**FIT5225 Cloud Computing and Security**

TAG STORE

Kshitij Pandey – 30764505

Chayan Agarwal – 30767717

Manik Guglani – 30768993

Somya Milendra Saxena - 30469406

Table of Contents

[AWS Architecture: 2](#_heading=h.c9c752k2llab)

[Role of Team Members: 3](#_heading=h.f403cbydepn1)

[User Manual: 4](#_heading=h.yge3wjea5ihv)

[Functionalities Covered 4](#_heading=h.2cfyn2ynz919)

[Steps for Registration 4](#_heading=h.1fob9te)

[Steps for Login 5](#_heading=h.i3jyzr8iktqz)

[Steps to Upload an image 6](#_heading=h.raptj7rz0b7y)

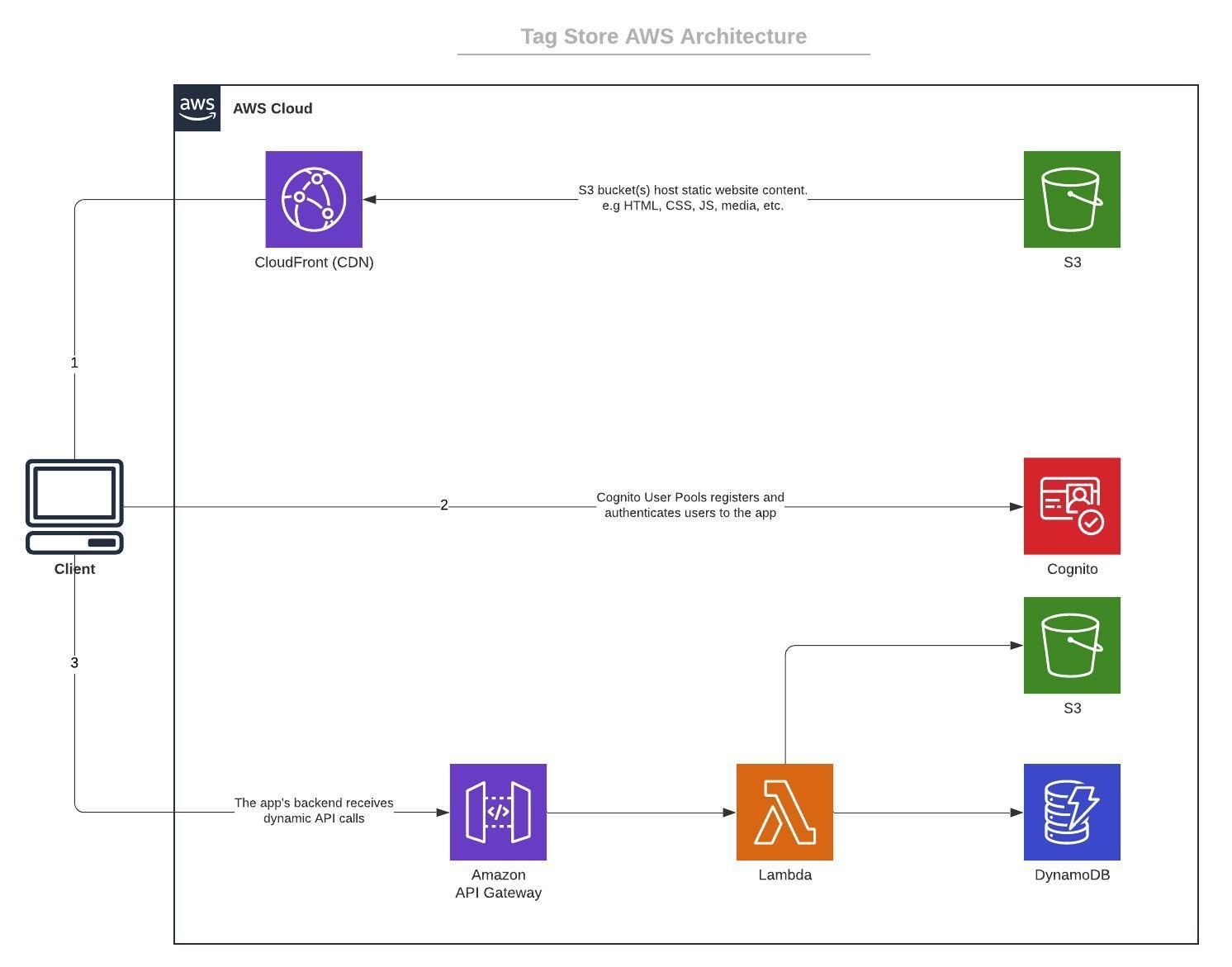
[Steps to Query tag 7](#_heading=h.ow67249500ur)

[GitHub Link: 8](#_heading=h.3znysh7)

[Improving Performance and Reducing Failure: 8](#_heading=h.tyjcwt)

[Real World Solution- Person detection: 8](#_heading=h.3dy6vkm)

## AWS Architecture:



Backend:

For this application we are using S3 buckets to store the assets like the images uploaded by the users and to store the static files of the website to be hosted. To authenticate the users we created a User Pool which lets us store the user details like email, first name, last name, etc. and manages that data for us. Cognito also provides a mechanism to sign up and login users and provides different ways to activate their accounts. We used email verification of Cognito to let users verify their email to gain access to our services. For authorization we are using Cognito Identity Pool which has roles and policies for authenticated users attached to it. IAM policies are used to provide fine grained access to the AWS resources so that they can access the other AWS resources as needed. For example, each of the Lambda functions have an IAM role and a policy attached to them which allows them to access either the S3 bucket or the DynamoDB table. Data for tags like the tag detected the URL associated with it is being stored in a DynamoDB table with the tag as the partition key. To expose our services outside the AWS account we are using the API Gateway which lets us create resources which can be attached to our services. Different HTTP methods like GET, POST can be created for each of the resources and the requests arriving on those methods can be forwarded to the respective AWS resources. For example, we created a resource named tag and created a POST method on it and the requests arriving on that method are redirected to the Lambda function which stores the images.

Frontend/UI:

We are using ReactJS to create the Single Page Application of our UI. To host our UI we are using S3 bucket hosting where AWS lets you host static websites. To deliver the website assets quickly and to get a secure URL for the Google Single Sign On we created a CloudFront distribution which points to the bucket with the UI assets.

## Role of Team Members:

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Contribution Percentage** | **Elements Contributed** |
| Kshitij Pandey | 25 | 1. Lambda function to query tags. 2. Creating the user interface. 3. Google Single Sign-On integration in UI. |
| Chayan Agarwal | 25 | 1. Creating a lambda function for detecting tags from the image. 2. Integrating Google Single Sign-On in Cognito. |
| Somya Milendra Saxena | 25 | 1. Preparing the database schema for storing tags. 2. Lambda function to store tags and URLs. |
| Manik Guglani | 25 | 1. Setting up the Cognito User Pool and Identity Pool. 2. Creating IAM policies for fine grained access. |

## 

While creating the Python Lambda function to detect the tags in images we saw that providing the dependencies like Numpy, OpenCV was not straightforward as compared to providing dependencies in NodeJS Lambda functions. To fix that issue we went through the Lambda function documentation and forum posts on Moodle which led us to create layers in our Lambda function and the function can pick up the dependencies from an S3 bucket defined in the layer.

The other challenge that we faced was integrating the Google Single Sign on with the Cognito. To enable the users of the User Pool to login through Google, Cognito asks for a secure URL to redirect the users after successfully logging in. We had to look for a lot of options and in the end we found that using CloudFront distribution to deploy your app gives you a domain attached with an SSL certificate. We used this domain in Cognito to redirect the user so that the propagation of Google sign in is a secure environment.

While creating the Lambda function to upload the image we faced an issue in encoding the image data while uploading it to S3. We found out that API Gateway did not support multipart form data which is the standard for sending files in an HTTP request. API Gateway team says they now provide support only for binary data which can only be used with Lambda integration to convert that binary data into its actual content. To overcome this issue we did not use multipart/form-data to send the file in the request, instead we converted the image file into its base64 encoded string and sent it to the Lambda function which decodes it back to a Buffer to upload the image to S3.

User Manual

### Steps for Registration:

Step 1: Click on the “Register” button available on the top right corner of the screen.

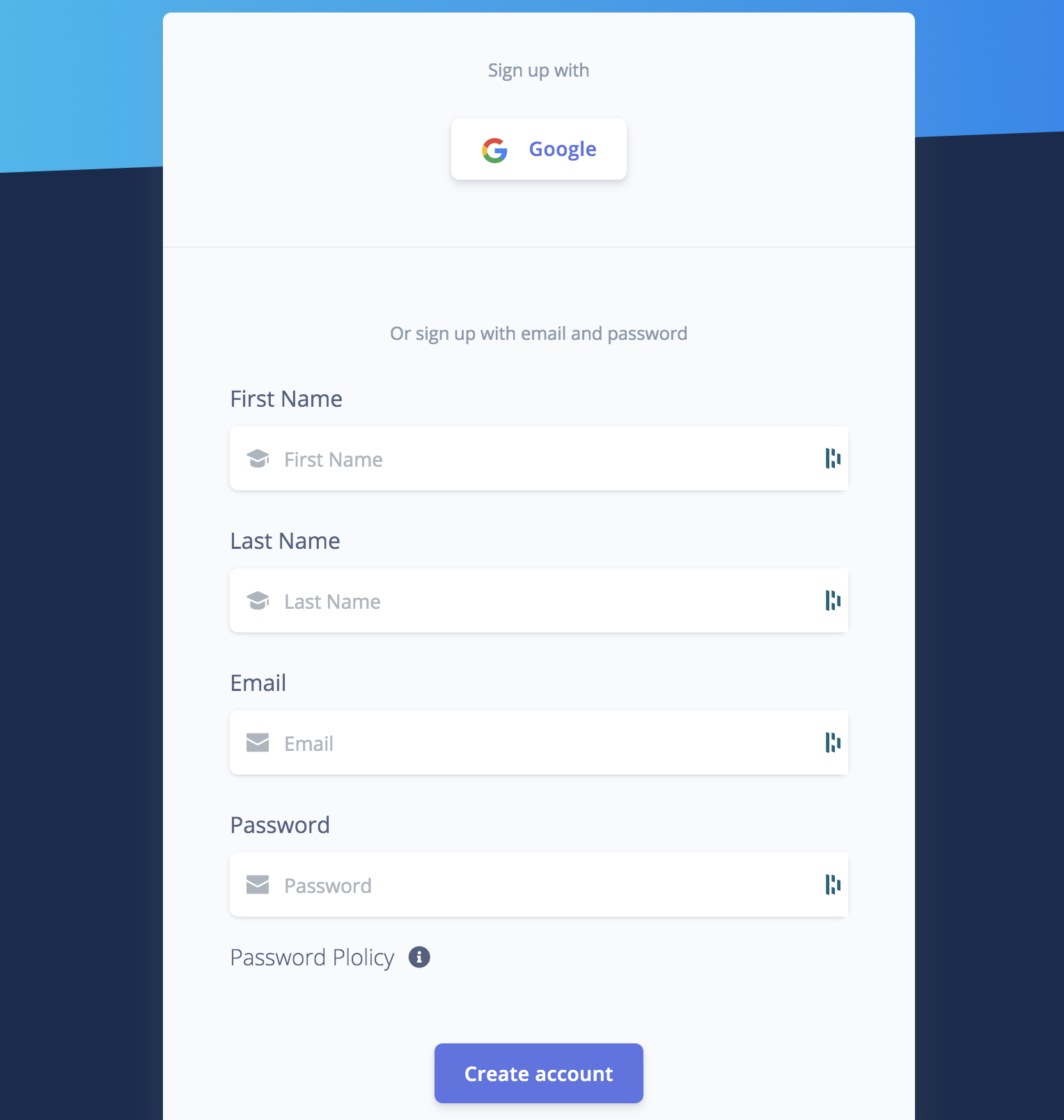


Fig 1. Registration

Step 2: Provide the details such as First Name, Last Name, email id, password and click on “Create account”. A verification link will be emailed to the user.

Step 3: Open account of the email address provided. Open the email from “Your verification link” and click on the hypertext “Verify Email” to verify the email address provided.

A screenshot of a cell phone

Description automatically generated

Fig 2: Registration Confirmation

## 

### Steps for Login:

Option1: Login by providing credentials.

Provide the username and password and click the “Sign in” button.

A screenshot of a cell phone

Description automatically generated

Fig 3: Sign-in with Email Id and password

Option 2: Login with google.

Click on the “Google” button and select the email id.

A screenshot of a cell phone

Description automatically generated

Fig 4. Sign-in with google option

### Steps to Upload image:

Step 1: Select the option “Upload Image”. Click on the button labelled “Drag 'n' drop your file here or click to select a file”.

A picture containing screenshot

Description automatically generated

Fig 5. Upload Image

## 

### Steps to Query tag:

Step 1: Select the option “Query Tag” from the navigation menu.

Step 2: Enter the tag to search and click on the “Search” button. The image of the entered tag will be displayed.

A picture containing screenshot

Description automatically generated

Fig 6. Enter tag

A screenshot of a social media post

Description automatically generated

Fig 7. Image Received

## GitHub Link: <https://github.com/Ashirogi-Muto/fit_5225_assignment_2>

## Improving Performance and Reducing Failure:

We have already made use of Amazon CloudFront which is a Content Delivery Network but we can enhance it in our application to make sure that the application’s performance is efficient with users from all over the world so that we can improve the response time. Writing an image recognition application in a Lambda with heavy weights is not really efficient since those weights and heavy configurations are loaded every time the Lambda function is executed. Instead we can use the AWS Rekognition service which is highly scalable and efficient for image recognition purposes. We can also use Kubernetes to create an image recognition service which is also scalable.

For heavy and recurring database read and write operations we can enable the auto scaling service on the DynamoDB table which will help us make the read and write queries super fast and depending on the number of requests AWS will elastically scale the read and write capacities.

To reduce the chances of failure we can make use of CloudWatch to monitor or services and create alarms in CloudWatch to make sure that if any service is malfunctioning quick action is taken to fix that service.

## Real World Solution:

## **Person detection**

The real-world application of this solution can be used for person detection through a CCTV generated footage or image. In a pandemic like this government agencies can make use of this service to batch upload images from their CCTV footage and the query for tag ‘person’ which can help them ensure that proper safety regulations are followed by the people.