Requirements Definition

*Dan’s Bagel Shop*

**Introduction and Context**

Currently orders can only be made in person at Dan’s Bagel Shop. With an online store Dan’s Bagel Shop will be more accessible, attracting a bigger consumer base. With a bigger consumer base there needs to be better tools for handling the workflow and inventory of Dan’s Bagel Shop. This project would provide tools for management and employees and provide an online storefront for customers to make purchases.

For store employees the project would keep track of orders to ensure they’re ready in time for the customers. Management will be able to view the store’s inventory anywhere, and order ingredients as needed. Customers will be able to place orders online and track their progress. They will also have access to their order history so they can easily order their favorite bagels.

**Users and their Goals**

This program contains four main users which are customers, chef, cashier, and manager. Below are some scenarios that outline these four users’ main goal(s). Refer to the use case diagrams for more info.

**Scenario 1**

1. A new customer registers for a new account providing their email, full name, and a password.
2. The customer then navigates to the menu page and chooses an item to order
3. If needed the customer adds funds to their account before finalizing order
4. The customer specifies when they want the bagel
5. Customer drives down and retrieves order from cashier at shop

**Scenario 2**

1. Chef arrives at the start of their shift and logs into their account
2. Chef views incoming orders and starts preparing whatever items were ordered
3. Chef updates order status accordingly and places order in pickup area when finished

**Scenario 3**

1. Cashier logs into account and looks at orders and which customer it is associated with.
2. As customers arrive to pick up orders, the cashier asks customers for their account id for verification.
3. After the customer leaves with their meal the cashier marks the order as completed.
4. Cashier throws away orders that haven’t been picked up in 30 minutes.

**Scenario 4**

1. Manager logs into account to view inventory status.
2. Manager places an order to replenish inventory based off analytics tools.
3. Manager creates new employee accounts for any new hires.

**Functional Requirements**

Most of the functional requirements for this project relate to what user accounts can do. The first item on the list defines what all accounts should be able to do. Items two through five define functions specific to a certain account type. There are four account types in our program which are: customer, chef, cashier, and admin(owner). The last items just detail how certain parts of the online interface will look.

1. **All Account Details**
   1. Each account must have an Email associated with it
   2. A first and last name must be provided for the account
   3. Passwords will be required
   4. All accounts will have a unique numeric ID
   5. Users will be able to change account info
      1. Passwords, name, and email
2. **Customer Account Details**
   1. $100 automatically added upon account creation
   2. Customers will be able to add funds to their account balance
   3. The ability to place orders online
   4. View order history
      1. Reorder from order history
   5. View order status
      1. Orders may be canceled until bagel status is “Ready”
      2. Orders may **not** be changed once placed.
3. **Chef Account Details**
   1. Will be able to see incoming orders
      1. Orders will be organized by pick up time
      2. The chef will be able to update the order status
      3. Orders will show up no later than 10 minutes before it needs to be ready
   2. Chefs will be able to update the inventory
      1. remove items that have been used for an order
      2. remove items that have spoiled
4. **Cashier Account Details**
   1. Be able to lookup customer’s numeric ID
   2. See list of orders and which customers they are associated with
5. **Admin (Owner) Account Details**
   1. Add and remove employee accounts
   2. Update the menu
   3. Order inventory
6. **Menu display**
   1. Able to be sorted by most popular items
      1. each item on the menu must keep track of how many times it was order in the last month
   2. Able to be sorted by price
   3. Default menu presentation will be sorted alphabetically within categories
      1. Menu categories are bagels, spreads, sandwiches, and beverages
   4. No premade combos, all items are sold individually
7. **Orders**
   1. Pick up times for an order may be specified
      1. Orders can only be placed one week in advance
      2. Orders without a specified pickup time will have a pickup time of when the order was placed plus 10 minutes
   2. Orders not picked up within 30 minutes of being ready are discarded
8. **Inventory**
   1. Items ordered for are the same as items sold on the menu
   2. Each item can be ordered individually
   3. Provide an analytics screen
      1. See what items need to be ordered
      2. See the sale figures of each item
9. **Existing Customers**
   1. Integrate existing customers into the new system
      1. Waiting for details, currently unknown how this existing database looks

**Non-functional Requirements**

This product will only be available as a website

1. Both front and back ends of the website will be developed with the Django framework
2. Meetings will be held Every Monday Wednesday, and Friday to review team progress
3. Goal milestones will be hammered out in high-level design
4. Team communication will occur in discord
5. Google drive will be used to store documents that the team worked on concurrently
6. A GitHub repository has been setup to keep track of code changes

**Future Features**

In the future we could develop a phone app which would allow customers to order from their phones. This would also have the added benefit of us being able to send them updates about their orders.

**Glossary**

Discord- a free instant messaging and VoIP application used for a wide variety of activities.

Django- a Python-based free and open-sourced web framework that follows the model-view template architectural pattern.