



BotReminder

A cloud based reminder for regular users.

Project By [GROUP 4]:-

Udatya Deb

Prakash Pratap Singh

Manan Mal

Project Report

In partial fulfilment for the award of the degree
Of
BACHELOR OF TECHNOLOGY
IN COMPUTER SCIENCE ENGINEERING
Under the guidance of MR. SHOUVIK SARKAR



ARDENT COMPUTECH PVT LTD (AN ISO 9001:2015 CERTIFIED)
SDF Bldg, Module 132, GP Block, Sector V, Bidhanagar, Kolkata- 700091

NETAJI SUBHASH ENGINEERING COLLEGE, KOLKATA
September– December 2021


- 
- Title of the Project: BotReminder (A cloud based reminder for regular users.)
 - Project Members
 - UDATYA DEB
 - PRAKASH PRATAP SINGH
 - MANAN MAL

Table of Contents

SI No	Name of the Main Topic	Sub Topics	Page No.
1	Introduction	1.1 Acknowledgement	1
		1.2 Objective of the Project	2
2	System Design	2.1 Overall Plan	3
		2.2 Amazon AWS DynamoDB	3.1
		2.3 Amazon LEX	4
		2.4 AWS Lambda	5
		2.5 Amazon Eventbridge and CloudWatch	6

Table of Content

SI No	Name of the Main Topic	Sub Topics	Page No.
3	Basic Coding	3.1 Python	7
4	Implementation and Testing	4.1 Introduction	16
		4.2 Objective of testing	19
		4.3 Test Cases	22
5	Snapshots	5.1 Few test cases and service snapshots	25
6	Conclusion	6.1 Conclusion of the Project	36
7	Bibliography	7.1 Reference links and sites	37

Acknowledgement

Success of any project depends largely on the encouragement and guidelines of many others. We take this sincere opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project work. Our heartfelt thanks to Mr. Sanjoy Banerjee , for providing us the opportunity to develop the project at Ardent Computech Pvt. Ltd

We would like to show our greatest appreciation to Mr. Shouvik Sarkar , Project Manager at Ardent , Kolkata. We always feel motivated and encouraged every time by his valuable advice and constant inspiration ; without his encouragement and guidance , this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees , project assistants and other members at Ardent Computech Pvt. Ltd for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project , was vital for the success of this project.

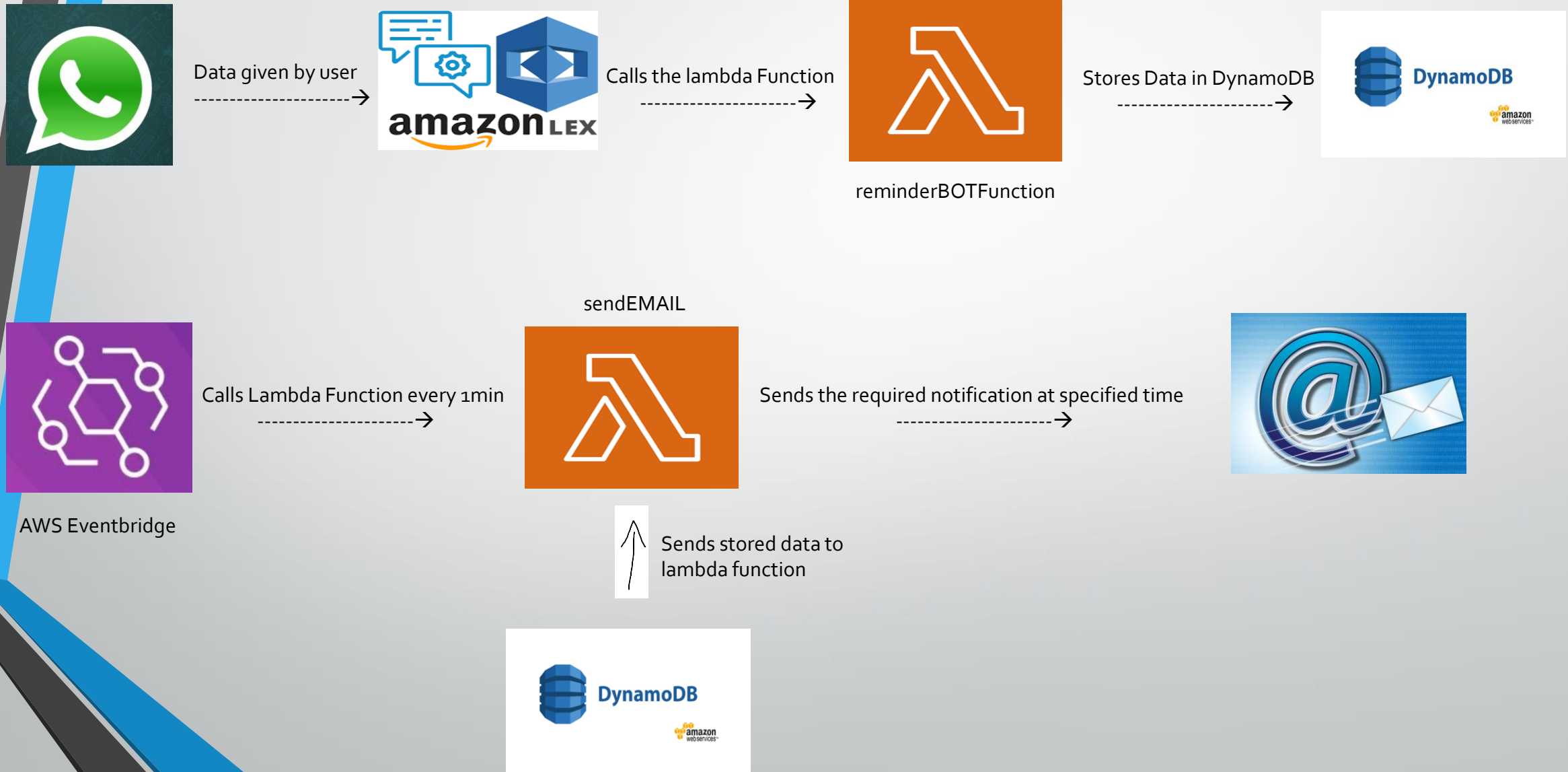


Objective of the project

The purpose of this project is to create a chatbot that can organize an individual's schedule for a day and can notify him at the time when a task is scheduled.

Overall Plan (System Design)

3



Amazon AWS Dynamodb



Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets offload the administrative burdens of operating and scaling a distributed database so that a user don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling.

DynamoDB also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data.

With DynamoDB, a user can create database tables that can store and retrieve any amount of data and serve any level of request traffic. He can scale up or scale down the tables' throughput capacity without downtime or performance degradation.

He can use the AWS Management Console to monitor resource utilization and performance metrics.

DynamoDB provides on-demand backup capability. It allows a user to create full backups of your tables for long-term retention and archival for regulatory compliance needs.

Amazon Lex



Amazon Lex is an AWS service for building conversational interfaces for applications using voice and text. With Amazon Lex, the same conversational engine that powers Amazon Alexa is now available to any developer, enabling any user to build sophisticated, natural language chatbots into any new and existing applications. Amazon Lex provides the deep functionality and flexibility of natural language understanding (NLU) and automatic speech recognition (ASR) so a person can build highly engaging user experiences with lifelike, conversational interactions, and create new categories of products.

Amazon Lex provides pre-built integration with AWS Lambda, and can easily be integrated with many other services on the AWS platform, including Amazon Cognito, AWS Mobile Hub, Amazon CloudWatch, and Amazon DynamoDB. Integration with Lambda provides bots access to pre-built serverless enterprise connectors to link to data in SaaS applications, such as Salesforce, HubSpot, or Marketo.

AWS Lambda



Lambda is a compute service that's used to run a code without provisioning or managing servers. Lambda runs the code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. With Lambda, a code can be run for virtually any type of application or backend service.

A developer organizes the code into Lambda functions. Lambda runs the function only when needed and scales automatically, from a few requests per day to thousands per second.

Lambda functions can be invoked using the Lambda API, or Lambda can run the functions in response to events from other AWS services. For example, we can use Lambda to:

- Build data-processing triggers for AWS services such as Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB.
- Process streaming data stored in Amazon Kinesis.
- Create your own backend that operates at AWS scale, performance, and security.

Amazon CloudWatch Events



Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in Amazon Web Services (AWS) resources. Using simple rules that can be quickly set up, a user can match events and route them to one or more target functions or streams. CloudWatch Events becomes aware of operational changes as they occur.

CloudWatch Events responds to these operational changes and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information. We can also use CloudWatch Events to schedule automated actions that self-trigger at certain times using cron or rate expressions.

A developer can configure the following AWS services as targets for CloudWatch Events:

- Amazon EC2 instances
- AWS Lambda functions
- Delivery streams in Amazon Kinesis Data Firehose
- Log groups in Amazon CloudWatch Logs
- Amazon ECS tasks
- Systems Manager Run Command
- Systems Manager Automation
- AWS Batch jobs

Basic codes (Python)

(For Storing information into the dynamodb from lex)

Code For Our Chatbot to Store Informations From User

import json

import datetime

import os

import time

import dateutil.parser

import logging

import boto3

import uuid

logger = logging.getLogger()

logger.setLevel(logging.DEBUG)

dyn_client = boto3.client('dynamodb')

TABLE_NAME = "tableReminder"

```
# def close(fulfillment_state, message):
# # def close(fulfillment_state, message,slots,intent_name):
#   response = {
# #     'sessionAttributes': session_attributes,
#     'dialogAction': {
#       'type': 'Close',
#       'fulfillmentState': fulfillment_state,
#       'message': message
#     }
#     # 'intentName': intent_name,
#     # "slots": {
#     #   "PhoneNo": slots['PhoneNo'],
#     #   "TimeReminder": slots['TimeReminder'],
#     #   "UserName": ["UserName"],
#     #   "topic": ['topic']
#     # }
#   }
#   return response
```

.....DynamDB save.....

9

```
def save(user_name, email, topic, timeReminder):
    id = str(uuid.uuid4())
    data = dyn_client.put_item(
        TableName=TABLE_NAME,
        Item={
            'id': {
                'S': id
            },
            'user_name': {
                'S': user_name
            },
            'timeReminder': {
                'S': timeReminder
            },
            'email': {
                'S': email
            },
            'topic': {
                'S': topic
            }
        }
    )
```

```
# def take(intent_request):  
#     slots = intent_request['currentIntent']['slots']  
#     user_name = slots['UserName']  
#     phone = slots['PhoneNo']  
#     topic = slots['topic']  
#     timeReminder = slots['TimeReminder']  
  
#     # .....Validation function.....  
  
#     # logger.debug(intent_request['invocationSource'])  
  
#     # if intent_request['invocationSource'] == 'DialogCodeHook':  
#     #     if len(phone)!=13:  
#     #         return "Invalid phone number"  
  
#     save(user_name, phone, topic, timeReminder)
```



```
# return close(  
#     # session_attributes,  
#     'Fulfilled',  
#     {  
#         'contentType': 'Plaintext',  
#         'content': 'Thanks {}, we have recorded your  
survey.'.format(user_name)  
#     },  
#     # slots,  
#     # intent_name  
# )
```

```
# def dispatch(intent_request):  
#     logger.debug('dispatch userId={},  
intentName={}'.format(intent_request['userId'],  
intent_request['currentIntent']['name']))  
#     # logger.debug('dispatch  
intentName={}'.format(intent_request['intentName']))  
#     # intent_name = intent_request['currentIntent']['name']  
#     # intent_name = intent_request['intentName']  
#     # if intent_name == 'botReminder':  
#     return take(intent_request)
```

```
def dispatch(event):
    # logger.debug(event)
    # logger.debug(event['currentIntent'])
    # logger.debug(event['currentIntent']['slots'])
    slots = event['currentIntent']['slots']
    user_name = slots['UserName']
    email = slots['Email']
    topic = slots['topic']
    timeReminder = slots['TimeReminder']
    save(user_name, email, topic, timeReminder)
    return {
        "dialogAction": {
            "type": 'Close',
            "fulfillmentState": 'Fulfilled',
            "message": {
                "contentType": "PlainText",
                "content": "Your reminder is succesfully set. Have a good
day!"
            }
        },
        # responseCard
    }
    # return response
```

```
# def lambda_handler(event, context):  
#     os.environ['TZ'] = 'Asia/Singapore'  
#     time.tzset()  
#     # logger.debug('event={}'.format('event'))  
#     logger.debug('event.bot.name={}'.format(event['bot']['name']))  
#     # logger.debug(event['invocationSource'])  
#     return dispatch(event)
```

```
def lambda_handler(event, context):  
    # logger.debug('event={}'.format(event))  
    return dispatch(event)
```

For accessing data from dynamodb and send the reminder mail

```
import boto3
import smtplib
from datetime import datetime
from zoneinfo import ZoneInfo
# import smtplib
from email.message import EmailMessage

def lambda_handler(event, context):
    client = boto3.resource("dynamodb")
    table = client.Table("tableReminder")
    tableReminder = table.scan()['Items']
    tableReminder_list = []
    i=0
    for x in tableReminder:
        tableReminder_list.insert(i,x)
        i+=1
    now = datetime.now(tz=ZoneInfo('Asia/Kolkata'))
    print(tableReminder_list)
```

```
for i in tableReminder_list:
    dt_string = now.strftime("%H:%M")
    a = i['timeReminder']
    # a = '14:12'
    if(dt_string == a):
        msg = EmailMessage()
        msg['Subject'] = 'Your Reminder'
        msg['From'] = 'BotReminder'
        msg['To'] = i['email']
        msg.set_content(f"Topic: {i['topic']} at {a}")

    server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
    server.login('botreminderaws@gmail.com', 'Pass@123')
    server.send_message(msg)
    server.quit
    # response = table.delete_item(Key={"id":i['id']})
    table.delete_item(Key={"id":i['id']})
```

IMPLEMENTATION AND TESTING

- A Software System Test Plan is a document that describes the objectives , scope, approach and focus of software testing effort.
- The process of preparing a test plan is a usual way to think the efforts needed to validate the software and ask ourselves, why will it be accepted.
- The complete document will help people outside the test group, understand why should they use the software and how will they be able to use it. Also they will get to know the process of the software.
- It should be thorough enough to be useful. Also everyone outside the test group will be able to read and understand it.

INTRODUCTION

17

- Testing is the process of running a system with the intention of finding errors.
- Testing enhances the integrity of a system by detecting deviations in design and errors in the system.
- Testing aims at detecting error-prone areas. This helps in the prevention of the errors in a system.
- Testing also adds value to the product by conforming to the user's requirements.
- The main purpose of Testing is to detect errors and error-prone areas in a system.
- Testing must be thorough and well planned. A partially tested system is as bad as an untested system. And the price of an untested and under-tested system is high.
- The implementation is the final and important phase. It involves user training, system testing in order to ensure successful running of the proposed system.
- The user tests the system and changes are made according to their needs.
- The Testing involves the Testing of the developed system using various kinds of data. While testing, errors are noted and correctness is the mode.

STEPS FOR SYSTEM TESTING :-

1. Test Environment Setup :- Create testing environment for the better quality of testing.
2. Create Test Case :- Generate test case for the testing process.
3. Create Test Data
4. Execute Test Case
5. Defect Reporting
6. Regression Testing
7. Log Defects
8. Retest

OBJECTIVES OF TESTING

The objective of our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. Our user interface to utilize these these functions, is designed to be user-friendly and provide an easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance and usability.

PROCESS OVERVIEW :-

The following represents the overall flow of the Testing Process :-

- Identify the requirements to be tested. All Test Cases shall be derived using the current Program Specifications.
- Identify which particular Test(s) will be used to test each module.

- Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
- Identify the expected results for each test.
- Document the test case configuration , test data , and expected results.
- Perform the test(s).
- Document the test data , test cases , and test configuration used during the testing process. The information shall be submitted via the Unit/System Test Report (STR).

- Successful unit testing is required before the unit is eligible for component integration/system testing.
-
- Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for the later technical analysis.
- Test Documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

TEST CASES

A Test case is a document that describes an input , action , or event and expected response , to determine if a feature of an application is working correctly. A Test case should contain particular such as test case identifier , test condition , test condition , input data.

Requirement expected results. The process of developing test cases can help find problems in the requirement or design of an application , since it requires completely thinking through the operation of the application.

TESTING STEPS :-

Unit testing :-

Unit Testing focuses efforts on the smallest unit of software design. This is known as Module Testing. The modules are tested separately. The Test is carried out during programming stage itself. In this step , each module is found to be working satisfactorily as regards to the expected output from the module.

INTEGRATION TESTING :-

Data can be lost across an interface. One module can have an adverse effect on another, sub functions, when combined, may not be linked in desired manner in major functions. Integration Testing is a systematic approach for constructing the program structure, while at the same time conducting test to uncover errors associated within the interface. The objective is to take unit tested modules and builds program structure. All the modules are combined and tested as a whole.

VALIDATION :-

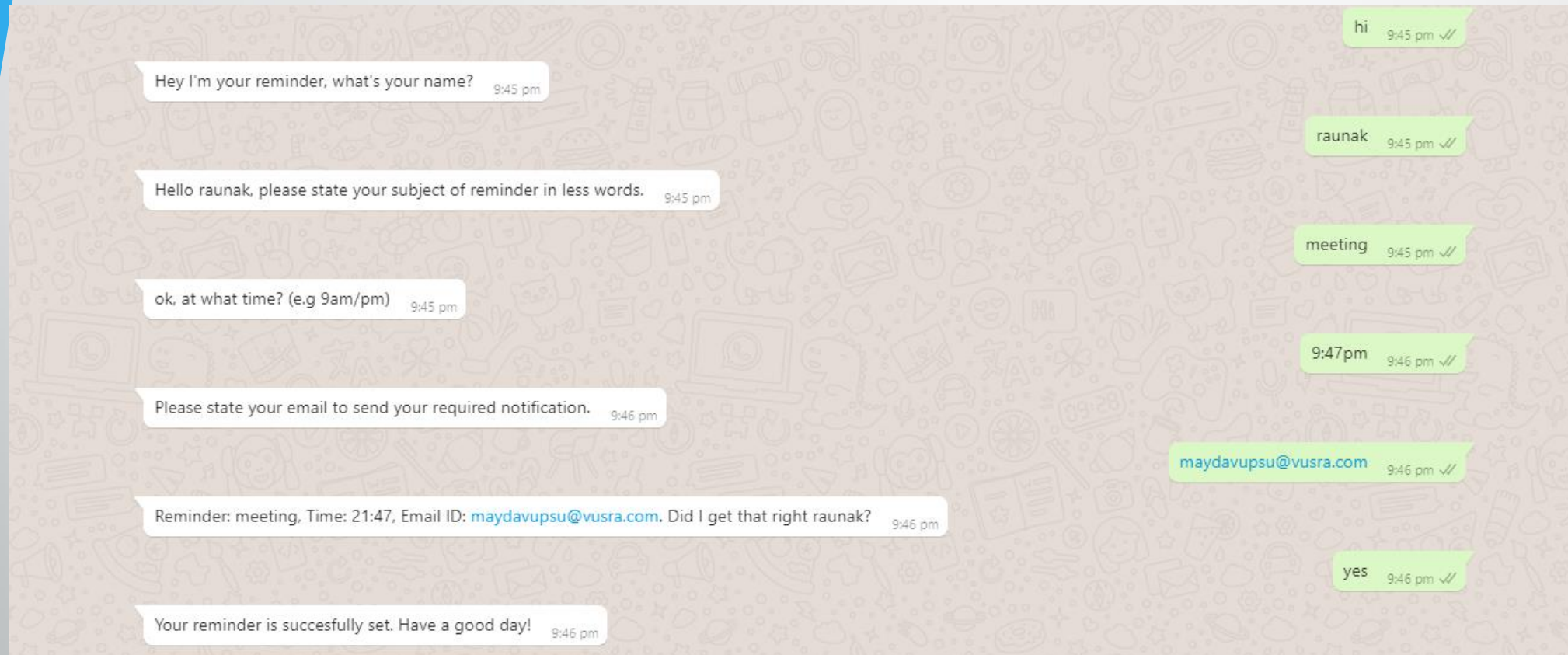
At the culmination of the integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begins in Validation Testing. Validation Testing can be defined in many ways. But a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists.

- The function or performance characteristics confirms to specification and are accepted.
- A deviation from specification is uncovered and a deficiency list is created.
- Proposed system under consideration has been tested by using validation test and found to be working satisfactorily.

Snapshots

Snapshots

- User with BOT conversation:-



Snapshots

- Amazon LEX Bot console :- (PART I)

botReminder Latest ▾

Build Publish ?

Editor Settings Channels Monitoring

Intents +

botReminder

Slot types +

No slots created

Error Handling

botReminder Latest ▾

Sample utterances ⓘ

e.g. I would like to book a flight. +

hello ✕

create a reminder ✕

hi ✕

► Lambda initialization and validation ⓘ

► Context ⓘ

▼ Slots ⓘ

Priority	Required	Name	Slot type	Version	Prompt	Settings
		e.g. Location	e.g. AMAZON.US_CITY ▾		e.g. What city?	+ ✕
1.	▾	<input checked="" type="checkbox"/>	UserName	AMAZON.US_FIRST_NAME ▾	Built-in ▾	Hey I'm your reminder, what's your name? ✕
2.	^ ▾	<input checked="" type="checkbox"/>	topic	AMAZON.AlphaNumeric ▾	Built-in ▾	Hello {UserName}, please state your subject of rer ✕
3.	^ ▾	<input checked="" type="checkbox"/>	TimeReminder	AMAZON.TIME ▾	Built-in ▾	ok, at what time? (e.g 9am/pm) ✕
4.	^	<input checked="" type="checkbox"/>	Email	AMAZON.EmailAddress ▾	Built-in ▾	Please state your email to send your required notif ✕

► Confirmation prompt ⓘ

▼ Fulfillment ⓘ

☒ AWS Lambda function ☐ Return parameters to client

Test Chatbot

Snapshots

- Amazon LEX Bot console :- (PART II)

The screenshot displays the Amazon LEX Bot console interface for configuring an intent. The top navigation bar includes the AWS logo, a search bar, and user information (Singapore, UDATYADEB). The left sidebar shows the 'botReminder' bot with tabs for Intents, Slot types, and Error Handling. The main area is titled 'botReminder Latest' and contains tabs for Editor, Settings, Channels, and Monitoring. The 'Editor' tab is active, showing the configuration for an intent named 'Email' (slot type: AMAZON.EmailAddress, built-in). The intent is currently in state 4. The configuration includes a confirmation prompt, fulfillment settings (AWS Lambda function: reminderBotFunction), and response options (Add Message, Enable response card, Wait for user reply). The 'Save Intent' button is visible at the bottom.

aws Services Search for services, features, blogs, docs, and more [Alt+S] Singapore UDATYADEB

< botReminder Latest Build Publish

Editor Settings Channels Monitoring

Intents +

botReminder

Slot types +

No slots created

Error Handling

4. ^ ✓ Email AMAZON.EmailAddress Built-in Please state your email to send your required notification

Confirmation prompt ⓘ

Fulfillment ⓘ

☒ AWS Lambda function ☐ Return parameters to client

Lambda function reminderBotFunction View in Lambda console

Version or alias Latest

Response ⓘ

+ Add Message

☐ Enable response card

☐ Wait for user reply
If the user says "no," the following message will be presented.

* Required

Save Intent Detach Intent

Snapshots

- Amazon DynamoDB:-

The screenshot displays the Amazon DynamoDB console interface. At the top, there's a navigation bar with the AWS logo, 'Services' link, a search bar, and regional settings (Singapore) and user identity (UDATYADEB). A blue banner at the top of the console area announces that the new DynamoDB console is now complete and becomes the default experience, encouraging user feedback.

The left-hand navigation pane shows the 'DynamoDB' section with links to Dashboard, Tables, Items (highlighted with a 'New' tag), PartiQL editor (also with a 'New' tag), Backups, Exports to S3, and Reserved capacity. Below this, the 'DAX' section includes links to Clusters, Subnet groups, Parameter groups, and Events. At the bottom of the pane, there's a feedback section with links to 'Tell us what you think' and 'Return to the previous console experience'.

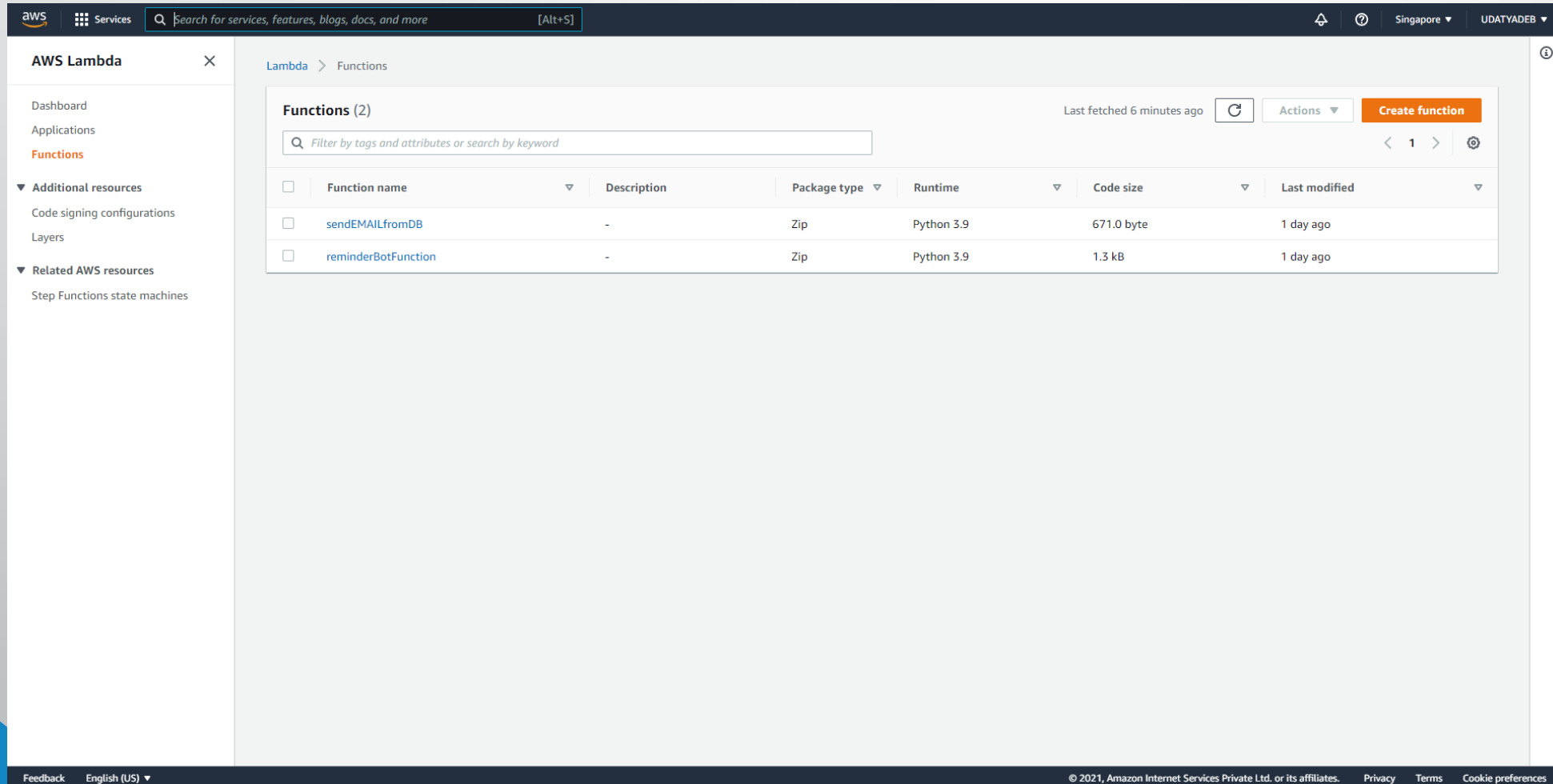
The main content area is titled 'Items' and shows the 'tableReminder' table selected. It includes a 'Tables (1)' sidebar with a search bar and a list of tables. The 'tableReminder' table is selected, and its details are shown on the right. The 'tableReminder' section has a 'View table details' button and a 'Scan' button. Below the 'Scan' button, there's a 'Table or index' dropdown menu set to 'tableReminder'. A 'Filters' section is also visible. The 'Run' button is highlighted, and a status message indicates 'Completed' with 'Read capacity units consumed: 0.5'.

The 'Items returned (1)' section shows a table with one item. The table has columns: id, email, timeRem..., topic, and user_name. The item has the following values: id: 001, email: abc@xyz.com, timeRem...: 00 am/pm, topic: beta testing, and user_name: tester.

The bottom of the console features a footer with 'Feedback', 'English (US)' language selection, and copyright information: '© 2021, Amazon Internet Services Private Ltd. or its affiliates.' along with links for 'Privacy', 'Terms', and 'Cookie preferences'.

Snapshots

- Amazon Lambda Functions:-



The screenshot displays the AWS Lambda console interface. The top navigation bar includes the AWS logo, a search bar, and the user's name 'UDATYADEB'. The left sidebar shows the 'AWS Lambda' section with options for Dashboard, Applications, Functions, and Additional resources. The main content area, titled 'Lambda > Functions', shows a list of functions. The list includes columns for Function name, Description, Package type, Runtime, Code size, and Last modified. Two functions are listed: 'sendEMAILfromDB' and 'reminderBotFunction', both using Python 3.9 runtime and Zip package type. The 'Last modified' column shows '1 day ago' for both functions. A 'Create function' button is visible in the top right corner of the function list area.

Functions (2) Last fetched 6 minutes ago [Actions](#) [Create function](#)

<input type="checkbox"/>	Function name	Description	Package type	Runtime	Code size	Last modified
<input type="checkbox"/>	sendEMAILfromDB	-	Zip	Python 3.9	671.0 byte	1 day ago
<input type="checkbox"/>	reminderBotFunction	-	Zip	Python 3.9	1.3 kB	1 day ago

© 2021, Amazon Internet Services Private Ltd. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Snapshots

- Amazon Lambda Function:- (reminderBotFunction)

The screenshot displays the AWS Lambda console interface for a function named 'reminderBotFunction'. The top navigation bar includes the AWS logo, 'Services' link, a search bar, and regional settings for 'Singapore' and 'UDATYADEB'. The breadcrumb trail shows 'Lambda > Functions > reminderBotFunction'. The function name 'reminderBotFunction' is prominently displayed at the top of the main content area, accompanied by 'Throttle', 'Copy ARN', and 'Actions' buttons. Below this, the 'Function overview' section provides details: a description of '-', last modified '1 day ago', and the function ARN 'arn:aws:lambda:ap-southeast-1:192333731437:function:reminderBotFunction'. It also shows 'Layers' as '(0)' and buttons for 'Add trigger' and 'Add destination'. A tabbed interface below the overview includes 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions', with 'Code' currently selected. The 'Code source' section shows a file explorer with 'reminderBotFunction' and 'lambda_function.py'. The code editor displays a Python function 'save' that interacts with a DynamoDB table. The code includes comments and a return statement. The bottom status bar shows '© 2021, Amazon Internet Services Private Ltd. or its affiliates.' along with links for 'Privacy', 'Terms', and 'Cookie preferences'.

aws Services Search for services, features, blogs, docs, and more [Alt+S] Singapore UDATYADEB

Lambda > Functions > reminderBotFunction

reminderBotFunction

Throttle Copy ARN Actions

▼ Function overview Info

reminderBotFunction

Layers (0)

+ Add trigger + Add destination

Description -

Last modified 1 day ago

Function ARN
arn:aws:lambda:ap-southeast-1:192333731437:function:reminderBotFunction

Code Test Monitor Configuration Aliases Versions

Code source Info Upload from

File Edit Find View Go Tools Window Test Deploy Changes deployed

Go to Anything (Ctrl-P)

Environment

reminderBotFunction

lambda_function.py

```
35 #  
36 #  
37 # return response  
38 # # .....DynamoDB save.....  
39  
40 def save(user_name, email, topic, timeReminder):  
41     id = str(uuid.uuid4())  
42     data = dyn_client.put_item(  
43         TableName=TABLE_NAME,  
44         Item={  
45             'id': {  
46                 'S': id  
47             },  
48             'user_name': {  
49                 'S': user_name  
50             },  
51             'timeReminder': {  
52                 'S': timeReminder
```

Feedback English (US) © 2021, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

Snapshots

- Amazon Lambda Function:- (sendEMAILfromDB)

The screenshot displays the AWS Lambda console interface for a function named 'sendEMAILfromDB'. The top navigation bar includes the AWS logo, 'Services', a search bar, and regional settings for 'Singapore' and 'UDATVADEB'. The breadcrumb trail shows 'Lambda > Functions > sendEMAILfromDB'. The function name 'sendEMAILfromDB' is prominently displayed, with buttons for 'Throttle', 'Copy ARN', and 'Actions'. Below this, the 'Function overview' section shows the function icon, name, and a 'Layers' section with '(0)' layers. A description field is empty, and the 'Last modified' timestamp is '5 minutes ago'. The 'Function ARN' is 'arn:aws:lambda:ap-southeast-1:192333731437:function:sendEMAILfromDB'. A trigger section shows 'EventBridge (CloudWatch Events)' as the selected trigger, with an 'Add trigger' button. A table with one row shows the trigger details. The 'Code source' section is active, displaying a code editor with the following Python code:

```
1 import boto3
2 import json
3 import smtplib
4 from datetime import datetime
5 from zoneinfo import ZoneInfo
6 from email.message import EmailMessage
7
8 def lambda_handler(event, context):
9     client = boto3.resource("dynamodb")
10    table = client.Table("tableReminder")
11    tableReminder = table.scan()['Items']
12    tableReminder_list = []
13    i=0
14    for x in tableReminder:
15        #client = boto3.client('sns', 'ap-southeast-1')
16        tableReminder_list.insert(i,x)
17        i+=1
18    now = datetime.now(tz=ZoneInfo('Asia/Kolkata'))
```

The code editor includes a file explorer on the left showing the project structure: 'sendEMAILfromDB' (folder) and 'lambda_function.py' (file). The bottom status bar shows '© 2021, Amazon Internet Services Private Ltd. or its affiliates.' and links for 'Privacy', 'Terms', and 'Cookie preferences'.

Snapshots

- Amazon EventBridge:- (PART I)

The screenshot displays the Amazon EventBridge console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and user information (Singapore, UDATYADEB). A left-hand sidebar lists various EventBridge features: Getting started, Event buses, Rules (highlighted in orange), Archives, Replays, Integration, Partner event sources, API destinations, Schema registry, and Schemas. Below this is a 'Documentation' link. The main content area is titled 'Rules' and includes a sub-header 'Amazon EventBridge > Rules'. A descriptive text states: 'A rule watches for specific types of events. When a matching event occurs, the event is routed to the targets associated with the rule. A rule can be associated with one or more targets.' Below this, there's a 'Select event bus' section with a dropdown menu currently set to 'default'. Further down, a 'Rules (1/1)' section contains a search bar, a status filter set to 'Any status', and a table of existing rules. The table has columns for Name, Status, Type, and Description. One rule is listed: 'sendEmail' with a status of 'Enabled' (indicated by a green checkmark icon), Type 'Scheduled Standard', and Description 'call function to send email at given time.' Action buttons (Refresh, Edit, Delete, Enable, and Create rule) are located above the table.

Amazon EventBridge

EventBridge - Learning content
Tell us what topics you would like to see more learning material for (tutorials, videos, blog posts, etc.).

Amazon EventBridge > Rules

Rules

A rule watches for specific types of events. When a matching event occurs, the event is routed to the targets associated with the rule. A rule can be associated with one or more targets.

Select event bus

Event bus
Select or enter event bus name

default

Rules (1/1)

Find rules Any status

	Name	Status	Type	Description
<input type="radio"/>	sendEmail	Enabled	Scheduled Standard	call function to send email at given time.

Refresh Edit Delete Enable Create rule

Snapshots

- Amazon EventBridge:- (PART II) Cloudwatch Logs

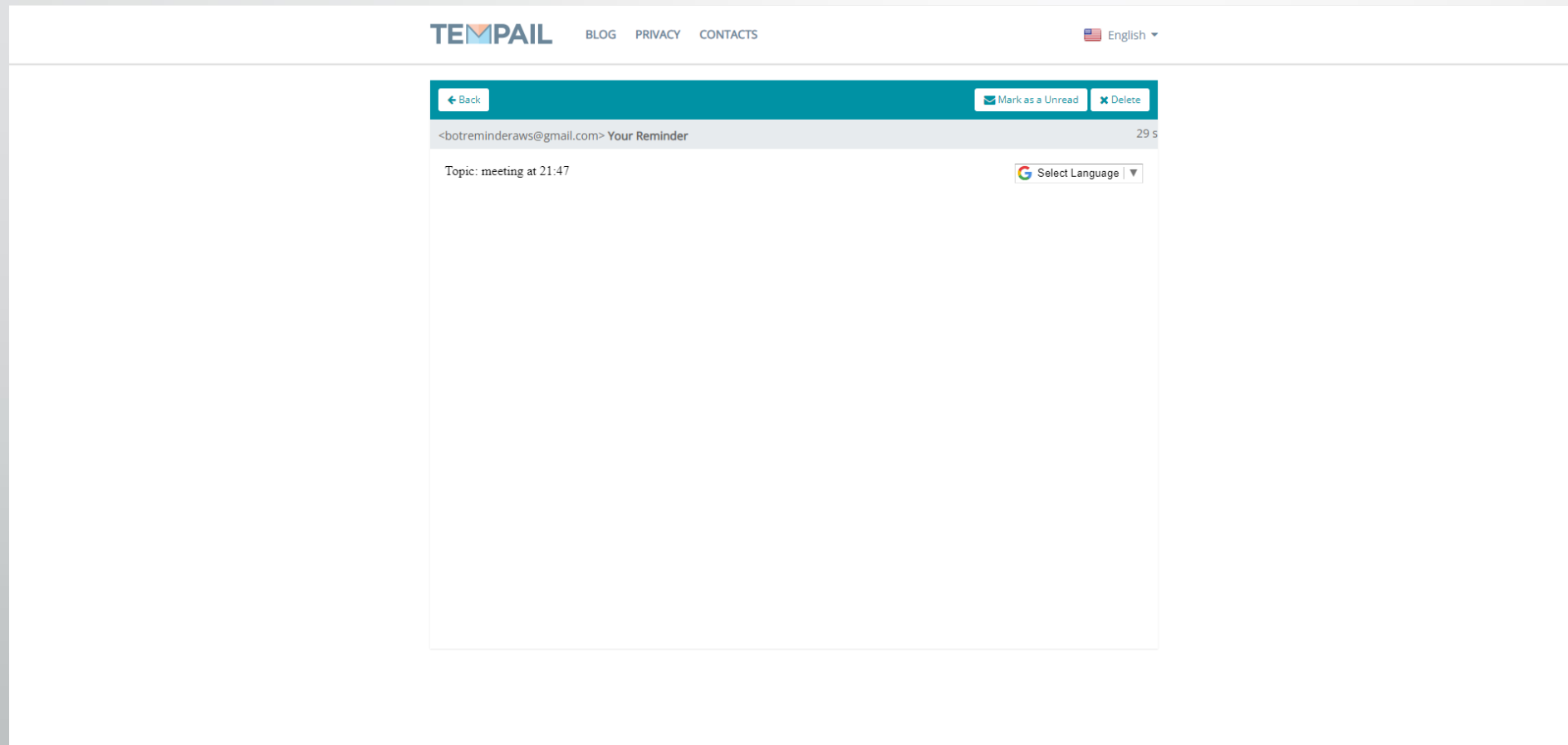
The screenshot displays the AWS CloudWatch console interface. On the left, a sidebar menu includes options for CloudWatch, Favorites, Dashboards, Alarms (with 2 alerts), Logs, Log groups, Logs Insights, Metrics, Events, Application monitoring, Insights, Settings, and Getting Started. The main content area shows a list of log events. Each event entry includes a timestamp, a message, and a details section. The messages are structured as JSON objects containing metadata like 'id', 'email', 'topic', 'timeReminder', and 'user_name'. The details section provides information about the request ID, duration, billed duration, memory size, and memory used.

Time	Message	Details
2021-12-08T18:26:39.581+05:30	START RequestId: b66f8417-4c7a-4073-9603-9ea892db8bd2 Version: \$LATEST	
2021-12-08T18:26:39.631+05:30	[{'id': '001', 'email': 'abc@xyz.com', 'topic': 'beta testing', 'timeReminder': '00 am/pm', 'user_name': 'tester'}]	
2021-12-08T18:26:39.634+05:30	END RequestId: b66f8417-4c7a-4073-9603-9ea892db8bd2	
2021-12-08T18:26:39.634+05:30	REPORT RequestId: b66f8417-4c7a-4073-9603-9ea892db8bd2 Duration: 49.85 ms Billed Duration: 50 ms Memory Size: 512 MB Max Memory Used: 68 MB	
2021-12-08T18:27:39.926+05:30	START RequestId: d2acbaa8-6da2-49cd-8832-0dbf68f31830 Version: \$LATEST	
2021-12-08T18:27:40.018+05:30	[{'id': '001', 'email': 'abc@xyz.com', 'topic': 'beta testing', 'timeReminder': '00 am/pm', 'user_name': 'tester'}, {'id': '5342f9b7-85dd-45c8-b03f-e177c01dc4d2', 'email': 'durtajar...'}]	
2021-12-08T18:27:42.399+05:30	END RequestId: d2acbaa8-6da2-49cd-8832-0dbf68f31830	
2021-12-08T18:27:42.399+05:30	REPORT RequestId: d2acbaa8-6da2-49cd-8832-0dbf68f31830 Duration: 2466.66 ms Billed Duration: 2467 ms Memory Size: 512 MB Max Memory Used: 68 MB	
2021-12-08T18:28:39.869+05:30	START RequestId: 178fc273-d680-4ba7-9945-b99df6992cba Version: \$LATEST	
2021-12-08T18:28:39.937+05:30	[{'id': '001', 'email': 'abc@xyz.com', 'topic': 'beta testing', 'timeReminder': '00 am/pm', 'user_name': 'tester'}]	
2021-12-08T18:28:39.939+05:30	END RequestId: 178fc273-d680-4ba7-9945-b99df6992cba	
2021-12-08T18:28:39.939+05:30	REPORT RequestId: 178fc273-d680-4ba7-9945-b99df6992cba Duration: 68.08 ms Billed Duration: 69 ms Memory Size: 512 MB Max Memory Used: 69 MB	
2021-12-08T18:29:39.782+05:30	START RequestId: 5b00472f-6169-4240-9704-fae03e0b7e52 Version: \$LATEST	
2021-12-08T18:29:39.850+05:30	[{'id': '001', 'email': 'abc@xyz.com', 'topic': 'beta testing', 'timeReminder': '00 am/pm', 'user_name': 'tester'}]	
2021-12-08T18:29:39.852+05:30	END RequestId: 5b00472f-6169-4240-9704-fae03e0b7e52	
2021-12-08T18:29:39.852+05:30	REPORT RequestId: 5b00472f-6169-4240-9704-fae03e0b7e52 Duration: 67.40 ms Billed Duration: 68 ms Memory Size: 512 MB Max Memory Used: 69 MB	
2021-12-08T18:30:39.856+05:30	START RequestId: af724161-3023-44c6-9a78-075f90bcd0c Version: \$LATEST	
2021-12-08T18:30:39.919+05:30	[{'id': '001', 'email': 'abc@xyz.com', 'topic': 'beta testing', 'timeReminder': '00 am/pm', 'user_name': 'tester'}]	
2021-12-08T18:30:39.921+05:30	END RequestId: af724161-3023-44c6-9a78-075f90bcd0c	
2021-12-08T18:30:39.921+05:30	REPORT RequestId: af724161-3023-44c6-9a78-075f90bcd0c Duration: 62.91 ms Billed Duration: 63 ms Memory Size: 512 MB Max Memory Used: 69 MB	
2021-12-08T18:31:39.623+05:30	START RequestId: 20aeb23e-12df-4686-ac59-f5d3bc2d0d83 Version: \$LATEST	
2021-12-08T18:31:39.675+05:30	[{'id': '001', 'email': 'abc@xyz.com', 'topic': 'beta testing', 'timeReminder': '00 am/pm', 'user_name': 'tester'}]	
2021-12-08T18:31:39.677+05:30	END RequestId: 20aeb23e-12df-4686-ac59-f5d3bc2d0d83	
2021-12-08T18:31:39.677+05:30	REPORT RequestId: 20aeb23e-12df-4686-ac59-f5d3bc2d0d83 Duration: 50.87 ms Billed Duration: 51 ms Memory Size: 512 MB Max Memory Used: 69 MB	
2021-12-08T18:32:39.857+05:30	START RequestId: 0c0ff9fb-e055-4c2e-bc5b-b780f5632080 Version: \$LATEST	
2021-12-08T18:32:39.916+05:30	[{'id': '001', 'email': 'abc@xyz.com', 'topic': 'beta testing', 'timeReminder': '00 am/pm', 'user_name': 'tester'}]	
2021-12-08T18:32:39.918+05:30	END RequestId: 0c0ff9fb-e055-4c2e-bc5b-b780f5632080	
2021-12-08T18:32:39.918+05:30	REPORT RequestId: 0c0ff9fb-e055-4c2e-bc5b-b780f5632080 Duration: 58.89 ms Billed Duration: 59 ms Memory Size: 512 MB Max Memory Used: 69 MB	
2021-12-08T18:33:39.517+05:30	START RequestId: fab09872-8e81-4870-a2d5-73de1821b019 Version: \$LATEST	
2021-12-08T18:33:39.574+05:30	[{'id': '001', 'email': 'abc@xyz.com', 'topic': 'beta testing', 'timeReminder': '00 am/pm', 'user_name': 'tester'}]	
2021-12-08T18:33:39.576+05:30	END RequestId: fab09872-8e81-4870-a2d5-73de1821b019	
2021-12-08T18:33:39.576+05:30	REPORT RequestId: fab09872-8e81-4870-a2d5-73de1821b019 Duration: 56.40 ms Billed Duration: 57 ms Memory Size: 512 MB Max Memory Used: 69 MB	
2021-12-08T18:34:39.898+05:30	START RequestId: c1c4526b-cfb3-43b8-979b-e55a5d14f472 Version: \$LATEST	
2021-12-08T18:34:39.967+05:30	[{'id': '001', 'email': 'abc@xyz.com', 'topic': 'beta testing', 'timeReminder': '00 am/pm', 'user_name': 'tester'}]	
2021-12-08T18:34:39.969+05:30	END RequestId: c1c4526b-cfb3-43b8-979b-e55a5d14f472	
2021-12-08T18:34:39.969+05:30	REPORT RequestId: c1c4526b-cfb3-43b8-979b-e55a5d14f472 Duration: 68.65 ms Billed Duration: 69 ms Memory Size: 512 MB Max Memory Used: 69 MB	

No newer events at this moment. [Auto retry paused](#). [Resume](#)

Snapshots

- EMAIL NOTIFICATION RECEIVED :-



Conclusion

The project has been appreciated by all the users in the organization. It is easy to use , since it uses the AWS provided in the user dialog. User friendly screens are provided. The usage of software increases the efficiency , decreases the effort. It has been thoroughly tested and implemented.

The project “BotReminder” is the ideal place for every person who wants to set a reminder , or create a schedule to get notified at specified time , just with a click sitting in his/her room.

The software collects all the required information from the user to provide a smooth experience. The software provides a reliable platform as it as it does NOT RETAIN any given data as soon as the user is notified through E-Mail. Though our project’s topic was simple, but the implementation of various AWS micro-services in our project has further enhanced our knowledge regarding the use of AWS platform, which I am sure will be useful in the future.

Bibliography

[Deleting Data from a DynamoDB Table using AWS Lambda – YouTube](#)

[How To Send Email In Python via smtplib – YouTube](#)

[AWS Lambda Tutorial | Use AWS CloudWatch Events to schedule Lambda function invocation – YouTube](#)

[Writing data to DynamoDB \(Getting started with AWS Lambda, part 7\) – YouTube](#)

[How to Integrate AWS lex bot with whatsapp – YouTube](#)

[Create Your Own Amazon Lex Chatbot - Full tutorial - YouTube](#)



Thank You.