DYNAMIC PRICING SAAS APPLICATION WITH STUDENTS DATA MANAGEMENT USING CLOUD COMPUTING

PRAJWAL P ACHARYA[1], RAGHAVENDRA S N[2], SRIHARI P N[3] , VENKATESH V[4],NITA MESHRAM[5]

1,2,3,4 Under Graduate Student, Department Of Computer Science and Engineering,
K S School of Engineering and Management
5 Assistant Professor, Department Of Computer Science and Engineering,
K S School of Engineering and Management

ABSTRACT
AWS is used to power a fully automated project, and contemporary design and workflows are integrated to minimize the need for manual maintenance. The entire application is constructed using the serverless computing technique. The O(n) standards, which are renowned for their speed and resiliency, are self-certified by algorithms. The main goal of these projects is security and compliance, and in order to prevent unusual attacks, we have used AWS S3 for storage. S3 is configured with Event Trigger so that when the trigger is invoked, the AWS Lambda function is called, and this service internally calls a python snippet that is intended for performing the calculations and writing the contents in the DynamoDB Database. When results are successfully processed by AWS Lambda, a mail is issued over SNS Subscription that includes all the processed data's specifics. To create a single robust automated program that can be modified for user interface scenarios and is outfitted with cutting-edge data processing methods employing cloud computing services that deal with high-velocity data and support serverless architecture. This model will replicate computations of student performance data sheets and reports by extracting data from various organized data formats and providing it to the final users. It is based on the "Pay as You Go" approach that has been adopted. The data for future credentials will be backed up and organized by this tiny program.

A. INTRODUCTION

1.1 OVERVIEW

In order to maximise the use of cloud services and automate the computation of data sheets/reports in order to avoid manual intrusion, this project proposes to automate the entire architecture. This mitigates human involvement and reduces dependencies inside the Android application.
AWS is used to power a fully automated project, and contemporary design and workflows are integrated to minimise the need for manual maintenance. The entire application is constructed using the serverless computing technique. The O(n) standards, which are renowned for their speed and resiliency, are self-certified by algorithms. We have used AWS S3 for storage, S3 is configured with Event Trigger so that when the trigger is invoked the AWS Lambda function is called so that this service internally invokes python snippet which is intended for making the calculations and writing the cont. Security and compliance is the main objective of these projects to avoid unusual attacks, several security tools such as AWS WAF(Web Application Firewall), AWS Shield are deployed to avoid Popular DDoS, SQL Injection, attacks.
To create a single robust automated program that can be modified for user interface scenarios and is outfitted with cutting-edge data processing methods employing cloud computing services that deal with high-velocity data and support serverless architecture. This model will replicate computations of student performance data sheets and reports by extracting data from various organized data formats and providing it to the final users. It is based on the "Pay as You Go" approach that has been adopted. The data for future credentials will be backed up and organized by this tiny program.

1.2 PURPOSE OF THE PROJECT

A straightforward user interface for entering datasets for calculations. A straightforward O(n) logic that speeds up timely execution by reducing iterations and looping relative to standard algorithms. Update of Results and Datasets Replication over various AZs, high availability, and reduced latency are all features of a DynamoDB database (Availability Zones). Result Button with a single click. Utilization of an effective and exception-handling algorithm that guarantees accurate results compilation and transmits the results after a click to a DynamoDB database operating on the AWS Cloud It uses an effective algorithm that avoids larger Iterations of datasets and with more compute efficiency and faster results algorithm with consistent results. Lambda (PaaS) Functions are used to trigger and update the values in DynamoDB database which is replicated across Different AZ (Availability Zones).

1.3 SCOPE OF THE PROJECT

As you can see in the following section, AWS Amplify is basically integrated with the Android SDK in Android Studio. Authentication is provided by the Cognito user pool after which the users upload the required format of the data. As soon as the user uploads the specific file in the required format, it is stored in Amazon S3. S3 is configured with Event Trigger so that when the trigger is invoked, the AWS Lambda function is called. The Lambda function is FaaS (Function as a Service), so this service internally invokes python code which is intended for calculation of data sheets/reports and writing the contents in the DynamoDB

Database, the explanation to each of the service is explained in detail below, after the results are calculated the event sends an SNS message (Simple Notification Service) message to the subscription which is configured that is also explained in detail in the below section. The Algorithms used are of O(n) complexity which reduces the overall time for execution of program and efficient utilization of memory.

1.4 DEFINITION

1.4.1 AMAZON WEB SERVICES

AWS (Amazon Web Services) is a division of Amazon that offers governments, businesses, and people metered pay-as-you-go cloud computing platforms and APIs. These web services for cloud computing offer a range of fundamental abstract technical infrastructure, distributed computing building pieces, and tools. One of these services is Amazon Elastic Compute Cloud (EC2), which enables users to have a virtual computer cluster at their disposal that is constantly accessible via the Internet. Most of the characteristics of a real computer are replicated by AWS's version of virtual computers, including hardware central processing units (CPUs) and graphics processing units (GPUs) for processing, local/RAM memory, hard-disk/SSD storage, a choice of operating systems, networking, and pre-loaded application software like web servers, databases and customer relationship management (CRM).

In this project, you'll discover how to use AWS Elastic Beanstalk and Amazon Relational Database Service to construct a high-availability LAMP stack web application (Amazon RDS). Linux, Apache, MySQL, and PHP make up the stack. With Elastic Beanstalk, all deployment-related tasks—including capacity provisioning, load balancing, auto-scaling, and application health monitoring—are handled automatically after you submit your code. Using simply customizable Auto Scaling options, Elastic Beanstalk automatically scales up and down your application according to its unique needs. Setting up, running, and scaling a relational database in the cloud is simple with Amazon RDS. It manages time-consuming database administration operations while offering affordable and scalable capacity.

1.4.2 DYNAMO DB

A fully managed messaging service for application-to-application (A2A) and application-to-person (A2P) communication, Amazon Simple Notification Service (Amazon SNS) is available. For high-throughput, push-based, many-to-many messaging across distributed systems, microservices, and event-driven serverless applications, the A2A pub/sub feature enables topics. Your publisher systems can fanout messages to several subscriber systems, such as Amazon SQS queues, AWS Lambda functions, HTTPS endpoints, and Amazon Kinesis Data Firehose, using Amazon SNS topics. You may reach a large number of users with messages by using SMS, mobile push, and email thanks to A2P capability. Therefore, email is delivered to the registered mail in the SNS Endpoint upon successful event triggering, such as PUT, PUSH, or GET.

1.4.3 SAAS (SOFTWARE AS A SERVICE)

A popular business concept is cloud computing, which gives customers online access to cloud resources whenever they need them. Cloud computing has poor reuse potential, however, because it lacks a standardised way of encoding knowledge that may provide clients with an all-inclusive solution for managing and building cloud applications. In this paper, a Semantic Agent as a Service (SAAS) that was created utilising modelling in the Unified Modeling Language is proposed. The SAAS architecture is built on research into multi-agent systems, cloud computing, and the semantic web. To create intelligent cloud computing applications, the architecture can be integrated with already-existing cloud service models like Software as a Service, Platform as a Service, and Infrastructure as a Service.

1.4.4 DYNAMIC PRICING

On-demand service and reserved service are the two categories into which the conventional cloud resource pricing models fall. The former is unfair to users with long usage times and low concurrency because it simply considers usage time. The latter does not take into account user resource consumption and charges all users the same amount. Therefore, in this work, we offer a dynamic pricing model that is flexible and takes into account both the maximum concurrency of resource consumption as well as the occupied time and resource usage of various users. Cloud computing is growing quite quickly. Instead of using a specific server, we can install various computer resources in the cloud thanks to cloud computing. To put it another way, there is no requirement for hardware equipment, which is particularly advantageous for enterprises or organisations as it frees up resources for essential operations rather to spending them on computer infrastructure and upkeep. A dynamic pricing scheme is required for both service providers and customers due to the fact that a service provided in the cloud may move between multiple resources while mobile users, such as smart devices, cause a dynamic change in resource usage.

B. PROBLEM IDENTIFICATION

2.1 PROBLEM STATEMENT

A straightforward user interface for entering datasets for calculations. A straightforward O(n) logic that speeds up timely execution by reducing iterations and looping relative to standard algorithms. Dataset and result updating to a DynamoDB database with high availability, low latency, and replication across several AZ (Availability Zones). Result Button with a single click.Utilization of an effective and exception-handling algorithm that guarantees accurate results compilation and transmits the results after a click to a DynamoDB database operating on the AWS Cloud. The application of an effective algorithm results in faster, more accurate, and consistent results by avoiding the need for extensive dataset iterations. Values in the replicated DynamoDB database that uses Lambda (PaaS) functions are triggered and updated (Availability Zones).

2.2 PROJECT SCOPE

Basically, AWS Amplify is linked with the Android SDK in Android Studio, and Cognito userpool provides authentication before users upload the necessary data in the manner indicated below. As soon as the user uploads the necessary information in the proper format, the file is saved in Amazon S3. The AWS Lambda function is called when the S3 is configured with an event trigger because it is a function as a service. This service internally calls python code that is designed to calculate data sheets and reports and write the contents to the DynamoDB database. The explanations for each service are provided in detail below, after the results are calculated the event sends an SNS message (Simple Notification Service) message to the subscription which is configured that is also explained in detail in the below section. The Algorithms used are of O(n) complexity which reduces the overal time for execution of program and efficient utilization of memory.

C. GOALS AND OBJECTIVES

3.1 PROJECT GOALS

To create a single robust automated program that can be modified for user interface scenarios and is outfitted with cutting-edge data processing methods employing cloud computing services that deal with high-velocity data and support serverless architecture. This model will replicate computations of student performance data sheets and reports by extracting data from various organized data formats and providing it to the final users. It is based on the "Pay as You Go" approach that has been adopted. The data for future credentials will be backed up and organized by this tiny program.

3.2 PROJECT OBJECTIVES

- ➢ To develop a web- or mobile-based application that experiences the end points and user interface.
- ➢ For the purpose of creating API gateways for various application interfaces.
- ➢ Using serverless technology, compute and synchronise the credential sets of data sheets and reports.
- ➢ Successfully using the cloud to notify the triggered execution and synchronise the data for the end users.
- ➢ To implement, evaluate, and develop a dynamic pricing plan for actual cloud-based businesses.

D. METHODOLOGY

4.1 Working-Flow of SGPA Calculator

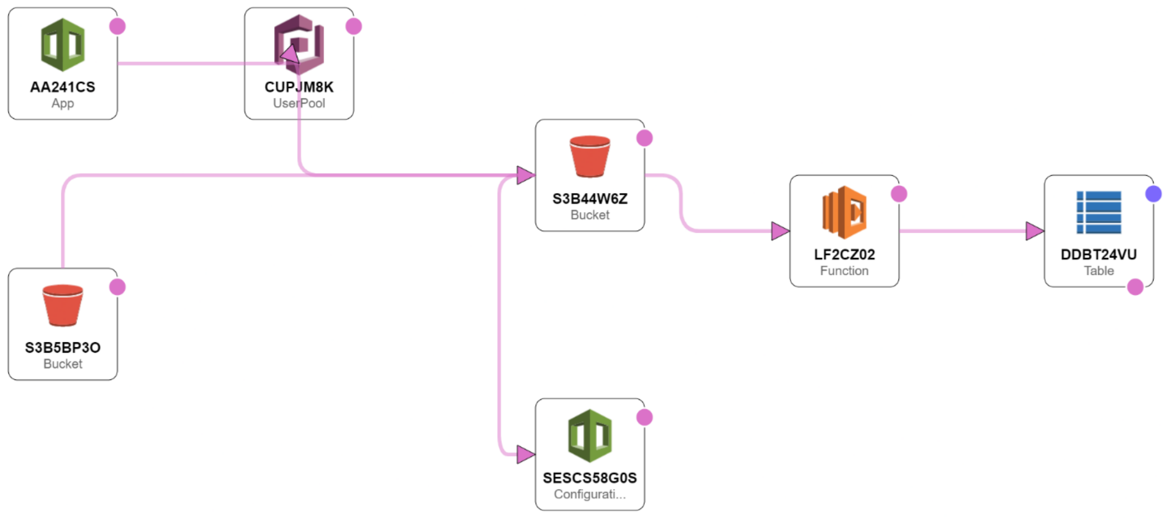

Fig 4.1 Workflow of SGPA Calculator

Basically, AWS Amplify is linked with the Android SDK in Android Studio, and Cognito userpool provides authentication before users upload the necessary data in the manner indicated below. As soon as the user uploads the necessary information in the proper format, the file is saved in Amazon S3. The AWS Lambda function is called when the S3 is configured with an event trigger because it is a function as a service. This service internally calls python code that is designed to calculate data sheets and reports and write the contents to the DynamoDB database. The explanations for each service are provided in detail below, after the results are calculated the event sends an SNS message (Simple Notification Service) message to the subscription which is configured that is also explained in detail in the below section. The Algorithms used are of O(n) complexity which reduces the overal time for execution of program and efficient utilization of memory..

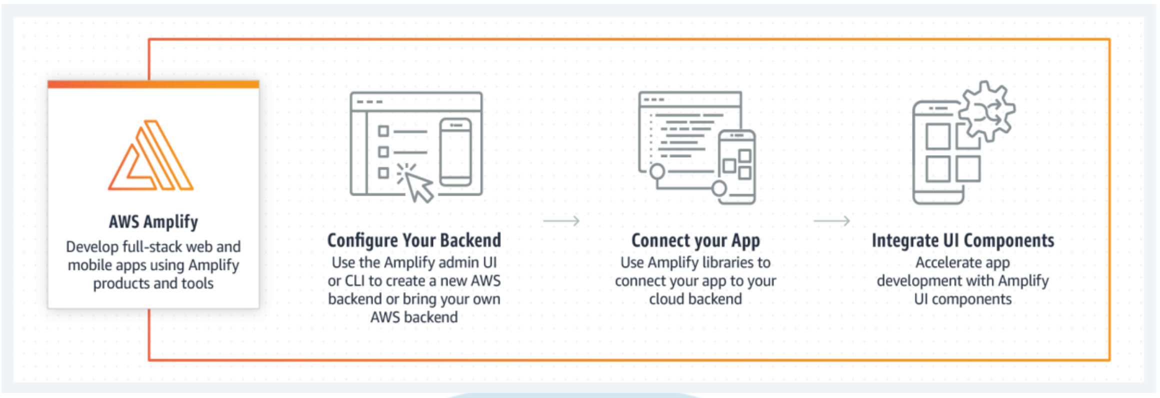
AWS Amplify
Develop full-stack web and mobile apps using Amplify products and tools
Configure Your Backend
Use the Amplify admin UI or CLI to create a new AWS backend or bring your own AWS backend
Connect your App
Use Amplify libraries to connect your app to your cloud backend
Integrate UI Components
Accelerate app development with Amplify UI components

Fig. 4.2: Amplify

- **Amplify:** Front-end web and mobile developers can build full-stack applications on AWS more quickly with the help of a suite of tools and services called AWS Amplify, which also gives developers the freedom to use the range of AWS services to further customise their apps. Popular web frameworks including JavaScript, React, Angular, Vue, and Next.js are supported by Amplify, as are mobile platforms like Android, iOS, React Native, Ionic, and Flutter.

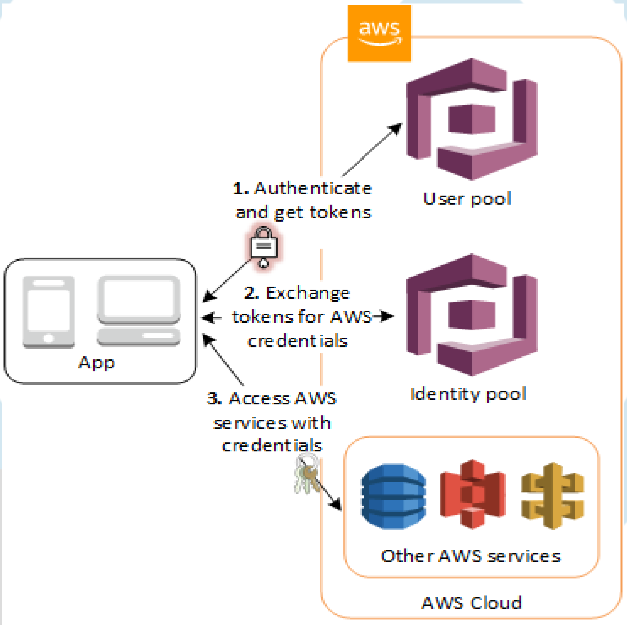
aws
User pool
1. Authenticate and get tokens
App
2. Exchange tokens for AWS credentials
Identity pool
3. Access AWS services with credentials
Other AWS services
AWS Cloud

Fig. 4.3: Cognito User Pool

- **Cognito User Pool:** For your web and mobile apps, Amazon Cognito offers user management, authentication, and authorization. Either directly with a username and password or through a third party like Facebook, Amazon, Google, or Apple, your users can sign in. User pools and identity pools are Amazon Cognito's two fundamental building blocks. User pools are user directories that give your app's users sign-up and sign-in alternatives. You can provide your users access to additional AWS services by using identity pools. User pools and identity pools may be used alone or jointly.

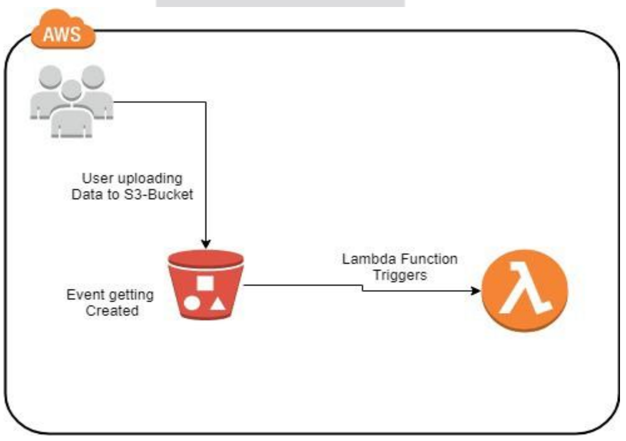

Fig. 4.4: AWS S3 Trigger

- **AWS S3 Trigger:** You can upload or remove files using the Amazon S3 service, which is used for file storage. As a result, whenever any files are uploaded to S3 buckets, AWS Lambda on S3 is triggered. The handler function in AWS Lambda serves as the function's starting point. The handler is in possession of the event specifics. Python was used to create the Lambda function, which computes SGPA using the user's raw data that is uploaded from the mobile application to S3. Trigger is then called, which executes the Lambda function.

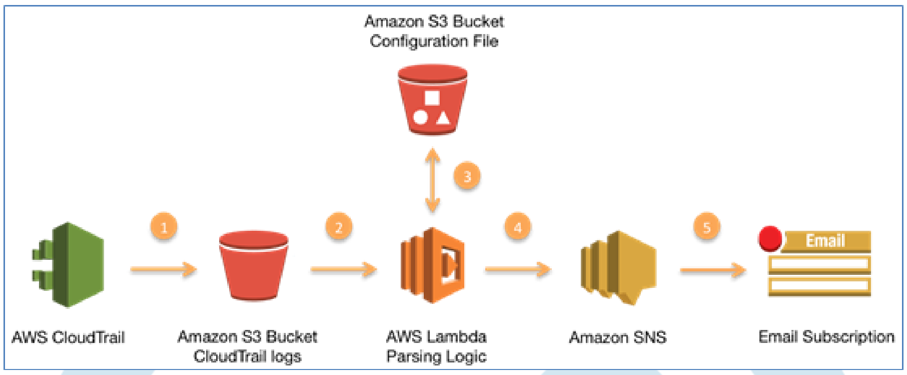

Fig. 4.5: SNS (Simple Notification Service)

- **SNS (Simple Notification Service):** A fully managed messaging service for application-to-application (A2A) and application-to-person (A2P) communication, Amazon Simple Notification Service (Amazon SNS) is available. For high-throughput, push-based, many-to-many messaging across distributed systems, microservices, and event-driven serverless applications, the A2A pub/sub feature enables topics. Your publisher systems can fanout messages to several subscriber systems, such as Amazon SQS queues, AWS Lambda functions, HTTPS endpoints, and Amazon Kinesis Data Firehose, using Amazon SNS topics. You may reach a large number of users with messages by using SMS, mobile push, and email thanks to A2P capability. Therefore, email is delivered to the registered mail in the SNS Endpoint upon successful event triggering, such as PUT, PUSH, or GET.

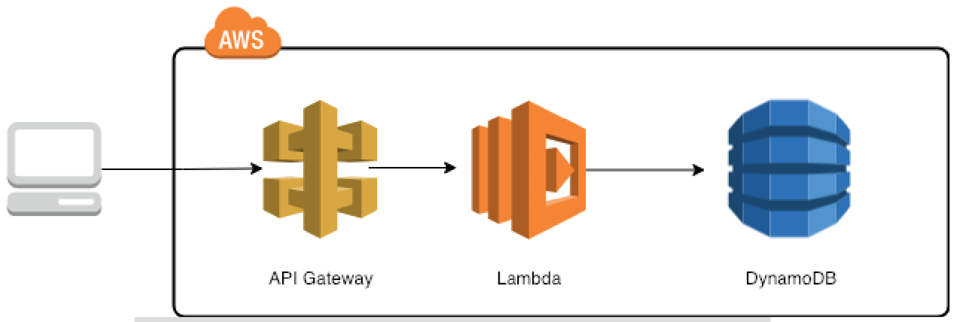

Fig. 4.6: Dynamo DB

- **Dynamo DB:** A fully managed messaging service for application-to-application (A2A) and application-to-person (A2P) communication, Amazon Simple Notification Service (Amazon SNS) is available. For high-throughput, push-based, many-to-many messaging across distributed systems, microservices, and event-driven serverless applications, the A2A pub/sub feature enables topics. Your publisher systems can fanout messages to several subscriber systems, such as Amazon SQS queues, AWS Lambda functions, HTTPS endpoints, and Amazon Kinesis Data Firehose, using Amazon SNS topics. You may reach a large number of users with messages by using SMS, mobile push, and email thanks to A2P capability. Therefore, email is delivered to the registered mail in the SNS Endpoint upon successful event triggering, such as PUT, PUSH, or GET.

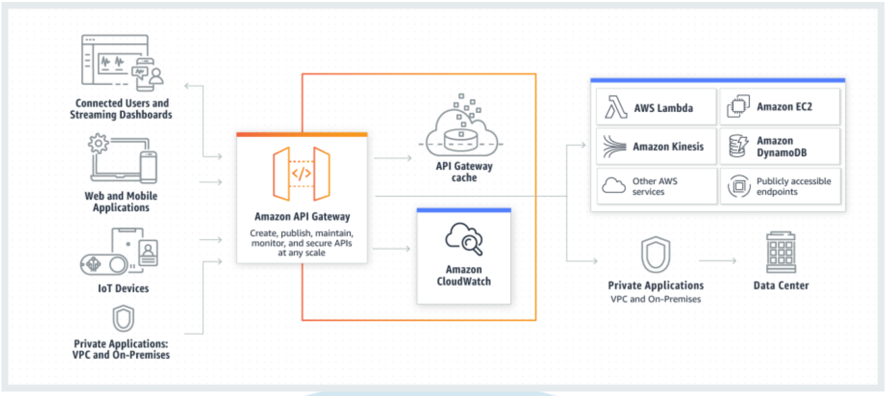
Connected Users and Streaming Dashboards
Web and Mobile Applications
IoT Devices
Private Applications: VPC and On-Premises
Amazon API Gateway
Create, publish, maintain, monitor, and secure APIs at any scale
API Gateway cache
Amazon CloudWatch
AWS Lambda
Amazon EC2
Amazon Kinesis
Amazon DynamoDB
Other AWS services
Publicly accessible endpoints
Private Applications
VPC and On-Premises
Data Center

Fig. 4.7: API Gateway

- **API Gateway:** In order to construct, publish, maintain, monitor, and protect APIs at any size, developers can use the fully managed service known as Amazon API Gateway. Applications use APIs as their "front door" to data, business logic, or functionality in your backend services. You can build WebSocket and RESTful APIs with API Gateway to enable apps with real-time two-way communication. Web apps, serverless workloads, and containerized workloads are all supported by API Gateway. Traffic management, CORS support, authorization and access control, throttling, monitoring, and API version management are all activities that API Gateway manages in order to accept and process up to hundreds of thousands of concurrent API calls. There are no setup fees or minimum charges for API Gateway.

E. APPLICATIONS

The fundamental strategy in this project is to give an email notification to the relevant users along with an immediate result. This initiative makes automatic grading possible, which lessens the workload on teachers. Other characteristics of this project tool, aside from this salient feature, make it quite valuable. The following list includes some of these qualities:

- The project's grading component continually improves by taking in fresh data. The capacity of the paper grader to modify its grading settings by scanning documents that have been manually graded ensures that the grading procedure doesn't become obsolete.
- The software's grading methodology is improved by the project graders' ongoing learning. A grade calculator is simple to incorporate into an already-existing virtual interface.

F. CONCLUSIONS & FUTURE WORK

5.1 CONCLUSION

The entire architecture becomes a fault tolerant and redundant application thanks to a resilient application developed on AWS, which is renowned for its agility and functionality. This provides a pay as you go model in the created architecture. A serverless architecture was used to build the complete system, which lowers the overall cost of maintenance and human intervention. On-demand service and reserved service are the two categories into which the conventional cloud resource pricing models fall. The former is unfair to users with long usage times and low concurrency because it simply considers usage time. The latter does not take into account user resource consumption and charges all users the same amount. Therefore, in this work, we offer a dynamic pricing model that is flexible and takes into account both the maximum concurrency of resource consumption as well as the occupied time and resource usage of various users. As a result, this dynamic pricing model might, on the one hand, assist consumers in reducing the cost of cloud resources. On the other hand, service providers' earnings are assured. If serverless technologies are employed, the system is optimized for effective data storage, delivery, and management and avoids overall load or stress on the server. The system was created using the aforementioned technologies for cost optimization and automation.

5.2 FUTURE ENHANCEMENTS

Additional optimization and UI enhancements are possible.Queuing implementation in the AWS Data loss prevention between producers and consumers dead letter queue implementation to manage unprocessed dataset Complete Atomization enhancements to prevent Outage SES (Simple Email Service) implementation to provide results to each recipient's email address automatically.

G. REFERENCES

1. https://docs.amplify.aws/lib/storage/upload/q/platform/js
2. https://docs.amplify.aws/lib/storage/download/q/platform/js
3. https://docs.amplify.aws/lib/storage/list/q/platform/js#protected-level-list
4. https://docs.amplify.aws/lib/storage/copy/q/platform/js#copy-protected-files-fromanother-user
5. https://docs.amplify.aws/lib/storage/remove/q/platform/js#private-level-remove
6. https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
7. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SQLtoNoSQL. WriteData.html

8. Zaid Al-Ali†, Speidel Goodarzy†, Ethan Hunter†, Sangtae Ha†, Richard Han†, Eric Keller†, Eric Rozner †University of Colorado Boulder IBM Research.Making Serverless Computing More Serverless:IEEE 11th International Conference on Cloud Computing, pages 456-479, 2018.
9. Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, Joseph E. Gonzalez, Raluca Ada Popa, Ion Stoica, David A. Patterson, 9 Feb, 2019.
10. Fadi Alzhouri and Anjali Agarwal. Dynamic pricing scheme: Towards cloud revenue maximization. IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), pages 168–173, 2015.
11. Amazon EC2 Pricing- http://aws.amazon.com/ec2/pricing/ .[Online; accessed February-2015].
12. Amazon EC2 Spot Instances Pricing- https://aws.amazon.com/ec2/spot/pricing/ .[Online; accessed May-2017].
13. Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, and Matei Zaharia. Above the clouds: A Berkeley view of cloud computing. Technical report, 2009. Serverless computing and applications- https://aws.amazon.com/serverless/ . Accessed: 2019-01-23.

IJRTI