ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable Vice Chancellor Dr. C. MUTHAMIZHCHELVAN, for being the beacon in all our endeavors. We would like to express my warmth of gratitude to our Registrar Dr. S. Ponnusamy, for his encouragement

We express our profound gratitude to our Dean (College of Engineering and Technology) Dr. T. V.Gopal, for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing Dr. Revathi Venkataraman, for imparting confidence to complete my course project

We wish to express my sincere thanks to Course Audit Professor Dr.M.LAKSHMI, Professor and Head, Data Science and Business Systems and Course Cordinator Dr.E. Sasikala, Associate Professor, Data Science and Business Systems for their constant encouragement and support.

We are highly thankful to our my Course project Internal guide Subject handling staff name , Designation , Department, for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to Student HOD name Department and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project

TABLE OF CONTENTS

1. ABSTRACT

Attendance Management System using AWS

The current project is a complex application of AWS functionalities that aims to generate an autonomous system for generating student attendance data from image data provided periodically (as in hour-by-hour, or any other custom required periods).

Within the project, AWS Rekognition was deemed the most suited by our team for its trainable facial recognition models that allows us to train for and store graphic identification parameters for a large student database via an intuitive Machine Learning model. Additionally Python has been also decided to be used for the local driver code, owing to its ease of use with the BOTO3 sdk from Amazon. For the ease of functionality and uniformity, python has also been elected for driving required intermediate Lambda functions that deal vitally in main info transfers among the model, database, and webapp.

During research pertaining to the project, preference of certain AWS toolds over multiple other similar-functionality-based ones, like Lambda over EC2, and S3 over EBS & EFS have been established, and due data has also been embellished accordingly within the report documentation under implementation notes

2. INTRODUCTION

As the current ever-changing timeline of technological advancements, it is imperative that the concept of automation is leveraged as much as possible where feasible. One such department that we chose to address was automating the universal classroom attendance system.

Fcial-recognition-based modern machine learning concepts were deemed best to be applied and integrated for this purpose. This has included the popular cloud service providers AWS and some of their provided tools/services for the same, including S3, Lambda, API Gateways, Rekognition, Python SDKs (boto3).

This project has been taken up by the following students:

- **Utkarsh Rastogi [RA1911028010019]**
- **Vaibhav Sharma [RA1911028010018]**
- **Shivendra Pratap Singh [RA1911028010017]**

3. REQUIREMENT ANALYSIS

3.1. Required Tangible Hardware Resources-

Camera – Required for taking periodic pictures of class

Local Machine – intermediate for captured pictures, medium for storing and running driver code, holding credentials for accessing results via webapp. Internet access required.

3.2. Required AWS / Soft Resources-

Driver code file/app along with authorized credentials [hosted on local machine] – responsible for extracting periodic images, uploading to allocated s3 bucket, checking them by trained AWS Rekognition model, inferring and passing the acquired results forward to AWS API Gateway.

AWS Rekognition Model [Trained] – AI model to be used for extracting and assimilating facial recognition data from images.

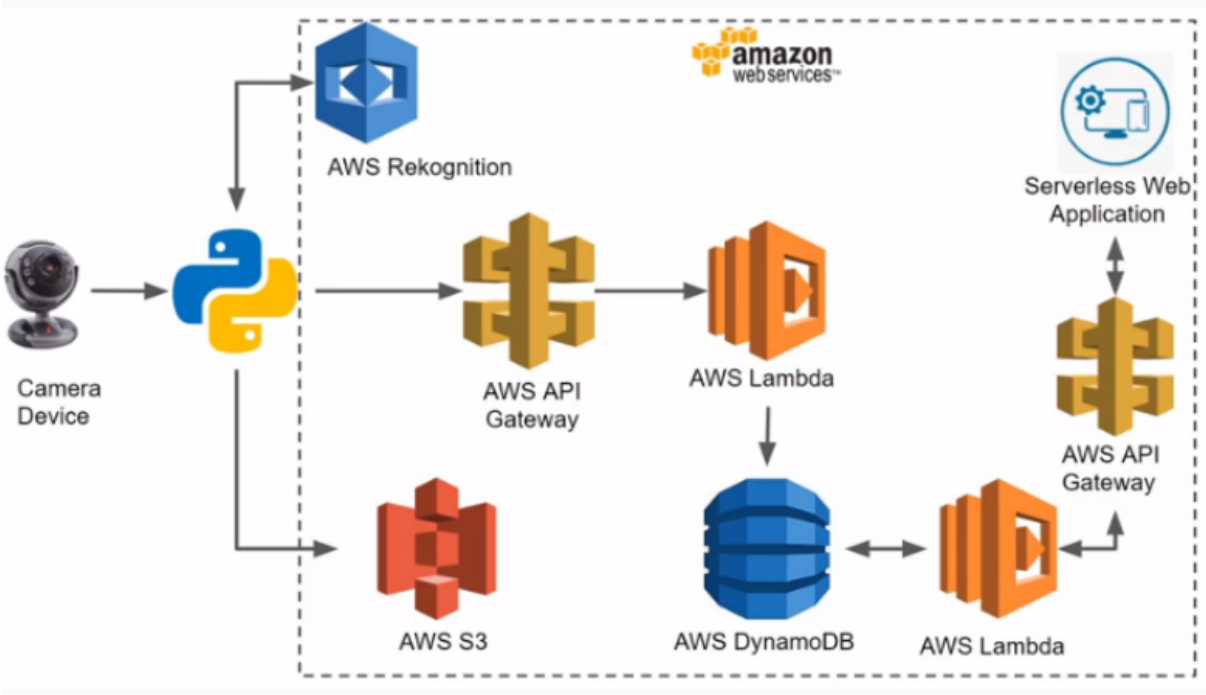
AWS S3 Bucket - Stores the images

AWS API Gateway – Connects external driver code and webapp to dynamoDB (via Lmbda instances)

AWS Lambda - Used to update attendace in DynamoDB and access data from the same to display on webapp

Web Application (Serverless) – Uses serverless Lambda Iinstances to access data from DynamoDB for attendance

4. ARCHITECTURE & DESIGN



Amazon Rekognition offers pre-trained and customizable computer vision (CV) capabilities to extract information and insights from your images and videos.

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications.

Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can store and

protect any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and software as a service (SaaS) applications, and only pay for what you use.

Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale. DynamoDB offers built-in security, continuous backups, automated multi-region replication, in-memory caching, and data export tools.

5. IMPLEMENTATION NOTES

Why S3

S3 -

- $0.23 per GB per Month
- Accessible from anywhere on internet using APIs (Public access via Block Public Access)
- Access Security based on IAM (uses bucket & user policies)
- No specific hierarchy in system, roles & policies widely flexible and customizable for different users and user
- groups for better access
- Slight latency (insignificant as our service does not depend on 100% dot time accuracy)
- No limit on no. Of objects stored
- *Optional extra definition* S3 Block Public Access provides controls across an entire AWS Account or at the
- individual S3 bucket level to ensure that objects never have public access, now and in the future. Public access
- is granted to buckets and objects through access control lists (ACLs), bucket policies, or both.

Possible Alternatives – EBS & EFS -

- Both Elastic Block Storage and Elastic File Storage plans are more suitable for low-latency interactive
- applications that require consistent & predictable performance, data backups, scalable data for analytical
- purposes etc.

Reason we still use S3 -

- s3 is overall cheapest for data storage alone, which is our primary concern here, making it very

conveniently

- accessible to the concerned authorized parties. Then there's also the unrequired parameters from EBS & EFS
- that they cannot be paired to function with Lambda instances, and that the additional nil-latency features are
- not relevant to the envisioned target functionality.

Why Lambda

AWS Lambda -

- Allows us to run a piece of code written on one of the supported programming languages – Java, JavaScript, or Python when a trigger linked to an event is fired.
- We don’t need to configure a virtual server and environment to run an application you have written.
- In essence, this is a serverless coded trigger, which allows us to focus on our application, rather than server management. Lambda is an implementation of Function as a Service (FaaS) by Amazon.

Possible Alternative – Amazon EC2 -

- service that allows for using virtual machines called EC2 instances in the cloud and providing scalability.
- Type of service – IaaS (Infrastructure as a service). Provides automatic scaling and load balancing.
- Can work in conjunction with most other Amazon web services

Reason we still use Lambda -

- Our application will require AWS services to run in {hosted on local machine/app} very dispersed and short intervals of time.
- Running EC2 which is costs on continuous hour-by-hour usage, idle time can build up to be exponentially more than the in-use time.

- Apart from this, however, EC2 does not provide virtually any latency in application boot and execution
- Running Lambda gives us a container limit of 15 mins max, and an expected latency of about 100ms for executing an app after sending a request.

Cost -

- EC2 t2.large -> $0.0928 per execution (Rounded to nearest hour)
- Lambda -> ($0.00001667 per each Gigabyte-second, rounded to nearest second)

Conclusion -

- Thus we prefer using Lambda due to synchronous, dispersed and quick execution requirements.
- In case in an alternate hypothetical scenario where the app was remotely in use for multiple institutions

6. EXPERIMENT RESULTS

As we observe, the system works to extract attendance data from passed periodical pictures and stores in within our retrievable database.

7. REFERENCES

- https://aws.amazon.com/sdk-for-python/
- https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
- https://boto3.amazonaws.com/v1/documentation/api/latest/index.html
- https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/rekognition.html
- https://docs.aws.amazon.com/code-samples/latest/catalog/code-catalog-python-example_code-rekognition.html
- https://gist.github.com/alexcasalboni/0f21a1889f09760f8981b643326730ff
- https://docs.aws.amazon.com/lambda/latest/dg/services-apigateway-tutorial.html