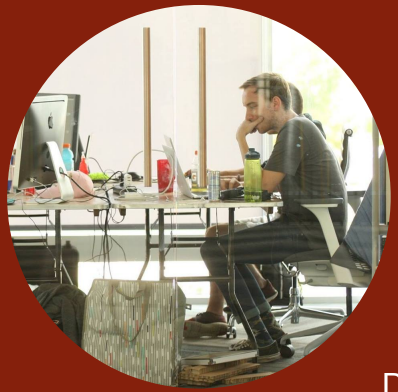
# EXPRESSO


Group: may1734
Email: may1734@iastate.edu
Website: may1734.sd.ece.iastate.edu
Advisor: Bill Adamowski

# The Team

Jonny Krysh



Lucas Collins

Jake Long

Derek Yu

Garret Meier

# Overview

1. The Problem
2. Solution Overview
3. Requirements Overview
4. Design Overview
5. Potential Risks
6. Status and Moving Forward

# Problem

- Ecommerce platforms can be difficult and expensive.
- Coffee roasters run on small margins.
- So, roasters aren’t able to sell their coffee online or are left with subpar solutions

# Existing Solutions

- High-profile coffee shops with custom software
- Expensive, luxury coffee distributors
- Narrow options and high barrier to entry

# Proposed Solution - High Level

## Customer

- Create an account
- Choose subscription from coffee bean types and local coffee providers.
- Choose time interval and quantity of subscription.

## Provider

- Receive empty boxes
- Keep inventory up-to-date.
- View and fulfill orders (including shipping them).

# Solution Decision 1: Decentralized Shipping

## Decentralized

- Beans will be fresher when shipped
- Saves time and money by not needing to ship to middle-man warehouse

VS

## Centralized

- Easier to track and ensure successful delivery of beans
- Easier to maintain inventory

# Solution Decision 2: Microservice Architecture

## Microservice

- Easily Scalable
- Easier to deploy
- Easier to split work between team members

VS

## Monolithic

- Generally one programming language and a small number of moving parts
- May be less difficult to code if code is not intended to be too big to maintain in the future

# Solution Decision 3: Web application

## Web App

- Developer benefits by removing necessity of mobile-first design patterns
- Customer would likely never use a mobile application.

VS

## Web App + Mobile App

- Provider can more easily track inventory using our application on a tablet or phone

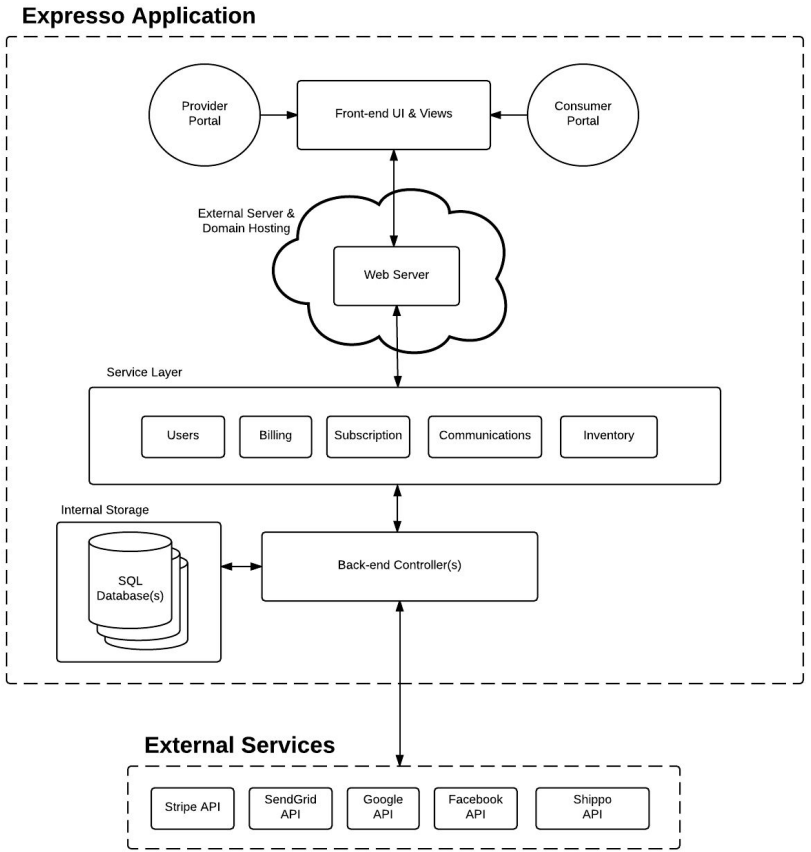
# Functional Requirements

- Providers
  - Post coffee beans
  - View orders
  - Receive payments
- Consumers
  - Subscribe to providers
  - Send payments
  - Set preferences for coffee types
  - View orders
- Both
  - Secure log in

# Acceptance Criteria

- Inventory
  - Track available and anticipated
  - Predicts future demand
- Delivery
  - Track orders
- Communication
  - Keep customers and providers informed
- Data Analysis
  - Personalize customer’s experience
- Billing
  - Send payments to providers
  - Receive payments from customers

# System Design

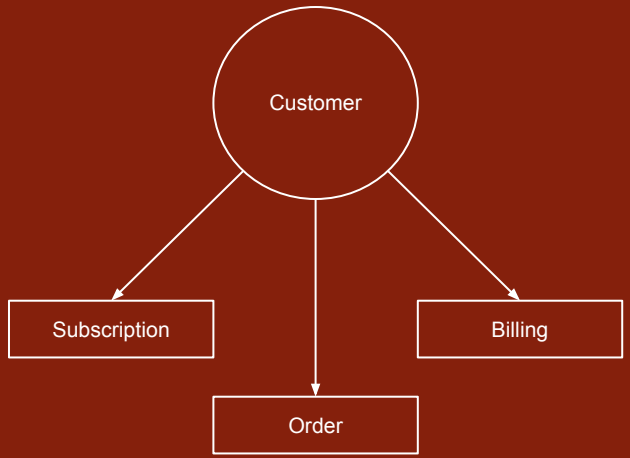


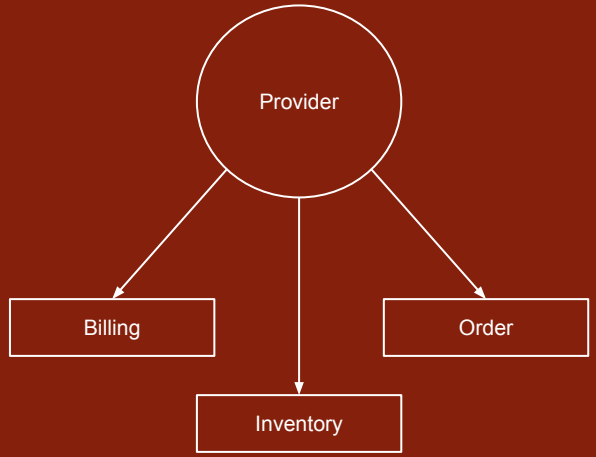
# Technology Used

- Frontend
  - ReactJS and Redux
  - Node.js as basic web server
- Backend
  - Golang for microservices
  - MySQL for database
  - RabbitMQ for queued services
- DevOps
  - DigitalOcean droplets
  - Docker for Orchestration

# User Service

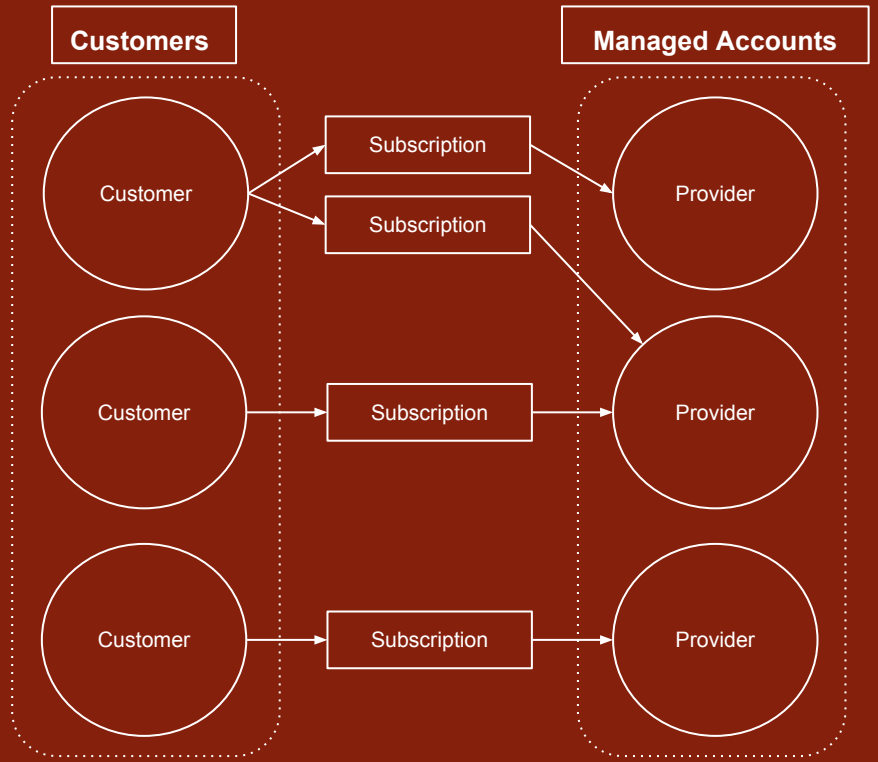
- Register and update consumers and providers
- Holds references to other services



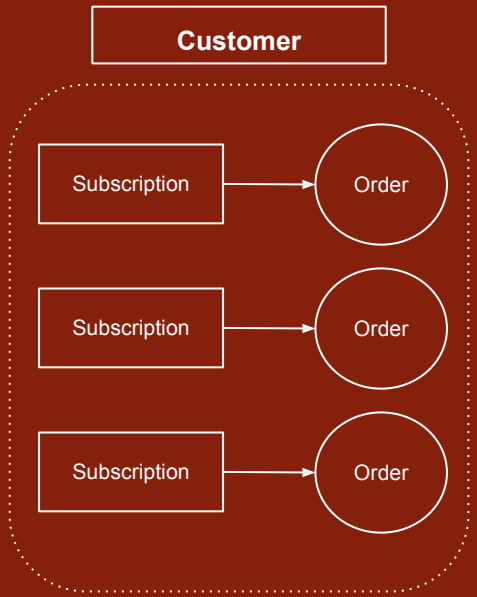
# Billing Service

- Utilizes Stripe's "Connect" API
  - Allows storage of Subscriptions to Providers by Customers
  - Unique key generation for every Customer and Provider
- Split into two main functions:
  1. **Customers** pay for **Subscriptions**
  2. **Providers** receive payments from **Subscriptions**
- Stripe Managed Accounts
  - Maintained programmatically by Expresso
  - Handle payments to Providers

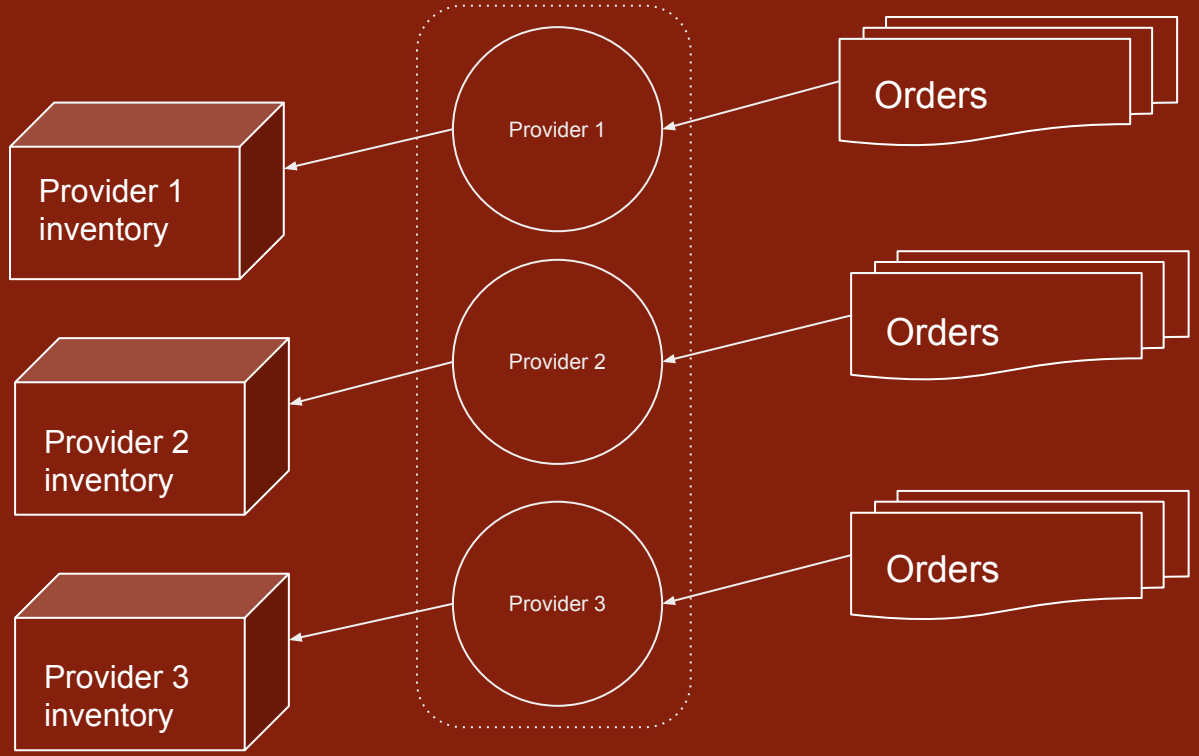


# Subscription Service

- Manages a customer's subscription(s)
- Each subscription associated with an order
- Order contains coffee information



# Inventory and Orders Service



# Inventory and Orders Service (Goals/Uses)

## Inventory

- Each provider will keep their inventory up-to-date.
- This will include their roasting schedule(s).
- The Web App will reference each inventory to keep from overselling product.
- Includes packing inventory

## Orders

- When a subscription is due to be fulfilled, an order will be created.
- Each provider will be notified of a set of orders to be fulfilled and shipped off.
- Providers will signal when they have shipped off an order.

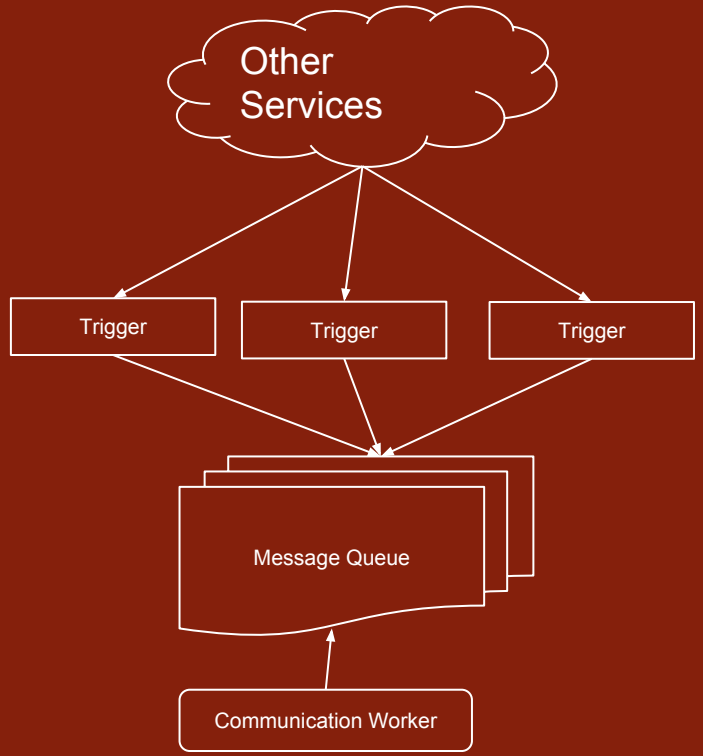
# Communication Service

## Goals:

- Send messages quickly and with minimal coupling to other services
- Be open for scaling to additional types of communication

## Solution:

- Use queued messages and triggers for communications



# Potential Risks

- Partnering with Local Providers
- Logistics
- Handling Sensitive Information
- Determining Cost

# Status and Moving Forward

- Individual Repos for each service
- Github Project for high level goals/deadlines
- Github Issues for individual bugs/features
- One Service owner, multiple contributors
    - Pull requests and code review for additions

# Questions?

# Testing Strategy

- Gofmt & Golint for code quality
- Golang’s built-in mock library for unit testing the microservices
- TravisCI for integration testing
- Xo (javascript linting) for consistent styling
- Alpha testing with roaster and customer involvement

# Timeline


15-Nov
29-Nov
13-Dec
27-Dec
10-Jan
24-Jan
7-Feb
21-Feb
7-Mar
21-Mar
4-Apr
18-Apr
2-May
16-May
Base Model (V1)
MVP (V2)
Client Verification
End-to-End Testing
Additional Features, stretch goals (V3)
Final Product (V4)
Client Presentations
Final Presentation of Product

# Market Research

**Bean Box** (https://beanbox.co/) - 21 of Seattle's roasters to distribute their coffee in consolidated boxes to Consumers monthly.

Fresh roasts delivered with free shipping and  high quality roasts

Lacks in customizability and cost effectiveness.

**Blue Bottle Coffee** (https://bluebottlecoffee.com/) - They run their entire vertical from sourcing beans through roasting and distribution.

Highest quality beans and mid-level prices,

Llacks variety and customizability.

**Counter Culture Coffee** (https://counterculturecoffee.com/) - complete vertical from purchasing beans to shipping the roasted coffees.

Customizable roasts and subscription offerings.

Not a solution for local roasters

**Ritual Coffee Roasters** (https://www.ritualroasters.com) - uses an external service called Shopify to handle their eCommerce.

Higher than almost all other options.

Lack the personalization of Counter Culture of Blue Bottle., who host their own services.