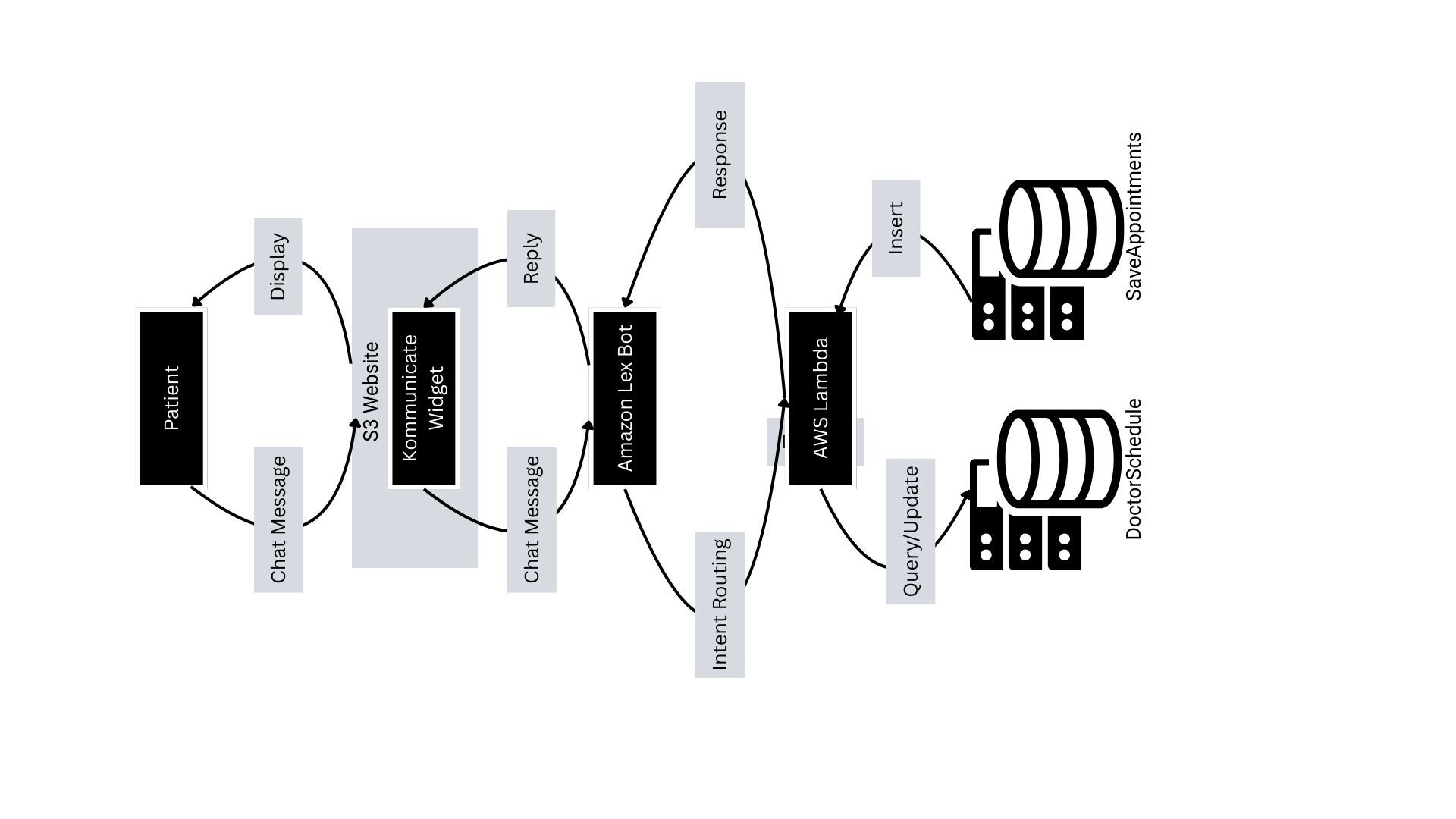
**Configuring Amazon Lex Chatbot for Clinic Booking System**

1. **Introduction**

This project integrates an Amazon Lex chatbot with AWS Lambda and DynamoDB to create an interactive conversational experience. The chatbot is deployed on a static website hosted on Amazon S3 and visually enhanced with Kommunicate for a modern chat interface.

1. **Architecture Overview**



1. **Create a Database in DynamoDB**

I created two DynamoDB tables (**DoctorSchedule** and **Appointments**) and used Python scripts to generate doctor schedules and save patient appointments. The entire setup is done on **Ubuntu Linux**, using a virtual environment for Python and **boto3** for AWS integration.

Step 1: Environment Setup on Ubuntu

1.1 Install Python & Pip (if not already installed)

sudo apt update

sudo apt install python3 python3-pip -y

1.2 Create a Python Virtual Environment

python3 -m venv myenv

source myenv/bin/activate

1.3 Install Boto3 inside the environment

pip install boto3

Step 2: AWS CLI Configuration

2.1 Install AWS CLI

sudo apt install awscli -y

2.2 Create an IAM User (via AWS Console)

* Create a new user (e.g., dynamodb-bot-user)
* Assign permissions: AmazonDynamoDBFullAccess
* Generate and copy: Access Key ID and Secret Access Key

2.3 Configure AWS CLI on Ubuntu

aws configure

Provide the values - Credentials are saved in ~/.aws/credentials

AWS Access Key ID [None]: <your-access-key>

AWS Secret Access Key [None]: <your-secret-key>

Default region name [None]: ap-southeast-1 # or your region

Default output format [None]: json

Step 3: Create DynamoDB Tables

3.1 Create **DoctorSchedule** Table

aws dynamodb create-table \

--table-name DoctorSchedule \

--attribute-definitions \

AttributeName=DoctorName,AttributeType=S \

AttributeName=Date\_Time,AttributeType=S \

--key-schema \

AttributeName=DoctorName,KeyType=HASH \

AttributeName=Date\_Time,KeyType=RANGE \

--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5

3.2 Create Appointments Table

aws dynamodb create-table \

--table-name Appointments \

--attribute-definitions \

AttributeName=AppointmentId,AttributeType=S \

--key-schema \

AttributeName=AppointmentId,KeyType=HASH \

--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5

### **Python Script – Generate Doctor Schedule**

#### **Save the Script**

Create a file generate\_schedule.py

vim generate\_schdeule.py - add code to this file

#### **Run the Script**

#### Activate the environment if not already: - source myenv/bin/activate

Run the code - python3 generate\_schedule.py

**Python Script – Save Appointments**

Create a file save\_appointment.py - vim save\_appointment.py

Run the script - python3 save\_appointment.py

Note - Use **deactivate** to exit the Python environment

1. **Creating Amazon Lex Chatbot for Booking Appointments**

This section explains how I created an **Amazon Lex V2 bot** for booking doctor appointments at our clinic. It includes:

* One intent: BookingAppointmentIntent
* Six slots for collecting user information
* A custom slot type for doctor names
* Built-in **confirmation prompt** before final booking
* Integration-ready design for Lambda & DynamoDB

Step 1: Create a Lex Bot (V2)

1. Go to Amazon Lex Console: <https://console.aws.amazon.com/lexv2>
2. Click “Create bot”
3. Fill in basic info:
   * **Bot name**: ClinicBookingBot
   * **IAM role**: Create a new one automatically
   * **Language**: English (US)
   * **Children’s data**: No
   * **Idle session timeout**: Default
   * Enable **Sentiment analysis** (optional)
4. Click **“Next”** → **“Create Bot”**

Step 2: Create the Intent — BookingAppointmentIntent

1. Go to your bot → Under *Intents*, click “Add intent” → “Create intent”
2. Name: BookingAppointmentIntent

### Step 3: Add Sample Utterances

In the intent, add a few examples like:

I want to book an appointment

Book a doctor

I need to see {DoctorName}

Book me with Dr. {DoctorName} on {Date} at {Time}

### **Step 4: Create Custom Slot Types**

#### **a. DoctorNameType (Custom)**

* Go to **Slot types → Create slot type**
* Name: DoctorNameType
* Add values like: Dr. Sarah Jones, Dr. Sophia Lee , Dr. David Park

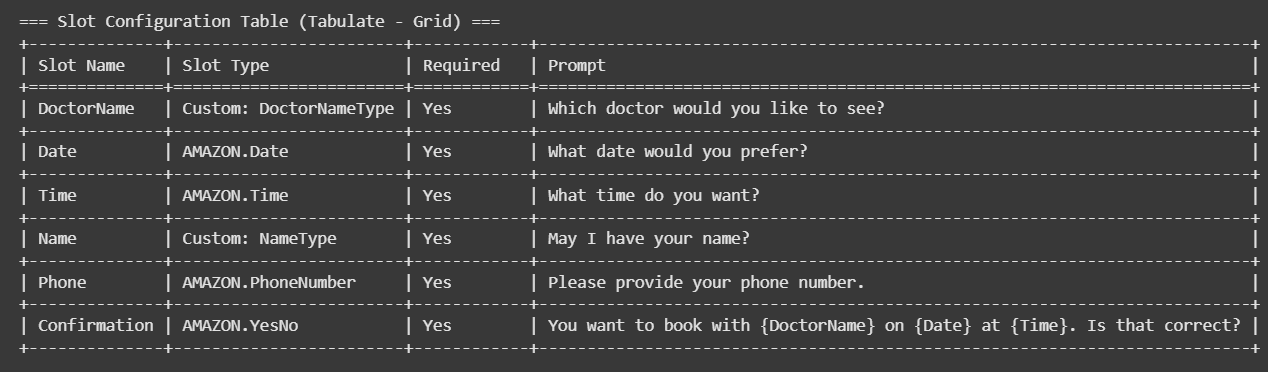
#### **b. NameType (Free text)**

#### Create another slot type: NameType

* Choose **"Don't add values"** → it will accept any name entered by the user

**Step 5: Define Slots**

Add the following **6 slots**, in the exact order you want Lex to ask

****

**Step 6: Enable Built-in Confirmation Prompt**

Scroll to the “Confirmation” section in the intent.

* Toggle on “Confirmation before fulfillment”
* Confirmation prompt: - If everything is correct, I will proceed with booking confirmation.
* Decline response: - okay! I will cancel the booking.

### **Step 7: Set Fulfillment**

Since we will connect to DynamoDB later via Lambda:

1. Under “Fulfillment,” select:  
   * **AWS Lambda function** (leave it empty for now)
   * Or just select “Return parameters to client” if you want to test without Lambda

5. Prepare Lambda Function (Python 3.12)

#### **Go to AWS Lambda Console**

* Click **Create function**
* Choose:
  + **Author from scratch**
  + **Function name**: BookAppointmentHandler
  + **Runtime**: Python 3.12
  + **Permissions**:
    - Create a new role with basic Lambda permissions
    - Then later manually attach:  
       AmazonDynamoDBFullAccess (or a custom policy with access to both tables)
* Add Python Code to Lambda - ClinicAppointmentLambda.py

### **Add DynamoDB Permissions to Lambda**

Go to:

1. **IAM Console** → Roles → select the role used by Lambda
2. Attach the policy AmazonDynamoDBFullAccess Or create a custom policy with limited access only to DoctorSchedule and Appointments.

3**. Connect Lambda to Lex**

1. Go back to **Lex bot**
2. Open BookingAppointmentIntent
3. Scroll to the **Fulfillment** section  
   Choose:

* **AWS Lambda function**
* Choose your BookAppointmentHandler
* Ensure Lambda permission is added (Lex will ask to grant invoke permissions)

1. Save the intent and build to test it again whether it works or not.

**4. Integrating Amazon Lex with Kommunicate Chat Widget**

This section documents how I connected **Amazon Lex** with **Kommunicate**, a customer support chat widget, and embedded it into my static website hosted on **Amazon S3**. Kommunicate serves as the front-end interface for the user to interact with the Lex bot deployed in the backend using **AWS Lambda** and **DynamoDB**.

### **Step 1: Create a Kommunicate Account**

1. Visit<https://www.kommunicate.io/>
2. Sign up for a free account (or log in if you already have one)
3. Navigate to the **Bot Integrations** section
4. Select **Amazon Lex** from the bot list

### **Step 2: Get Amazon Lex Bot Info**

Before connecting to Kommunicate, ensure:

* Lex bot is **deployed and published**
* The **region**, **bot name**, and **bot alias** are noted

### **Step 3: Connect Lex Bot in Kommunicate**

1. In Kommunicate dashboard:  
   * Go to **Bot Integrations → Amazon Lex**
   * Click **Integrate Bot**
2. Fill in:
   * **Bot Name**: your Lex bot’s name
   * **Bot Alias**: (usually live)
   * **AWS Access Key ID** and **Secret Access Key** (can create an IAM user with AmazonLexFullAccess for this)
   * **Region** where Lex bot is deployed
3. Click **Integrate -**  Kommunicate will now handle routing user messages to your Lex bot.

### **Step 4: Embed Chatbot in Static Website**

1. In the left-hand menu, go to Settings ⚙️
2. Click on Install (under Settings)
3. Under the Website tab, you’ll see a section labeled:  
    Add chat to your website
4. Copy the JavaScript embed code from Kommunicate
5. Paste this inside the <head> or at the end of your index.html file

**5. Configure static website on S3**

Step 1: Create a New S3 Bucket

1. Go to the AWS S3 Console:<https://s3.console.aws.amazon.com/s3/>
2. Click Create bucket
3. Fill in:
   * Bucket name: e.g., medicareclinic.com
   * Region: Choose your preferred region
4. Uncheck: “Block all public access” (important for website access)
5. Acknowledge the warning checkbox
6. Click Create bucket

### Step 2: Upload Website Files

1. Click into your new bucket
2. Click Upload
3. Choose index.html (website file) and click upload

### Step 3: Enable Static Website Hosting

1. In the bucket, go to the Properties tab
2. Scroll down to Static website hosting
3. Click Edit
4. Choose:
   * Enable
   * Hosting type: Host a static website
   * Index document: index.html
   * (Optional) Error document: error.html
5. Click Save changes

### Step 4: Make the Files Public

By default, S3 blocks public access, so we need to:

#### **a. Update Bucket Policy**

1. Go to the **Permissions** tab
2. Scroll to **Bucket policy**
3. Click **Edit**
4. Paste the following policy (replace your-bucket-name):

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "PublicReadGetObject",

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::clinic-chatbot-site/\*"

}

]

}

5. Click Save changes

## **Final Step: Test the Chatbot Integration**

To verify that everything works:

1. **Open your S3 static website URL** (e.g., http://clinic-chatbot-site.s3-website-us-east-1.amazonaws.com)
2. The **Kommunicate chat widget** should automatically appear in the bottom right corner of your page.
3. Click the widget and try sending a message like:  
     
    "I want to book an appointment with Dr. Sarah Jones"
4. The bot should respond using:
   * **Amazon Lex** for natural language understanding
   * **AWS Lambda** for business logic
   * **DynamoDB** for fetching/storing schedule or appointment data
5. If the chatbot completes the conversation and confirms a booking, that means everything is working

#### 

#### 