AWS Lambda Functions for LangGraph Integration

Overview

You will create 4 AWS Lambda functions:

1. `get_ticket_details`
2. `get_failure_details`
3. `fix_failure_steps`
4. `search_failure_bedrock` (invokes Mistral via Bedrock)

Pre-requisites

- AWS account with permissions to create Lambda functions
- IAM role with permissions for Lambda, CloudWatch Logs, and Bedrock (for #4)
- Python 3.11 runtime
- Region: `us-east-1` (recommended for Bedrock support)

Step-by-Step Instructions

Step 1: Create IAM Role for Lambda

1. Go to **IAM > Roles > Create Role**
2. Select **AWS Service > Lambda**
3. Attach the following permissions:
    - `AWSLambdaBasicExecutionRole`
    - `AmazonBedrockFullAccess` (for `search_failure_bedrock`)
4. Name it: `lambda-genai-execution-role`

Step 2: Create Lambda Function – `get_ticket_details`

1. Go to **Lambda > Create function**
2. Name: `get_ticket_details`
3. Runtime: Python 3.11
4. Role: `lambda-genai-execution-role`
5. Paste code:

```
def lambda_handler(event, context):
    ticket_id = event.get("ticket_id", "TICKET001")
    return {
        "ticket_id": ticket_id,
```

```
        "description": "Application crash observed in production",
        "failure_code": "FAIL-503"
    }
```

6. Deploy and test with:

```
{
  "ticket_id": "TICKET98765"
}
```

Step 3: Create Lambda Function – `get_failure_details`

1. Create new Lambda: `get_failure_details`
2. Paste code:

```
def lambda_handler(event, context):
    failure_code = event.get("failure_code", "FAIL-503")
    return {
        "failure_code": failure_code,
        "category": "Application Failure",
        "message": "503 Service Unavailable due to backend overload"
    }
```

3. Test with:

```
{
  "failure_code": "FAIL-503"
}
```

Step 4: Create Lambda Function – `fix_failure_steps`

1. Create new Lambda: `fix_failure_steps`
2. Paste code:

```
def lambda_handler(event, context):
    failure_code = event.get("failure_code", "FAIL-503")
    return {
        "failure_code": failure_code,
        "steps": [
            "Restart backend services",
            "Verify load balancer configuration",
            "Check database availability"
```

```
        ]
    }
```

3. Test with:

```
{
  "failure_code": "FAIL-503"
}
```

Step 5: Create Lambda Function – `search_failure_bedrock`

1. Create new Lambda: `search_failure_bedrock`
2. Paste code:

```
import boto3
import json
from botocore.exceptions import ClientError

def lambda_handler(event, context):
    # Input will be JSON with failure detail
    failure_info = event.get("failure_details", {})
    prompt = f"Explain the failure in detail: {json.dumps(failure_info)}"

    client = boto3.client("bedrock-runtime", region_name="us-east-1")
    model_id = "mistral.mistral-large-2402-v1:0"

    body = {
        "prompt": f"<s>[INST] {prompt} [/INST]",
        "max_tokens": 500,
        "temperature": 0.5
    }

    try:
        response = client.invoke_model(
            modelId=model_id,
            body=json.dumps(body),
            contentType="application/json",
            accept="application/json"
        )
        result = json.loads(response['body'].read())
        return {
            "statusCode": 200,
            "explanation": result['outputs'][0]['text']
        }
    except ClientError as e:
        return {
            "statusCode": 500,
```

```
            "error": str(e)
        }
```

3. Test with:

```
{
  "failure_details": {
    "failure_code": "FAIL-503",
    "message": "503 Service Unavailable"
  }
}
```

Test All 4 Functions Individually

You can test each Lambda from the console by using the sample payloads above.
