Criterion A: Planning

Word Count: 448

Problem Statement:

Mrs. Krupali Kavle, my client, finds it difficult to manage all the orders she gets from customers and to calculate the total revenue and profits from all her sales. She is looking for a solution that simplifies the process of ensuring all the orders are delivered, easily store and retrieve them, ensure that the payment was received from the customer and also keep track of her inventory.

Description of the Problem:

Mrs. Krupali Kavle is a partner at Royal Elevators—a small scale company in Mumbai, India. As her passion, she also operates a small-scale business at the side, called Krupa's Foods, where she sells spices, modaks[1], jaggery blocks, mangoes (seasonally), and other consumables.

Based on our discussion during our initial interview[2], she does not have any current system in place to track her orders' statuses or inventory. She ends up scrolling through her WhatsApp messages to find who has placed the order and still has not received it, which is rather cumbersome and prone to human error of missing out on customers. She is looking for a piece of software that will solve this issue and improve her overall operations of her business.

Rationale for the Solution:

Mrs. Krupali Kavle, would require an application that will have a place that will let her store the current inventory and how much it costed along with a place to store the orders that can be categorized by: placed, delivered, completed (payment received). This management of stock and orders should be possible separately for all different products she sells like mangoes, spices, modaks, etc.

[1] A modak is a sweet Indian dumpling filled with coconut and jaggery, often made during festivals.

[2] Initial Interview can be found in *Appendix A: Initial Interview*.

The solution could have been in the form of a website or as a native application. However, my client insisted on being able to manage her orders without being connected to the internet due to issues with network reception at her home location. Additionally, my client uses a Mac and an iPhone as her primary devices, so for the best possible experience, developing a native application was the way to go. I chose to make the application in SwiftUI, a declarative user interface framework built specifically for Apple platforms, which will allow me to create a performant piece of software that can work without an active internet connection. Swift, an open-source protocol-oriented programming language, will be used for the logic; it is clean, safe, and memory-efficient. For the database, I will use SwiftData—a framework for persistent storage built on top of SQLite. It brings all the great features from SQLite and simplifies them in a modern format that integrates seamlessly with Swift and SwiftUI. I have also been working with Swift for over five years, which means I'm familiar with the language, allowing me to build the best possible solution for my client.

Success Criteria:

1. A user friendly and intuitive interface that makes the application easy to use with a minimal learning curve.
2. Should support iPhone and Mac.
3. The app should have a high bar for data privacy & security.
4. Data should be synced across devices owned by the client.
5. The user should be able to switch between sections of the app for each product they sell.
6. The amount of available stock should be able to be stored and must be easily accessible
7. Stock should be able to be deleted.
8. Backordering must be allowed, and the client should be prompted to add more stock based on the backorder quantity.
9. Stock must automatically be recalculated when new orders are placed.
10. Orders should contain details about the person who placed it, who took the order, related date and times, the address of the delivery, quantity ordered, order notes, amount paid or expected to be paid.
11. An AI-powered feature that can take a screenshot from a WhatsApp chat and fill-in the Order Form based on available context.
12. If an order is being placed at a loss, the client should be alerted about how much money they are losing out on. But placing orders for free should be allowed since the client often sends out their products as gifts.
13. Order details should be able to be edited after placing the order in case the customer asks for a change in quantity or price.
14. Orders can be marked as delivered or completed
15. Orders on the home page should be grouped based on their current delivery or payment status.
16. Orders should be able to be deleted.
17. When adding a new customer, their details can be imported directly from the system contacts app without having to type in the details manually.

18. Customer details should be able to be edited later to accommodate changes in basic details of the customer such as phone numbers or addresses.
19. The amount paid for all orders in total can be used to calculate the total revenue and compared against the cost of sales.
20. Total profits and total revenue should be presented in the form of chart reports that can be set to selected time ranges.
21. Version Control via GitHub to rollback changes if required.
22. Once an order has been placed, and the payment has been made, an invoice can be generated and shared with the customer.
23. Validation Checks
    - i. A product cannot be added until the name and icon are specified.
    - ii. An order cannot be placed until the quantity purchased and customer details are specified.
    - iii. A customer cannot be added to the records unless the name, phone number, and address for the customer is provided.
    - iv. A stock order cannot be added until the quantity is specified.
    - v. An existing customer cannot be deleted until all associated orders are removed.
    - vi. When generating chart reports for revenue and profits, the date range cannot be after the current date, and the start date must always precede the end date.
24. An app icon should be designed to represent the app on the user’s device home screen.