Dan’s Bagel Shop - FooBar

# **Project Overview/Summary**

Dan’s Bagel Shop has hired us; FooBar, to build a web based application that will be used to operate a soon to open bagel store. There will be a single store location which will prepare and distribute bagels to eager customers. Our task is to develop the aforementioned application, such that it meets the specifications provided to us by the client (Dan from Dan’s Bagel Shop).

Dan’s Bagel Shop would like to make the process of placing, obtaining, and paying for bagel orders a quick, easy, and convenient process. Customers will be able to place orders online or through kiosks at the store. When orders are placed they will be sent to cooks who will prepare the order using the application to access inventory. When an order is ready to be picked up, the application will notify the customer. When the customer arrives a cashier will use the application to dispense the order and charge the customers Dan’s Bagel account. If the customer enjoys their bagel the application will make reordering quick and easy.

The application will also help the business run smoothly. A database will store account information, orders, and inventory. There will be Manager, Customer, Chef and Cashier accounts. The application will track inventory as well as sales information. Information regarding a customer rewards program will also be tracked. The Manger will be able to generate reports, view analytics, and track inventory information.

# **Team Organization**

Team Philosophies:

Don’t be afraid to ask questions

Egoless Programming

If you believe there is a better way, bring it up in discord or a team meeting

|  |  |
| --- | --- |
| Member Name | Roles (Subject to change) |
| Braysen Goodwin | Back End Developer, Back End Code Reviewer |
| Nathan Johnson | Back End Developer |
| Jared Scott | Front End Developer, Front End Code Reviewer |
| Peter Allen | Front End Developer |

# **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**

See the README.md in the Git repository.

# **Configuration Management**

See the README.md in the Git repository.

# **Team Planning**

We will be using an agile development method for this project. We will have sprint planning every Monday after class between 2:30pm-3:00pm to determine tasks for the week, and standup/scrum on Wednesday and Friday after class between 2:30pm and 3:00pm.

# 

# 

# **Risk Analysis**

In case we need to do a full rundown of Risk Analysis for a specific requirement below is the template we will follow.

###### Task: Base Template

Likelihood: (Will it be completed)

Severity: (how important)

Consequences: (if not completed what does it affect)

Work-around: (Ways that can make up for this loss if any)

Difficulty:

Impact:

Pros:

Cons:

###### Task: Make a Web Based Application

Likelihood: High

Severity: High

Consequences: Failure to fulfil contract

Work-around: None

###### Task: Make an Account

Likelihood: High

Severity: High

Consequences: Failure to fulfil contract

Work-around: Make a guest system so customers don’t need accounts.

###### Task: Make an Inventory Database

Likelihood: High

Severity: medium (it is an aspect that Dan’s Bagel shop wants but they can manage without.)

Consequences: Failure to fulfil contract

Work-around: Shop continues with a paper system.

###### Task: Make an Account Database

Likelihood: High

Severity: medium

Consequences: Failure to fulfil contract

Work-around: all users use a guest account to order.

###### Task: Manager can view and change databases

Likelihood: High

Severity: medium

Consequences: Changes would need to be forwarded to us or another IT company to change databases

Work-around: None

###### Task: Orders will deduct from Inventory Database and user Money

Likelihood: High

Severity: medium

Consequences: inventory will not be accurate to menu

Work-around: website menu will need to be adjusted as inventory changes

###### Task: User can create and edit own Account details

Likelihood: High

Severity: medium

Consequences: changes will need to be submitted through the shop

Work-around: changes can be submitted to the company

###### Task: Rewards program implementation

Likelihood: medium

Severity: low

Consequences: No Rewards system will be in place on the database

Work-around: Paper punch card system can be in place if Rewards program is wanted.

###### Task: Manager can change account details.

Likelihood: High

Severity: High

Consequences: Managing employees will go to whoever can change the databases

Work-around: all changes will need to be done manually in the database files. Or there will be employee accounts that employees sign into to manage their job rather than personal accounts.

###### Task: Notifications displayed for Orders

Likelihood: High

Severity: Medium

Consequences: Customers won’t be notified and pick up may be hard

Work-around: have estimated wait times displayed.

###### Task: Customers can order 2 days in advance.

Likelihood: Medium

Severity: low

Consequences: Orders can be set at any time if the system doesn't say otherwise.

Work-around: Have policies in place to protect the company in case of software or hardware failure to fulfil orders that are far out in time.

###### Task: Customers can cancel order

Likelihood: Medium

Severity: Medium

Consequences: Customers may call in holding up the automated system.

Work-around: Customers will have to call in to cancel an order.

###### Task: Customers can save orders for future use

Likelihood: Medium

Severity: low

Consequences: Customers will have to manual do all orders.

Work-around: Have premade orders for all users to select.

###### Task: Menu is adjusted for most popular

Likelihood: low

Severity: low

Consequences: None

Work-around: manually adjust the menu to what wants or should be sold at top.

###### Task: Suggestion or comment box is available for managers to see and customers to leave

Likelihood: low

Severity: low

Consequences: Customers can't provide feedback

Work-around: feedback slips provided in store.

###### Task: Inventory is marked as expired when the item is expired.

Likelihood: low

Severity: low

Consequences: Employees will have to manually input in the system changes to inventory.

Work-around: easy made deduction system to match the actual in shop inventory.