SRS Documentation

Bloom & Brew Café

Group A- Brooke Spangler, Camila Urena, Kylee Grasela, Brittany Fetters

11-13-24

[**1.1 Introduction and Purpose**](#_sz9kkg7zcn2z) **2**

[**1.2**](#_h0q18i77u1fz) **Overview 2**

[**1.3 Use Cases**](#_kj525z8z82d) **2**

[**2.1 Project Scope**](#_j7z68isrzdhk) **3**

[**2.2**](#_6cppzingxd91) **Goals and Objectives:****3**

[**2.3 Constraints**](#_omjvn4sazt1d) **4**

[**3.1**](#_rq4cqo3qap7x)**Requirements 4**

[**3.2**](#_letab3oldwt) **Manager View 5**

[**3.3**](#_1mbx20hv50i) **Menu Management 5**

[**4**](#_qupxwrbflwrd) **System Design 6**

[**4.1**](#_xliabjvw1v1o) **Entities and Relationships 6**

[**4.2**](#_bdbbhs9sthg) **UML Diagrams 7, 8, 9**

[**5**](#_jiibx3grpx8d) **System Architecture 10**

[**5.1**](#_thkrja1bt9y3) **Architecture Overview 10**

[**5.2**](#_a1uisryr46j) **Component Design 10**

[**6**](#_tcjz1fgg3zwd) **Testing and Validation 11**

[**6.1**](#_ns22vqroqwrp) **Basic Test Cases 11, 12**

[**6.2**](#_83p5xwwx8gk3) **Ensuring System Correctly Displays Hardcoded Menu Items 13**

[**6.3**](#_7ulwcvj1dzxd) **Function Criteria 13, 14**

[**7**](#_ot4kzybw56pn) **More Functionalities 14**

[**8**](#_8qedjy4v9kpo) **Conclusion and Approval****14**, **15**

### 

### 

### 

### 

### 

### 

### 

### **1. Introduction**

### 1.1

• **Purpose of the Document:**

This document will clearly outline the requirements, scope, entities, attributes, and system design for a restaurant management project such as the Bloom and Brew Cafe.

### 1.2

• **Project Overview:**

The Bloom and Brew app provides a seamless, calming ordering experience by guiding users through a thoughtfully designed interface that reflects the café's values. Upon opening the app, customers are greeted by the gentle colors and organic visuals of the Bloom and Brew logo, setting a welcoming tone. Users can explore the menu via clearly labeled categories—like “Coffee & Tea,” “Bakery,” and “Smoothies”—or quickly reorder past favorites for convenience. Selecting an item brings up a customization screen where users can tailor their drinks by choosing milk types, flavors, and temperature, all with eco-friendly options like oat milk and minimal packaging. With easy add-ons like seasonal ingredients or an extra shot of espresso, each order is crafted to reflect both user preferences and Bloom and Brew’s commitment to sustainability. The app even allows customers to opt for eco-friendly packaging or skip disposable utensils, reinforcing the café’s value of Earth, and environmental values. Each step is designed to feel warm, natural, and simple, allowing customers to quickly place their order and focus on the experience of enjoying it.

### 1.3

• **Audience and Use Cases:**

**Primary Users:**

**Customers / Users:**

Browse menu, customize / place orders.

**Secondary Users:**

**Staff**:

Receive and prepare orders, rely on the app to track and take orders.

**Manager / Administrator** :

Monitor apps functionality, manage orders

**Stakeholders:**

**Café Owner** :

Monitor app performance, make sure it aligns with the café's branding.

**Tech Team**:

Building and maintaining the app, functionalities of the app.

**Marketing** :

Promoting the app to customers and handling customer feedback.

**2. Project Scope**

### 2.1

• **Define Scope:** Our system for Bloom and Brew will include customer order management, menu viewing and client-side database integration but will not include integration with complex billing or alternate forms of logins such as google login or linkedin.

### 2.2

• **Goals and Objectives:** The Bloom and Brew App will mainly focus on giving customers an easy platform to navigate the menu, customize and place orders. It will have features like personalizing orders, eco-friendly options, and loyalty rewards program. Customers can save their payment details and reorder favorites. Also it will have a secure payment process for smooth transactions.

### 2.3

• **Constraints**: The app will only be available in the U.S. and the user base will only be for local people. The app will not support any form of delivery whether it be third-party or in-house.

### **3. Requirements**

### 3.1

• **Features, Capabilities, and Behavioral Requirements:**

#### **Customer Order Placement**

**Feature Description:**

* Enables customers to browse the menu and place orders online or in-store.

Capabilities:

* Customers should be able to view the full menu, including item details (ingredients, prices, sizes, and customization options like "add whipped cream").
* A cart system should allow customers to review and modify selected items before checkout.
* Multiple payment options should be supported (credit/debit cards, digital wallets, and potentially cash for in-store orders).

**Data Requirements:**

* Input Data: Customer name, contact information (for order confirmations), and payment details.
* System Outputs: Order confirmation number, and digital receipt.

**Behavior:**

* On interaction, the system should display a clean, intuitive interface.
* Upon checkout, the system validates inputs (e.g., payment details, contact info) and confirms the order with a unique identifier.
* Real-time updates on order status, e.g., "in preparation" or "ready for pickup."

### **3.2 Manager View**

**Feature Description:**

* Provides managers with an interface to monitor, process, and manage orders in real-time.

**Capabilities:**

* Display all incoming orders in a dashboard format, categorized by status (e.g., "new," "in preparation," "completed").
* Ability to mark orders as complete, flag issues, and oversee staff workflow.
* Access to customer feedback and system analytics (e.g., sales trends, peak times).

**Data Requirements:**

* Input Data: Manager credentials for secure login; ability to input updates or comments on order statuses.
* System Outputs: Live updates on order status, inventory notifications (e.g., "low stock"), and summarized reports for business insights.

**Behavior:**

* Orders should appear in a time-sorted list, with color codes or filters for priority.
* Managers should receive alerts for delayed or flagged orders.
* Seamless integration with inventory systems to manage stock efficiently.

### **3.3 Menu management**

**Feature Description:**

* Allows managers to create, update, and delete menu items and categories, ensuring flexibility for seasonal or promotional items.

**Capabilities:**

* Adding new items, including images, detailed descriptions, price points, and availability times (e.g., “Available 8 AM – 11 AM” for breakfast items).
* Editing existing items to update ingredients, pricing, or availability.
* Temporarily marking items as "out of stock" with automated updates for customers placing orders.

**Data Requirements:**

* Input Data: Item name, description, pricing, category (e.g., "pastries," "beverages"), and stock status.
* System Outputs: Updated menu display for customers and internal systems.

**Behavior:**

* Changes should reflect instantly for both customer-facing and manager-facing systems.
* The interface should allow managers to preview the menu before publishing updates to ensure accuracy.
* Provide recommendations based on sales trends (e.g., suggesting popular items for promotions).

### **4. System Design**

### 4.1

### **Entities and Relationships**

### Bloom and Brew

| **Object** | **Attributes** | **Methods** |
| --- | --- | --- |
| Customers / Users:  Browse menu, customize / place orders. | Customer ID | AddItem(Item:Item) -Adds items to their cart. Can also add things to customize their drinks. |
| Manager / Administrator :  Monitor apps functionality, manage orders, | Manager ID | GetOrderDetails()-  This will give them details about their orders or subscription. |
| Employee/ workers:  Make the food/drinks: accept and confirm mobile orders. | Worker ID | RemoveItem(item:Item)- delete something from the customer's subscription that they don't want anymore |
| Order: the items they got, receipt. | Order Status ID | GetOrderDetails()- gives them the information on their mobile receipt, such as time it will be ready, price they paid, etc… |
| Payment: $5 month ( premium get rid of ads, access to more features, like choose your scammer) make most profit from social media. | Payment ID | CalculateTotalAmount()- Gives them the total amount and they pay online |

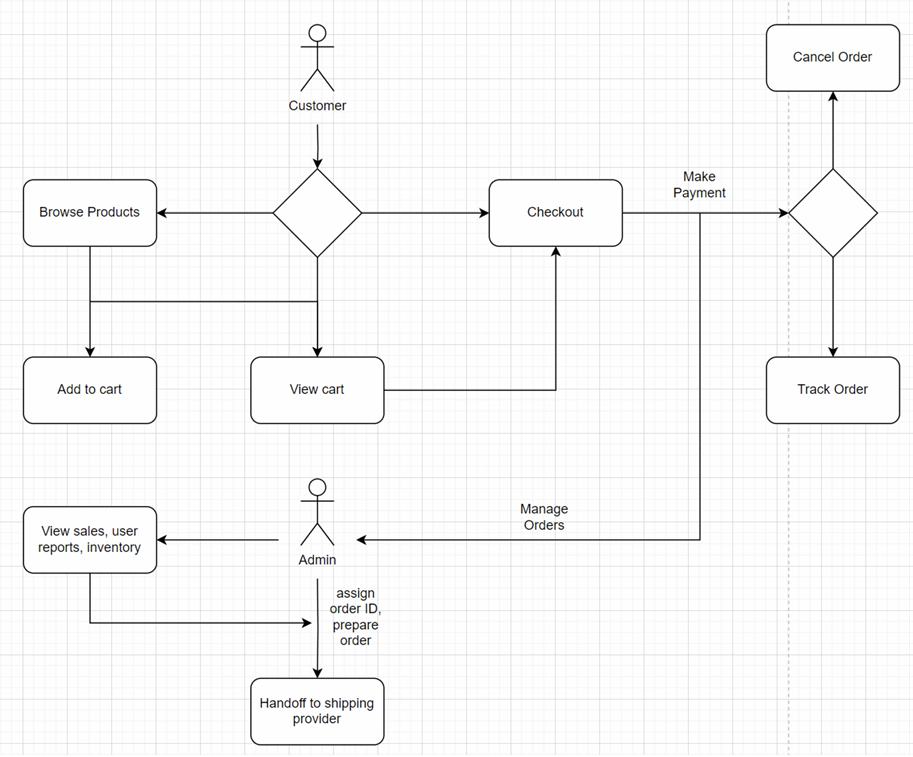
### 4.2

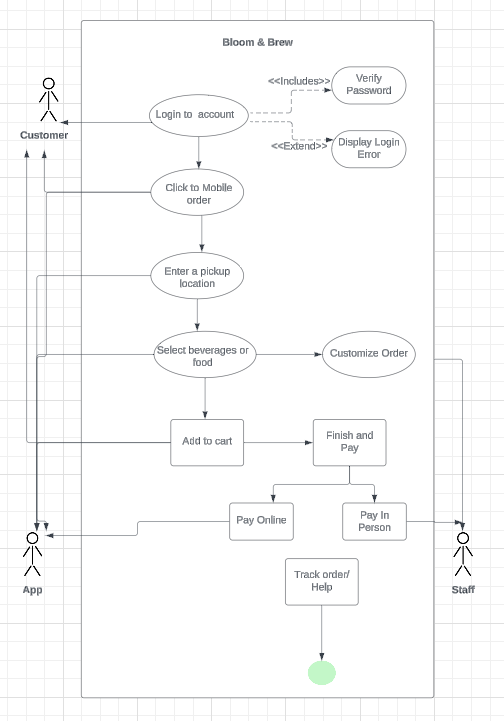
### **UML Diagrams**

• **Class Diagrams:**

• **Sequence Diagrams (if needed):** If helpful, include sequence diagrams to outline

interaction flows, such as the process of placing an order or updating an order.





### **5. System Architecture**

### 5.1

## **Architecture Overview**

The user interface, where managers and customers will interact, will run on client-side. This is where the customers will browse the menu, customize orders, and place orders. Also where staff and managers view and manage the orders. The backend will store data and interact with customers orders. It will manage the customer’s orders, update statues, and perform most operations. It will have a simple database to keep order details, menu items, and customer information.

**System Flow**

1. Customer does an action (place an order / customize item) on the app.
2. The app sends the request to the system (back-end) for order processing.
3. The back-end processes / takes the request and handles the data. Then gives a response that confirms order.
4. The client will see the displayed results as the user interface updates from the data inputted.

## **Component Design**

### 5.2

**Customer Component:**

*Define*

Lets customers browse the menu, allows them to customize orders (choose flavor or milk type), allows customers to place orders and views order status.

*Interact*

The customer selects the item, customizes the item, and then places the order. The Order Processing component takes the order and the customer component shows the order confirmation and status.

**Manager / Staff Component:**

*Define*

Let managers see all active and new orders. Allows managers to update the order status like, “in progress” or “completed”. Lets managers add or remove items from the menu.

*Interact*

The manager component gets the order information from the Order Processing Component. When the manager updates the order status it gets sent to the Order Processing Component and then it's shown in the Customer Component.

**Order Processing Component:**

*Define*

It processes orders, stores them, calculates prices, and updates order status.

*Interact*

When a customer places an order the Order Processing Component processes it like, calculate the price and store the order. It will also track the status of orders and send updates to both the Customer Component and Manager Component.

### **6. Testing and Validation**

### **Basic Test Cases for Validating Functions**

### 6.1

### **Testing Customer Order Placement**

* **Test Case 1.1**: Validate successful order placement.
  + **Steps**:
    - Navigate to the ordering interface.
    - Add items to the cart, customize options (e.g., size, additional toppings), and proceed to checkout.
    - Enter valid customer details (name, email, payment info).
    - Submit the order.
    - **Expected Result**: Order confirmation displays, including order ID, estimated pickup time, and itemized receipt.
* **Test Case 1.2**: Handle invalid inputs.
  + **Steps**:
    - Attempt to submit an order with missing or incorrect payment details.
    - **Expected Result**: An error message appears, such as "Invalid payment details. Please try again."
* **Test Case 1.3**: Check for "out-of-stock" items.
  + **Steps**:
    - Add an item marked as "out of stock" to the cart.
    - Attempt to checkout.
    - **Expected Result**: A prompt informs the customer, "Item is unavailable," and prevents order submission.

#### **. Checking Manager's Ability to View and Update Orders**

* **Test Case 2.1**: Validate order display.
  + **Steps**:
    - Log in as a manager.
    - Navigate to the order management dashboard.
    - **Expected Result**: Orders appear categorized by status (e.g., "new," "in preparation").
* **Test Case 2.2**: Update order status.
  + **Steps**:
    - Select an order from the dashboard.
    - Update its status to "completed."
    - **Expected Result**: Status changes to "completed," and customers are notified (via app/website or email).
* **Test Case 2.3**: Handle flagged orders.
  + **Steps**:
    - Mark an order as "flagged for review."
    - **Expected Result**: A flagged notification is sent to relevant parties.

### 6.2

#### **Ensuring System Correctly Displays Hardcoded Menu Items**

* **Test Case 3.1**: Validate menu display.
  + **Steps**:
    - Access the menu page as a customer.
    - **Expected Result**: All hardcoded menu items appear with correct names, prices, descriptions, and availability.
* **Test Case 3.2**: Temporarily remove an item.
  + **Steps**:
    - Log in as a manager.
    - Mark an item as "out of stock."
    - Check the customer-facing menu.
    - **Expected Result**: The item is no longer visible, displayed as none or marked as unavailable.
* **Test Case 3.3**: Add a new item.
  + **Steps**:
    - Log in as a manager.
    - Add a new item with all required fields (name, description, price, image).
    - Check the updated menu.
    - **Expected Result**: The new item appears on the customer-facing menu.

### **Function Criteria**

### 6.3

#### **1. Customer Order Placement**

* Orders must generate unique order IDs.
* System must validate inputs (e.g., payment details) and handle errors gracefully.
* Customers must receive real-time updates on order status.

#### **2. Manager Order View/Update**

* Managers should see all active orders in real-time.
* Status changes must update accurately across all interfaces.
* Alerts should trigger for flagged orders or system issues.

#### **3. Menu Management**

* Updates (e.g., adding, editing, or marking items out of stock) must reflect on the customer-facing menu immediately.
* All menu items must include accurate descriptions, pricing, and availability.

### **7. More Functionalities**

* **Customer Feedback Form**
* **Build Your Own Beverage**

### **8. Conclusion and Approval**

• **Summary:**

In today's fast-paced world, consumers increasingly seek spaces that offer relaxation, connection, and nourishment for both mind and body. Cafés like Bloom and Brew fulfill this need by providing a serene atmosphere where patrons can connect with nature through celestial-themed aesthetics and naturally inspired offerings. Research shows that coffee and a positive café environment not only benefit physical health but also enhance emotional well-being and productivity.

The objective is to have a calming and convenient ordering experience. The order process is smooth, user friendly, where the customer can easily browse the menu, customize/ personalize their orders without confusion. The Bloom and Brew app will give a seamless, engaging, and convenient experience for the customers. The café will build customer loyalty, increase sales, and even have a more efficient operation by this app. The system will expand the café’s experience beyond the physical location, from the celestial aesthetic, to the unique and immersive experience all from an app. It will enhance the customers experience and at the same time make the process run a lot quicker for ordering and preparing orders. The system can be used by customers daily, to browse the menu, customize orders, and place orders for pickup. At the same time the customers can have the option of reordering their favorites and keeping track of their past orders.

In essence, Bloom and Brew meets a growing need for mindful, health-oriented café spaces, combining the benefits of coffee with a nature-centered, community-focused environment that nurtures both individual well-being and social interaction.

- Version: 1.1

Date: 2024-11-15

Changes Made: Initial draft of the SRS document.

Author: Brooke Spangler, Camila Urena, Kylee Grasela, Brittany Fetters