

Problem Statement

In most school WhatsApp groups, crucial messages like exam schedules, early closures, holidays or PTMs often get buried in hundreds of casual chats. Parents and Students may miss critical information that directly affect their routines.

With just a few AWS services – S3, Lambda, Bedrock, developed a scalable and intelligent pipeline that transforms unstructured chat noise into insightful school notifications.

This is a POC, and the solution can be developed to read messages from a WhatsApp Group in real time or **Microsoft teams** in real time and extract key announcements using Generative AI and **Amazon Quicksight** can be used for Visualization.

Ethical Note

1. Do not use for commercial purpose without WhatsApp approval
2. Avoid storing sensitive information
3. Ensure all have agreed
4. Do it only for your own groups

Tech Stack

1. Amazon S3 for WhatsApp txt chat files (raw input)
2. Amazon S3 - Store output in JSON format after post-processing
3. Amazon Lambda for parsing raw data and process LLM response and saves output to S3 bucket
4. Use Amazon Bedrock for extracting key messages

Step 1

1. Create S3 Buckets
 - a. school-chat-uploads for WhatsApp .txt uploads
 - b. school-chat-insights – for storing structured results
 - c. If you need to access via web/app uncheck all public access (Optional)
 - d. Enable versioning if required (optional)

	Name ▲	AWS Region ▼	IAM Access Analyzer ▼	Creation date ▼
<input type="radio"/>	school-chat-insights	US East (N. Virginia) us-east-1	View analyzer for us-east-1	June 11, 2025, 12:24:23 (UTC+05:30)
<input type="radio"/>	school-chats-uploads	US East (N. Virginia) us-east-1	View analyzer for us-east-1	June 11, 2025, 12:23:59 (UTC+05:30)

Step 2

1. Create a Lambda Function
2. Select Runtime as Python
3. Provide a meaningful name like **WhatsAppChatProcessor**
4. Create an IAM role with the necessary permissions
 - a. S3 ReadOnlyAccess
 - b. S3 FullAccess to write output
 - c. AmazonBedRock Full access or minimum permissions to invoke
5. Create a Function
6. Provide a meaningful name. I will use the same name **WhatsAppChatProcessor**
7. Select the IAM role from the drop down
8. Create a function
9. In the Code Source, Enter the Python Code
10. Add the trigger
11. Trigger Configuration will be S3
12. Select the Bucket Name for upload
13. The Event type will be put
14. Select the recursive invocation and click on Add

Step 3

1. Enable Amazon Bedrock Anthropic Claude 2.1
2. Make a note of the model Id
3. Replace the model Id in the Lambda function
4. Deploy the lambda function

Step 4

1. After deploying, test the event
2. Go to Test section
3. Create a new event
4. Provide an event name
5. Select template as Hello World
6. In the Event Json enter the following code
7. Click on Test

Event JSON

```
1 * {}
2 * "Records": [
3 *   {
4 *     "s3": {
5 *       "bucket": {
6 *         "name": "school-chats-uploads"
7 *       },
8 *       "object": {
9 *         "key": "WhatsApp Chat with [REDACTED].txt"
10 *       }
11 *     }
12 *   }
13 * ]
14 *
15 *
```

8. If you encounter an error Timeout, then in the lambda function – **General Configuration change the timeout to 15 sec or more if needed.**
9. If you get a max_token error then refer the document
 - a. <https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-anthropic-claude-text-completion.html>

Step 5

1. In the insights folder you will find the json extract .
2. Below is the extract of Whatsapp Text Messages Uploaded

```
11/06/25, 12:40 - Messages to yourself are end-to-end encrypted. No one else, not even WhatsApp, can read, listen to, or share them. Learn more.
11/06/25, 12:40 - Appu: School exam on 1st July
11/06/25, 12:41 - Appu: School function on Saturday
11/06/25, 12:41 - Appu: 🍕
11/06/25, 12:41 - Appu: 🍷
11/06/25, 12:53 - Appu: Teacher: Tomorrow the school will close at 1:30 PM due to staff meeting.
11/06/25, 12:53 - Appu: Parent: Thank you!
11/06/25, 12:53 - Appu: Teacher: History class test on Monday.
```

3. This is the python code what messages to extract

```
Extract messages related to: school closure, tests, holidays, PTM, early dismissals, exams.
```

4. This is the output in json format

Here are the key school-related announcements I extracted from the messages:

```
[
  {
    "date": null,
    "type": "exam",
    "message": "School exam on 1st July"
  },
  {
    "date": null,
    "type": "early_dismissal",
    "message": "Tomorrow the school will close at 1:30 PM due to staff meeting."
  },
  {
    "date": null,
    "type": "test",
    "message": "History class test on Monday."
  }
]
```

5. I haven't coded to extract the dates or infer the timestamps and date you will find Null in the above output.

1. IAM Role Creation

Name, review, and create

Role details

Role name

Enter a meaningful name to identify this role.

WhatsAppChatProcessor

Maximum 64 characters. Use alphanumeric '+' '@' '_' characters.

Description

Add a short explanation for this role.


Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use letters [A-Z and a-z], numbers [0-9], tabs, new lines, or any of the following characters: _+*, @-./[]!#\$%^&*(){}~"

2. IAM Role Creation - Permissions

Step 2: Add permissions

Permissions policy summary

Policy name 	Type	Attached as
AmazonBedrockFullAccess	AWS managed	Permissions policy
AmazonS3FullAccess	AWS managed	Permissions policy
AmazonS3ReadOnlyAccess	AWS managed	Permissions policy

3. Lambda Function

Create function [info](#)

Choose one of the following options to create your function.

☒ Author from scratch
Start with a simple Hello World example.

☐ Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

☐ Container image
Select a container image to deploy for your function.

Basic information

Function name

Enter a name that describes the purpose of your function.

WhatsAppChatProcessor

Function name must be 1 to 64 characters, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime [info](#)

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.13

Architecture [info](#)

Choose the instruction set architecture you want for your function code.

☐ arm64

☒ x86_64

Permissions [info](#)

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Existing role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions

☒ Use an existing role

☐ Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

WhatsAppChatProcessor

4. Lambda Function – Python Code

```
lambda_function.py X
lambda_function.py
1  import boto3
2  import re
3  import json
4
5  bedrock = boto3.client("bedrock-runtime", region_name="us-east-1")
6  s3 = boto3.client("s3")
7
8  def extract_messages(content):
9      messages = []
10     lines = content.splitlines()
11     for line in lines:
12         match = re.match(r'^(\d{1,2}/\d{1,2}/\d{2,4}), (\d{1,2}:\d{2}) - (.*)?: (.*)', line)
13         if match:
14             date, time, sender, message = match.groups()
15             messages.append(message)
16     return messages
17
```

5. Add a Trigger – When uploading the WhatsApp.txt

