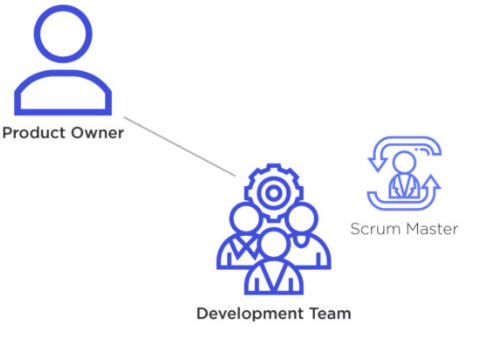
Dan’s Bagels

# Project Overview

This project aims to build a system for operating a bagel shop. The system will be used to run the storefront of the bagel shop for both the customers and employees. It is the only way in which customers will place and manage orders, and where customers will manage their funds. The system will also be used by employees to view orders, change order status, and manage inventory. It will be capable of running on web browsers on a PC.

# Team Organization

The team plans to use an agile generalist team structure. This allows a small team where each member has more cross-functionality rather than overly specific roles or titles. The members of the team will contribute to the project in many ways rather than being specialists in specific areas. Egoless programming is also planned to be followed due to every member playing a wide role. Dan Watson is the client. All members are part of the development team, with Jake Pope being the scrum master to begin the project. These roles are subject to change as the project moves on.



Client

# Software Development Process

The development will be broken up into five phases. Each phase will be a little like a Sprint in an Agile method and a little like an iteration in a Spiral process. Specifically, each phase will be like a Sprint, in that work to be done will be organized into small tasks, placed into a “backlog”, and prioritized. Then, using on time-box scheduling, the team will decide which tasks the phase (Sprint) will address. The team will use a Scrum Board to keep track of tasks in the backlog, those that will be part of the current Sprint, those in progress, and those that are done.

Each phase will also be a little like an iteration in a Spiral process, in that each phase will include some risk analysis and that any development activity (requirements capture, analysis, design, implementation, etc.) can be done during any phase. Early phases will focus on understanding (requirements capture and analysis) and subsequent phases will focus on design and implementation. Each phase will include a retrospective.

|  |  |
| --- | --- |
| **Phase** | **Iteration** |
| 1. | Phase 1 - Requirements Capture |
| 2. | Phase 2 - Analysis, Architectural, UI, and DB Design |
| 3 | Phase 3 - Implementation, and Unit Testing |
| 4 | Phase 4 - More Implementation and Testing |

We will use Unified Modeling Language (UML) to document user goals, structural concepts, component interactions, and behaviors.

# Communication policies, procedures, and tools

The team will communicate frequently using the following tools:

* Zoom
* Project Team Group Message

Zoom will be used for longer meetings, and the group message should be used for short updates, reminders, and questions. All group members are expected to attend all scheduled meetings unless otherwise specified. The same zoom information will be used unless otherwise specified.

# Risk Analysis

**Only one storefront needs to be managed and the website will correspond with the physical workplace.**

There is a very high likelihood of this requirement being met. If this requirement is not met, the severity of the consequences would be extreme. The consequences of failing would be a failed project. There are no workarounds.

**Store inventory should be editable such that goods used to create bagels can be added as they are physically stocked and removed as they are used to create meals and sold to customers.**

There is a high likelihood of this requirement being met. The severity of this requirement not being met is high. The consequences would be not being able to tell customers when items are out of stock. A workaround would be to manually mark items as out of stock on the menu.

**Facilitate the steps that take place in the purchase of a bagel.**

There is a very high likelihood of this requirement being met. The severity of this requirement not being met is extreme. The consequence of failure would be the failure of the whole projects. There are no workarounds for this requirement.

**Customers can see order status after placing order.**

There is a high likelihood of this requirement being met. The severity of this requirement not being met is high. The consequence of failure would be not having a system where multiple actors are able to access order info. A workaround would be only showing when orders or complete, or just to give a fixed estimated wait time.

**Send an email to the customer when their order is ready.**

There is a low likelihood of this requirement being met. The severity of this requirement not being met is very low. There are no real consequences of not meeting this requirement, especially if the requirement of updating order status is met. A workaround would be only having the status updated without any notification.

**Menu to order from should be dynamic and updatable.**

There is a high likelihood of this requirement being met. The severity of this requirement not being met is high. A workaround would be editing the menu as a developer every time you want to change the items.

**Be able to handle orders being ready within five minutes.**

There is a high likelihood of this requirement is being met. The severity of this requirement not being is moderate. A workaround would be putting all orders in a queue and letting the chef choose when to start an order.

**Support different types of users of the system.**

There is a very high likelihood of this requirement being met. The severity of this requirement not being met is very high, as it would make the system insecure and hard to manage. A partial workaround would be only having an employee account and a customer account.

**Accounts have access to relevant information for their role.**

There is a very high likelihood of this requirement being met. The severity of this requirement not being met is extreme, as it would greatly affect security. There are no workarounds.

**Customer accounts can order, cancel orders, create an account, add money, and view order status.**

There is a very high likelihood of this requirement being met. The severity of failure is very high, as this is the basis of the whole user experience. There are no workarounds.

**Bagel chef accounts can see orders, update inventory, and update order status.**

There is a very high likelihood of this requirement being met. The severity of failure is somewhat high, as this requirement is closely connected with other requirements. A workaround would be only meeting one of these requirements, such as being able to update order status.

**Cashier accounts can view order status, charge the customer account, and discard abandoned orders.**

There is a very high likelihood of this requirement being met. The severity of failure is somewhat high, as this requirement is closely connected with other requirements. There are no real workarounds for this requirement because this is the entire purpose of the cashier role.

**Manager accounts can approve account creation, edit inventory, and order inventory.**

There is a very high likelihood of this requirement being met. The severity of failure is somewhat high, as this requirement is closely connected with other requirements. There are no real workarounds for this requirement because this is the entire purpose of the manager role.

**The system should store names, contact information, account balance, identification number, password, and additional data.**

There is a very high likelihood of this requirement being met. The severity of failure is very high, as this involves the security of the system. There are no workarounds

**The systems should be accessible through a web browser.**

There is an extremely high likelihood of this requirement being met. The severity of failing to meet this requirement is extremely high because it is the format requested by the client. A workaround would be using a different platform, but it is not preferred.

**The system should record analytics about the site and system.**

There is a low likelihood of this requirement being made. The severity of failing to meet the requirement is low. A workaround would be letting analytics about purchases be made manually by staff.

# Configuration Management

See the README.md in the Git repository.