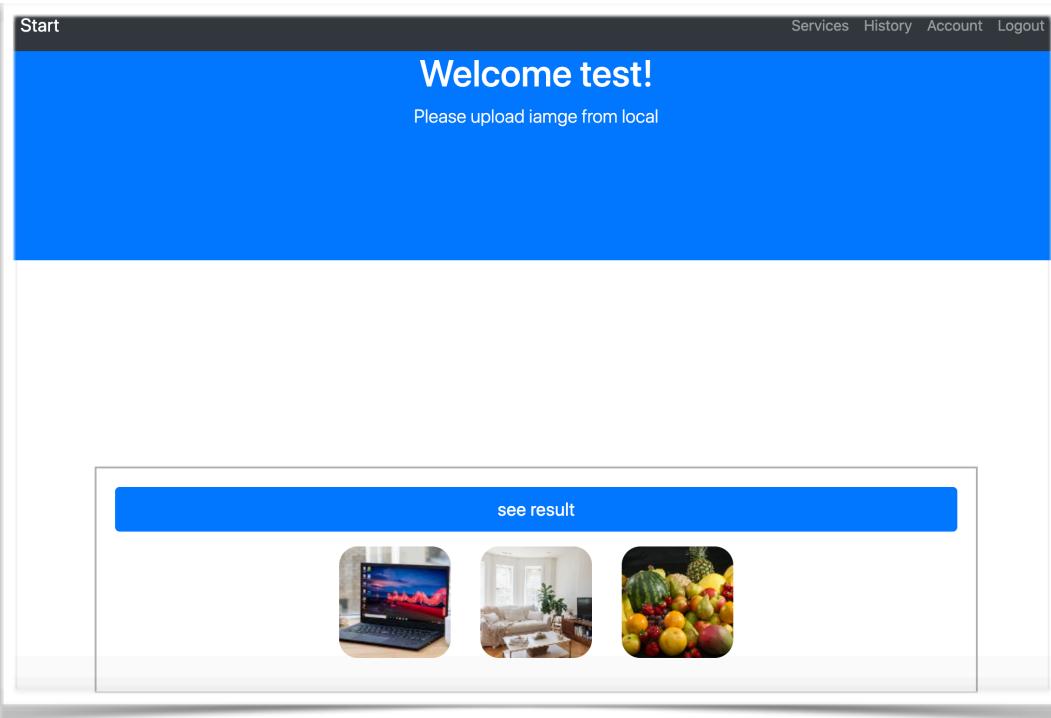
Recover your Account

Username

send new password

Back to Sign in Page

Enter the username to reset the password, the new password will automatically send to the registration email address



User can drag an image from local or click the box to select image to upload

The application allow user to upload multiple images

Click "see result" to see the detection results

HISTORY IMAGE & LINKS



Transportation

Bicycle

Vehicle

Bike

[Delete this image and links](#)

Result and history will showed this way

Click "Delete this image and links" to delete all data from S3 and dynamoDB

Click "Bicycle" will open the link on amazon

The screenshot shows the Amazon.ca homepage with a search bar containing 'Bicycle'. The top navigation bar includes 'Hello Select your address', a search icon, and account options like 'Hello, Sign in Account & Lists Returns & Orders'. A blue button on the right says 'Cart' with a '0' icon. The main search results area displays a grid of bicycle-related products. On the left, there's a sidebar with filters for 'Eligible for Free Shipping', 'Department' (Sports & Outdoors), 'Avg. Customer Review' (4.5 stars and up), and 'Brand' (Eurobike, Schwinn, YaeCCC, Soozier, DJ Bikes, NCM, Jetson). The main content area shows three cycling light products by 'DON PEREGRINO' with prices of \$29.99, \$25.49 (a 'Limited time Deal'), and \$29.99. Below these are four different bicycle models displayed horizontally.

Example shopping website on Amazon

Services History Account Logout

We have navigation bar for scrolling the web page.

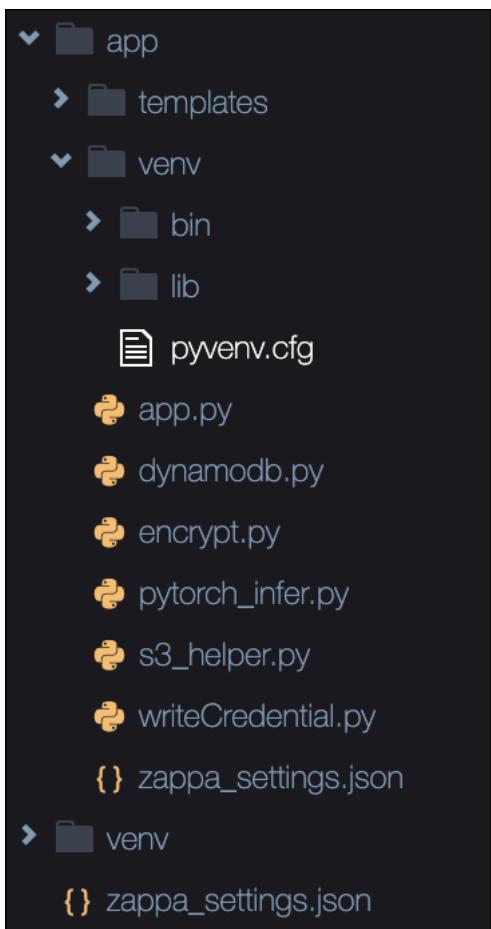
Change Password

New Password

Change Password

On the bottom of the index page, enter the new password and click “change password”
Will change the login password

Architecture of application



All html, css, javascript files are stored in templates directory.

app.py does the routing for flask and interact with dynamodb.py, encrypt.py, pytorch_infer.py, s3_helper.py.

dynamodb.py connect the application with AWS dynamoDB, create table for account info and image info

encrypt.py does the hashing for password

pytorch_infer.py does the image detection for uploaded pictures by calling AWS Rekognition

s3_helper.py upload, download and delete data from S3

Background Process

send_mail: send the database email to administrator

dynamoDB_monitor: return the database information

analyze_data: format the data from dynamoDB and help generate email message

Call while loop and loop every 60 sec to monitor the dynamoDB information.

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with various navigation links like 'Buckets', 'Storage Lens', 'AWS Organizations settings', and 'Feature spotlight'. The main area is titled 'Objects (23)' and contains a table of files. The table has columns for Name, Type, Last modified, Size, and Storage class. The files listed are all named 'a_...' followed by a unique identifier and have a type of 'jpeg'.

	Name	Type	Last modified	Size	Storage class
	a_1607660573.jpeg	jpeg	December 10, 2020, 23:22 (UTC-05:00)	12.8 KB	Standard
	a_1607660582.jpeg	jpeg	December 10, 2020, 23:23 (UTC-05:00)	6.1 KB	Standard
	a_1607660592.jpeg	jpeg	December 10, 2020, 23:23 (UTC-05:00)	437.7 KB	Standard
	a_1607674742.jpeg	jpeg	December 11, 2020, 03:19 (UTC-05:00)	202.5 KB	Standard
	a_1607674822.jpeg	jpeg	December 11, 2020, 03:20 (UTC-05:00)	202.5 KB	Standard
	a_1607674899.jpeg	jpeg	December 11, 2020, 03:21 (UTC-05:00)	202.5 KB	Standard

S3 screenshot

The screenshot shows the AWS DynamoDB console with two tables: 'Accounts' and 'Images'. Both tables have 'Items' tabs selected. The 'Accounts' table has items for users 'a', 'aa', and 'b'. The 'Images' table has items for user 'a' with file names like 'a_1607660573.jpeg', 'a_1607660582.jpeg', and 'a_1607660592.jpeg'. Each item includes fields for 'user_name', 'img_name', 'category', and 'url'.

user_name	img_name	category	url
a	a_1607660573.jpeg	3	https://www.amazon.ca/s?k=Vehicle https://www.amazon.ca/s?k=
a	a_1607660582.jpeg	3	https://www.amazon.ca/s?k=Sport https://www.amazon.ca/s?k=F
a	a_1607660592.jpeg	3	https://www.amazon.ca/s?k=Couch https://www.amazon.ca/s?k=

Accounts table and
Images table from
DynamoDB

Cost Model

Assumption: each average user login to the website 2 times a day, save 50 images (avg size 0.5M per image) for 6 months, click the shopping links 3 times for every login, delete 10 history images for 6 months. Each lambda request only needs 512 MB memory, time 6ms.

Calculation example for 10 users:

Number of login requests: $10 \times 2 \times 3 \times 6 \times 30 = 10800$

Number of access data request: $10 \times 60 = 600$

Approximate request count ≈ 12500

Lambda & API Gateway cost = $\$0.000000083 \times 12500 \times 6 = \0.0006225

S3 usage: $10 \times 50 \times 0.5 / 1024 = 0.24414\text{G}$

S3 & DynamoDB cost: $0.24414 \times 0.023 = \$0.00002246$

number AWS Rekognition image: $10 \times 50 = 500$

AWS Rekognition cost = $\$0.001 \times 500 = \0.5

Total cost $\approx \$0.501$

Calculation example for 1000 users:

Number of login requests: $1000 \times 2 \times 3 \times 6 \times 30 = 1080000$

Number of access data request: $1000 \times 60 = 600$

Approximate request count ≈ 1250000

Lambda & API Gateway cost = $\$0.000000083 \times 1250000 \times 6 = \0.06225

S3 usage: $1000 \times 50 \times 0.5 / 1024 = 24.414\text{G}$

S3 & DynamoDB cost: $24.414 \times 0.023 = \$0.002246$

number AWS Rekognition image: $1000 \times 50 = 50000$

AWS Rekognition cost = $\$0.001 \times 50000 = \50

Total cost $\approx \$50.1$

Calculation example for 1000,000 users:

Number of login requests: $10000000 \times 2 \times 3 \times 6 \times 30 = 1080000000$

Number of access data request: $10000000 \times 60 = 600000000$

Approximate request count ≈ 1250000000

Lambda & API Gateway cost = $\$0.000000083 \times 1250000000 \times 6 = \62.25

S3 usage: $1000000 \times 50 \times 0.5 / 1024 = 24414\text{G}$

S3 & DynamoDB cost: $24414 \times 0.023 = \$2.246$

number AWS Rekognition image: $1000000 \times 50 = 50000000$

AWS Rekognition cost = $\$0.001 \times 50000000 = \50000

Total cost $\approx \$50100$