```
import json
import boto3
import datetime
import os

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('my_url_shortner')  # Replace with your DynamoDB table name
domain = os.environ['domain']

def base62_encode(number):
    characters = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
    base62_encoded = ""
    while number > 0:
        remainder = number % 62
        base62_encoded = characters[remainder] + base62_encoded
        number //= 62
    return base62_encoded or "0"

def lambda_handler(event, context):
    try:
        print(event)
        # Check if the 'body' attribute exists in the event object
        if 'body' in event:
            data = json.loads(event['body'])
        else:
            # If 'body' attribute doesn't exist, assume the event object itself is the request data
            data = event

        long_url = data.get('longUrl', '')  # Get 'longUrl' from the request data

        # Calculate expiration timestamp (24 hours from now)
        url_created_time=datetime.datetime.now()

        print(url_created_time)

        expiration_timestamp = int((datetime.datetime.now() +
datetime.timedelta(hours=24)).timestamp())

        print(expiration_timestamp , "from the value")


        # Generate a unique shortened key using custom base62 encoding

        original_number = int(datetime.datetime.now().timestamp())  # You can use any unique number
as the original number

        shortened_key = base62_encode(original_number)

        print(shortened_key)

        # Construct the shortened URL with API Gateway endpoint

        shortened_url = f'{domain}{shortened_key}'


        # Store mapping in DynamoDB

        table.put_item(

            Item={

                'shortkey' : shortened_key,

                'shortURL': shortened_url,

                'longURL': long_url,

                'ExpirationDate': expiration_timestamp

                #'created_at':url_created_time

            }

        )


        # Return shortened URL and expiration timestamp

        response = {

            'statusCode': 200,

            'body': json.dumps({'shortenedUrl': shortened_url, 'expirationTimestamp':
expiration_timestamp})

        }

        print(response)
```

```
    return response
except Exception as e:
    # Handle any exceptions and return an error response as a JSON-formatted string
    error_response = {
        'statusCode': 500,
        'body': json.dumps({'error': str(e)})
    }
    return error_response
```