DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

# CSCI 5409 Cloud Computing

# Project

Submitted by:
Team 10


| Team Member | Banner ID | Mail |
|---|---|---|
| Anqi Chen | B00838586 | anqi.chen@dal.ca |
| Hardik Patel | B00854244 | hr325484@dal.ca |
| Jay Mandaliya | B00851416 | jy665593@dal.ca |
| Ridham Dabhi | B00853506 | ridhamdabhi@dal.ca |
| Vivek Sakariya | B00848519 | vv329933@dal.ca |

## Table of Contents

## Table of Tables and Figures

## Summary

- **Delivery files**
  - Proj-Team10-Report.pdf
  - Proj-Team10-TasksForm.pdf
  - Proj-Team10-CollaborationTools.zip
  - Proj-Team10-Demo-Cases.pdf
- **Tasks achieved**
  - Our scenario includes three independent companies organized together through orders.
  - Data are stored on three different DBs on clouds.
  - ORM is used to access information in two of the databases.
  - Deployed Bakery Company on AWS with serverless computing.
  - Containers are used for the deployment of Supply and Order Companies.
  - Used XA transactions [4] to achieve functional requirements across all companies.
  - Tested every endpoint with Postman.
  - Used the agile software development method based on Sprints and Scrum.
- **Excellence**
  - In addition to the functional requirements, we added a few features like registration and added email verification function for user authentication.
  - Instead of deploying two companies on clouds, we deployed three companies using different cloud platform.
  - We used diverse functions existing in node.js (like nodemailer) to expand our knowledge and understanding of the framework [2].

## Business Scenario

As restaurants and bakery shops become more and more digitally driven, they're turning to next-generation order management solutions to help them bridge the gap between their physical and online operations. We have designed an order application where users are able to add or view an order, for three companies to manage their own resources and interact with other companies. If an authenticated user makes an effective order online, the order system should then send the order to both the bakery and the warehouse to prepare for the order.

UI for the order website should be user-friendly so that our customers can easily make an order on the platform. For our company, the security of user identity is also important, we must ensure the one who make operations on websites are real customers, we will verify users by sending an account activation link through an email.

The frontend of the Bakery Company website will be used by the staff of the bakery company, so the UI is simple and straightforward. All the operations to manage dishes and orders can be easily reached through the navigation bar and buttons on homepage.

The frontend of the Ingredients Company will be used by the staff of the ingredients supply company, so the UI is simple and straightforward. All the operations to manage ingredients and orders can be easily accessed through the navigation bar and buttons on homepage.

## Software Overview

### A. Bakery Company

The basic function of Bakery Company is to manage available products and also accept orders from the Order App. The operations on the website are listed as below:

- List products
- Add product
- Edit product
- Accept Order

- The web pages can be seen here:

  http://bakery-host.s3-website-us-east-1.amazonaws.com

### B. Ingredients Company

The basic function of Ingredients Company is to manage available ingredients which can be supplied for Bakery Company, and also accept orders from the Order App. The operations on the website are listed as below:

- List ingredients
- Add ingredients
- Edit ingredient

- Accept Order

- The web pages can be seen here:

  https://ingredientsappcloud.azurewebsites.net/

## C. Order App

The basic function of Order Company is to manage orders from users, send the corresponding requests for Bakery and Ingredients Companies. The operations on the website are listed as below:

- Login/logout
- Register and verify user status
- Add Order
- View Orders

- The web pages can be seen here:

  https://orderappcloud.herokuapp.com/

### Software Frameworks

Bakery Company uses AWS serverless computing platform, while OrderApp and Ingredients Company are deployed on cloud (Azure and Heroku respectively) using Containerization.

All the three companies use SailsJS as a real-time MVC Framework for Node.js. The serverless flow chart (Figure 1) and the MVC software framework (Figure 2) are shown below.

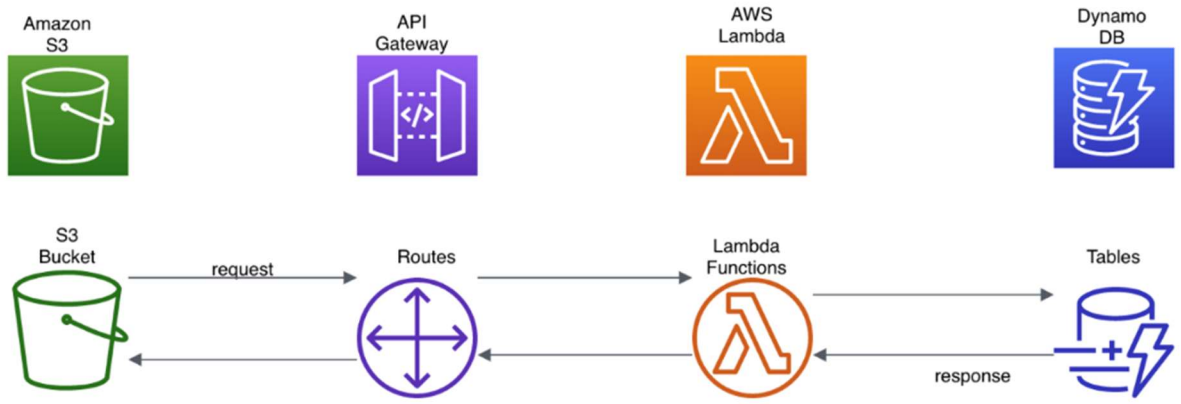


**Figure 1: Flow chart of AWS Serverless Computing drawn using draw.io [5]**

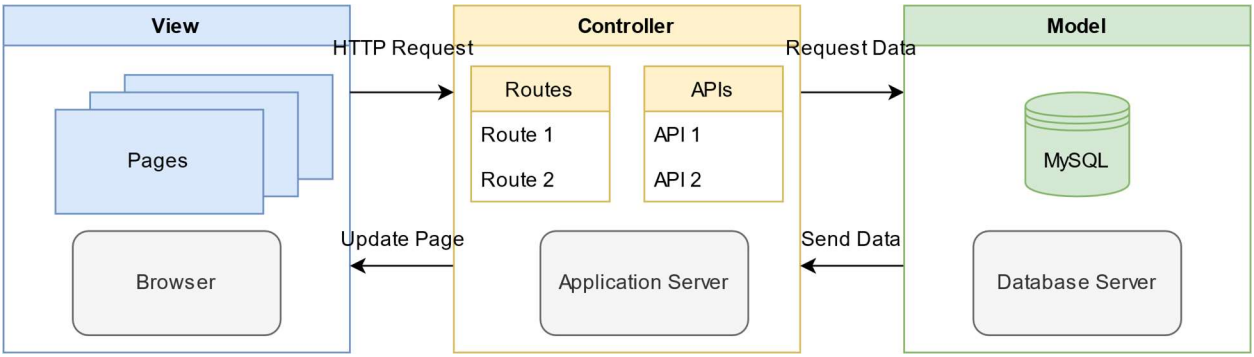


**Figure 2: Software Architecture of SailsJS MVC Framework drawn using draw.io [5]**

## RESTful API Endpoints

**Table 1: API endpoints for the bakery company**

| Bakery | | | |
|---|---|---|---|
| **Description** | **URL** | **Method** | **Parameter** |
| Get All Products | *https://v8nokd46w9.execute-api.us-east-1.amazonaws.com/default/getAllProducts* | GET | - |
| Get Product with Id | *https://v8nokd46w9.execute-api.us-east-1.amazonaws.com/default/getProductById* | POST | id |

**Table 2: API endpoints for the supply company**

| Supply | | | |
|---|---|---|---|
| **Description** | **URL** | **Method** | **Parameter** |
| Get All Ingredients | https://ingredientsappcloud.azurewebsites.net/list | GET | - |
| Get Ingredient with Id | https://ingredientsappcloud.azurewebsites.net/ getIngredientByID | POST | id |

## Steps to Complete the Task

### A. Ingredients Application

1. Created a web application for Company Y to manage Parts because it was is independent of other web applications.
    a. Created “Ingredients” API and endpoints to perform CRU operation on Ingredients.
    b. Created “Order” API and endpoints to record orders.
    c. Tested endpoints with local MySQL database.
2. Created a managed relational database instance of AWS RDS on AWS.
    a. Tested endpoints of the ingredients webapp with managed relational database.
3. Containerized the webapp of Ingredients Supply Company and deployed on MS Azure.

### B. Bakery Application

1. Created the web application for the Bakery to manage bakery products. It is dependent on Ingredients Company (A product can only have parts that exist in Company Y’s inventory). Created the web application for Company Y to manage Parts because it was is independent of other web applications.
    a. Created “Product” (addProduct, editProduct, deleteProduct, getAllProduct, getProduct,ById) and “Orders” APIs in AWS API Gateway.
    b. Deployed the APIs and created a role used in our lambda function to provide access at different levels.
    c. Created the corresponding Lambda functions to request data.
    d. Tested the function with mock response and integrated API with the AWS Lambda.
2. Created tables on AWS DynamoDB and make the necessary changes.

a. Created the table of “Product”.
b. Created the table of “Ingredient”.
c. Created the table of “Order”.
d. Changed Lambda function code to send request for data in the DynamoDB and accepts the response.
e. Changed API Gateway in in Method Request and Integration Request.

3. Tested on postman for all the APIs created using API Gateway.

## C. Order Application

1. Created the web application for Order App to manage Orders. This is dependent on both Companies, the Bakery and Ingredients.
    a. Created “Order” API and endpoints to record order details.
    b. Created “User” API and endpoints to perform user authentication.
    c. Tested endpoints with local MySQL database.
2. Created a managed relational database instance of AWS RDS on AWS.
    b. Tested endpoints of the ingredients webapp with managed relational database.
3. Containerized the webapp of Order App and deployed on Heroku.

## Collaboration Summary

We have used a few collaboration tools, namely, FCS Gitlab code repository, MS Teams, Trello, and minutes of meetings to make decisions and plans informed to every group member.

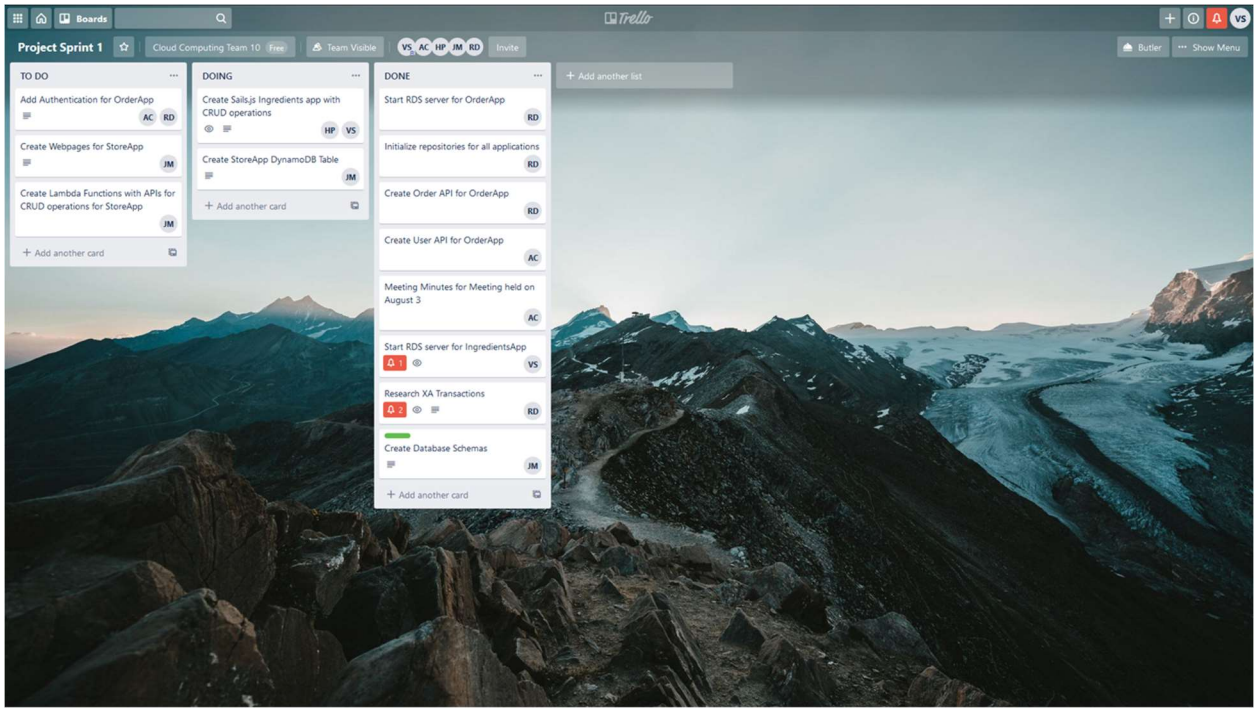


**Figure 3: Trello task management for Sprint 1**

The above screenshot defines various tasks assigned to each of the group member based on the choice and voting of individuals. It includes a task based on the research to be done for XA transactions and even a description of meetings recorded. The other task related to database initialization, API creation and other minor contributions are also listed defined for the board.

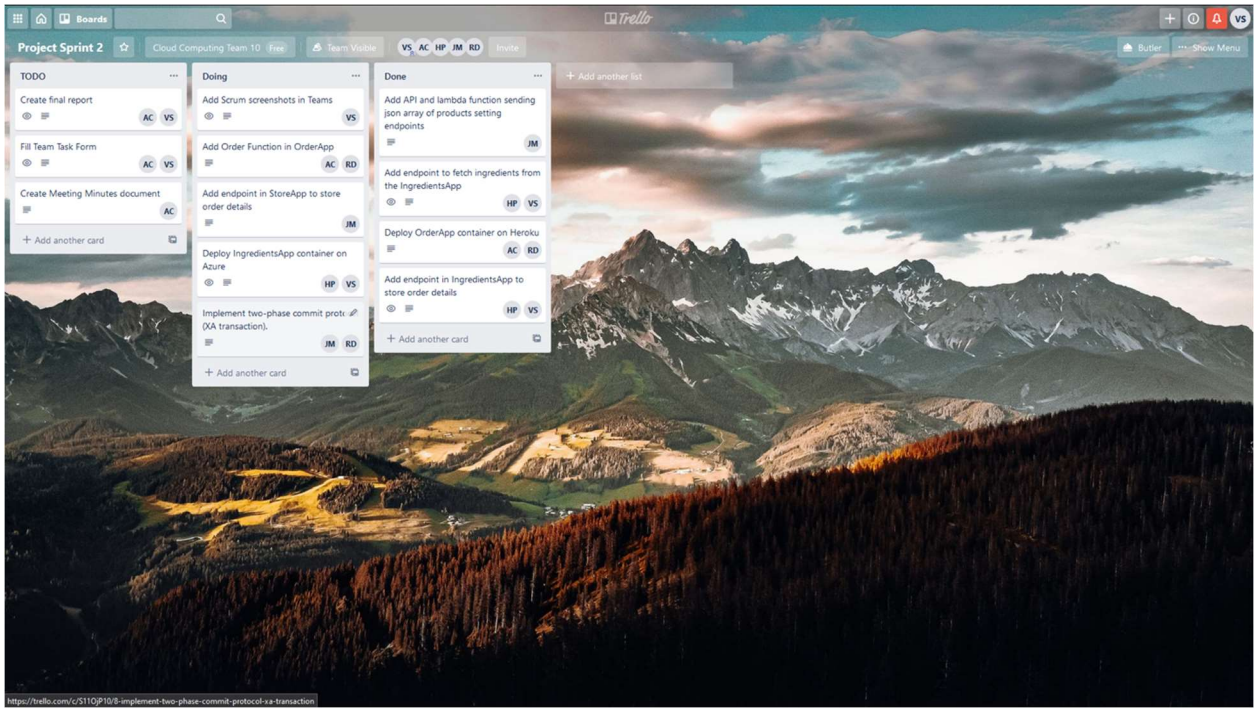


**Figure 4: Trello task management for Sprint 2**

The above given snapshot explains the details about the tasks assigned in Sprint 2. The important tasks like final report completion, task form, meeting minutes, implementation of XA transaction, and Scrum screenshot are included in this part of project.

All the screenshots of Trello and the copies of meeting minutes can be found in a separate file named 'Proj-Team10-CollaborationTools.zip'.

The assigned tasks and responsibilities of each group member are described in the file called 'Proj-Team10-TasksForm.pdf', please refer to that file for more details.

## Code Repositories

We used FCS GitLab as the platform for us to upload and commit changes of the code.

- GitLab repository for Company X could be found here:

  https://git.cs.dal.ca/rdabhi/csci5409-project-storeapp.git

- GitLab repository for Company Y could be found here:

  https://git.cs.dal.ca/rdabhi/csci5409-project-ingredientsapp.git

- GitLab repository for Order Application could be found here:

  https://git.cs.dal.ca/rdabhi/csci5409-project-orderapp.git

### Access Information

We have given our repositories internal access, using which anyone within Dalhousie University can access the repositories. We did this to make sure that any TA or Professor Bodorik can access the repositories.

## Resources-related Information

### Cloud resources

- **AWS serverless computing** (S3 Buckets, API Gateway, Lambda functions, DynamoDB)
  Used for the implementations of Bakery Company.
- **Microsoft Azure**
  Used for Ingredients Company to be deployed as a container.
- **Heroku**
  Used for Order Company to be deployed as a container.

### Security Mechanisms

For security reasons, we stored user passwords in hashed form, which guards against the possibility that someone who gains unauthorized access to the database can retrieve the passwords of every user in the system. A one-time password (OTP) is an automatically generated numeric or alphanumeric string of characters that authenticates the user for login session [1].

We have used these two ways to avoid the pitfalls of a pair of static username and password stored in the database used for user identification. If a user register as a new customer, we will send a security link including a random string of secret token to the user email, to check the validation. Before a successful verifying process, the user is marked as not active by default. After the verification, the user will be marked active and then can log in successfully and make operations on the website [3].

## Database details

### Database Schema

We have designed three databases to store information for each company separately. The one used for ingredients supply company is designed to have two tables ‘Ingredient’ and ‘Order’ to manage the quantity of each ingredient. The one used for Bakery Company has ‘Product’, ‘Ingredient’ and ‘Order’, while the one for OrderApp has two tables ‘User’ and ‘Order’ to manage the process of making an order (shown in table 3).

**Table 3: Database Schema of three companies**

| **Supplies** | |
|---|---|
| Ingredient | *(id, name, qoh)* |
| Order | *(id, ingredient_id, quantity)* |
| **Bakery** | |
| Product | *(id, name, price)* |
| Ingredient | *(id, product_id, ingredient_id, ingredient_name, quantity)* |
| Order | *(id, product_id, quantity, user_id, time)* |
| **OrderApp** | |
| User | *(id, email, active, password, secretToken)* |
| Order | *(id, product_id, quantity, user_id, time, status)* |

## Database access

- **Bakery Company DynamoDB**

  NoSQL on AWS, no need to access via endpoints.

- **Supply Company RDS**

  Endpoint: inventoryappcloudproject.cfoav4vcqlvs.us-east-1.rds.amazonaws.com

  User: group_10

  Password: group_10

- **Order App RDS**

  Endpoint: orderapp.cpajqlojz9ry.ca-central-1.rds.amazonaws.com

  User: group_10

  Password: group_10

## References

[1] “What is one-time password (OTP)? - Definition from WhatIs.com,” *SearchSecurity*. https://searchsecurity.techtarget.com/definition/one-time-password-OTP [Accessed Aug. 09, 2020].

[2] B. Awad, “Confirmation Email with Node.js,” *Youtube,* Online Video: https://www.youtube.com/watch?v=76tKpVbjhu8 [Accessed Aug. 09, 2020].

[3] “Build a One-Time Password (OTP) Service Using the Dispatch API - Vonage Developer Blog.” https://www.nexmo.com/blog/2019/02/07/build-an-otp-service-dr [Accessed Aug. 09, 2020].

[4] “MySQL :: MySQL 5.7 Reference Manual :: 13.3.7 XA Transactions.” https://dev.mysql.com/doc/refman/5.7/en/xa.html [Accessed Aug. 09, 2020].

[5] draw.io, “draw.io - Diagrams For Everyone, Everywhere,” draw.io. https://drawio-app.com/ [Accessed Jul. 03, 2020].