# A Mini Project Report

on

"Web form using AWS S3 Lambda and API Gateway"

Submitted to

CLOUD COMPUTING LAB
(20BT61231)

**BACHELOR OF TECHNOLOGY**

**IN**
**INFORMATION TECHNOLOGY**

*Submitted by*


| | |
|---|---|
| **Y.SUSMITHA** | **21121A12B9** |
| **Y.UPENDRA** | **21121A12C0** |
| **M PRASHANTHI** | **21121A12C1** |
| **A MOHAMMAD YUNUS** | **22125A1201** |
| **G SIVAMANI** | **22125A1202** |
| **K SHUAIB KHAN** | **22125A1203** |




**DepartmentofInformationTechnology**
**SREEVIDYANIKETHANENGINEERINGCOLLEGE**
(AUTONOMOUS)
(AffiliatedtoJNTUA, Ananthapuramu,ApprovedbyAICTE,AccreditedbyNBA&NAAC)
Sree SainathNagar,Tirupati–517102,A.P.,INDIA

2023-2024

## ABSTRACT

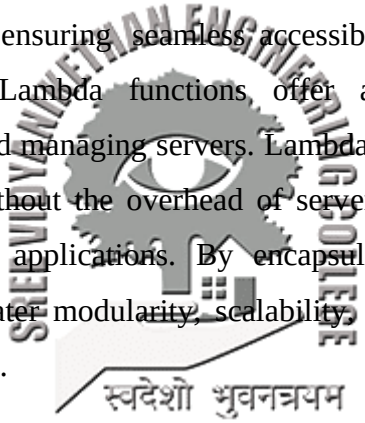
This abstract explores the integration of Amazon Web Services (AWS) S3, Python Lambda functions, and an API Gateway to develop a streamlined solution for a simple web form. AWS S3 offers scalable storage for various data types, making it an ideal choice for storing form submissions securely. Python Lambda functions provide serverless compute power, enabling the processing of form data without the need for managing server infrastructure. The API Gateway acts as a bridge between the web form and the Lambda function, facilitating seamless communication between the frontend and backend components. The integration process involves setting up an S3 bucket to store HTML, CSS, and JavaScript files constituting the web form. Lambda functions are created to handle form submissions, validating input data and storing it in S3. The API Gateway is configured to trigger the Lambda function upon receiving HTTP requests from the web form. Through this setup, users can submit data via the web form, which is then processed by the Lambda function and stored securely in S3. It highlights the simplicity and efficiency of utilizing AWS services in combination with Python Lambda functions to create a robust solution for web form handling. The serverless architecture ensures scalability, cost-effectiveness, and ease of maintenance, making it suitable for various applications requiring form submission functionality. Overall, this integration demonstrates the power and flexibility of AWS for developing modern web applications.



**KEYWORDS:** AWS S3, Python Lambda, PAI gateway, Serverless architecture, HTTP request, Backend processing.

## INTRODUCTION

In today's digital landscape, the ability to swiftly develop and deploy web applications is crucial for businesses striving to maintain a competitive edge. Cloud computing platforms, such as Amazon Web Services (AWS), offer a myriad of services and tools to streamline the development process, allowing developers to focus on innovation rather than infrastructure management. Among these services, AWS Simple Storage Service (S3), AWS Lambda, and API Gateway stand out as powerful components for building scalable and efficient web applications. This introduction delves into the integration of AWS S3, Python Lambda functions, and API Gateway to create a straightforward yet robust web form. This amalgamation of services enables developers to architect solutions that are highly adaptable, cost-effective, and easily maintainable. AWS S3 serves as the cornerstone for storing and managing data in the cloud. With its unparalleled scalability, durability, and low- latency access, S3 provides an ideal solution for hosting static assets, such as HTML, CSS, and client-side JavaScript files, which are essential for web form development. By leveraging S3's intuitive interface and global infrastructure, developers can effortlessly upload, retrieve, and manage files, ensuring seamless accessibility and reliability for end-users. Complementing AWS S3, Python Lambda functions offer a serverless computing paradigm that eliminates the need for provisioning and managing servers. Lambda functions allow developers to execute code in response to various events without the overhead of server maintenance, thereby enabling rapid prototyping and deployment of web applications. By encapsulating business logic within Lambda functions, developers can achieve greater modularity, scalability, and cost- efficiency, as resources are allocated dynamically based on demand.




API Gateway acts as the gateway between client applications and backend services, facilitating seamless communication and data exchange. By defining RESTful APIs and integrating with Lambda functions, API Gateway enables developers to expose endpoints for processing incoming requests from web forms. This allows for real-time validation, authentication, and processing of form submissions, ensuring data integrity and security throughout the interaction lifecycle. The synergy between AWS S3, Python Lambda, and API Gateway becomes particularly evident when building a simple web form. Consider a scenario where a company wishes to collect user feedback through a web-based form. By leveraging S3 to host the frontend assets, Lambda functions to handle form submissions, and API Gateway to orchestrate communication between the frontend and backend, developers can quickly develop and deploy a scalable solution that meetsthe company's requirements. The process begins with designing and implementing the frontend components of the web form using HTML, CSS, and JavaScript. These files are then uploaded to an S3 bucket, which serves as the static website hosting platform. Upon accessing

the web form, users can input their feedback andsubmit the form, triggering an HTTP request to the API Gateway endpoint associated with the Lambda function.

## ADVANTAGES

**Scalability**: AWS services like S3, Lambda, and API Gateway are highly scalable. As your web form receives more traffic or data submissions, these services can automatically scale to accommodate the increased load without requiring manual intervention.

**Cost-effectiveness:** With AWS Lambda, you only pay for the compute time your function consumes, and with S3, you pay for the storage used. This pay-as-you-go model can be cost-effective, especially for low to moderately used web forms, as you don't have to provision or pay for servers when the form is not in use.

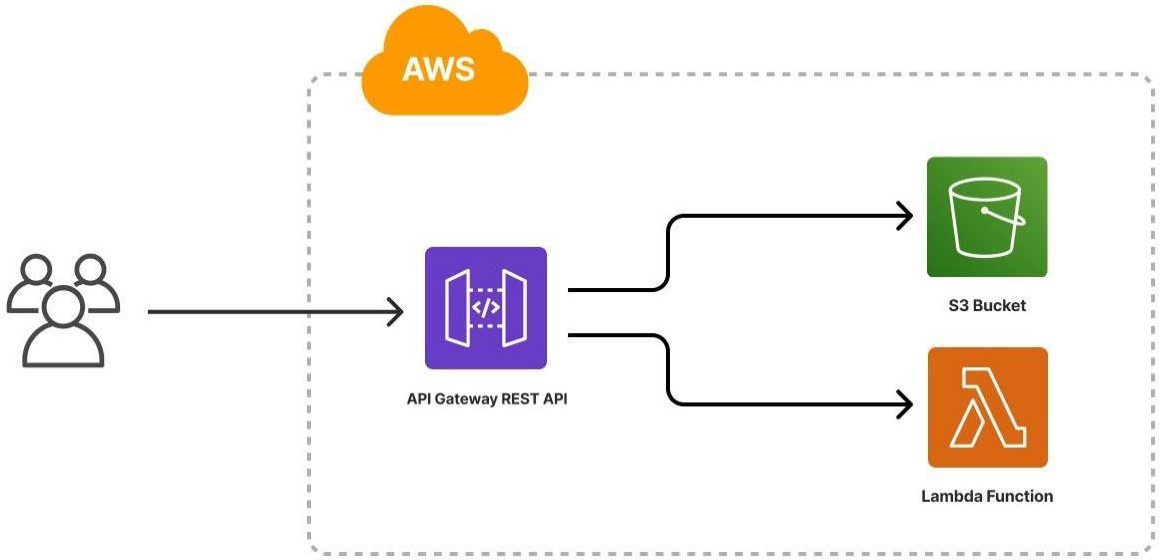
**Serverless architecture**: This setup is based on a serverless architecture, meaning you don't have to manage servers. AWS takes care of server provisioning, maintenance, and scaling, allowing you to focus on developing your web form and application logic without worrying about infrastructure management.

**Ease of deployment and maintenance:** Developing and deploying Lambda functions and configuring API Gateway endpoints is relatively straightforward, especially if you're already familiar with AWS services. Additionally, since there are no servers to manage, there's less overhead in terms of maintenance and updates.

**Integration with other AWS services:** AWS provides a wide range of services that can integrate seamlessly with S3, Lambda, and API Gateway. For example, you can easily integrate your web form with other AWS services such as DynamoDB for storing form submissions, SNS for sending notifications, or SES for sending emails based on form submissions.

**Security:** AWS offers various security features to protect your web form and data. You can leverage AWS Identity and Access Management (IAM) to control access to your AWS resources, configure encryption for data at rest and in transit .

## RESOURCE DIAGRAM




## RESOURCES LIST

**AWS Well-Architected Framework:** Learn about best practices for designing and deploying applications on AWS byexploring the AWS Well-Architected Framework. Understand key concepts such as security, reliability, performance efficiency, cost optimization, and operational excellence.

**AWS Developer Forums:** Join the AWS Developer Forums to connect with other developers, ask questions, and shareinsights and best practices related to building applications on AWS. Get help with troubleshooting issues and optimizing your architecture.

**AWS Training and Certification:** Enroll in AWS training courses and certification programs to deepen your understanding of AWS services and solutions. Explore courses on serverless computing, API development, and AWSarchitecture to enhance your skills.

**AWS Blogs and Whitepapers:** Stay updated with the latest announcements, case studies, and best practices by readingthe AWS blogs and whitepapers. Explore topics relevant to serverless computing, web development, and cloud architecture.

## PROCEDURE

**Deploying Weather Research and Forecasting on a virtual machine . Here's a general procedure:**

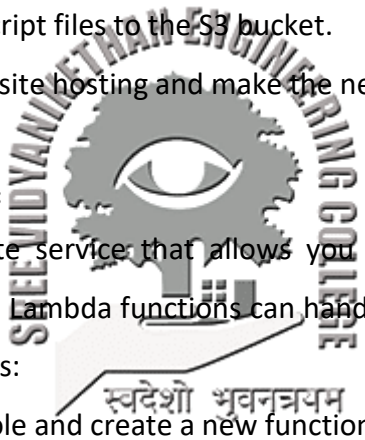
**1. Understanding AWS S3:**

AWS S3 is a scalable object storage service designed to store and retrieve any amount of data from anywhere on the web.It provides developers with a highly durable and available storage infrastructure. In the context of building a web form, S3 can be used to host static web content, such as HTML, CSS, and JavaScript files.

**2. Setting up AWS S3 for Hosting:**

To get started, create an AWS account if you haven't already and navigate to the AWS Management Console. Then,follow these steps:

- Create an S3 bucket with a unique name to host your web form files.
- Upload your HTML, CSS, and JavaScript files to the S3 bucket.
- Configure the bucket for static website hosting and make the necessary files public.




**3. Creating Python Lambda Functions:**

AWS Lambda is a serverless compute service that allows you to run code without provisioning or managing servers. In this case, Python Lambda functions can handle form submissions and process data. Here's how to create Lambda functions:

- Navigate to the AWS Lambda console and create a new function.
- Write Python code to handle form submissions, validate inputs, and perform any necessary operations.
- Configure triggers for the Lambda function, such as API Gateway.

**4. Integrating with API Gateway:**

AWS API Gateway enables you to create, deploy, and manage APIs for your web applications. It acts as a front door for accessing Lambda functions and other AWS services. Follow these steps to set up API Gateway:

- Navigate to the API Gateway console and create a new API.
- Define endpoints for handling HTTP requests, such as POST requests from your web form.
- Configure integration with Lambda functions to process incoming requests.
- Deploy the API to make it accessible via a publicly accessible URL.

**5. Securing Your Web Form:**

Security is paramount when building web applications. Consider implementing the following measures to secure yourweb form:

- Use HTTPS to encrypt data transmitted between the client and server.
- Implement input validation on both the client and server sides to prevent malicious inputs.
- Utilize AWS Identity and Access Management (IAM) to control access to your S3 bucket and Lambda functions.
- Consider implementing rate limiting and authentication mechanisms to protect against abuse and unauthorized access.

## SOURCE CODE

Here's a basic example of a serverless contact form using AWS S3, Lambda, and API Gateway:

**HTML Form (index.html):**

```
f<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Contact Form</title>
</head>
<body>
  <h1>Contact Form</h1>
  <form id="contactForm">
    <label for="name">Name:</label><br>
    <input type="text" id="name" name="name"><br>
    <label for="email">Email:</label><br>
    <input type="email" id="email" name="email"><br>
    <label for="message">Message:</label><br>
    <textarea id="message" name="message"></textarea><br>
    <button type="submit">Submit</button>
```

```
    </form>

    <script>
      document.getElementById('contactForm').addEventListener('submit',  async
        function(event) {event.preventDefault();
        const formData = new FormData(this);
        const  response  =  await
          fetch('YOUR_API_ENDPOINT', {method: 'POST',
          body: formData
        });
        if (response.ok) {
          alert('Message sent successfully!');
        } else {
          alert('Failed to send message.');
}
});
    </script>
 </body>
 </html>
```

Replace 'YOUR_API_ENDPOINT' with the actual endpoint of your API Gateway.

**Lambda Function (lambda_function.py):**

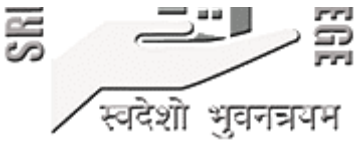
```
import boto3

def lambda_handler(event,
    context):s3 =
    boto3.client('s3')
    bucket_name = 'YOUR_S3_BUCKET_NAME'

    # Extract form data from the POST
    requestname =
    event['body']['name']
```

```
    email = event['body']['email']
    message =
    event['body']['message']
    # Save form data to an S3 object
    s3.put_object(Bucket=bucket_name, Key=f'contact-form/{email}.txt', Body=f'Name: {name}\nEmail:
{email}\nMessage: {message}')
    return {
        'statusCode': 200,
        'body': 'Message sent successfully!'
    }
```

## OUTPUT


BYU Hawaii
iMacros
Teaching
Other
Personal
Fundamentals...
Your First Form
salkdjfowijesd
First Name
Jeff
Last Name
Sure
Country
Tonga
Subject
Submit
Oftentimes a copywrite and other information goes here


BYU Hawaii
iMacros
Teaching
Other
Personal
Fundamentals...
{"country": "tonga", "firstname": "Jeff", "lastname": "Sure", "subject": "salkdjfowijesd"}


SRI
EC
स्वदेशो भुवनत्रयम

## CONCLUSION

In conclusion, the integration of Amazon Web Services (AWS) S3, Python Lambda functions, and an API Gateway to create a simple web form presents a powerful and scalable solution for various web application needs. This architecture leverages the flexibility and reliability of AWS services to efficiently handle data storage, processing, and communication. Firstly, AWS S3 serves as a robust storage solution for storing form submissions and associated data. With its high availability, durability, and scalability, S3 ensures that data is securely stored and easily accessible whenever needed. Additionally, features like versioning and encryption enhance data protection and compliance with security standards. Python Lambda functions play a crucial role in processing form submissions and executing custom logic in response to events triggered by the API Gateway. Leveraging AWS Lambda allows for serverless computing, eliminating the need to provision or manage servers, and scaling automatically to handle varying workloads efficiently. Moreover, the ease of deploying and updating Lambda functions enables rapid iteration and development of the application. The API Gateway acts as a bridge between the frontend web form and the backend Lambda functions, facilitating seamless communication and data exchange. By defining RESTful APIs and integrating with Lambda functions, the API Gateway enables flexible routing, authentication, and validation of incoming requests. This ensures that data is transmitted securely and reliably between the client and backend services.