AWS Backend Codes

Architecture

- API Gateway -> Lambda -> DynamoDB
- S3 Bucket for Picture Uploads

Stores Lambda Function

- CRUD Operations for Stores

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ScanCommand,
  PutCommand,
  GetCommand,
  DeleteCommand,
  UpdateCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const haversineDistance = (lat1, lon1, lat2, lon2) =%3E {
  const toRad = (value) => (value * Math.PI) / 180;
  const R = 6371; // Radius of the Earth in km

  const dLat = toRad(lat2 - lat1);
  const dLon = toRad(lon2 - lon1);
  const lat1Rad = toRad(lat1);
  const lat2Rad = toRad(lat2);

  const a =
    Math.sin(dLat / 2) * Math.sin(dLat / 2) +
    Math.sin(dLon / 2) * Math.sin(dLon / 2) * Math.cos(lat1Rad) *
Math.cos(lat2Rad);
  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
  const distance = R * c;
  return distance;
};
const dynamo = DynamoDBDocumentClient.from(client);

const tableName = "LastCallSG";

const currentLocation = {
  latitude: 1.29508,
  longitude: 103.848953
};

export const handler = async (event, context) => {
  let body;
  let statusCode = 200;
  const headers = {
    "Content-Type": "application/json",
  };

  try {
    switch (event.routeKey) {
      case "DELETE /stores/{id}":
        await dynamo.send(
          new DeleteCommand({
            TableName: tableName,
            Key: {
              id: event.pathParameters.id,
            },
          })
        );
        body = `Deleted item ${event.pathParameters.id}`;
        break;
      case "GET /stores/{id}":
        body = await dynamo.send(
          new GetCommand({
            TableName: tableName,
            Key: {
              id: event.pathParameters.id,
            },
          })
        );
        body = body.Item;
        break;
      case "GET /stores":
        const scanResult = await dynamo.send(new ScanCommand({ TableName:
tableName }));
        const stores = scanResult.Items;

        stores.forEach(store => {

          const storeLocation = {
            latitude: store.storeLatitude,
            longitude: store.storeLongitude
          };
          store.distanceFromCurrentLocation = haversineDistance(
            currentLocation.latitude,
            currentLocation.longitude,
            storeLocation.latitude,
            storeLocation.longitude
          );
        });

        stores.sort((a, b) => a.distanceFromCurrentLocation -
b.distanceFromCurrentLocation);
        body = stores;
        break;
      case "PUT /stores":
        let requestJSON = JSON.parse(event.body);
        await dynamo.send(
          new PutCommand({
            TableName: tableName,
            Item: {
              id: requestJSON.id,
              price: requestJSON.price,
              name: requestJSON.name,
            },
          })
        );
        body = `Put orders ${requestJSON.id}`;
        break;
      default:
        throw new Error(`Unsupported route: "${event.routeKey}"`);
    }
  } catch (err) {
    statusCode = 400;
    body = err.message;
  } finally {
    body = JSON.stringify(body);
  }

  return {
    statusCode,
    body,
    headers,
  };
};
```

Orders Lambda Function

- Create new order
- Read new order

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ScanCommand,
  PutCommand,
  GetCommand,
  DeleteCommand,
  UpdateCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const dynamo = DynamoDBDocumentClient.from(client);

const ordersTableName = "LastCallSG-Order";
const storesTableName = "LastCallSG";

export const handler = async (event, context) =%3E {
  let body;
  let statusCode = 200;
  const headers = {
    "Content-Type": "application/json",
  };

  try {
    switch (event.routeKey) {
      case "DELETE /orders/{id}":
        await dynamo.send(
          new DeleteCommand({
            TableName: ordersTableName,
            Key: {
              id: event.pathParameters.id,
            },
          })
        )
        body = `Deleted item ${event.pathParameters.id}`;;
        break;
      case "GET /orders/{id}":
        const getResult = await dynamo.send(
          new GetCommand({
              TableName: ordersTableName,
              id: event.pathParameters.id,
            }))
        body = body.Item;
        break;
      case "GET /orders":
        body = await dynamo.send(
          new ScanCommand({ TableName: ordersTableName })
        );
        body = body.Items;
        break;
      case "POST /orders":
        let requestJSON = JSON.parse(event.body);

        // Create new order in LastCallSG-Order table
        await dynamo.send(
          new PutCommand({
            TableName: ordersTableName,
            Item: {
```

```
            id: requestJSON.id,
            username: requestJSON.username,
            contact: requestJSON.contact,
            email: requestJSON.email,
            quantity: requestJSON.quantity,
            price: requestJSON.price,
            totalPrice: requestJSON.totalPrice,
            discount: requestJSON.discount,
            storeLogo: requestJSON.storeLogo,
            storeTitle:requestJSON.storeTitle,
            itemName: requestJSON.item.name,
            store:requestJSON.store,
            isPaid:requestJSON.isPaid,
          },
        })
      );

      // Update item quantity in LastCallSG table
      const store = requestJSON.store;
      const itemToUpdate = requestJSON.item;

      const updatedItems = store.items.map(storeItem => {
        if (storeItem.name === itemToUpdate.name) {
          storeItem.quantity = Math.max(0, storeItem.quantity -
requestJSON.quantity);
        }
        return storeItem;
      });

      await dynamo.send(
        new UpdateCommand({
          TableName: storesTableName,
          Key: {
            id: store.id,
          },
          UpdateExpression: "SET #items = :updatedItems",
          ExpressionAttributeNames: {
            "#items": "items",
          },
          ExpressionAttributeValues: {
                ":updatedItems": updatedItems,
              },
            })
          );

          body = `Post item ${requestJSON.id}`;
          break;
        case "PUT /orders":
          let updateRequestJSON = JSON.parse(event.body);
          await dynamo.send(
            new PutCommand({
              TableName: ordersTableName,
              Item: {
                id: updateRequestJSON.id,
                price: updateRequestJSON.price,
                name: updateRequestJSON.name,
              },
            })
          );
          body = `Put item ${updateRequestJSON.id}`;
          break;
        default:
          throw new Error(`Unsupported route: "${event.routeKey}"`);
      }
    } catch (err) {
      statusCode = 400;
      body = err.message;
    } finally {
      body = JSON.stringify(body);
    }

    return {
      statusCode,
      body,
      headers,
    };
  };
```

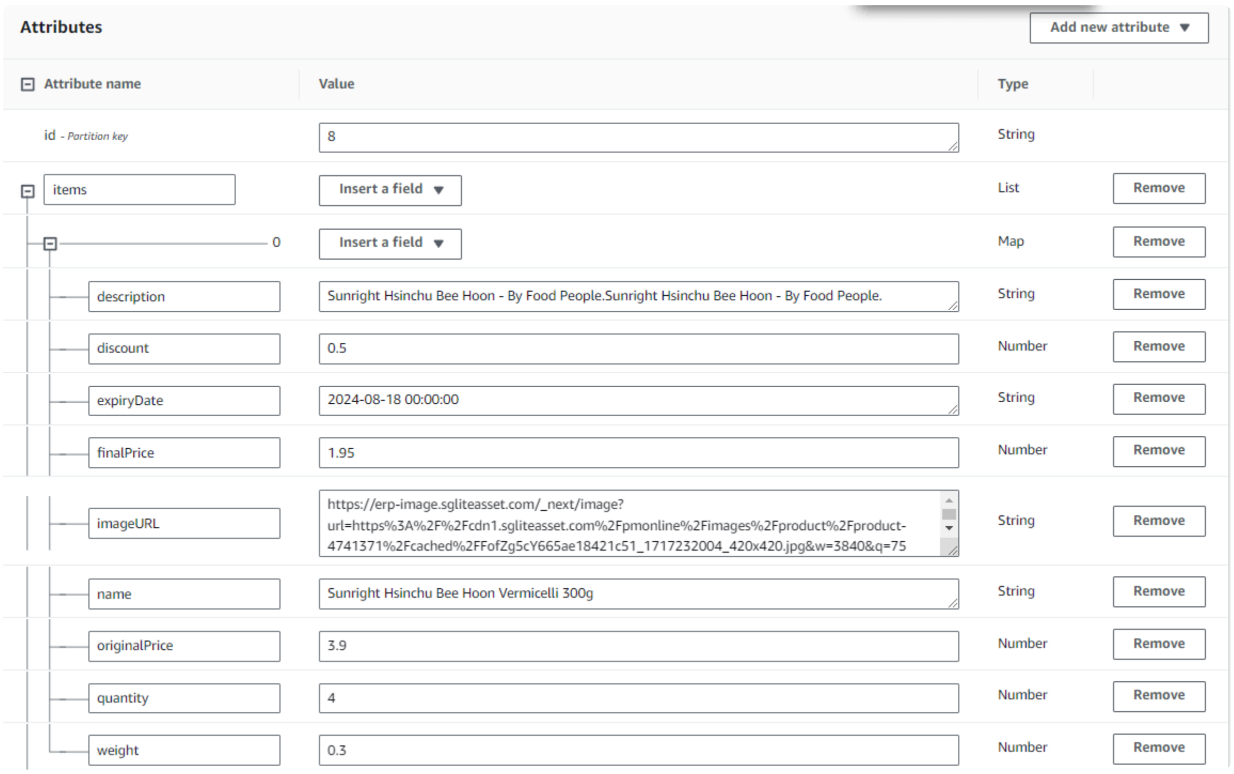
User Registration Lambda Function

- No authentication

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  GetCommand,
  PutCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const dynamo = DynamoDBDocumentClient.from(client);

const tableName = "LastCallSG-users";

export const handler = async (event) =%3E {
  let body;
  let statusCode = 200;
  const headers = {
    "Content-Type": "application/json",
  };

  try {
    const requestJSON = JSON.parse(event.body);

    switch (event.routeKey) {
      case "POST /register":
        // Check if user already exists
        let existingUser = await dynamo.send(
          new GetCommand({
            TableName: tableName,
            Key: {
              username: requestJSON.username,
            },
          })
        );

        if (existingUser.Item) {
          throw new Error("User already exists");
        }
      // Add new user
      await dynamo.send(
        new PutCommand({
          TableName: tableName,
          Item: {
            username: requestJSON.username,
            password: requestJSON.password,
            email:requestJSON.email,
          },
        })
      );
      body = { message: "User registered successfully" };
      break;

    case "POST /login":
      // Get user from the database
      let user = await dynamo.send(
        new GetCommand({
          TableName: tableName,
          Key: {
            username: requestJSON.username,
          },
        })
      );

      if (!user.Item || (requestJSON.password !== user.Item.password)) {
        throw new Error("Invalid username or password");
      }

      body = {
        message: "Login successful",
        email: user.Item.email, // Include the email in the response
      };
      break;



    default:
      throw new Error(`Unsupported route: "${event.routeKey}"`);
  }
  } catch (err) {
    statusCode = 400;
    body = { error: err.message };
  } finally {
    body = JSON.stringify(body);
  }

  return {
    statusCode,
    body,
    headers,
  };
};
```

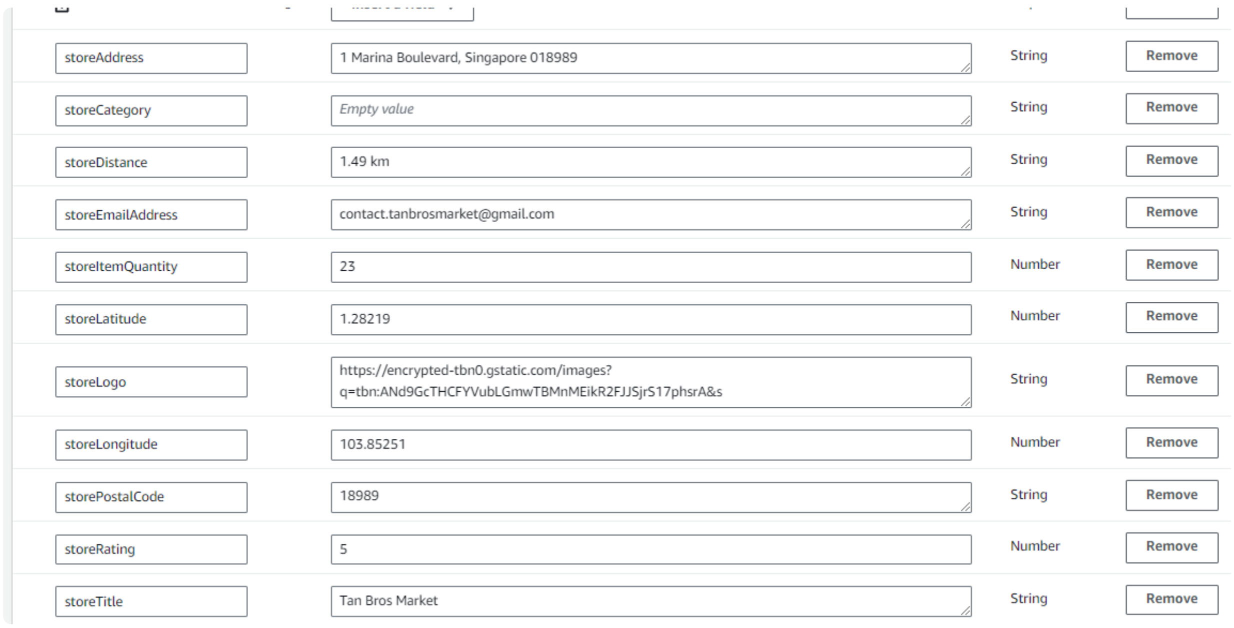
Store DB

- Store Items - Stored as an array of items within the store itself



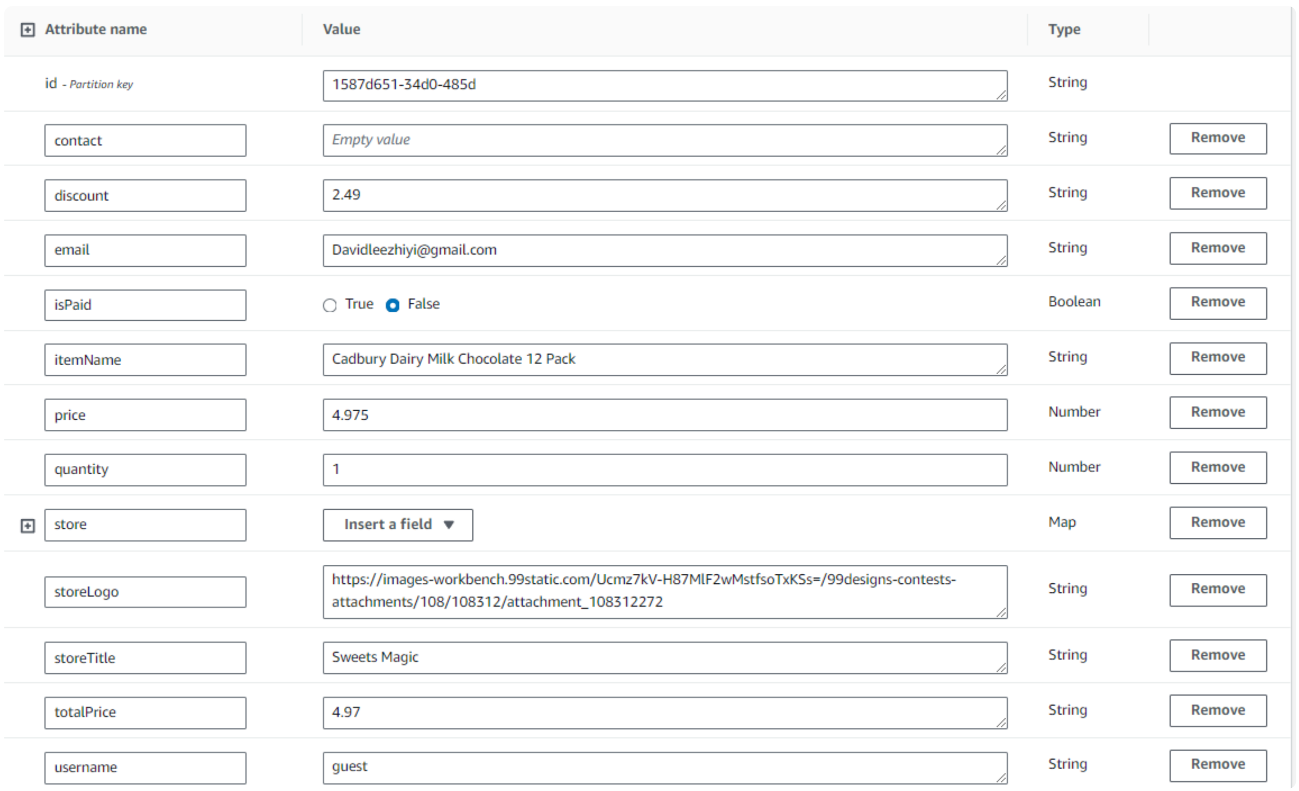
Attribute name	Value	Type
id - *Partition key*	8	String
items		List
0		Map
description	Sunright Hsinchu Bee Hoon - By Food People.Sunright Hsinchu Bee Hoon - By Food People.	String
discount	0.5	Number
expiryDate	2024-08-18 00:00:00	String
finalPrice	1.95	Number
imageURL	https://erp-image.sgliteasset.com/_next/image?url=https%3A%2F%2Fcdn1.sgliteasset.com%2Fpmonline%2Fimages%2Fproduct%2Fproduct-4741371%2Fcached%2FFofZg5cY665ae18421c51_1717232004_420x420.jpg&w=3840&q=75	String
name	Sunright Hsinchu Bee Hoon Vermicelli 300g	String
originalPrice	3.9	Number
quantity	4	Number
weight	0.3	Number

- Individual Store



storeAddress	1 Marina Boulevard, Singapore 018989	String	Remove
storeCategory	*Empty value*	String	Remove
storeDistance	1.49 km	String	Remove
storeEmailAddress	contact.tanbrosmarket@gmail.com	String	Remove
storeItemQuantity	23	Number	Remove
storeLatitude	1.28219	Number	Remove
storeLogo	https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTHCFYVubLGmwTBMnMEikR2FJJSjrS17phsrA&s	String	Remove
storeLongitude	103.85251	Number	Remove
storePostalCode	18989	String	Remove
storeRating	5	Number	Remove
storeTitle	Tan Bros Market	String	Remove

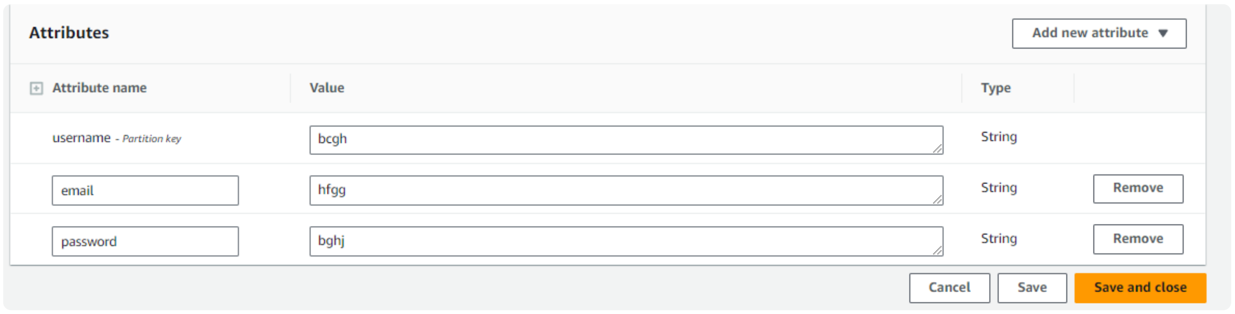
Orders DB



Attribute name	Value	Type	
id - *Partition key*	1587d651-34d0-485d	String	
contact	*Empty value*	String	Remove
discount	2.49	String	Remove
email	Davidleezhiyi@gmail.com	String	Remove
isPaid	True [✓ False]	Boolean	Remove
itemName	Cadbury Dairy Milk Chocolate 12 Pack	String	Remove
price	4.975	Number	Remove
quantity	1	Number	Remove
store	Insert a field ▼	Map	Remove
storeLogo	https://images-workbench.99static.com/Ucmz7kV-H87MlF2wMstfsoTxKSs=/99designs-contests-attachments/108/108312/attachment_108312272	String	Remove
storeTitle	Sweets Magic	String	Remove
totalPrice	4.97	String	Remove
username	guest	String	Remove

Users DB

- Light weight, no authentication



Attributes

Attribute name	Value	Type	
username - *Partition key*	bcgh	String	
email	hfgg	String	Remove
password	bghj	String	Remove