

PROJECT REPORT

Image to PDF converter

Anshul Choudhary (E22CSEU0952)

Mohit Jakhar(E22CSEU0941)

Akshit Naagar(E22CSEU0944)

A report submitted in part fulfilment of the degree of

B.Tech in Computer Science

Supervisor: Dr. Naveen Kumar



BENNETT
UNIVERSITY
THE TIMES GROUP

NAAC **GRADE A+**
ACCREDITED UNIVERSITY

School of Computer Science Engineering and Technology

Bennett University

April 29,2025

1. Introduction

This project presents a serverless web application that allows users to upload multiple images and convert them into a single PDF document. It leverages a set of AWS cloud services to create a scalable, secure, and fully managed solution. The application demonstrates the effective use of serverless architecture for handling real-world workloads with minimal infrastructure overhead.

2. Problem Statement

Creating PDFs from images is a common requirement for many users (e.g., students, professionals). Desktop software solutions may not be available or easy to use. This project provides an easy-to-access, browser-based alternative powered entirely by cloud services without maintaining any backend server infrastructure.

3. Objectives

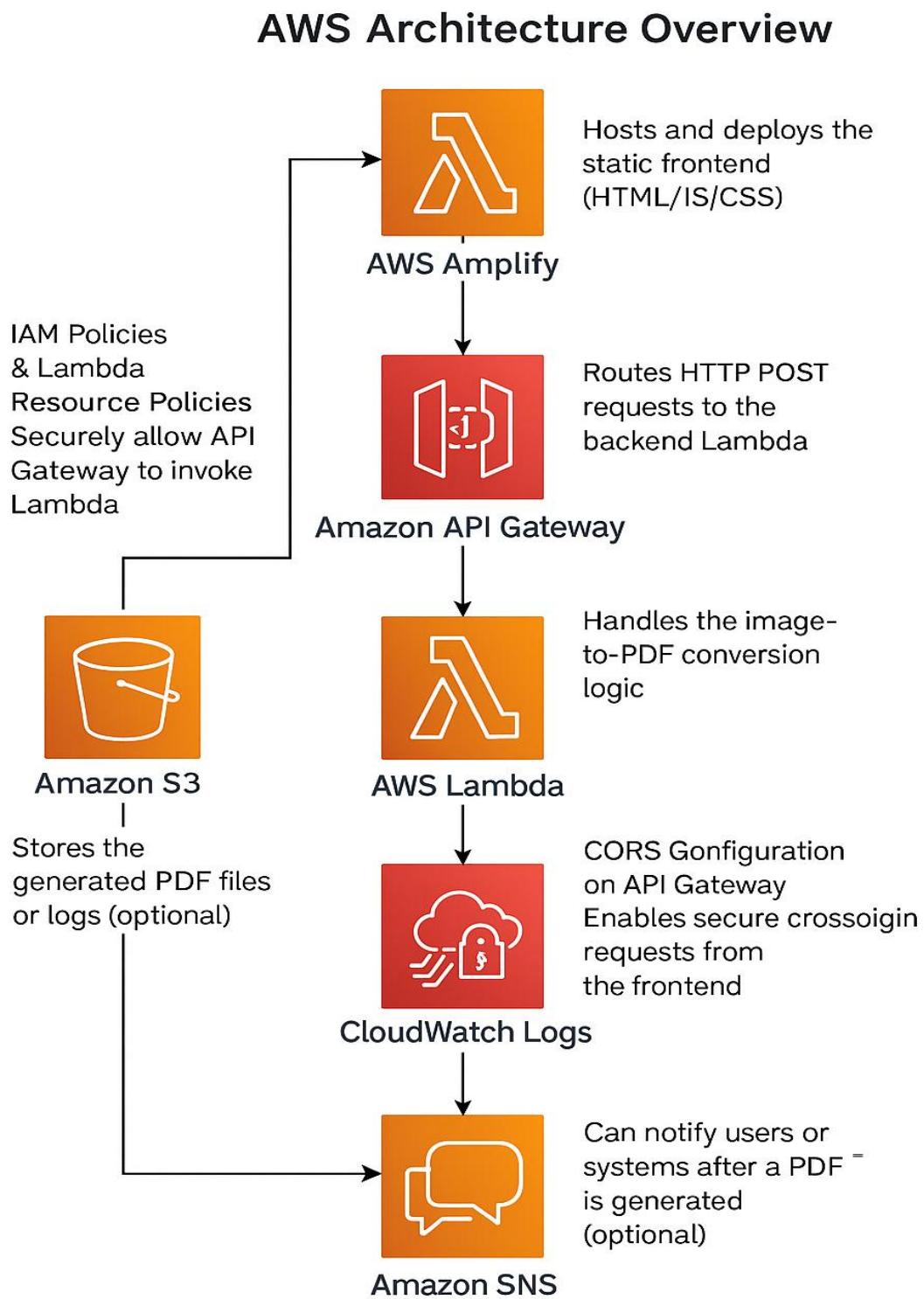
- Allow users to upload and preview multiple images via a browser.
- Convert uploaded images into a single PDF using cloud-based processing.
- Use serverless infrastructure to ensure scalability, low cost, and ease of maintenance.
- Securely store the resulting PDF files and monitor application activity.

4. AWS Services Used

AWS Service	Purpose
AWS Lambda	Image-to-PDF processing logic executed serverless
Amazon API Gateway	Exposes REST endpoint to trigger Lambda function
Amazon S3	Stores output PDF files and logs
AWS Amplify	Hosts the frontend (HTML/CSS/JS)
IAM	Controls access between API Gateway, Lambda, and S3
CloudWatch	Logs and monitors Lambda execution

5. Architecture Diagram

The application is designed using the following architecture:

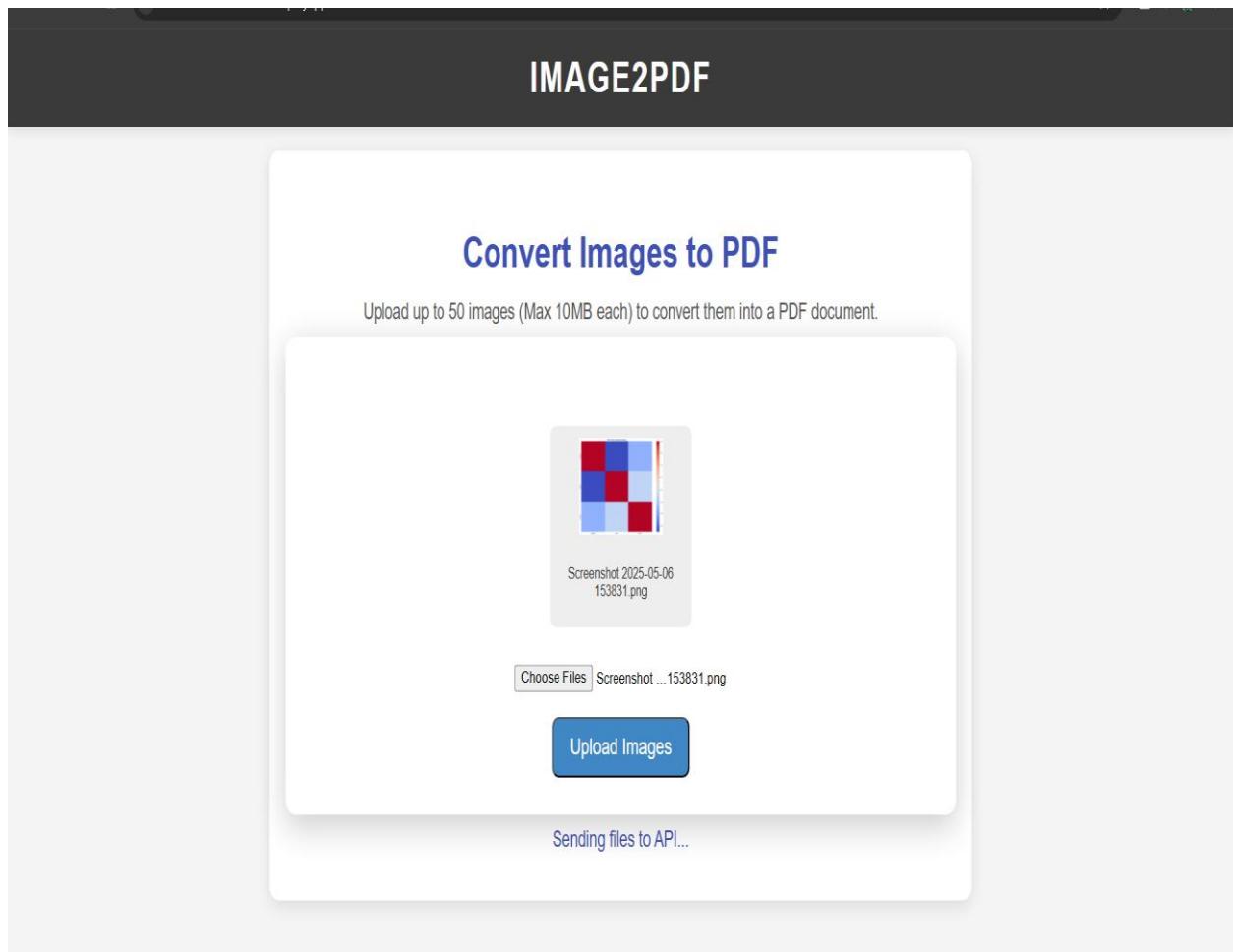


AWS Architecture

6. System Workflow

1. Frontend (User Interaction):

Hosted on Amplify, allows users to select and preview images.



2. Backend (Lambda Trigger):

On submission, images are sent to an API Gateway endpoint.

The screenshot shows the AWS API Gateway interface. On the left, there's a sidebar with options like 'API Gateway', 'APIs', 'Custom domain names', 'Domain name access associations', 'VPC links', 'Usage plans', 'API keys', 'Client certificates', and 'Settings'. The main area is titled 'APIs (1/1)' and contains a table with one row. The table columns are 'Name', 'Description', 'ID', 'Protocol', 'API endpoint type', and 'Created'. The single row shows 'ImageToPdfAPI' as the Name, with other details like ID (2h9sgaaum3), Protocol (REST), API endpoint type (Regional), and Created (2025-05-09). There are buttons for 'Delete' and 'Create API' at the top right of the table area. At the bottom of the page, there are links for 'CloudShell', 'Feedback', and copyright information: '© 2025, Amazon Web Services, Inc. or its affiliates.' followed by 'Privacy', 'Terms', and 'Cookie preferences'.

Name	Description	ID	Protocol	API endpoint type	Created
ImageToPdfAPI		2h9sgaaum3	REST	Regional	2025-05-09

3. Image Processing:

The API Gateway invokes a Lambda function that converts the images into a single PDF.

The screenshot shows the AWS Lambda function configuration page for 'ImageToPdfFunction'. The top navigation bar includes the AWS logo, a search bar, and user information for 'Anshulchoudhary'. The main content area displays the function details:

- Function Overview:** Shows the function name 'ImageToPdfFunction', a diagram icon, and a 'Layers' section indicating 0 layers.
- Destinations:** An 'API Gateway' destination is listed with 2 triggers. Buttons for '+ Add destination' and '+ Add trigger' are present.
- Description:** A brief description is provided.
- Last modified:** The function was last modified 21 hours ago.
- Function ARN:** arn:aws:lambda:us-east-1:762233738049:function:ImageToPdfFunction
- Function URL:** A link to the function's URL.

On the right side, there is a 'Tutorials' tab with a 'Create a simple web app' section. It includes a brief description and a list of steps:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

Buttons for 'Learn more' and 'Start tutorial' are available. The bottom navigation bar includes tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is selected.

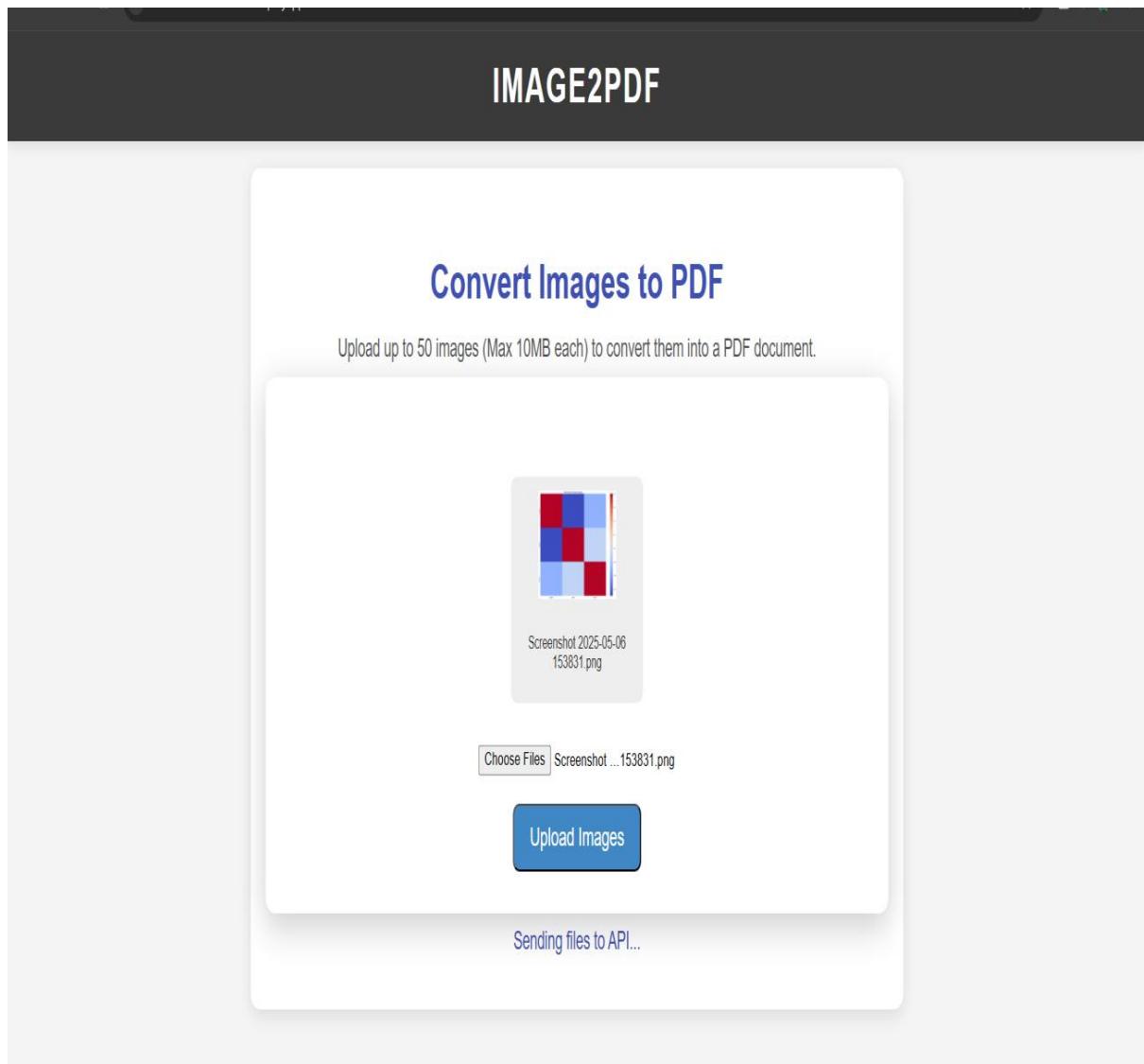
4. PDF Storage:

The generated PDF is stored in Amazon S3.

The screenshot shows the AWS S3 console interface. The left sidebar has a 'General purpose buckets' section with links for Directory buckets, Table buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and IAM Access Analyzer for S3. It also includes sections for Block Public Access settings and Storage Lens, and a Feature spotlight section with 11 items. The main content area shows a bucket named 'image2pdf-akshitnaagar'. The 'Objects' tab is selected, showing one object named 'uploads/'. There are buttons for Actions (Copy S3 URI, Copy URL, Download, Open, Delete, Create folder, Upload), a search bar for 'Find objects by prefix', and a table for listing objects. The table columns are Name, Type, Last modified, Size, and Storage class. The object 'uploads/' is listed as a Folder. The top navigation bar includes the AWS logo, a search bar, and user information for 'Anshulchoudhary'.

5. Status Response:

A success or error message is sent back to the frontend.



7. Project Directory Structure

plaintext

Copyedit

📦 image-to-pdf-converter

 └───(frontend/)

 | └───index.html

 | └───style.css

 | └───backend.js

 └───lambda/

 | └───image_to_pdf.py

 └───README.md

8. Deployment Steps

- **Lambda Function:**

Deploy `image_to_pdf.py` through the AWS Lambda Console with proper IAM execution roles and S3 access.

- **API Gateway:**

Create an HTTP POST endpoint integrated with Lambda and configure CORS.

- **Frontend (Amplify):**

Deploy the static files (`index.html`, `style.css`, `backend.js`) through AWS Amplify.

- **Testing:**

Use a browser or Postman to test the workflow.

9. Security Considerations

- CORS restricted to the Amplify domain.
- File type and size validation implemented in Lambda.

- IAM roles and resource policies set with the principle of least privilege.
- Optional usage plans or API keys can limit abuse.

10. Testing Summary

Test Case	Result
Upload multiple images	Passed
PDF generated successfully	Passed
Invalid file type rejected	Passed
Large image timeout handled	Partial (can improve with longer Lambda timeout)
Error message on failure	Passed

11. Challenges Faced

- Handling CORS configuration properly across services.
- Image encoding and ordering before PDF conversion in Lambda.
- Ensuring correct IAM roles to allow secure access between services.

12. Conclusion

This project demonstrates how a full-stack application can be developed and deployed using serverless technologies. The use of AWS services like Lambda, API Gateway, and Amplify reduces the need for infrastructure management and improves scalability. It is a cost-effective, user-friendly solution for real-world document processing.

13. Future Improvements

- Add user authentication with AWS Cognito.
- Provide downloadable links or email PDF results.
- Support for more image formats and drag-and-drop UI.
- Add database support (e.g., DynamoDB) to log user actions and analytics.