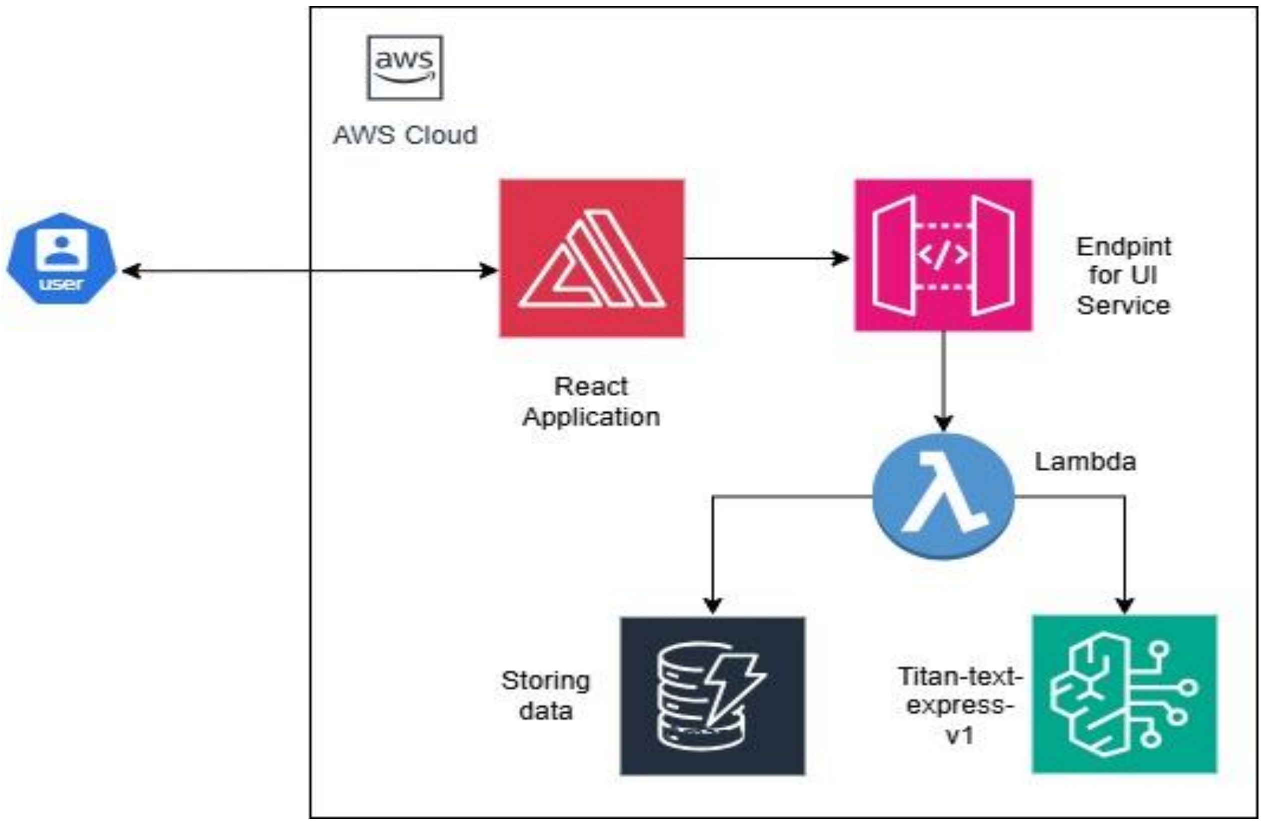
🧠 AI-Powered Text Summarizer (Serverless on AWS)

🚀 Project Overview

The AI Summarizer is a fully serverless, scalable application that generates intelligent and concise summaries of input text using Amazon Bedrock. To optimize performance and minimize redundant AI calls, results are cached using DynamoDB. The solution exposes a REST API using API Gateway, and provides an interactive React.js frontend hosted via AWS Amplify.

Git Repo: https://github.com/Utkarshlearner/aws-ai-text-summarizer



✅ Key Features

🔍 AI-Powered Summarization using Amazon Bedrock foundation models (e.g. Titan).

🛠️ Fully Serverless architecture built with AWS Lambda and API Gateway.

⚡ Low Latency Caching with DynamoDB to avoid repeated calls for the same input.

🌐 CORS-Enabled REST APIs for seamless frontend-backend communication.

💻 React.js Frontend with real-time input/output interaction.

📦 Infrastructure as Code (IaC) using AWS CloudFormation.

🏗️ Architecture Diagram

```
[React.js Frontend]
      |
      ▼
[API Gateway (REST)]
      |
      ▼
[AWS Lambda Function]
      |
      ├── Check DynamoDB for Cached Summary
      └── If Not Found → Call Amazon Bedrock (Titan.)
           |
           ▼
      Save Summary in DynamoDB → Return to Frontend
```

🛠️ Tech Stack

Service / Tech ---->	Purpose
Amazon Bedrock ---->	AI/ML summarization using foundation models
AWS Lambda ---->	Backend business logic (Python)
API Gateway ---->	Secure RESTful endpoints
Amazon DynamoDB --->	Caching layer for summarized content
React.js ---->	Frontend user interface
AWS Amplify --->	Hosting for React frontend
CloudFormation --->	IaC to provision the infrastructure

⚙️ Prerequisites

Before you begin, ensure you have the following installed:

✅ An AWS Account

✅ AWS CLI installed and configured

✅ Node.js and npm installed (for React frontend)

✅ Python 3.x installed (for Lambda backend)

✅ Basic permissions to use Amazon Bedrock, Lambda, API Gateway, and DynamoDB

🧪 How It Works

User inputs text in the React app.

The input is sent via API Gateway to a Lambda function.

Lambda checks DynamoDB for a cached summary.

If not found, it calls Amazon Bedrock for summarization.

The summary is stored in DynamoDB and returned to the user.

📁 Project Structure

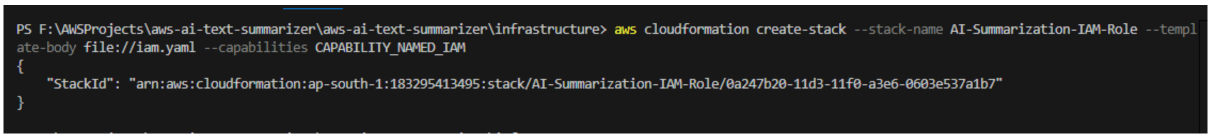
```
aws-ai-text-summarizer/
 ├── frontend/            # React.js frontend
 ├── infrastructure/      # Infrastructure as Code (IaC) templates
 |   ├── amplify.yaml       # Amplify
 |   ├── apigateway.yaml    # API Gateway configuration
 |   ├── dynamodb.yaml      # DynamoDB table definition
 |   ├── iam.yaml           # IAM roles and policies
 |   └── lambda.yaml        # Lambda function resources
 ├── deployment_steps/    # Steps for Deployment
 └── README.md            # Project overview and usage
```

🚀 Deployment Steps

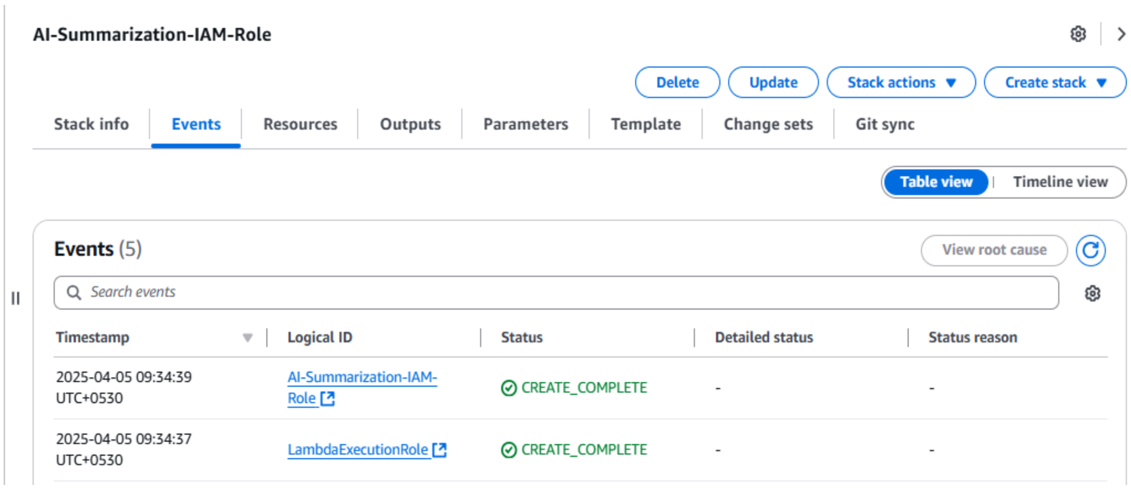
Pre-requisite: Ensure your AWS CLI is configured (aws configure) before executing the steps below.

1. **Create IAM Role**

   aws cloudformation create-stack --stack-name AI-Summarization-IAM-Role --template-body file://iam.yaml --capabilities CAPABILITY_NAMED_IAM



```
PS F:\AWSProjects\aws-ai-text-summarizer\aws-ai-text-summarizer\infrastructure> aws cloudformation create-stack --stack-name AI-Summarization-IAM-Role --templ
ate-body file://iam.yaml --capabilities CAPABILITY_NAMED_IAM
{
    "StackId": "arn:aws:cloudformation:ap-south-1:183295413495:stack/AI-Summarization-IAM-Role/0a247b20-11d3-11f0-a3e6-0603e537a1b7"
}
```



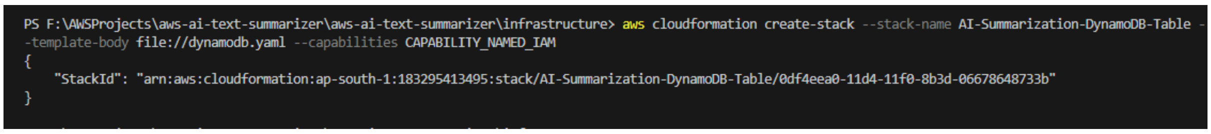
AI-Summarization-IAM-Role

Events (5)

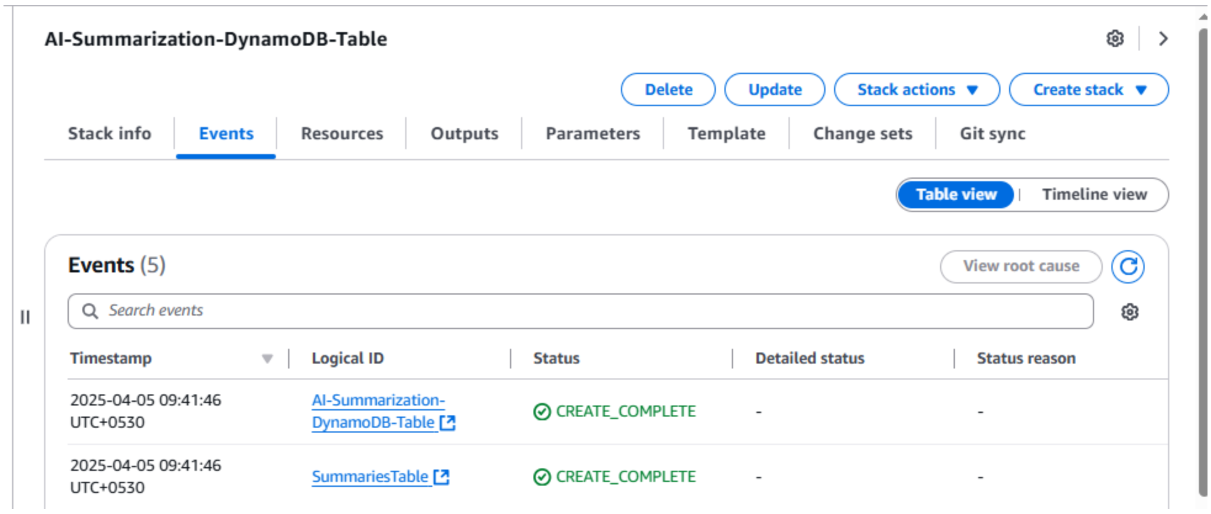
Timestamp	Logical ID	Status	Detailed status	Status reason
2025-04-05 09:34:39 UTC+0530	AI-Summarization-IAM-Role	✓ CREATE_COMPLETE	-	-
2025-04-05 09:34:37 UTC+0530	LambdaExecutionRole	✓ CREATE_COMPLETE	-	-

2. **Create DynamoDB Table**

   aws cloudformation create-stack --stack-name AI-Summarization-DynamoDB-Table --template-body file://dynamodb.yaml --capabilities CAPABILITY_NAMED_IAM



```
PS F:\AWSProjects\aws-ai-text-summarizer\aws-ai-text-summarizer\infrastructure> aws cloudformation create-stack --stack-name AI-Summarization-DynamoDB-Table -
-template-body file://dynamodb.yaml --capabilities CAPABILITY_NAMED_IAM
{
    "StackId": "arn:aws:cloudformation:ap-south-1:183295413495:stack/AI-Summarization-DynamoDB-Table/0df4eea0-11d4-11f0-8b3d-06678648733b"
}
```

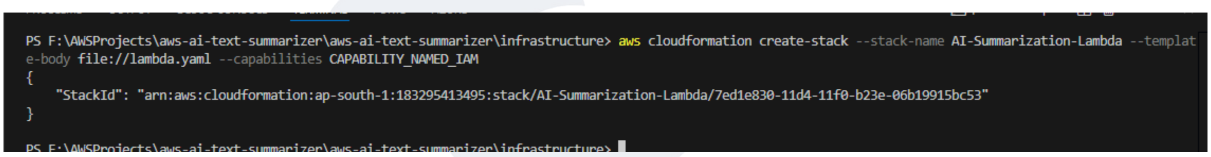


AI-Summarization-DynamoDB-Table

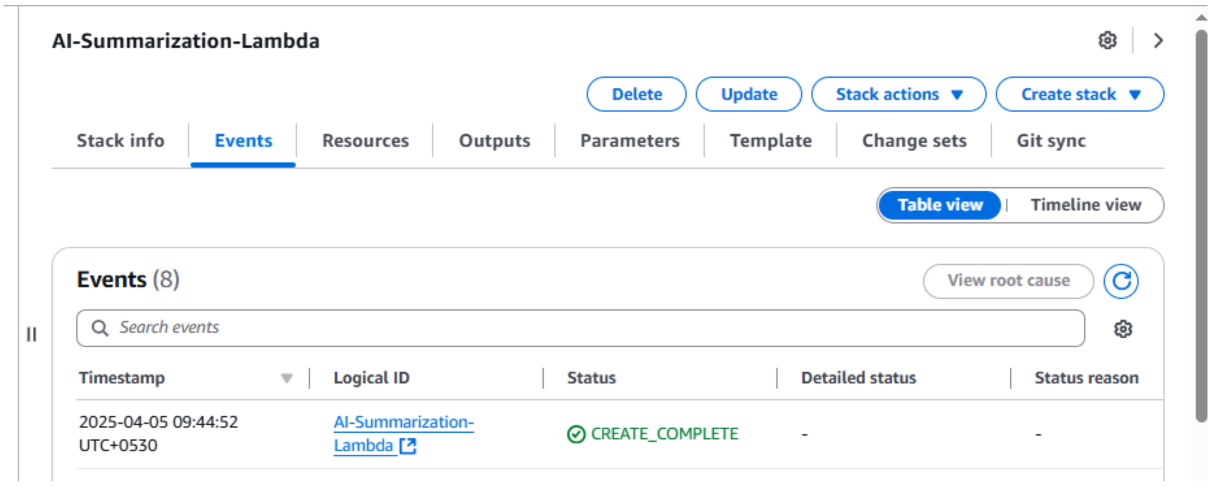
Events (5)

Timestamp	Logical ID	Status	Detailed status	Status reason
2025-04-05 09:41:46 UTC+0530	AI-Summarization-DynamoDB-Table	✓ CREATE_COMPLETE	-	-
2025-04-05 09:41:46 UTC+0530	SummariesTable	✓ CREATE_COMPLETE	-	-

3. **Create Lambda Function**
   aws cloudformation create-stack --stack-name AI-Summarization-Lambda --template-body file://lambda.yaml --capabilities CAPABILITY_NAMED_IAM



```
PS F:\AWSProjects\aws-ai-text-summarizer\aws-ai-text-summarizer\infrastructure> aws cloudformation create-stack --stack-name AI-Summarization-Lambda --templat
e-body file://lambda.yaml --capabilities CAPABILITY_NAMED_IAM
{
    "StackId": "arn:aws:cloudformation:ap-south-1:183295413495:stack/AI-Summarization-Lambda/7ed1e830-11d4-11f0-b23e-06b19915bc53"
}

PS F:\AWSProjects\aws-ai-text-summarizer\aws-ai-text-summarizer\infrastructure>
```



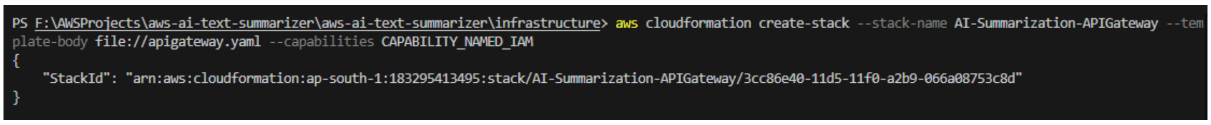
AI-Summarization-Lambda

Events (8)

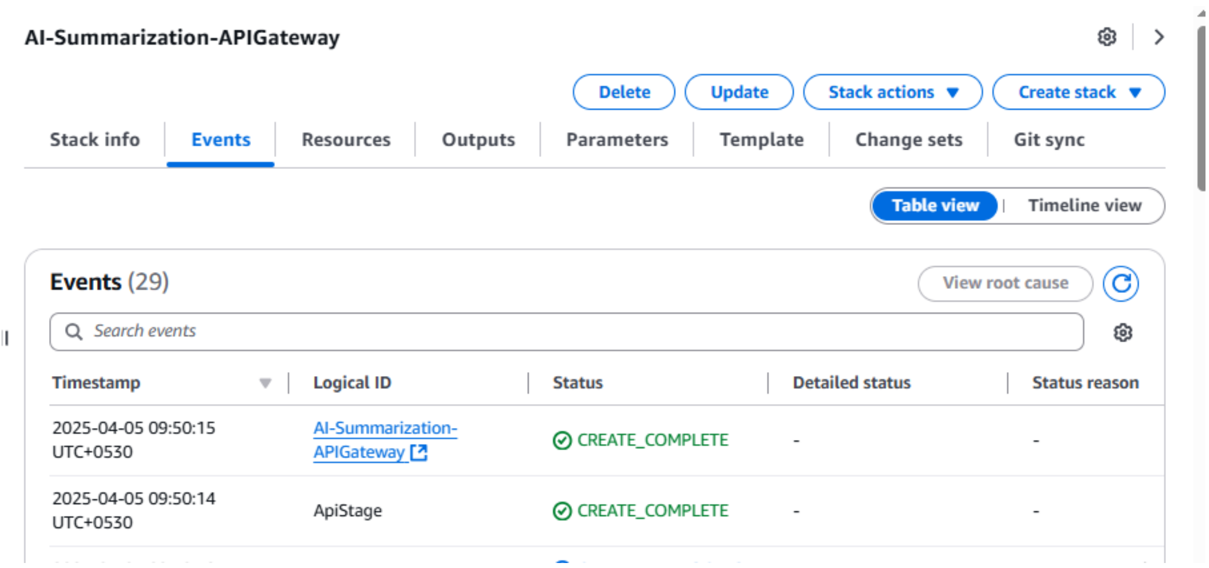
Timestamp	Logical ID	Status	Detailed status	Status reason
2025-04-05 09:44:52 UTC+0530	AI-Summarization-Lambda	✓ CREATE_COMPLETE	-	-

4. **Create API Gateway**

   aws cloudformation create-stack --stack-name AI-Summarization-APIGateway --template-body file://apigateway.yaml --capabilities CAPABILITY_NAMED_IAM



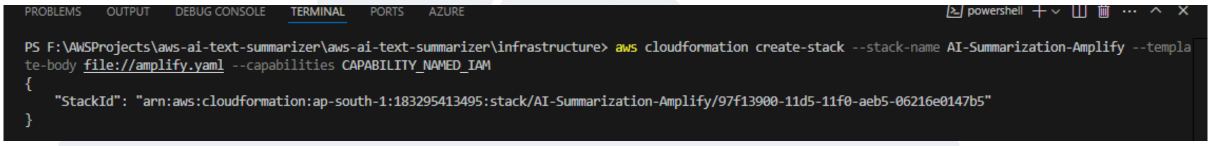
```
PS F:\AWSProjects\aws-ai-text-summarizer\aws-ai-text-summarizer\infrastructure> aws cloudformation create-stack --stack-name AI-Summarization-APIGateway --tem
plate-body file://apigateway.yaml --capabilities CAPABILITY_NAMED_IAM
{
    "StackId": "arn:aws:cloudformation:ap-south-1:183295413495:stack/AI-Summarization-APIGateway/3cc86e40-11d5-11f0-a2b9-066a08753c8d"
}
```



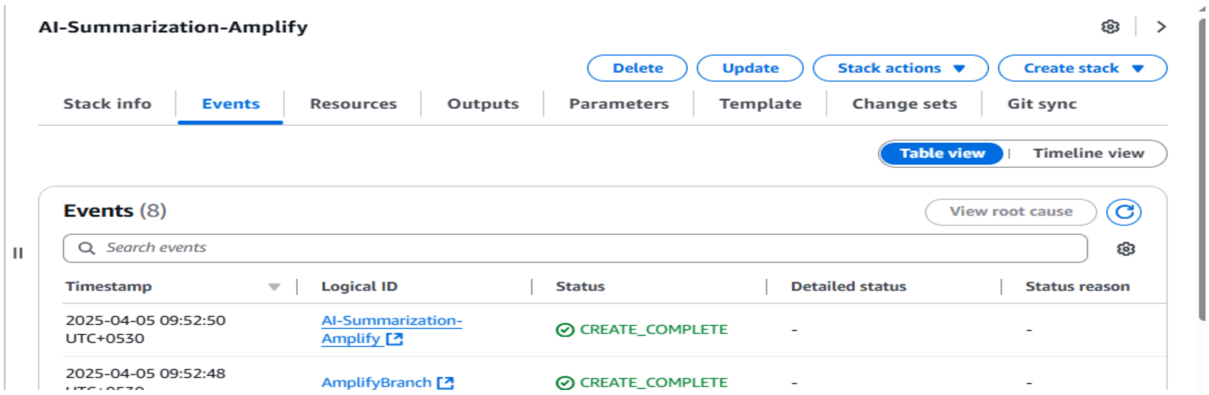
Timestamp	Logical ID	Status	Detailed status	Status reason
2025-04-05 09:50:15 UTC+0530	AI-Summarization-APIGateway	CREATE_COMPLETE	-	-
2025-04-05 09:50:14 UTC+0530	ApiStage	CREATE_COMPLETE	-	-

5. Create Amplify

aws cloudformation create-stack --stack-name AI-Summarization-Amplify --template-body file://amplify.yaml --capabilities CAPABILITY_NAMED_IAM



```
PS F:\AWSProjects\aws-ai-text-summarizer\aws-ai-text-summarizer\infrastructure> aws cloudformation create-stack --stack-name AI-Summarization-Amplify --templa
te-body file://amplify.yaml --capabilities CAPABILITY_NAMED_IAM
{
    "StackId": "arn:aws:cloudformation:ap-south-1:183295413495:stack/AI-Summarization-Amplify/97f13900-11d5-11f0-aeb5-06216e0147b5"
}
```



Timestamp	Logical ID	Status	Detailed status	Status reason
2025-04-05 09:52:50 UTC+0530	AI-Summarization-Amplify	CREATE_COMPLETE	-	-
2025-04-05 09:52:48	AmplifyBranch	CREATE_COMPLETE	-	-

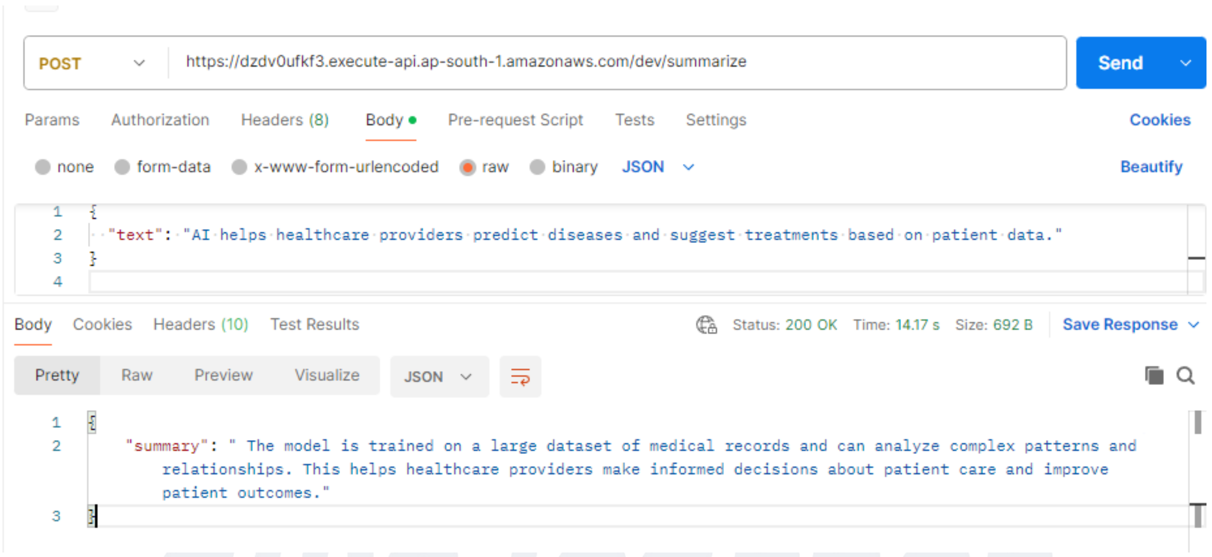
✅ Testing Your AI Summarizer API via Postman

After deploying your API Gateway stack, get your **API endpoint URL** from:

- CloudFormation (**AI-Summarization-APIGateway**) Outputs
- Or from API Gateway (**AI-Summarizer-API**) Console → Your API → Stages → Invoke URL

Example:

https://xyz456.execute-api.ap-south-1.amazonaws.com/prod/summarize

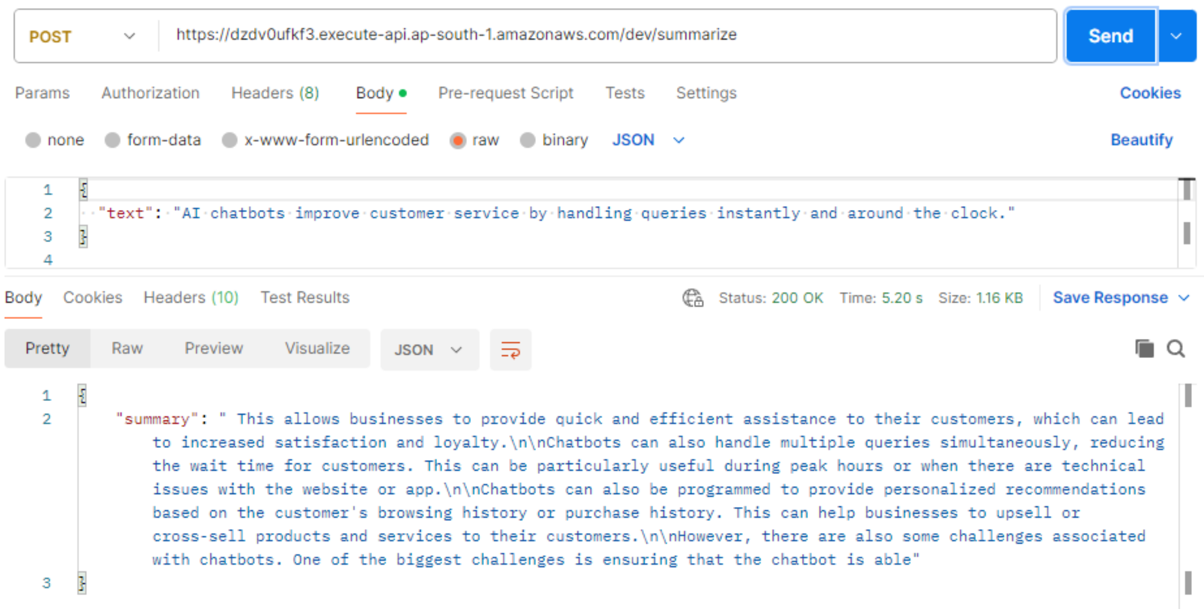


https://dzdv0ufkf3.execute-api.ap-south-1.amazonaws.com/dev/summarize

```
{
  "text": "AI helps healthcare providers predict diseases and suggest treatments based on patient data."
}
```

Status: 200 OK Time: 14.17 s Size: 692 B

```
{
    "summary": " The model is trained on a large dataset of medical records and can analyze complex patterns and
        relationships. This helps healthcare providers make informed decisions about patient care and improve
        patient outcomes."
}
```



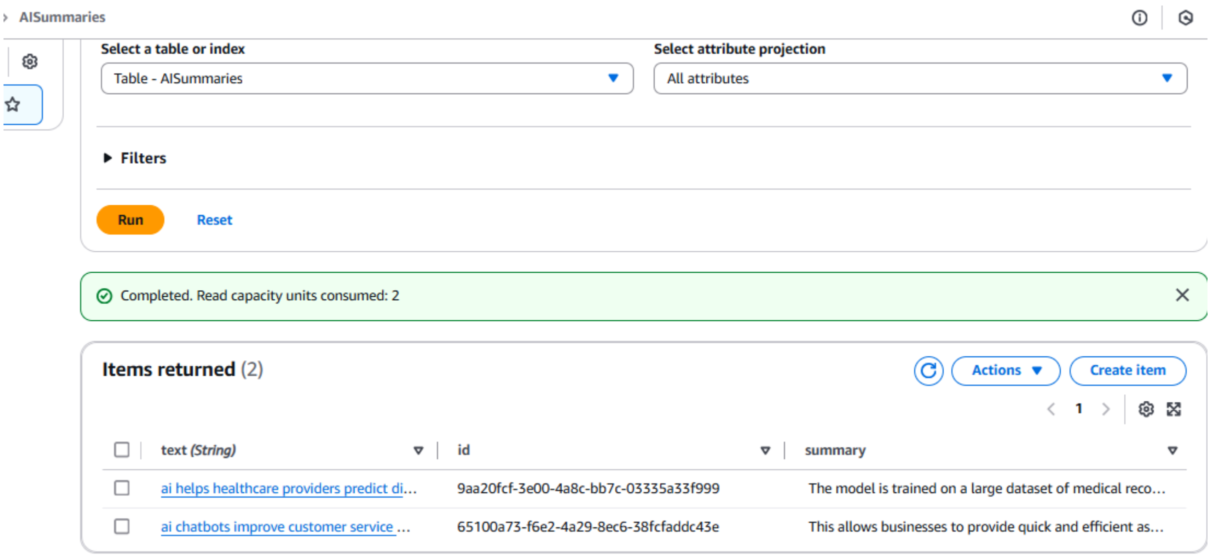
https://dzdv0ufkf3.execute-api.ap-south-1.amazonaws.com/dev/summarize

```
{
  "text": "AI chatbots improve customer service by handling queries instantly and around the clock."
}
```

Status: 200 OK Time: 5.20 s Size: 1.16 KB

```
{
    "summary": " This allows businesses to provide quick and efficient assistance to their customers, which can lead
        to increased satisfaction and loyalty.\n\nChatbots can also handle multiple queries simultaneously, reducing
        the wait time for customers. This can be particularly useful during peak hours or when there are technical
        issues with the website or app.\n\nChatbots can also be programmed to provide personalized recommendations
        based on the customer's browsing history or purchase history. This can help businesses to upsell or
        cross-sell products and services to their customers.\n\nHowever, there are also some challenges associated
        with chatbots. One of the biggest challenges is ensuring that the chatbot is able"
}
```

DynamoDB:



AISummaries

Select a table or index: Table - AISummaries

Select attribute projection: All attributes

Filters

Run Reset

Completed. Read capacity units consumed: 2

Items returned (2)

text (String)	id	summary
ai helps healthcare providers predict di…	9aa20fcf-3e00-4a8c-bb7c-03335a33f999	The model is trained on a large dataset of medical reco…
ai chatbots improve customer service …	65100a73-f6e2-4a29-8ec6-38fcfaddc43e	This allows businesses to provide quick and efficient as…

✅ Building Frontend Application

1) npm install

```
F:\AWSProjects\aws-ai-text-summarizer\aws-ai-text-summarizer\frontend>npm install
npm warn ERESOLVE overriding peer dependency
```

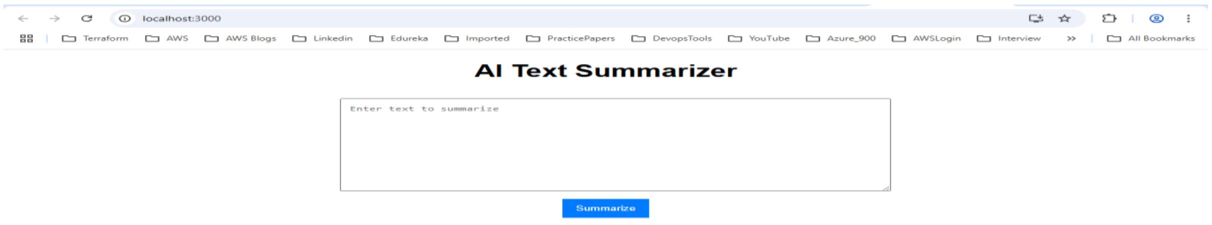
2) Update API Endpoint in App.tsx

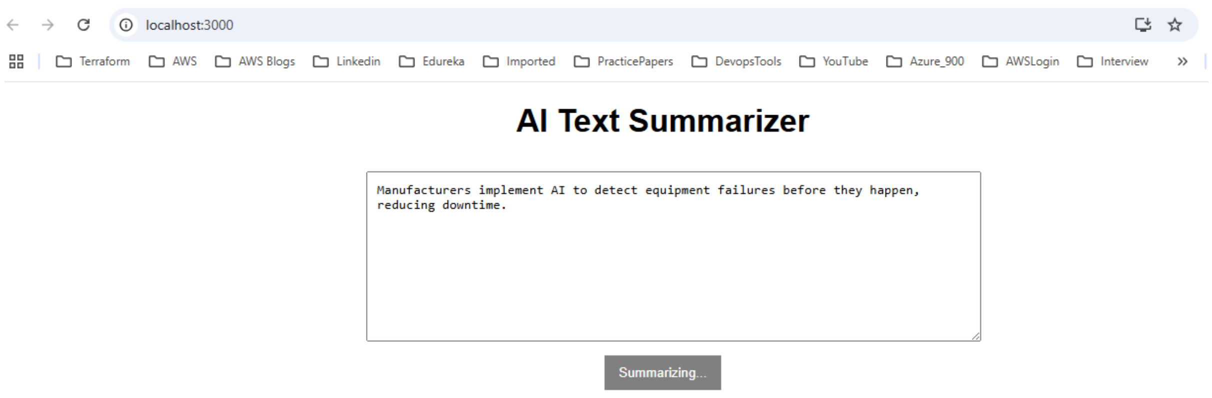
Path → frontend → src → App.tsx

Example:

const API_URL = " https://xyz456.execute-api.ap-south-1.amazonaws.com/prod/summarize"

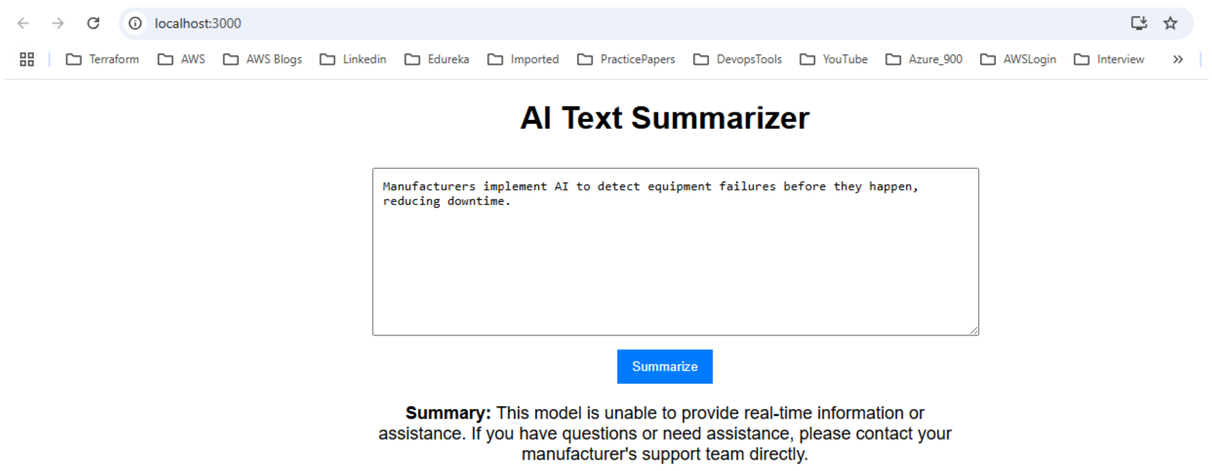
3) Starting React Server in local (npm start)


localhost:3000
Terraform AWS AWS Blogs Linkedin Edureka Imported PracticePapers DevopsTools YouTube Azure_900 AWSLogin Interview All Bookmarks
AI Text Summarizer
Enter text to summarize
Summarize



AI Text Summarizer

Manufacturers implement AI to detect equipment failures before they happen, reducing downtime.



AI Text Summarizer

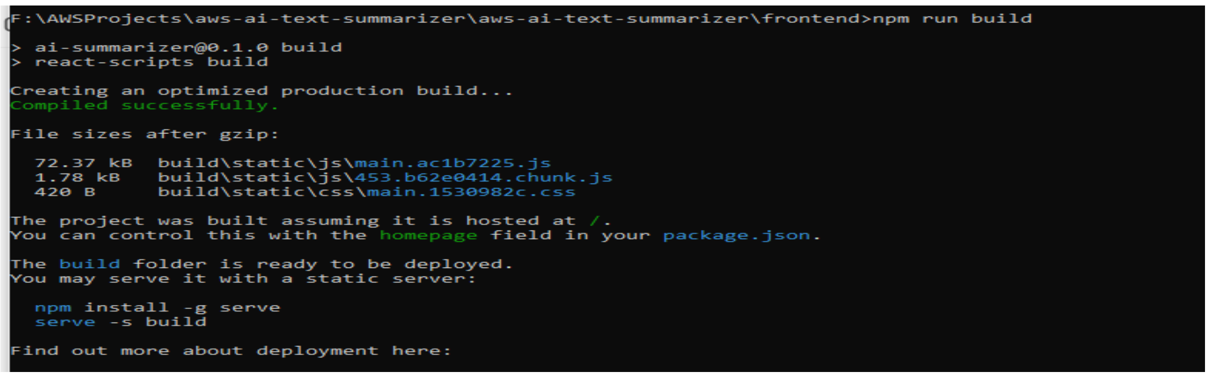
Manufacturers implement AI to detect equipment failures before they happen, reducing downtime.

Summary: This model is unable to provide real-time information or assistance. If you have questions or need assistance, please contact your manufacturer's support team directly.

Exactly! 🎯 Once your app works locally (including API integration), you can build and deploy it to AWS Amplify. Here is how to go from local to live:

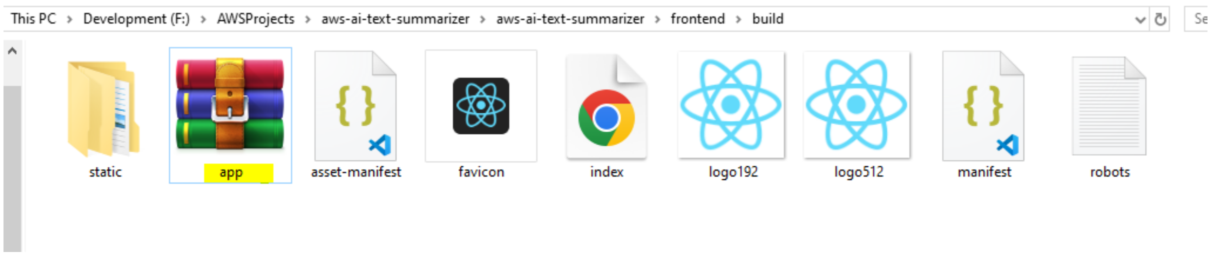
🚀 Deploying to Amplify from Local

1) Build the React App (npm run build)



```
F:\AWSProjects\aws-ai-text-summarizer\aws-ai-text-summarizer\frontend>npm run build

> ai-summarizer@0.1.0 build
> react-scripts build

Creating an optimized production build...
Compiled successfully.

File sizes after gzip:

  72.37 kB  build\static\js\main.ac1b7225.js
  1.78 kB   build\static\js\453.b62e0414.chunk.js
  420 B     build\static\css\main.1530982c.css

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  npm install -g serve
  serve -s build

Find out more about deployment here:
```

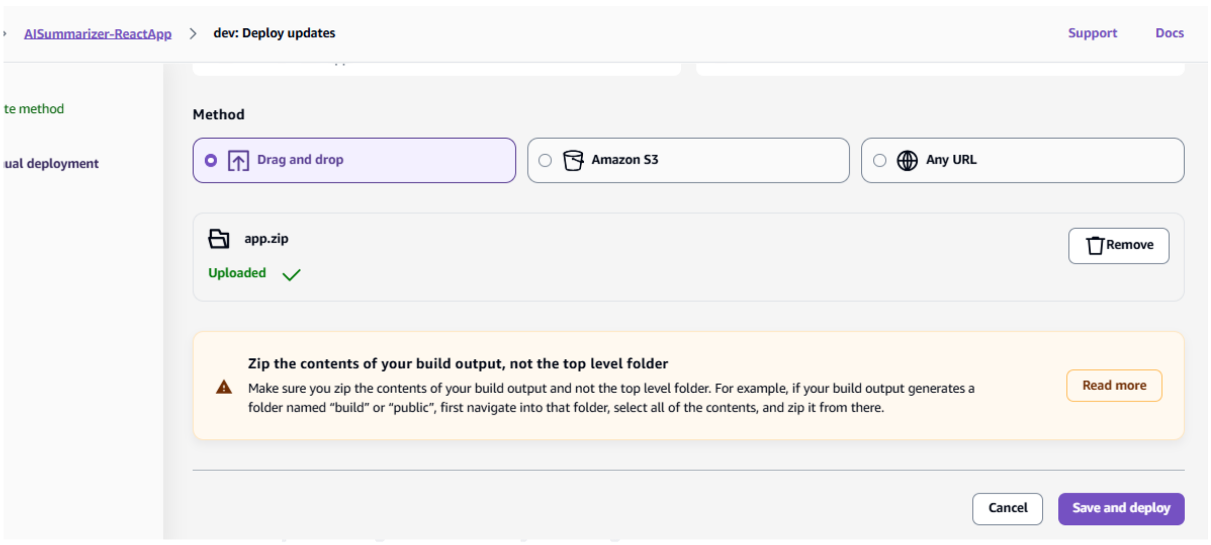
2) Zip the Contents of build



This PC > Development (F:) > AWSProjects > aws-ai-text-summarizer > aws-ai-text-summarizer > frontend > build
static
app
asset-manifest
favicon
index
logo192
logo512
manifest
robots

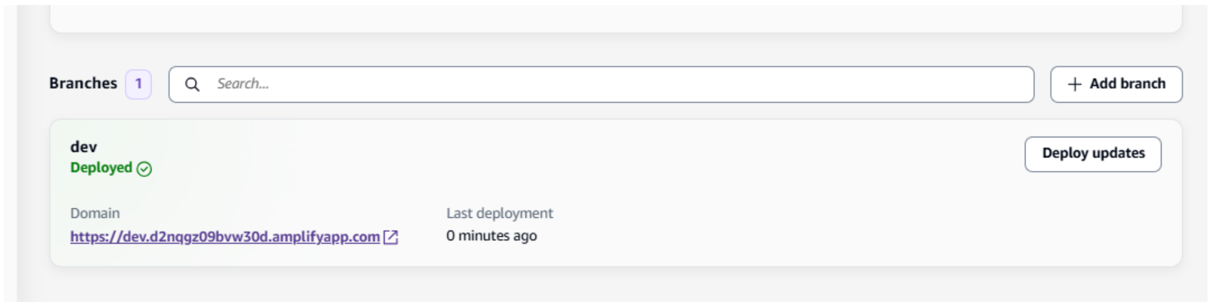
3) Deploy via Amplify Console (Drag & Drop)

Go to the AWS Console → Amplify → Your App
Click on **"Hosting environments"**.
Look for the **“Deploy Updates”** section.



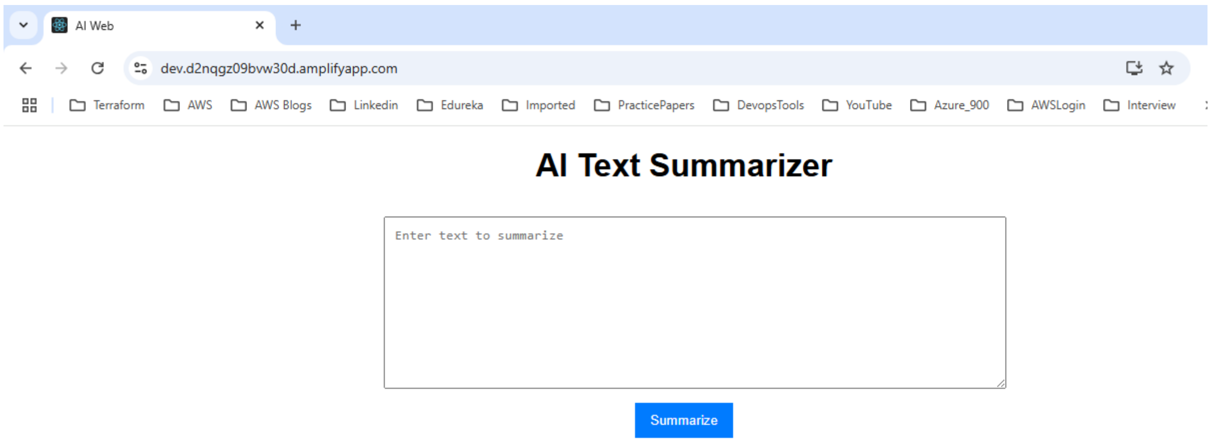
Zip the contents of your build output, not the top level folder
Make sure you zip the contents of your build output and not the top level folder. For example, if your build output generates a folder named “build” or “public”, first navigate into that folder, select all of the contents, and zip it from there.

Once Deployed AWS Amplify will give public URL:



Domain
https://dev.d2nqgz09bvw30d.amplifyapp.com

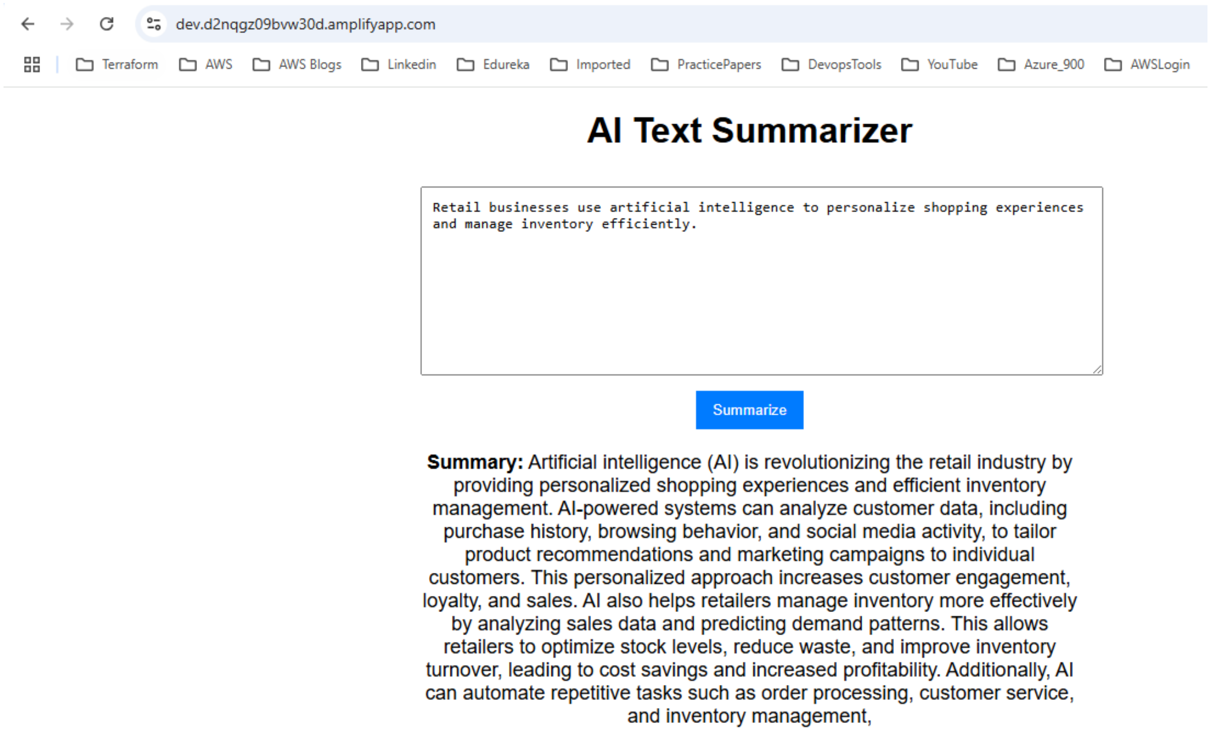
Last deployment
0 minutes ago


AI Web

dev.d2nqgz09bvw30d.amplifyapp.com

AI Text Summarizer

Enter text to summarize


dev.d2nqgz09bvw30d.amplifyapp.com

AI Text Summarizer

```
Retail businesses use artificial intelligence to personalize shopping experiences
and manage inventory efficiently.
```

Summary: Artificial intelligence (AI) is revolutionizing the retail industry by providing personalized shopping experiences and efficient inventory management. AI-powered systems can analyze customer data, including purchase history, browsing behavior, and social media activity, to tailor product recommendations and marketing campaigns to individual customers. This personalized approach increases customer engagement, loyalty, and sales. AI also helps retailers manage inventory more effectively by analyzing sales data and predicting demand patterns. This allows retailers to optimize stock levels, reduce waste, and improve inventory turnover, leading to cost savings and increased profitability. Additionally, AI can automate repetitive tasks such as order processing, customer service, and inventory management,

Clean-Up Steps to Delete Stacks & Resources

Delete CloudFormation Stacks (Delete One by One)

aws cloudformation delete-stack --stack-name AI-Summarization-Amplify

aws cloudformation delete-stack --stack-name AI-Summarization-APIGateway

aws cloudformation delete-stack --stack-name AI-Summarization-Lambda

aws cloudformation delete-stack --stack-name AI-Summarization-DynamoDB-Table

aws cloudformation delete-stack --stack-name AI-Summarization-IAM-Role

Check for Leftover Resources

Manually check (and delete if needed):

- Lambda Functions
- Amplify
- API Gateway
- DynamoDB Tables
- IAM Roles/Policies
- CloudWatch Log Groups

Final Check: Billing Dashboard

Go to the **AWS Billing Console** → **Cost Explorer** or **Free Tier Usage Alerts** to ensure no surprise charges.

✍️ Author

Utkarsh Rastogi

LinkedIn → https://www.linkedin.com/in/rastogiutkarsh

Blogs → https://awslearner.hashnode.dev