Bjorn's Brew

Menu Ordering Application

Software System Design

Sean Ward
svw5927@psu.edu

Table of Contents

Business Description

Bjorn's Brew is a small chain of coffee shops operating in the greater Salt Lake City area. "Bjorn's" as it is affectionately referred to by its customers, aims to provide high quality coffee, tea, pastries and other breakfast items to its customers at reasonable prices. Bjorn's has made a great name for themselves in Salt Lake City by providing their goods through both drive through services and sit down locations.

Locations

Foothill Drive
Foothill Drive is primarily drive through based. They have a single drive up payment window that is serviced through two drive up menu locations. These menu locations employ speakers and microphones for the customers to communicate their orders to the barista's. Foothill Drive also uses a walk up counter where customers can place their orders directly to a barista if they are not in their cars.

Highland Drive
Highland Drive is primarily drive through based. They have 2 drive up payment windows that are each serviced through their own drive up menu locations. These menu locations employ speakers and microphones for the customers to communicate their orders to the barista's. Highland Drive also uses a walk up counter where customers can place their orders directly to a barista if they are not in their cars.

Business Requirements

Problem Statement

As the popularity of Bjorn's Brew increases, the amount of patrons that they are servicing each day is greatly increasing. Since their locations rely so heavily on their drive through ordering experience, this can lead to long lines of cars and increased wait times for their patrons. Their ordering process is starting to develop a bottleneck at their drive up menus where customers order their items verbally through the microphone.

Proposed Solution

Bjorn's would like to create an online menu ordering application where customers are able to place their orders prior to arriving at any of their locations. These online orders will be placed using a mobile device. The orders will then be sent to the barista team where they are displayed on an order list tablet. Here the baristas will be able to manage and assemble orders prior to the customer arriving, allowing for a speedy pickup process. This application will allow for faster order fulfillment leading to less wait time for the customers.

Business Goals

Reduce wait times for patrons
Reduce friction in ordering process
Spread order requests over larger period of time
Streamline menu item management

Functional Requirements

User Interface

The system shall provide an interface for users to order goods.
The system shall allow users to select the location where they will be ordering from.
The system shall display a menu (based on location) of all the goods that Bjorn's Brew provides.
The system shall allow users to select items from the menu
The system shall allow users to pre order their selected items
The system shall allow users to choose their pick up time.
The system shall allow users to pre pay for their order.
The system shall integrate with the current POS System.

Employee Interface

The system shall provide an interface for employees to receive orders
The system shall organize orders based on pickup time
The system shall allow orders to be marked as "assembling"
The system shall allow orders to be marked as "assembled"
The system shall allow order to be marked as fulfilled

Non-Functional Requirements

Usability

The system must be highly usable. To encourage patrons of Bjorn's Brew to start placing their orders through the application, all friction in the application ordering process must be minimum.

Performance

Due to the fast nature of drive through ordering, the system must maintain a high level of performance. Orders placed must be displayed on orders tablet within 1 second of processing.

Security

The system will be handling sensitive data (customer billing info, credit cards) and must therefore be highly secure. All credit card data must be encrypted to protect the business and the customer.

Scalability

Bjorn's Brew has seen rapid growth in the past 10 years projections are showing that the business will continue to grow at this rate. Bjorn's requires an application that will scale with their business as they continue to add new locations and receive more customers.

System Design

Use Case Analysis and Diagram

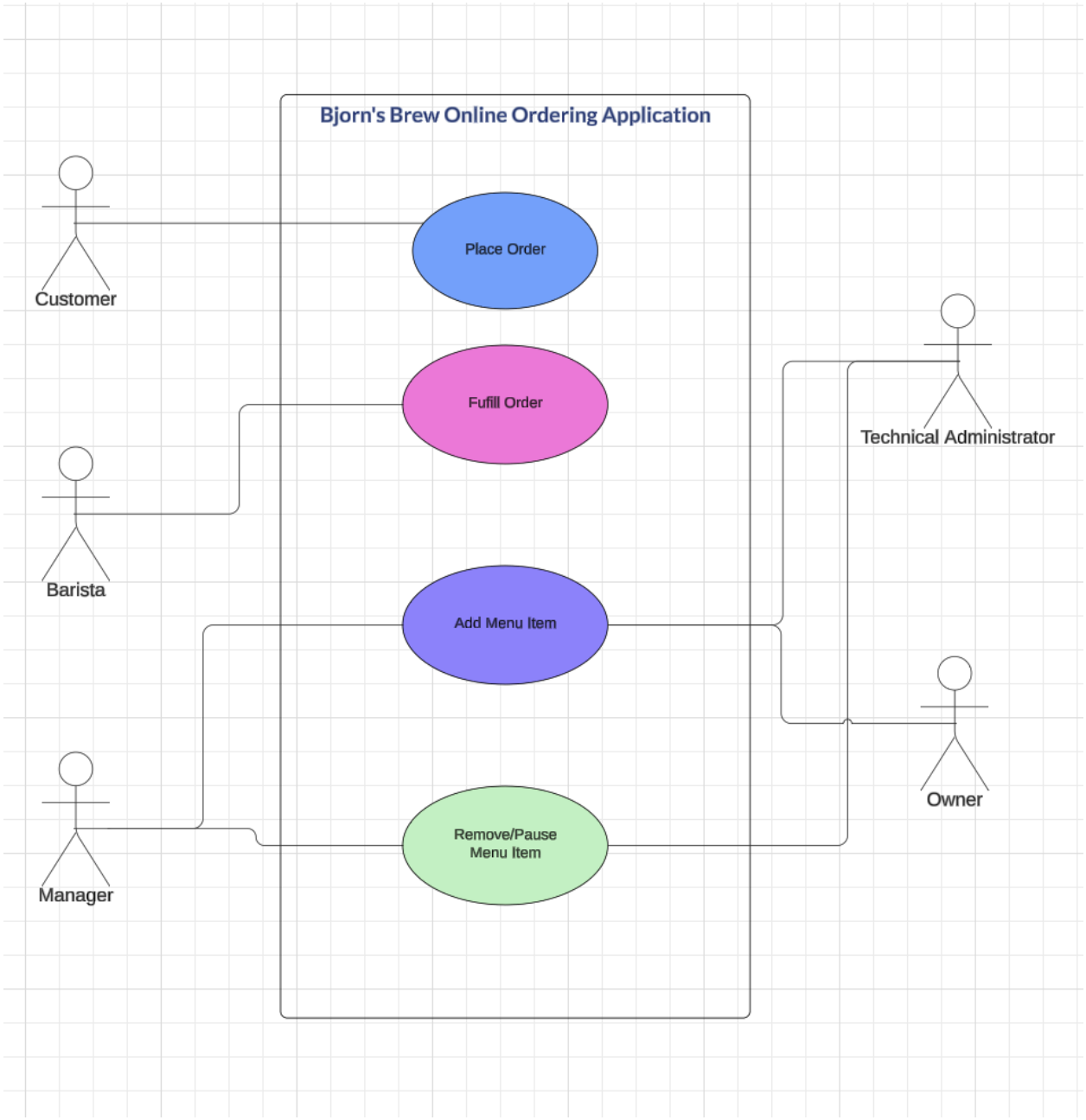
Actors

Type	Actor	Goal Description
Primary		
	Customer	Order their items with the least amount of friction as possible. Wait in line for the shortest amount of time. Feel comfortable using an application to place their orders.
	Barista	Deliver the order to the customer as efficiently as possible. Receive orders in a well organized manner.
	Manager	Ensure a productive workplace environment for the workers. Provide excellent service to customers.
Secondary		
	Technical Administrator	Ensure stability of the application. Assist Owners in changes to app configuration.
	Owner	Increase bottom line. Streamline process to be able to provide as many orders as possible efficiently. Promote brand trust to patrons.

Use Cases

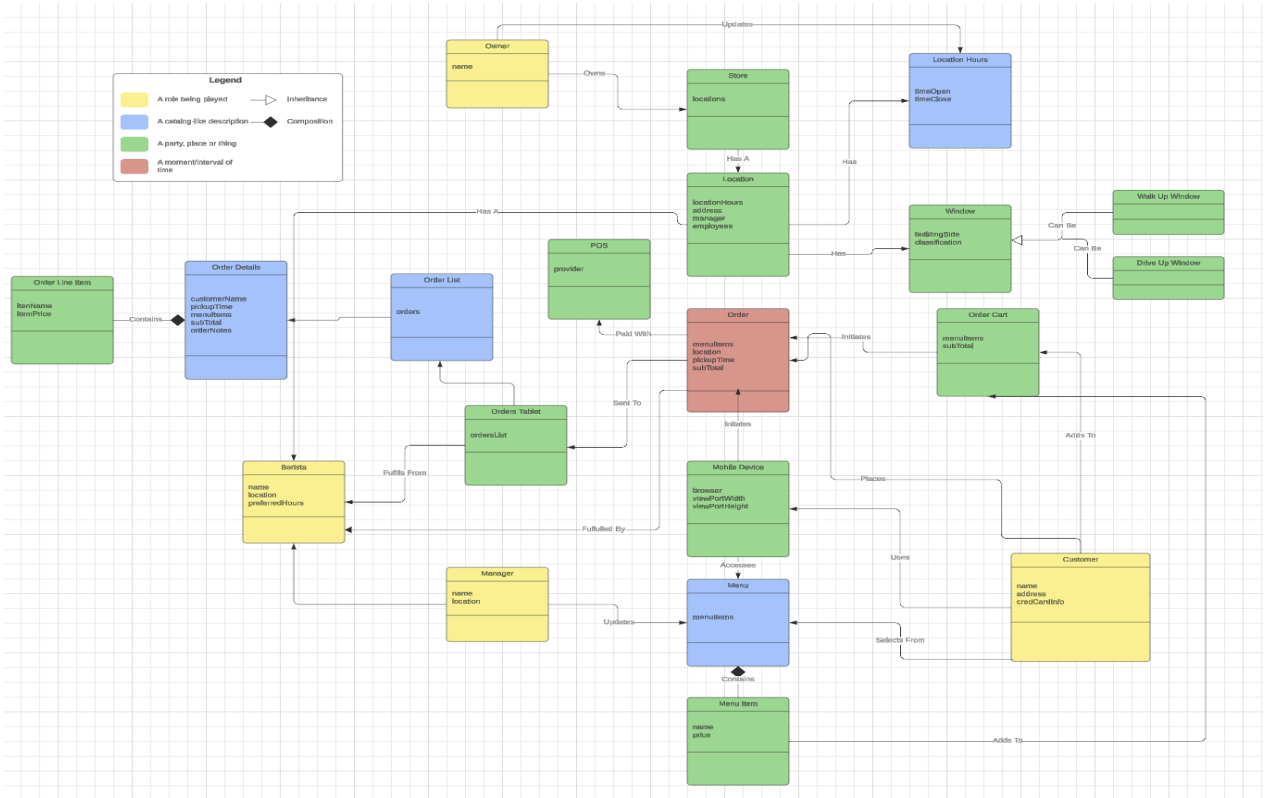
Use Case 1 - Place Order
Use Case 2 - Fulfill Order
Use Case 3 - Add Menu Item
Use Case 4 - Remove/ Pause Menu Item

Figure 1: Use Case Diagram



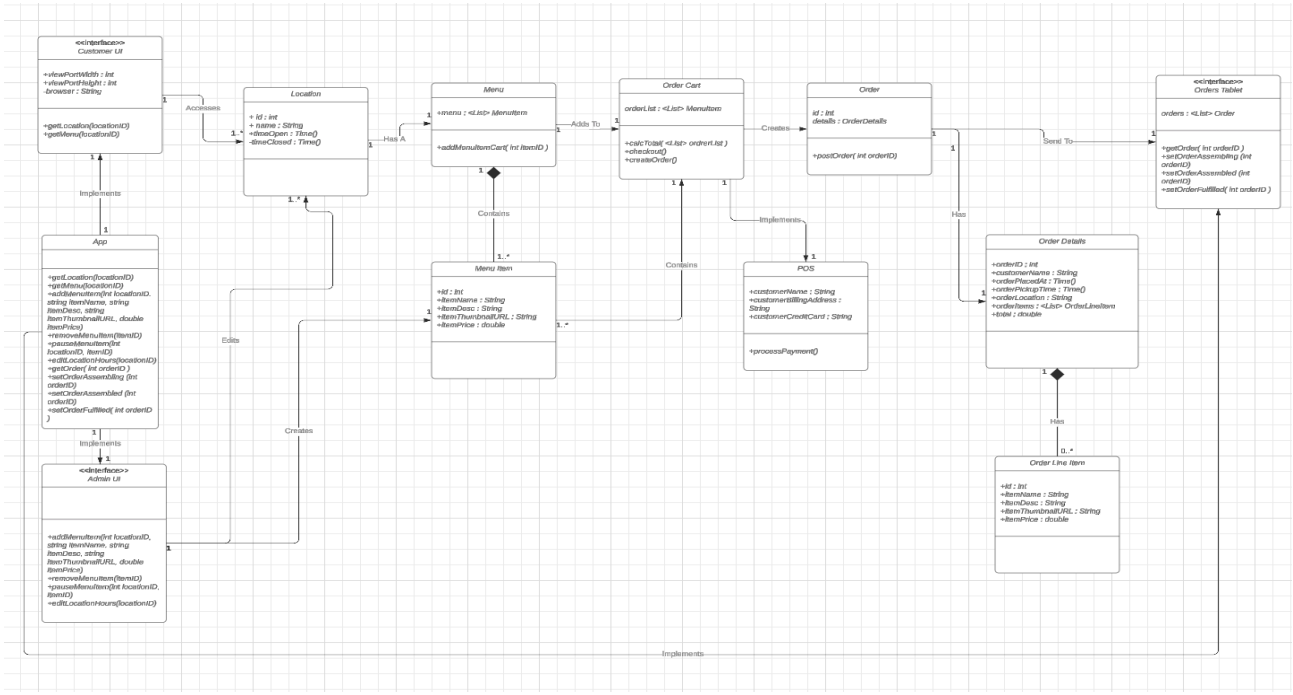
Domain Model

Figure 2: Domain Model



Class Diagram

Figure 3: Design Class Diagram

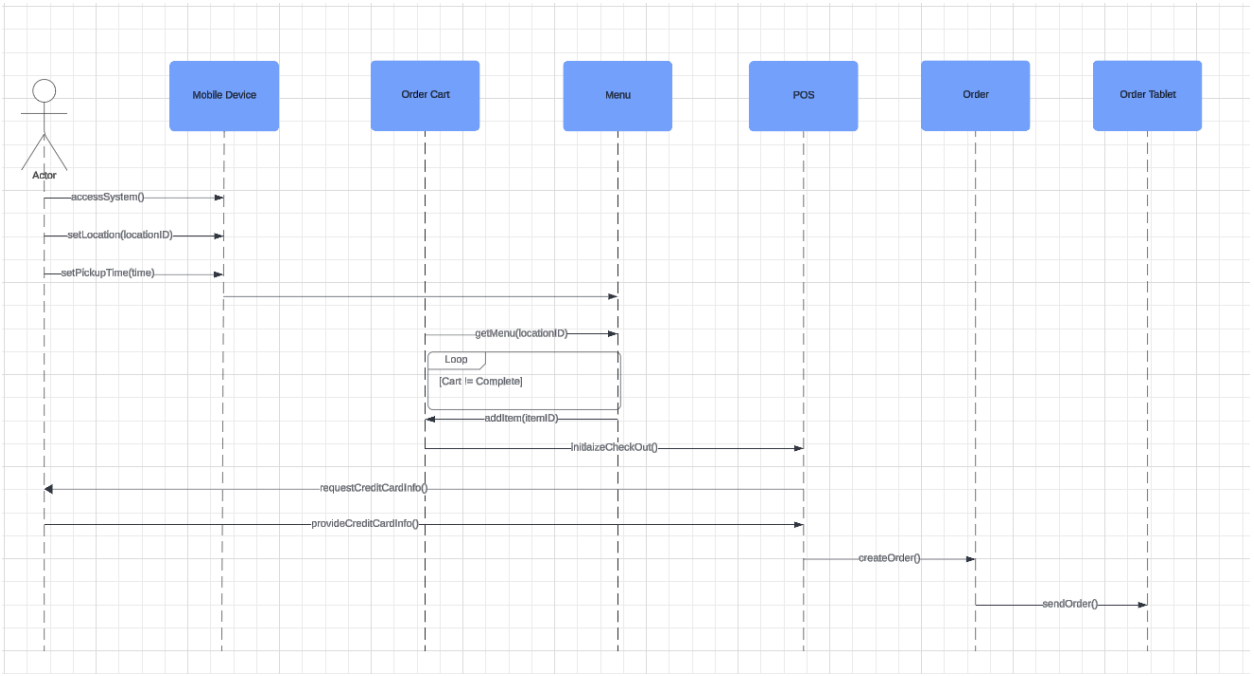


Sequence Diagrams

Use Case Scenario 1 - Place Order

1. Customer accesses system through mobile device
2. Customer selects New Order
3. Customer selects location
4. Customer selects pickup time
5. Customer is directed to location menu
6. Customer selects menu item
7. Customer adds menu item to cart
8. Customer repeats steps 6 and 7 until satisfied with their cart
9. Customer selects checkout
10. Total in calculated and displayed to the customer
11. Customer inputs their credit card information
12. Customer's card is charged
13. Order details are sent to the location

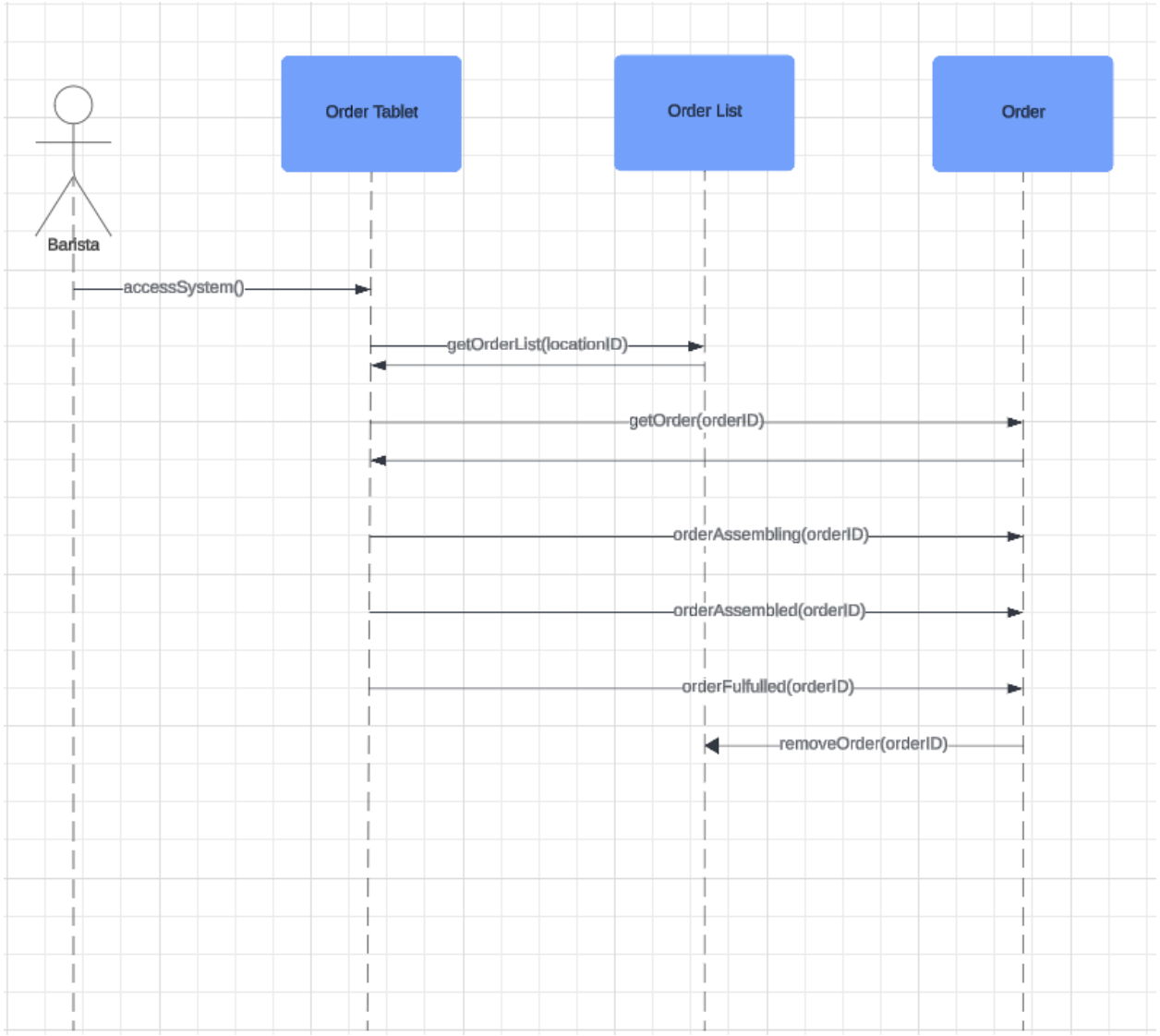
Figure 4: Sequence Diagram - Use Case 1



Use Case Scenario 2 - Fulfill Order

1. Barisa accesses system through orders tablet
2. Barista selects order based on pickup time
3. Order details are displayed to the barista
4. Barista marks the order as “assembling”
5. Barisa assembles the order
6. Barista marks order as “assembled”
7. Customer arrives at fulfillment window
8. Barista provides customer with their assembled order
9. Barista marks the order as “fulfilled”
10. Order is removed from Order List

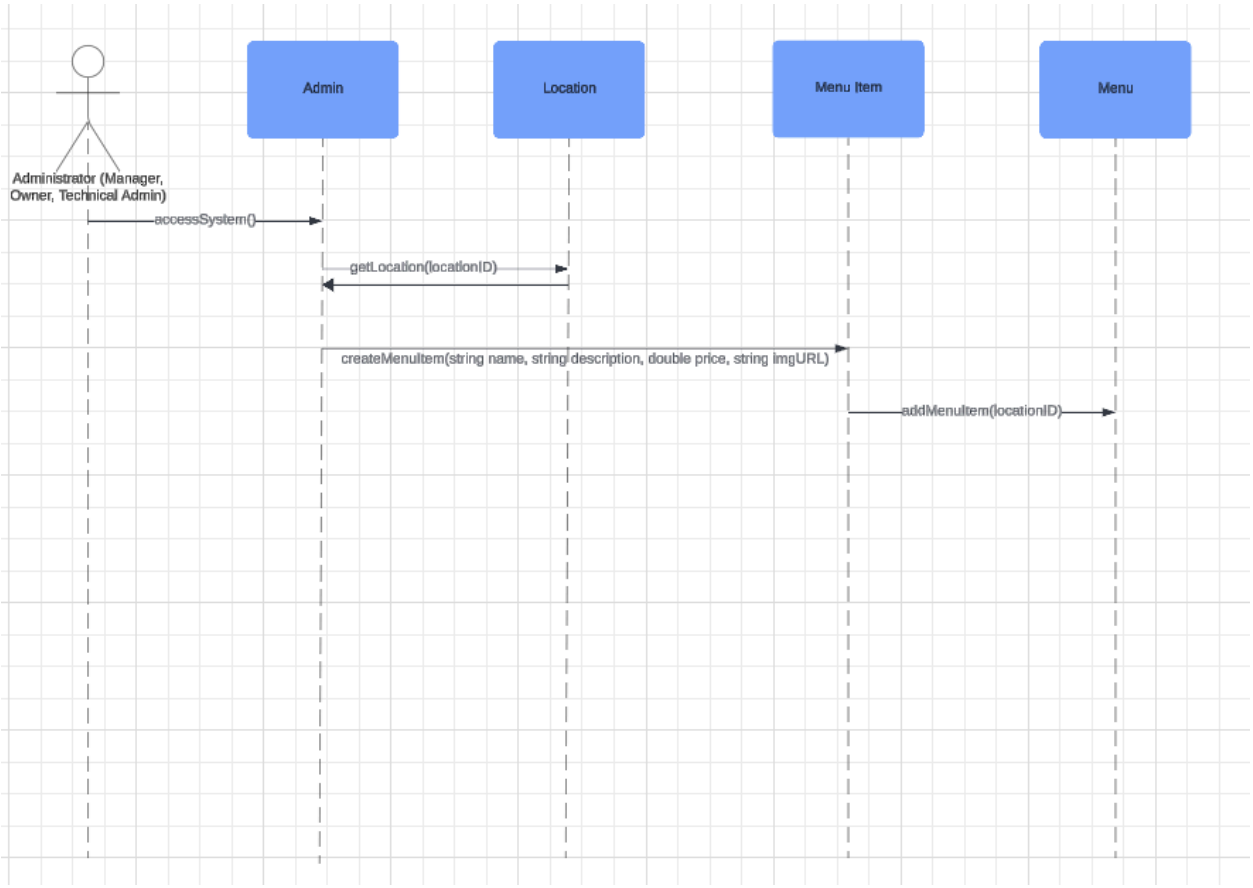
Figure 5: Sequence Diagram - Fulfill Order



Use Case Scenario 3 - Add Menu Item

1. Administrator logs in to system backend
2. Administrator selects location
3. Administrator selects add menu item
4. System provides new menu item form
5. Administrator inputs menu item name, menu item price, menu item description, menu item thumbnail
6. Administrator submits new menu item

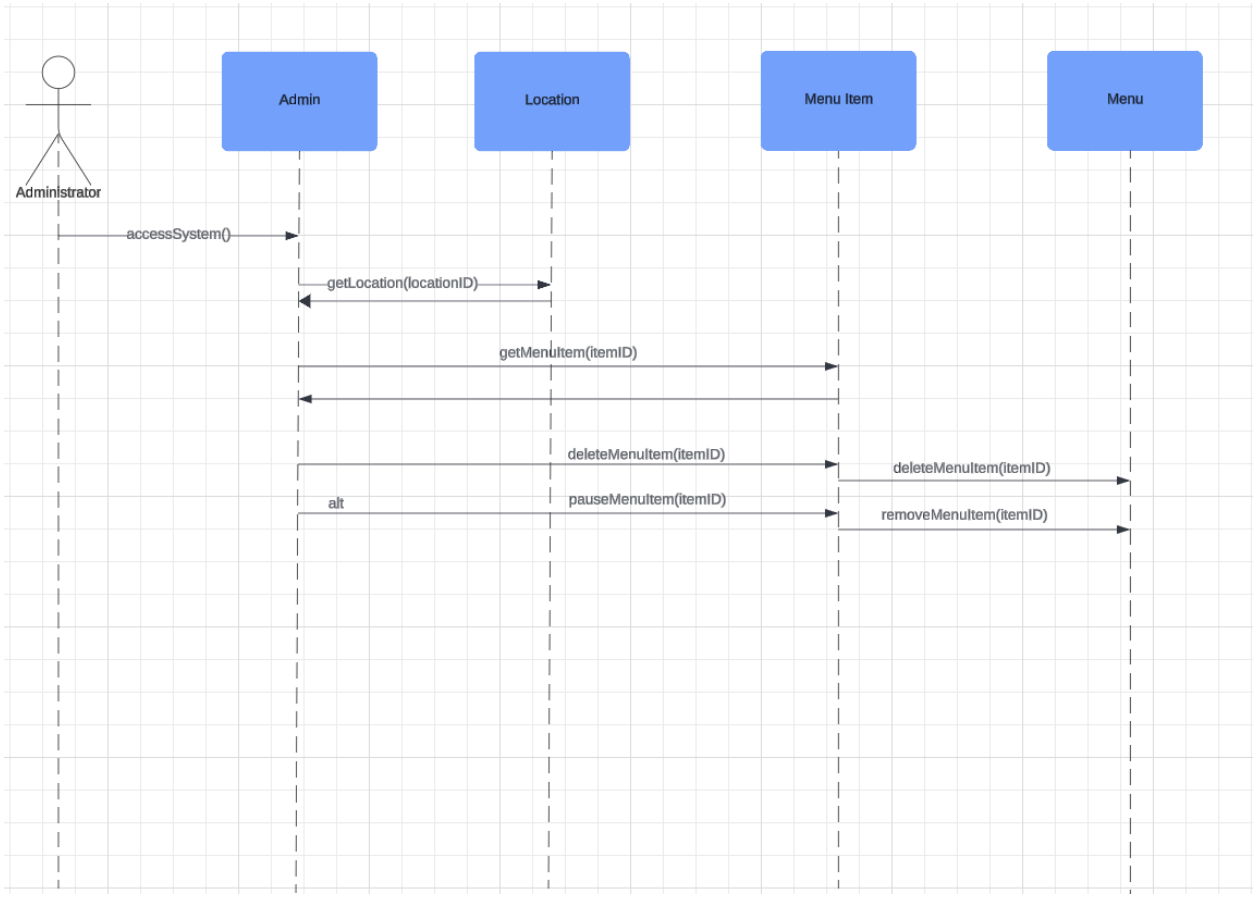
Figure 6: Sequence Diagram - Add Menu Item



Use Case Scenario 4 - Remove/Pause Menu Item

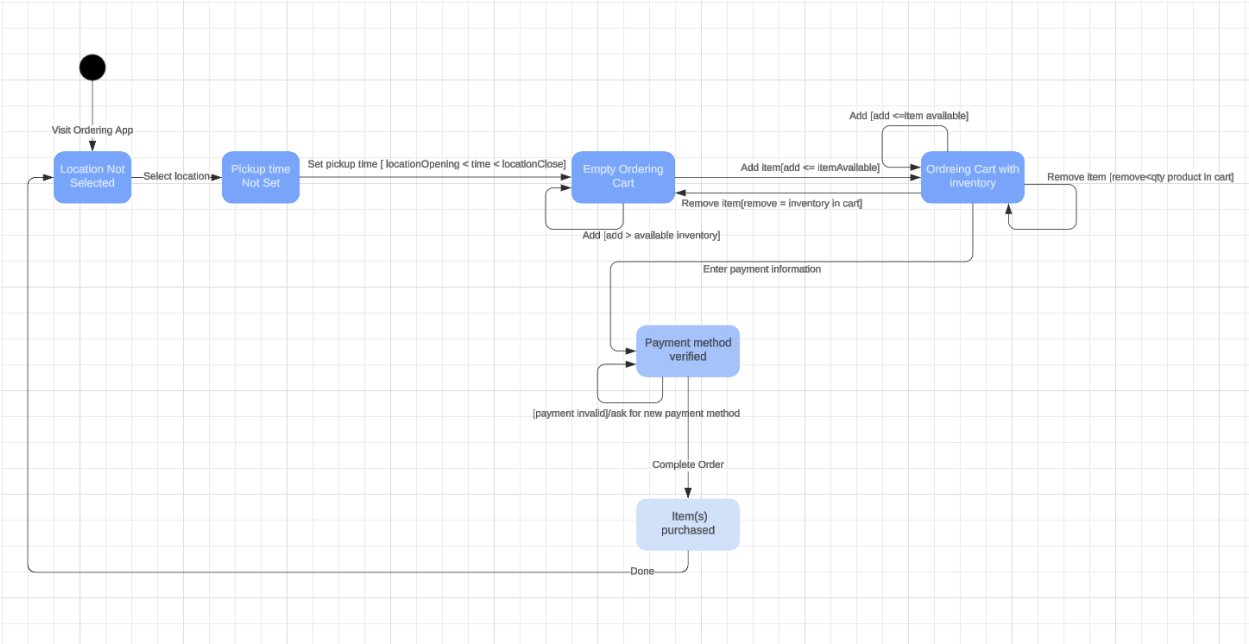
1. Administrator logs into system backend
2. Administrator selects location
3. Administrator accesses location menu
4. Administrator selects a menu item
5. Administrator selects remove item
6. System deletes menu item
7. Administrator selects pause item
8. System removes the menu item from the menu, does not delete the item in the database.

Figure 7: Sequence Diagram - Add Menu Item



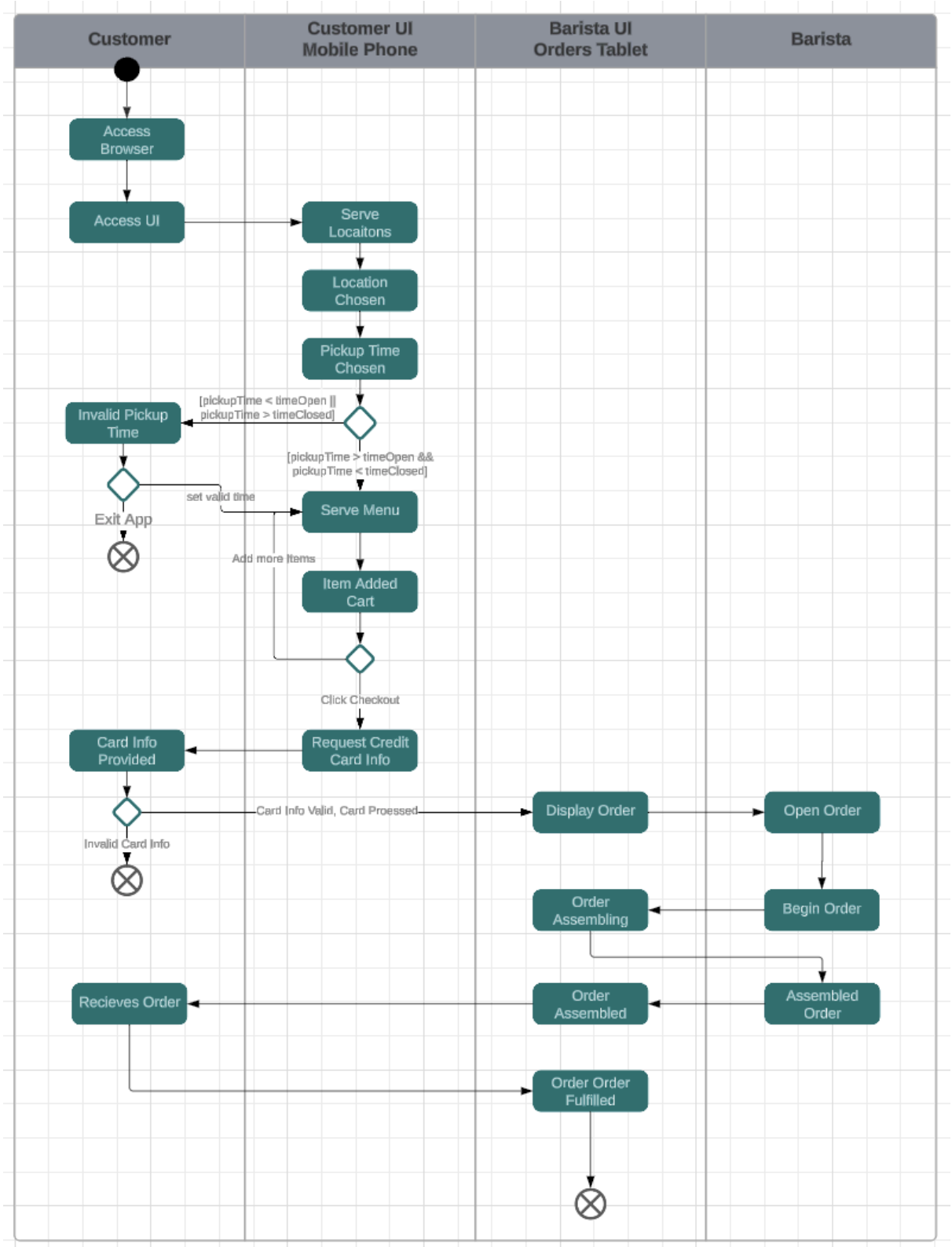
State Diagram

Figure 8: State Diagram - Use Case 1



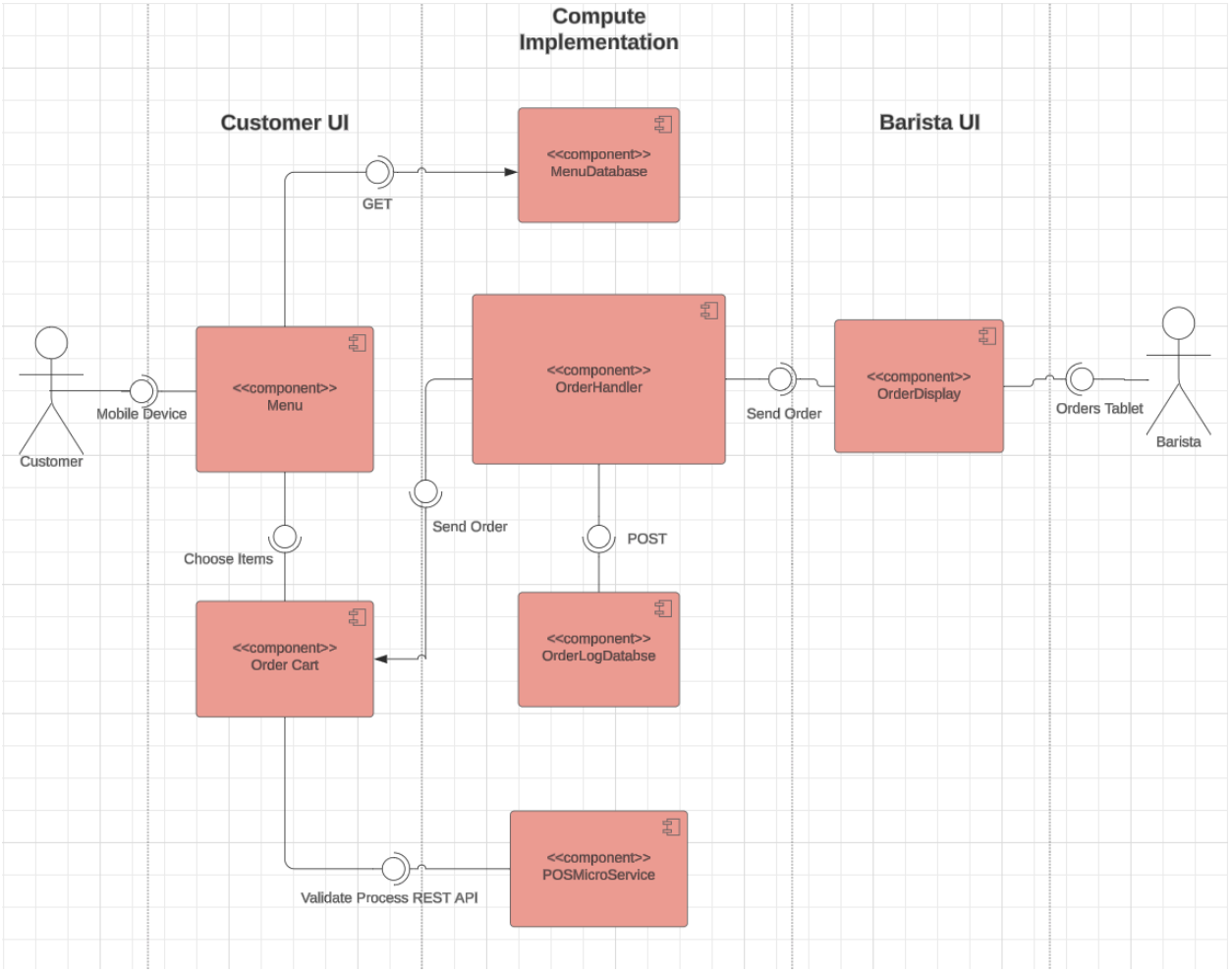
Activity Diagram

Figure 9: Activity Diagram - Use Case 1 & Use Case 2



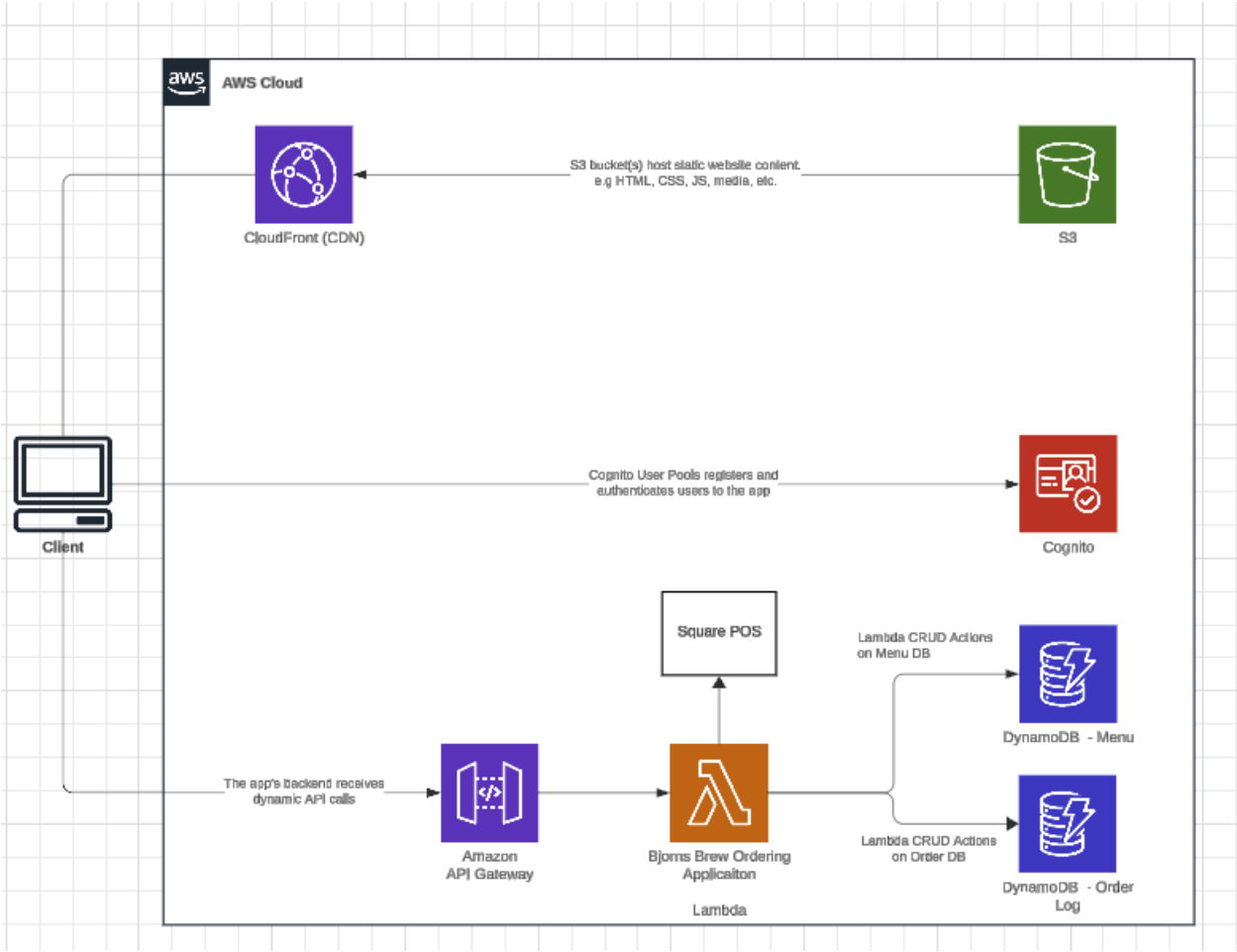
Component Diagram

Figure 10: Component Diagram



Deployment Diagram

Figure 11: Deployment Diagram



Decisions and Rationale

The Bjorn's Brew Ordering App will be deployed on an AWS Serverless Hosting Environment. This decision was made for several reasons:

- The lightweight nature and simple functionality of this application lends itself to a serverless architecture.
- This application is event driven. The functionality of the application will be able to be dissolved into functions triggered by events. (ie, New Order).
- AWS provides a multitude of Microservices that can be used to strengthen the application.
- Being a small chain of coffee shops, Bjorn's Brew would like to keep their operational costs as low as possible in terms of the ordering application. Using a serverless architecture will allow for "pay-per-use" of functions. Meaning more functions used means more orders coming in. This will allow for ease of scalability.

Microservices Used

- AWS S3 Bucket will host static web application content
- AWS CloudFront will serve static content and leverage caching for faster performance.
- Amazon Cognito will allow for user authentication, management of different user types within the application, Admin, Customer, Barista, Owner.
- Amazon API Gateway will send and receive dynamic content, menus, order cart, new orders, order state updates.
- Amazon Lambda will manage CRUD operations handling the ordering of items, and the displaying of orders.
- DynamoDB will provide a data store for Store, Location, Menu data.
- DynamoDB will also provide a data store for Order data.
- Square POS will be implemented for its ease of integration with AWS, high level of support and adoption, and current use as Bjorn’s brick and mortar POS syste.

Skeleton Classes

Data Models

```
public class Location{
	public int ID;
	public String name;
	public String address;
	public Time timeOpen;
	public Time timeClosed;
	public Menu menu;
}

public class Menu{
	public <List> MenuItem items;

	public void addMenuItemCart(int itemID){}
}


public class MenuItem{
	public int itemID;
	public String itemName;
	public String itemDesc;
	public String itemThumbnailUrl;
	pubic double itemPrice;
}

public class Order{
	public int id;
	public OrderDetails details;

	public void postOrder(int orderID){}
}


public class OrderDetails{
	public int orderID;
	public String customerName;
	public Time orderPlacedAt;
	public Time orderPickupTime;
	public String orderLocation;
	public <List>OrderLineItem orderItems;
	public double total;
}

public class OrderLineItem{
	public int itemID;
	public String itemName;
	public String itemDesc;
	public String itemThumbnailUrl;
	pubic double itemPrice;
}
```

Interfaces

```
public class OrderDispay{
	public <List> Order orders;

	public void getOrder( int orderID){}
	public void setOrderAssembling(int orderID){}
	public void setOrderAssembed(int orderID){}
	public void setOrderFulfilled(int orderID){}
}

public class CustomerUI{
	public int ViewPortWidth;
	public int ViewPortHieght;
	public String browser;

	public void getLocation(locationID){}
	public void getMenu(locationID){}
}

public class CustomerUI{

	public void addMenuItem( int locationID, String itemName, String itemDesc, String itemThumbnailURL, double itemPrice)
	public void removeMenuItem( int itemID)
	public void pauseMenuItem( int itemID)
	public void editLocationHours (int locationID)
	public void updateLocationHours (Time timeOpen, Time timeClosed)
}
```