Serverless-CRM Web Application Using AWS

Introduction:

CRM systems are essential for organizations to manage customer data, track interactions, and improve relationships. Traditional CRM applications require continuous server management and incur high infrastructure costs. To overcome these challenges, this project implements a **serverless CRM system** using AWS cloud services. The application eliminates the need for managing physical or virtual servers and leverages AWS-managed services for seamless scalability and performance.

Objective:

The main objectives of this project are:

- To design and develop a serverless CRM web application using AWS services.
- To implement user authentication (Login and Registration).
- To store and retrieve customer data securely using DynamoDB.
- To utilize AWS Lambda and API Gateway for backend logic and API communication.
- To deploy and host the frontend on Amazon S3 for high availability and scalability.

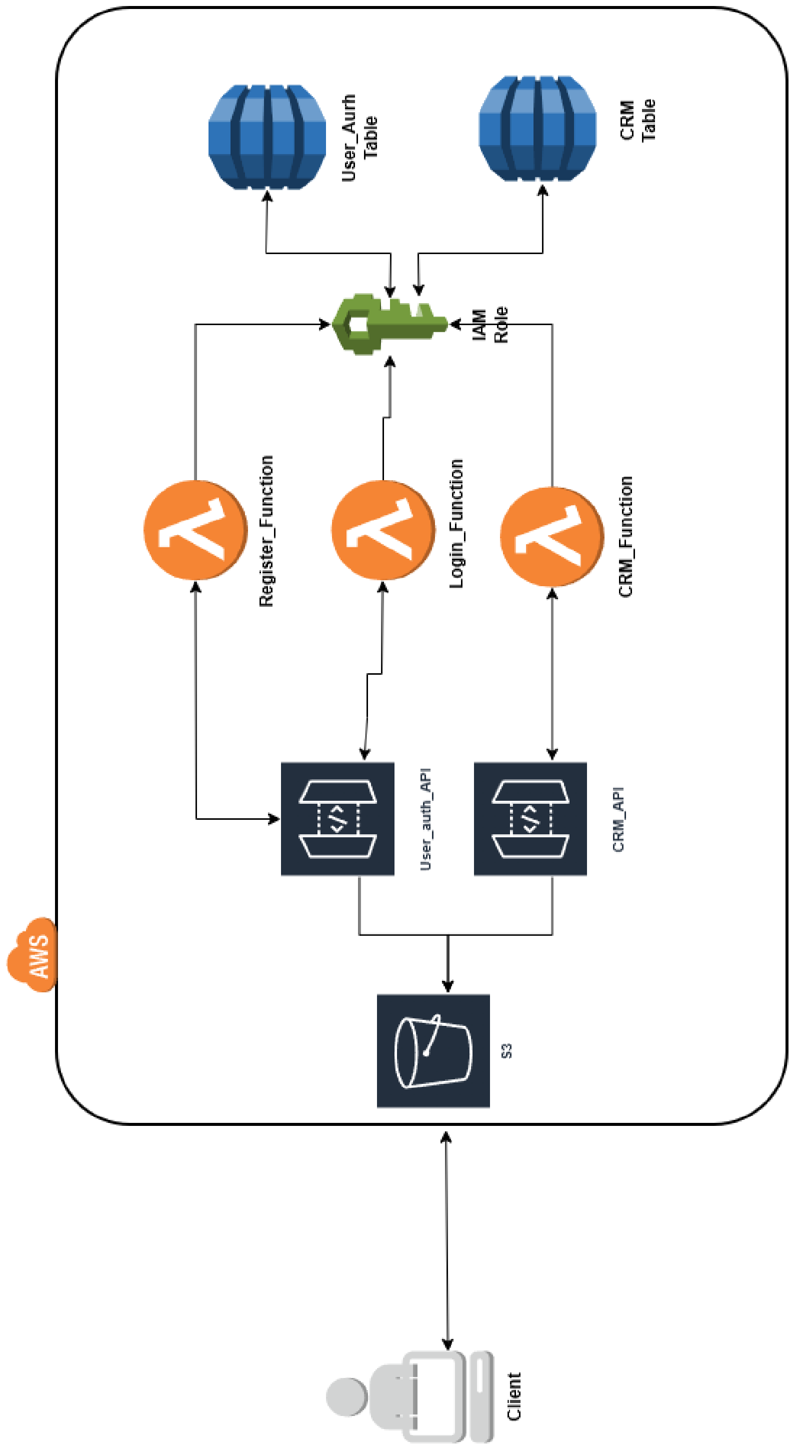
System Architecture:

The architecture of the CRM application is built entirely on AWS serverless components.

Key Components:

1. **Amazon S3 (Simple Storage Service):**
   Hosts the static frontend of the CRM application (HTML, CSS, JavaScript). It serves the web interface to users through a public endpoint.
2. **Amazon API Gateway:**
   Acts as the main entry point for client requests. It exposes RESTful APIs that communicate with AWS Lambda functions.
3. **AWS Lambda:**
   Executes backend logic such as user registration, login validation, and CRUD operations on customer data.
   Lambda functions are stateless, event-driven, and automatically scale based on demand.
4. **Amazon DynamoDB:**
   A NoSQL database that stores user information, customer details, and other CRM data. It supports fast read/write performance with automatic scaling.
5. **AWS IAM (Identity and Access Management):**
   Controls access between services by defining roles and policies for Lambda, S3, and DynamoDB.

System Architecture Diagram:



Modules Description:

1. User Authentication Module

- **Registration Page:**
  Allows new users to create an account by submitting their email and password. The data is validated and securely stored in DynamoDB. A Lambda function handles this process.
- **Login Page:**
  Authenticates users by validating credentials from DynamoDB. If successful, the user is granted access to the dashboard.

2. Dashboard Module

- Displays user-specific data, such as customer records or activity logs.
- Allows adding, updating, and viewing CRM-related information.

3. Backend Module (AWS Lambda)

- Each Lambda function corresponds to a specific API (e.g., `/register`, `/login`, `/addCustomer`).
- Returns JSON responses to the frontend through API Gateway.

4. API Gateway Module

- Defines and exposes REST APIs for frontend communication.
- Provides secure endpoints for user authentication and data handling.

5. Database Module (DynamoDB)

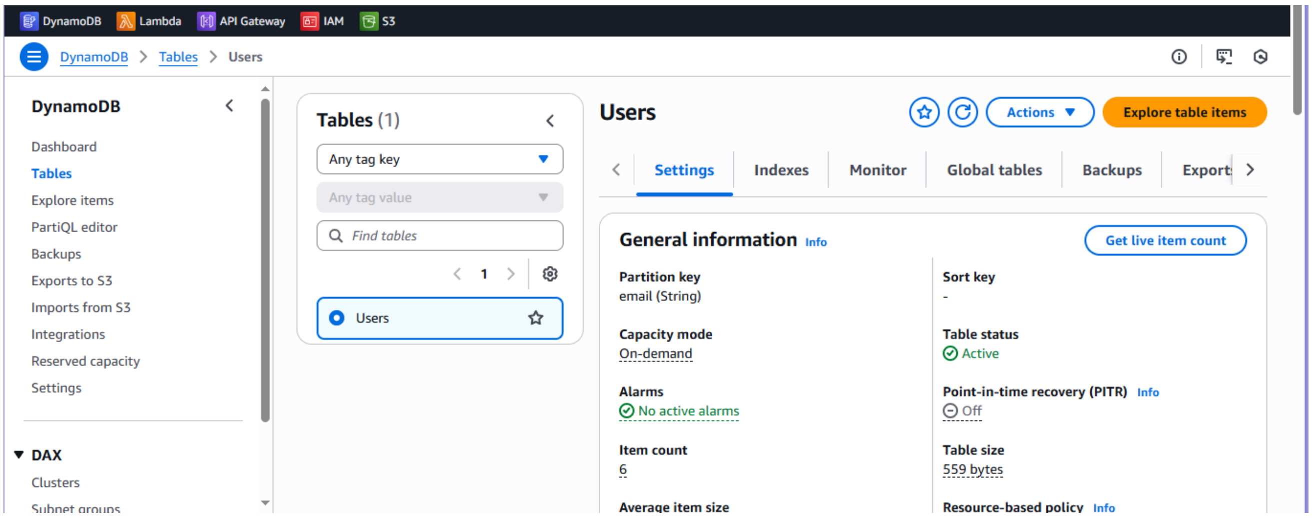
- Stores all user and CRM-related data in tables.
- Uses primary keys and indexes for fast data retrieval.
- Automatically scales according to the number of requests.

Steps Involved in Solving the Project Problem Statement:

The project aimed to build a Cloud-Based Serverless- CRM system with a serverless architecture to manage customer interactions efficiently. The following steps were taken to implement the solution:

Step1: Set Up the DynamoDB Table:

- Created a DynamoDB table named CustomerTable with customerID as the partition key. This Table Used to store user customer details.
- Created a DynamoDB table named Users with email as the partition key. This Table Used for User Authentication and Store user Email and Password.
- Configured on-demand capacity to handle dynamic workloads .



Partition key
email (String)

Sort key
-

Capacity mode
On-demand

Table status
Active

Alarms
No active alarms

Point-in-time recovery (PITR) Info
Off

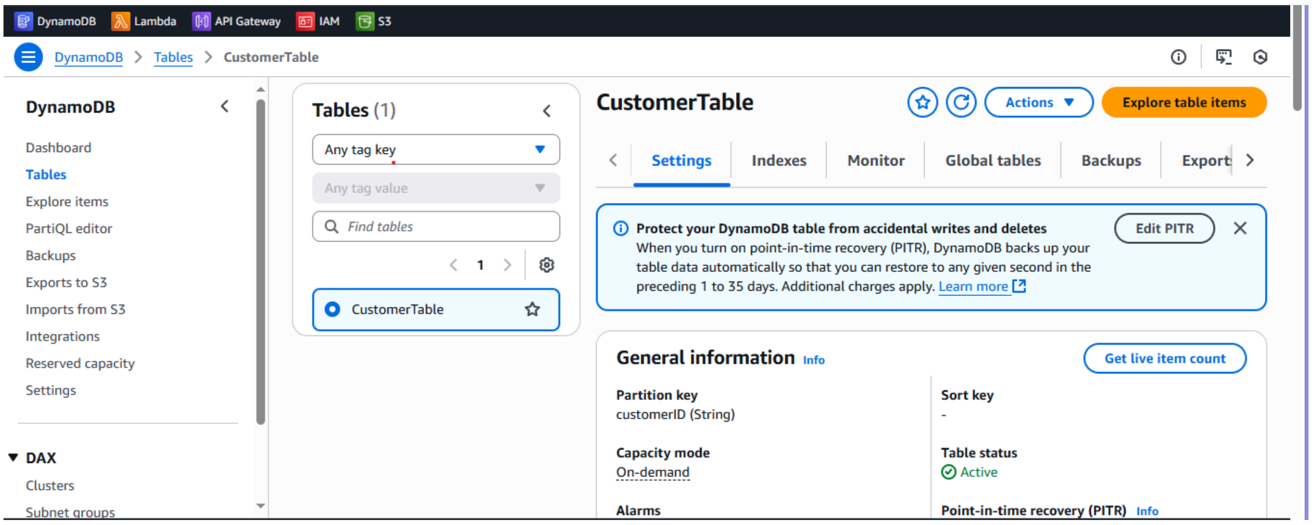
Item count
6

Table size
559 bytes

Average item size

Resource-based policy Info

Users Table for User Authentication



Protect your DynamoDB table from accidental writes and deletes
When you turn on point-in-time recovery (PITR), DynamoDB backs up your table data automatically so that you can restore to any given second in the preceding 1 to 35 days. Additional charges apply. Learn more

Partition key
customerID (String)

Sort key
-

Capacity mode
On-demand

Table status
Active

Alarms

Point-in-time recovery (PITR) Info

Customer Table for Store Customer Data

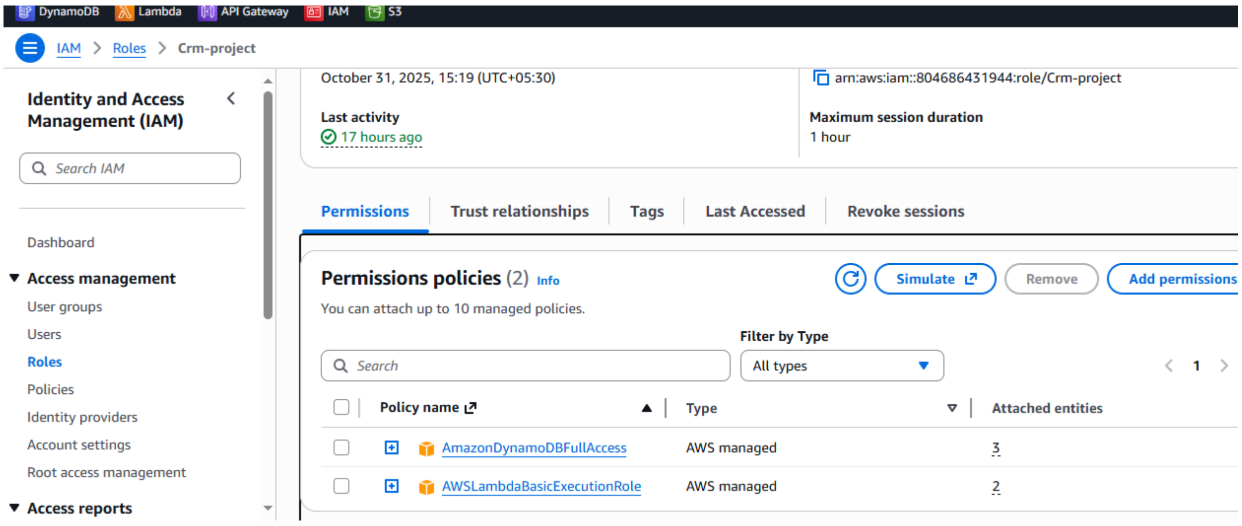
Step 2: Create IAM role for Lambda Function:

Created an **IAM role** for Lambda with permissions to:

- Access DynamoDB tables.
- Write logs to CloudWatch for monitoring.

Policies:

- **AmazonDynamoDBFullAccess**
- **AWSlambdaBasicExecutionRole**



October 31, 2025, 15:19 (UTC+05:30)

arn:aws:iam::804686431944:role/Crm-project

Last activity: 17 hours ago

Maximum session duration: 1 hour

Permissions policies (2) Info

You can attach up to 10 managed policies.

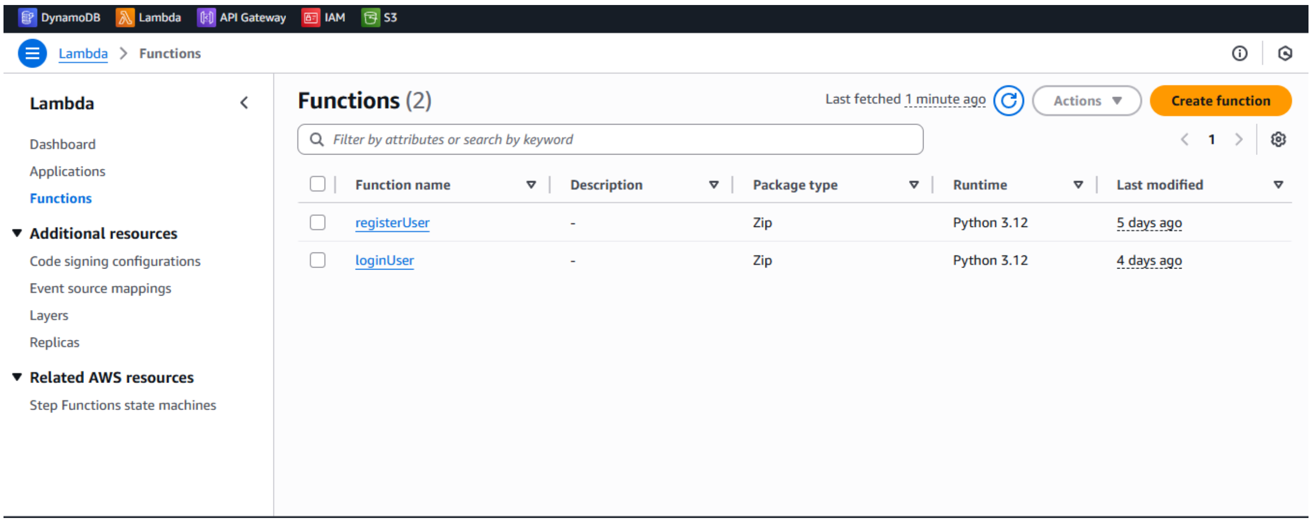
Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	3
AWSLambdaBasicExecutionRole	AWS managed	2

IAM role for Lambda

Step 3: Create a Lambda Function:

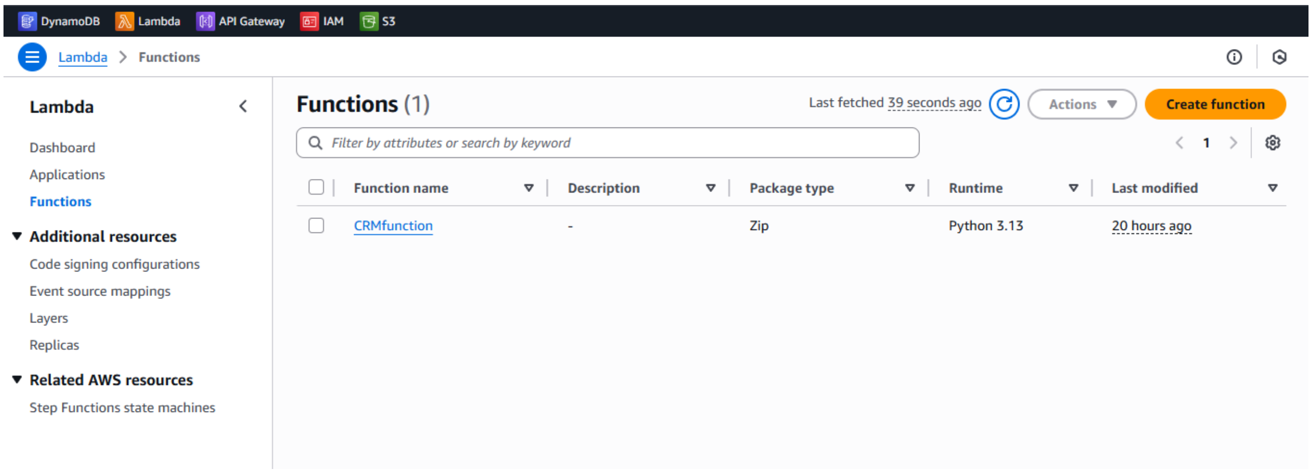
Created **Lambda functions** in Python to handle:

- **User Registration** – Inserts user data into DynamoDB.
- **User Login** – Verifies credentials from DynamoDB.
- **Customer Data Management** – Adds, updates, or retrieves CRM data.



Function name	Description	Package type	Runtime	Last modified
registerUser	-	Zip	Python 3.12	5 days ago
loginUser	-	Zip	Python 3.12	4 days ago

User Registration & Login Functions



Function name	Description	Package type	Runtime	Last modified
CRMfunction	-	Zip	Python 3.13	20 hours ago

CRM Function for store Customer Data

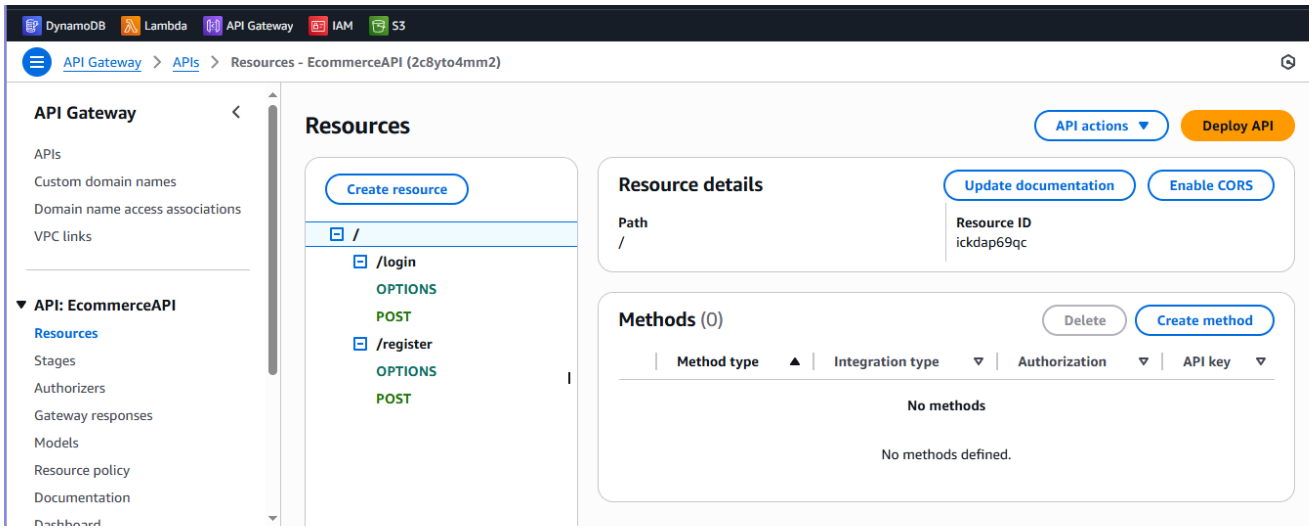
Step 4: API Creation With AWS API Gateway :

Created a **REST API** in API Gateway.

- Defined resources and methods (POST/GET) for each function:

➢ `/register` → LambdaRegisterFunction
➢ `/login` → LambdaLoginFunction
➢ `/add-customer`→ LambdaCRMFunction
➢ `/delete-customer`
➢ `/get-all-customer`
➢ `/get-customer`
➢ `/update-customer`

- Enabled **CORS (Cross-Origin Resource Sharing)** to allow the frontend hosted on S3 to access the API.
- Deployed the API to a **stage (e.g., “prod”)** and obtained the **Invoke URL** for integration.


DynamoDB Lambda API Gateway IAM S3

API Gateway > APIs > Resources - EcommerceAPI (2c8yto4mm2)

Resources

/
/login
OPTIONS
POST
/register
OPTIONS
POST

Resource details
Path: /
Resource ID: ickdap69qc

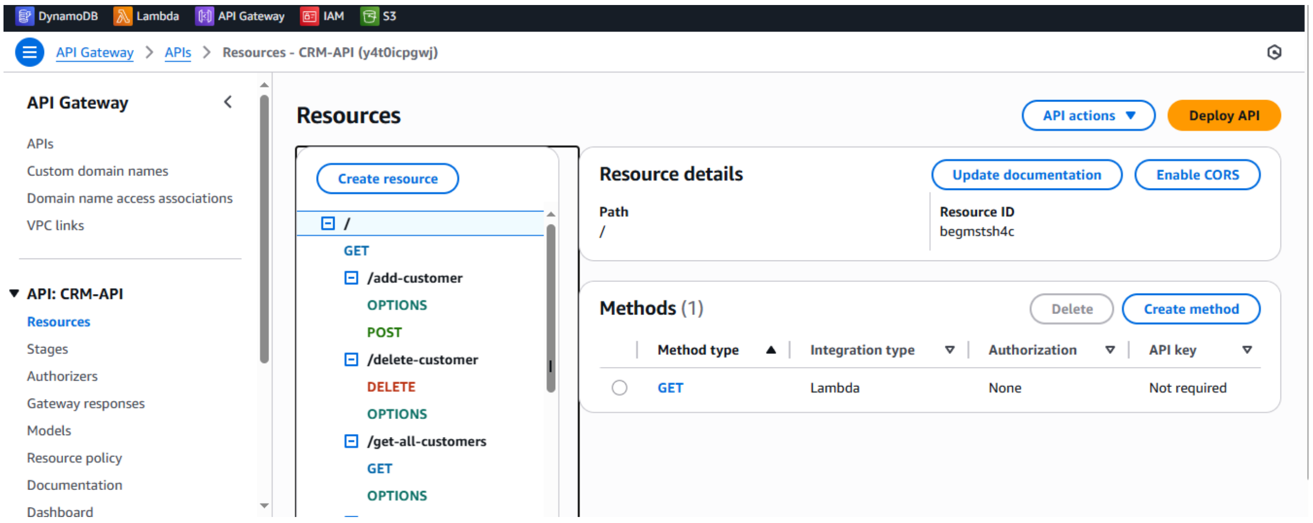
Methods (0)

Method type	Integration type	Authorization	API key

No methods

No methods defined.

API for Register & Login


DynamoDB Lambda API Gateway IAM S3

API Gateway > APIs > Resources - CRM-API (y4t0icpgwj)

Resources

/
GET
/add-customer
OPTIONS
POST
/delete-customer
DELETE
OPTIONS
/get-all-customers
GET
OPTIONS

Resource details
Path: /
Resource ID: begmstsh4c

Methods (1)

Method type	Integration type	Authorization	API key
GET	Lambda	None	Not required

API for CRM Application

Step 5: Frontend Devlopment :

Devloped the web interface using **HTML, CSS, and JavaScript**.

Designed the following pages:

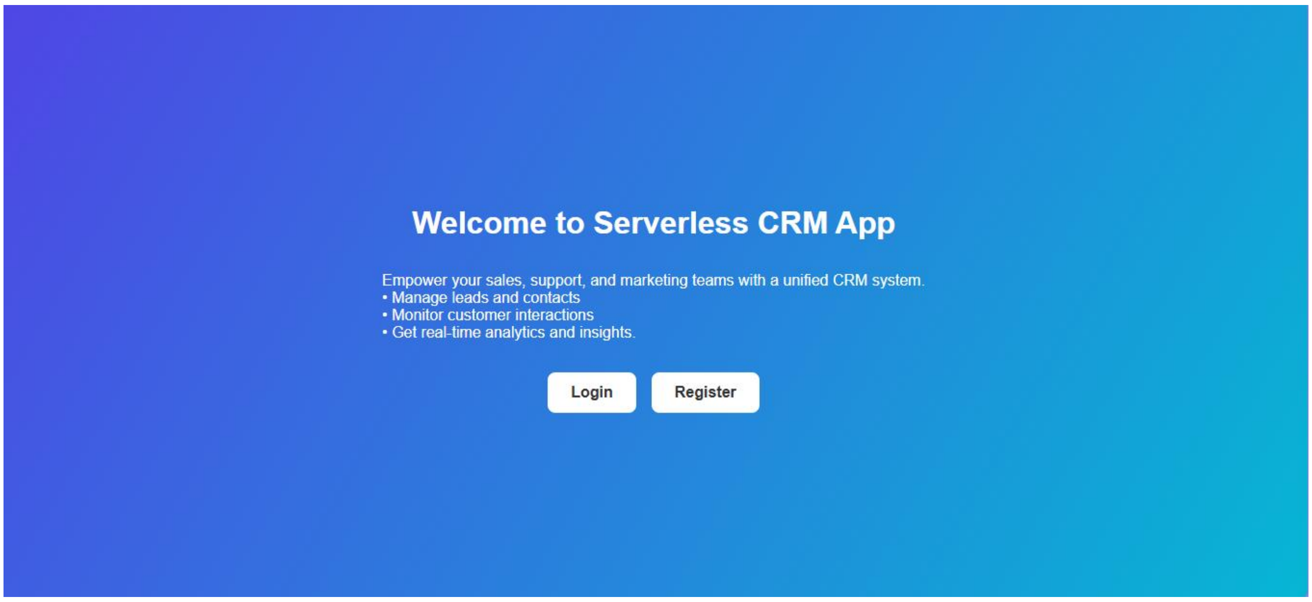
- **Home Page** – Introduction to the CRM system.
- **Register Page** – Allows new users to sign up.
- **Login Page** – Authenticates existing users.
- **Dashboard** – Displays CRM data after login.

- Integrated frontend scripts to send HTTP requests (fetch API calls) to AWS API Gateway endpoints.

Step 6: Hosted The Application on Amazon S3 :

- Created an **S3 bucket** to host the static web application files (HTML, CSS, JS).
- Enabled **Static Website Hosting** and uploaded all necessary frontend assets.
- Configured the **bucket policy** to allow public read access for web content.
- Verified that the website was accessible globally using the **S3 website endpoint**.
- Integrated the S3 frontend with the **API Gateway Invoke URL** to enable complete end-to-end interaction between the UI and backend.

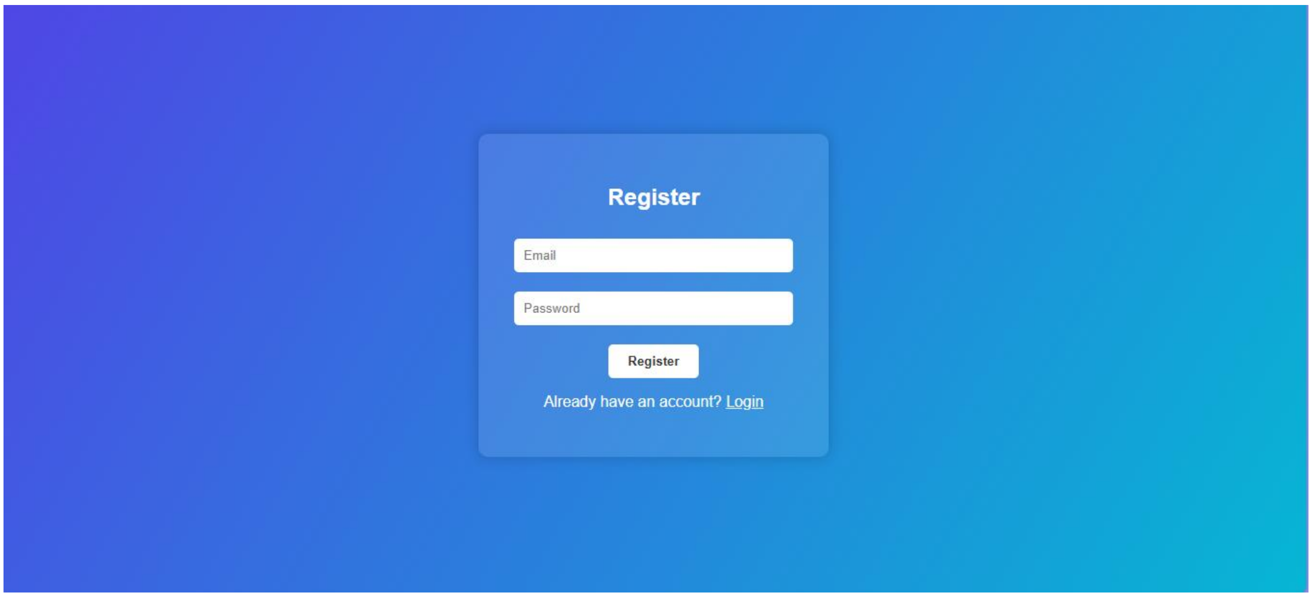
Final Output:



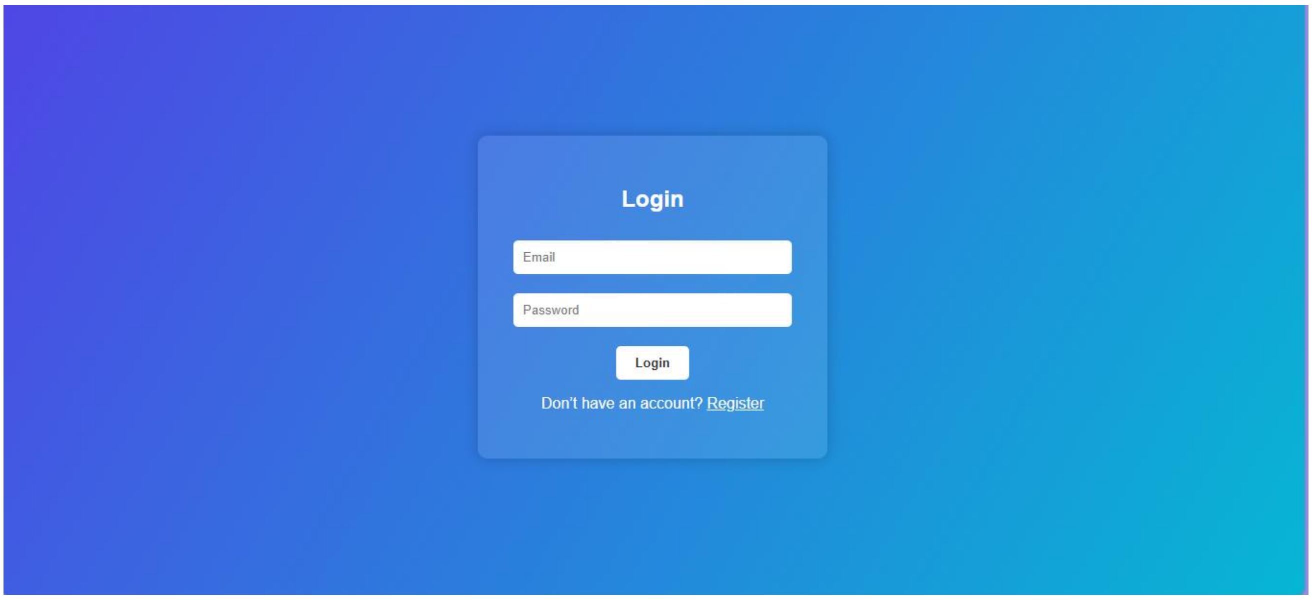
Welcome to Serverless CRM App

Empower your sales, support, and marketing teams with a unified CRM system.
• Manage leads and contacts
• Monitor customer interactions
• Get real-time analytics and insights.

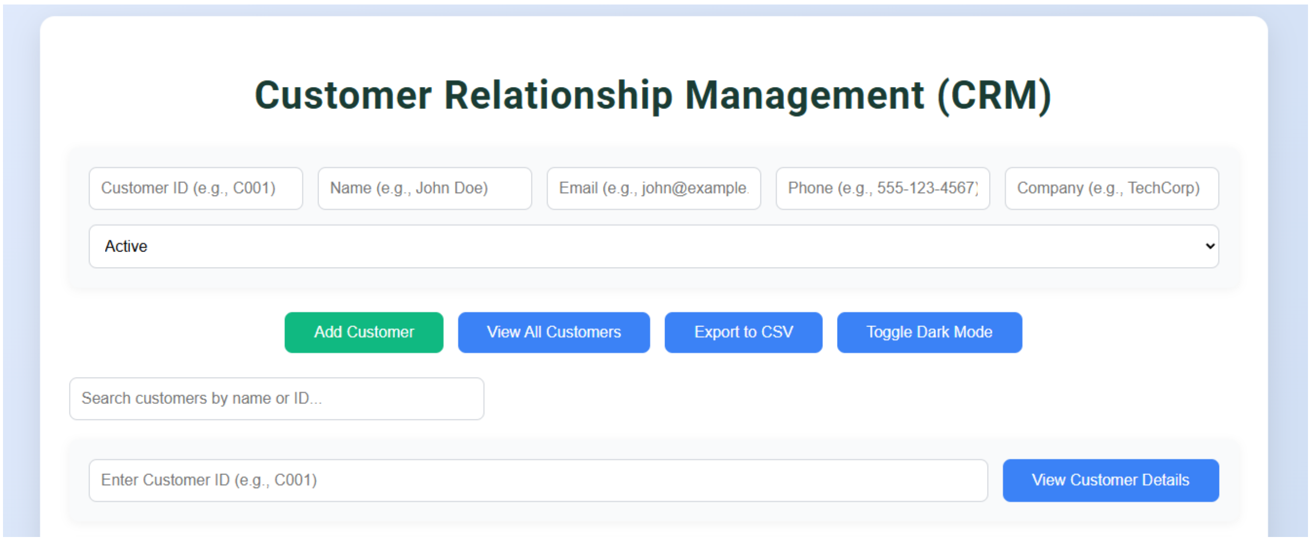
Home page For CRM



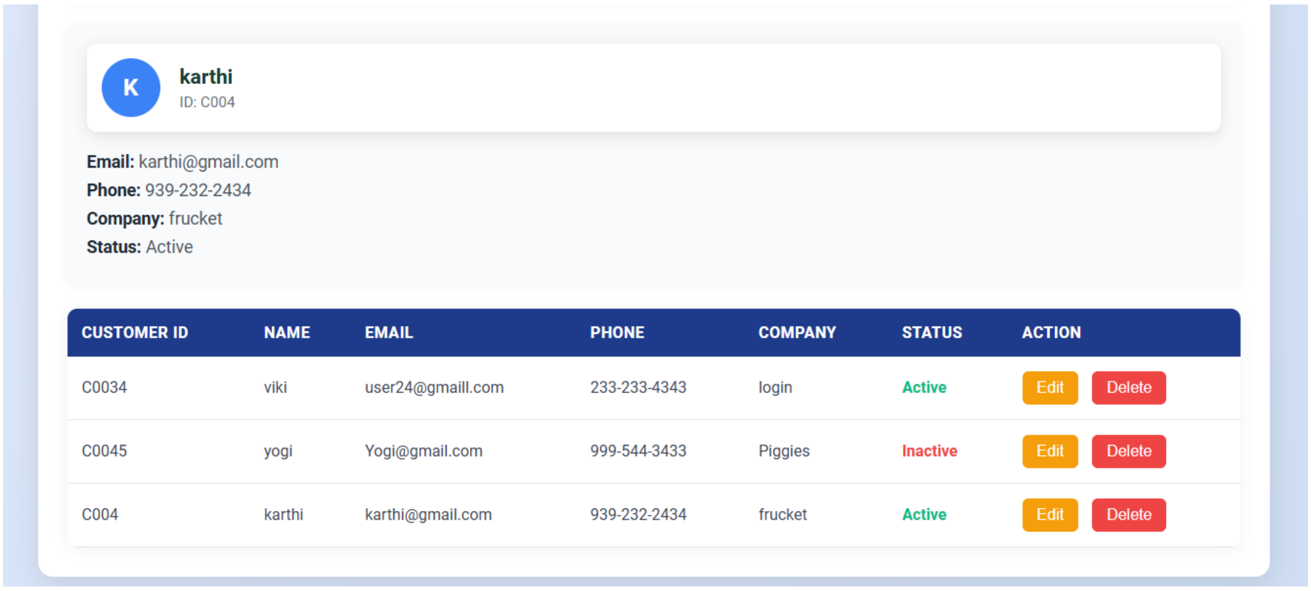
Register page For CRM



Login Page For CRM



CRM Application page


karthi
ID: C004

Email: karthi@gmail.com
Phone: 939-232-2434
Company: frucket
Status: Active

CUSTOMER ID	NAME	EMAIL	PHONE	COMPANY	STATUS	ACTION
C0034	viki	user24@gmaill.com	233-233-4343	login	Active	Edit Delete
C0045	yogi	Yogi@gmail.com	999-544-3433	Piggies	Inactive	Edit Delete
C004	karthi	karthi@gmail.com	939-232-2434	frucket	Active	Edit Delete

CRM Customer List

Conclusion:

The Serverless CRM Web Application successfully demonstrates the power and flexibility of AWS cloud services in building scalable, cost-effective, and efficient business solutions. By leveraging AWS Lambda, API Gateway, DynamoDB, S3, and IAM, the system eliminates the need for traditional server management while ensuring high availability and security.

The integration of **JWT-based authentication** adds a secure user access layer, protecting sensitive customer data and allowing only authorized users to interact with the CRM system. The **serverless architecture** ensures automatic scaling with minimal operational overhead, while **API Gateway and Lambda** enable a modular and event-driven design.

Overall, the project achieves the goal of creating a **reliable, secure, and fully managed CRM system** that can easily be extended with additional features like analytics dashboards, notification systems, and AI-based customer insights in the future. This implementation highlights the potential of AWS serverless technology in modern web application development.