**Full Stock: Restaurant Ordering and Automated Inventory Management System**

**Team 8**

Cameron Nguyen, Nathan Carney, Rachel Freedman, and Sam Song

Table of Tables, Contents, and Figures

[**Development Recommendation**](#_9oty7veh5pg) **2**

[**Table 1.1 -** Alternative Matrix](#_f7gmw8rkf7ag) **4**

[**Architecture Design**](#_e42p7gftbcqy) **5**

**Figure 2.1 -** [Client / Server Architecture](#_pp7m74iajhcc) **6**

[**Table 2.1 -** Nonfunctional Requirements](#_22cb5agmy215) **7**

[**Table 2.2 -** Specific Nonfunctional Requirements for Full Stock](#_dfwlhe2evcdb) **8**

[**Storyboard**](#_jwhgu6o9kize)**s 10**

[**Figure 3.1 -** Restaurant Storyboard](#_nu0le08nprdl) **10**

[**Figure 3.2 -**](#_sdzpswdwoj3r) User Storyboard **11**

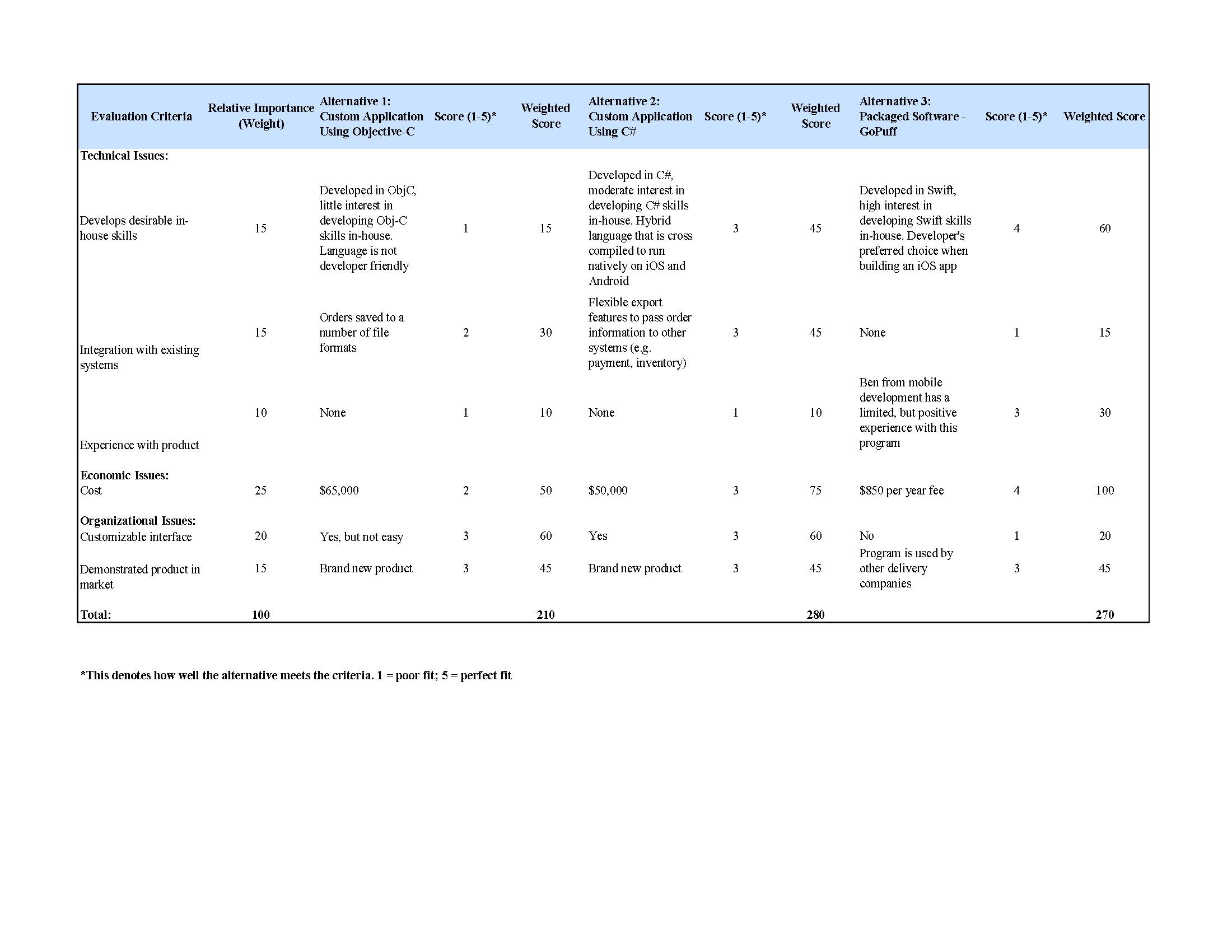
**Figure 3.3 -** [**User Interface Design -** Local Restaurant](#_i7vlqt4uqn52) **12**

**Figure 3.4 -** [**Interface Design -** Customer/Supplier](#_qvg2a8tezmy) **14**

# **Development Recommendation**

At FullStock, we took the design phase incredibly seriously. A considerable amount of time was spent in determining our preferred system acquisition strategy. We carefully considered the nonfunctional business requirements as they influence the system requirements that drive the design of the system’s architecture. Extensive planning went into our new system’s performance as we explored: how fast the automated inventory system will operate, what its capacity should be, and of course its reliability and availability. Additionally, it was important that we were able to create the most secure system possible that meets industry standards for encryption, authentication, and virus control. To deliver this, custom development was needed. There was no doubt that developing a custom application from the ground up would enable the teams at FullStock to have complete control over not only the way that it looks, but its functionality as well. We wanted to streamline the ordering process that allows for a safe contactless pickup, while also updating a local restaurant’s inventory levels, menus, and prices in real time. Seamless integration with our payment system that satisfies PCI DSS (Payment Card Industry Data Security Standard) compliance was essential. A beautiful app that works on both iOS and Android devices, and meets these highly specialized requirements is doable by building a system from scratch. No existing packaged software is capable of meeting these unique requirements. Furthermore, these packages are entirely reliant on an outside vendor for maintenance and future enhancements. Despite the potential cost and time savings, this was not the right solution for us. Similar problems arise with outsourcing development, as it relies on vendor support, increases the security risk of losing confidential information, and reduces control over future development. Most importantly, in-house professionals are denied the opportunity to gain new skills. A custom development gives us exactly what we want, and offers developers the freedom and creativity to solve business problems as they see fit. At FullStock, past customer orders will serve as a strategic enabler. FullStock’s future plans involve utilizing the information from the system to better understand the needs of our customers and local restaurants. Later updates to the system will include the incorporation of data-mining software and new geographic information systems to help facilitate marketing research. Building a custom application makes these changes a lot easier, and whose components ultimately take advantage of current technologies in order to support our strategic efforts. Lastly, creating a system in-house builds technical skills and functional knowledge within the company. The more our developers interact with business users, the greater their understanding of the business as a whole, and an enhanced ability to align information systems with strategies and needs. Future projects applying similar technologies become easier as a result. We are confident that the additional expenditures and time spent on a custom development will lead to a world-class application unlike any other.

## Table 1.1 - **Alternative Matrix**



An Alternative Matrix was used to organize the pros and cons of the design alternatives so that the FullStock team could determine the best solution. It combines several feasibility analyses into one matrix which makes comparisons between alternatives easier. Weights were added for different parts of the matrix to demonstrate that certain criteria (e.g. development of in-house skills) are more valuable than others. The project team and approval committee recognize that all three alternatives have positive points, however alternative 2 shows the most promise as its programming language works for multiple platforms (iOS and Android). A hybrid app can potentially save significant time and costs because the in-house team would only be responsible for a single development as opposed to two (i.e. Java for Android and Swift for iOS). The expense is reasonable with no recurring costs, and there is moderate interest from in-house to develop C# skills.

# Architecture Design

Our team chose an n-tiered architecture design for FullStock. An n-tiered design for use over a two or three-tiered because of its scalability. The separate servers provide the allowance of the shared load. Our team does not want the server to crash while consumers use it. This type of design is common for e-commerce. Our team decided on using a thin client-server architecture to reduce the overhead and maintenance. The restaurant industry can be hectic and we did not want our clients to worry about maintaining the server. As seen in the non-functional requirements figure, our team’s system requirements support a thin architecture.

The following list are reasons for a thin client-server architecture:

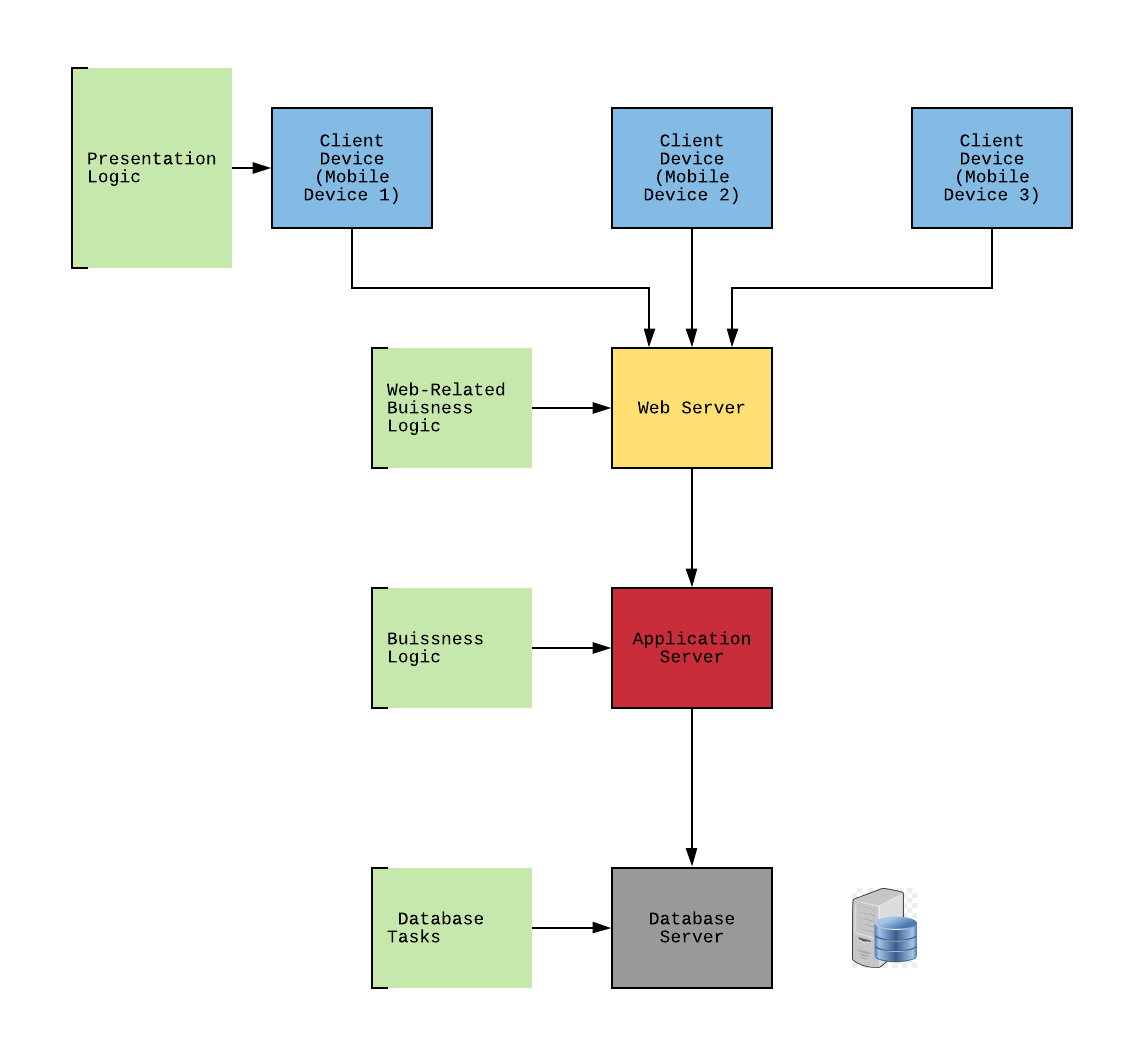
**Operational Requirements** - Our application can be integrated with a desktop and have vast portability requirements. Data from the application is constantly being transferred to the restaurant’s own systems.

**Performance Requirements** - Client-server architecture is best suited for our application. This type of architecture is scalable meaning that companies can easily change the hardware to the speed requirements. Also if a server crashes, then another server can easily be used. Users will have an easier time operating FullStock.

**Security Requirements** - A client-based server provides more advanced authentication tools to ensure that the users and the restraunts personal information won’t be stolen.

**Cultural and Political Requirements** - Separating the presentation logic from the application logic and data allows more languages to be used and more customization. The restaurant industry attracts a variety of people from different backgrounds. It is important to accommodate all types of employees. Having a thin-server architecture allows an easier separation.

## Client / Server Architecture



**Figure 2.1 -** The client / server architecture system is pictured above. This shows FullStock’s n-tiered architecture design. This also shows where the different logics and database tasks reside.

## Table 2.1 - Nonfunctional Requirements

| **Requirements** | **Server-Based** | **Thin Client - Server** | **Thick Client - Server** |
| --- | --- | --- | --- |
| **Operational Requirements** |  |  |  |
| System Integration Requirements | Yes | Yes |  |
| Portability Requirements |  | Yes |  |
| Maintainability Requirements | Yes | Yes |  |
| **Performance Requirements** |  |  |  |
| Speed Requirements |  | Yes |  |
| Capacity Requirements |  | Yes |  |
| Availability/Reliability Requirements | Yes | Yes |  |
| **Security Requirements** |  |  |  |
| High System Value | Yes | Yes |  |
| Access Control | Yes |  |  |
| Encryption/Authentication |  | Yes |  |
| Virus Control | Yes |  |  |
| **Cultural/Political Requirements** |  |  |  |
| Multilingual Requirements |  | Yes |  |
| Customization Requirements |  | Yes |  |
| Making Unstated Norms Explicit |  | Yes |  |
| Legal Requirements | Yes | Yes |  |

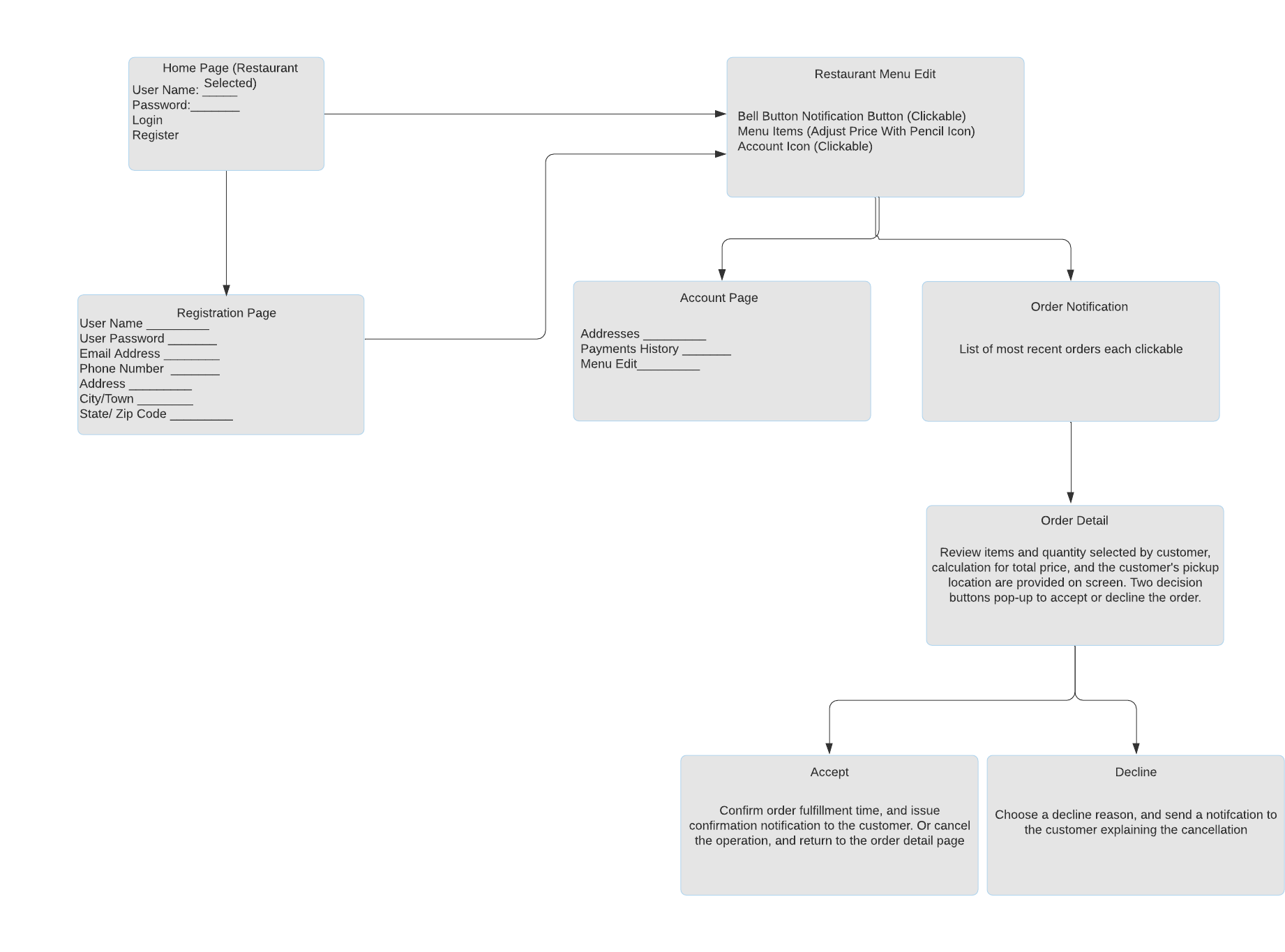
The table above shows the breakdown of the different requirements and the relationship between our team’s chosen thin client server architecture.

## Table 2.2 - Specific Nonfunctional Requirements for Full Stock

| **Requirements** | **Number** | **Description** |
| --- | --- | --- |
| **Operational Requirements** |  |  |
| Technical | 1.1 | The system will work with the internet. |
|  | 1.2 | Consumers can use the application via their mobile device. |
| System Integration | 1.3 | The system needs to use MS to properly track the inventory quantities. |
|  | 1.4 | The system needs to work with the application to properly update the restaurants ordering system and inventory database. |
| Portability | 1.5 | The system will be able to work with multiple operating systems and mobile devices (iphone, android, etc.) |
| Maintainability | 1.6 | The system will be updated as needed. |
| **Performance Requirements** |  |  |
| Speed | 2.1 | Response time must be less than 5 seconds. |
| Capacity | 2.2 | Maximum of 50 users at a time when placing an order for food. |
| Availability/Reliability | 2.3 | System will be accessible during the restaurant's business hours. |
| **Security Requirements** |  |  |
| High System Value | 3.1 | No special system value requirements are anticipated. |
| Access Control | 3.2 | Consumers can access their accounts using their mobile app. Via username and password. |
| Encryption/Authentication | 3.3 | Payment is secure and information must be re-entered each time. |
| Virus Control | 3.4 | Virus threats are constantly monitored. The application cannot be downloaded if the user has a virus on their mobile device. |
| **Cultural/Political Requirements** |  |  |
| Multilingual | 4.1 | Restaurants have the option to switch the applications language to Spanish, English, and Chinese to accommodate different types of consumers. |
| Customization | 4.2 | Each application is customized to the particular restaurant. |
| Unstated Norms | 4.3 | The application is currently only in the US. There are currently no unstated norms. |
| Legal | 4.4 | No special legal requirements are anticipated. |

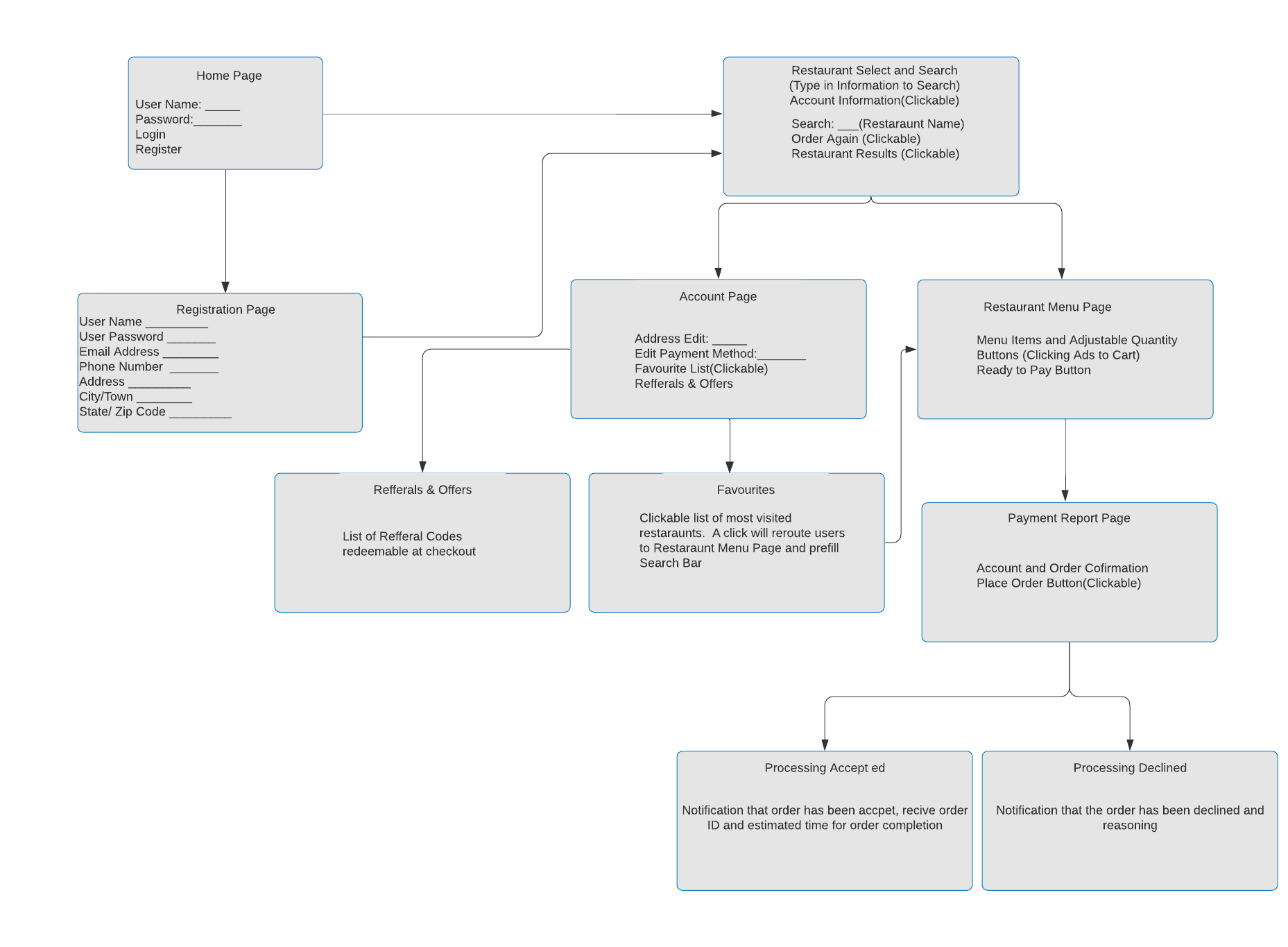
The table above shows a more in depth look at the requirements for Full Stock.

# Restaurant Storyboard



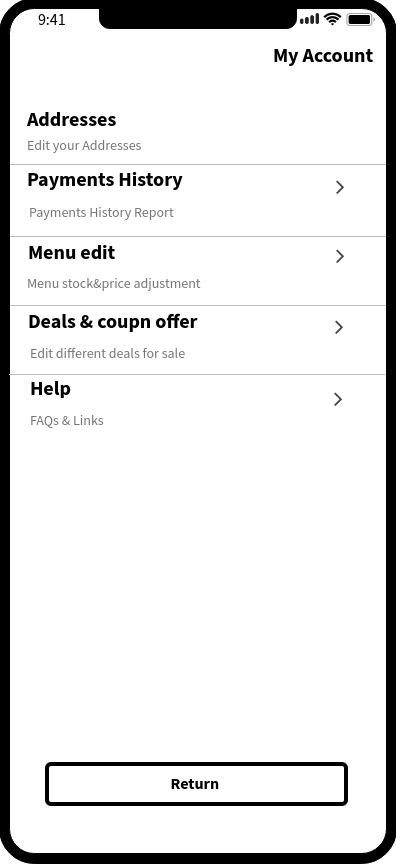
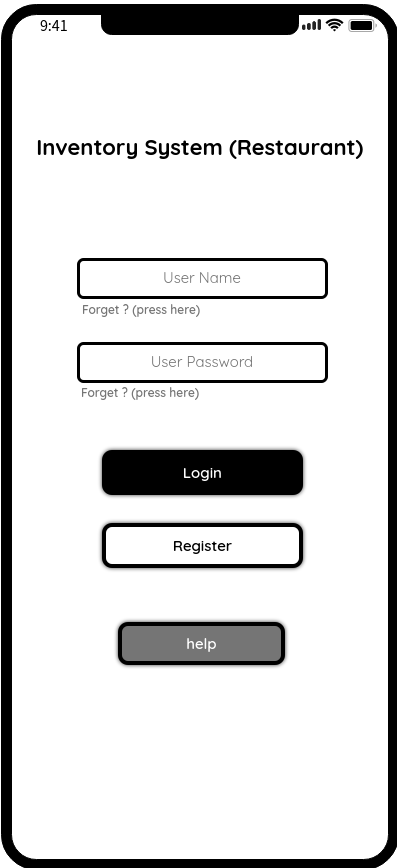
## **Figure 3.1 -** This storyboard showcases the restaurant worker’s journey using the app from registration to order completion.

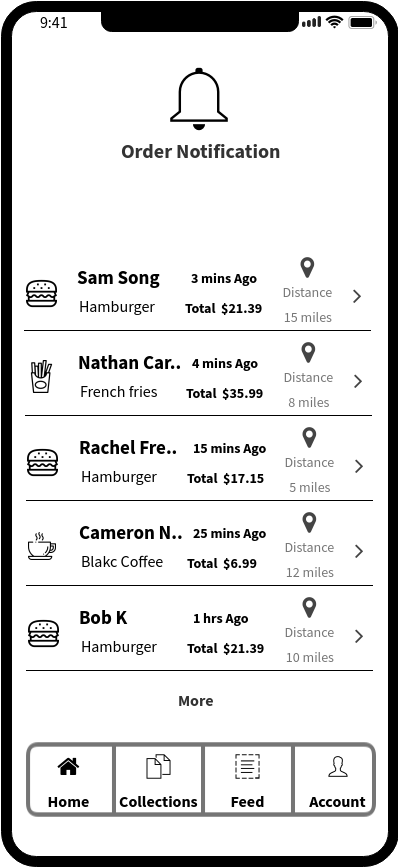
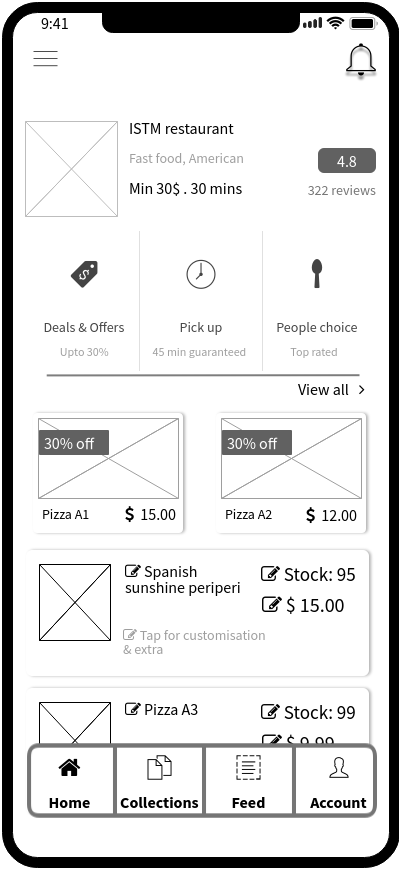
1. **User Storyboard**

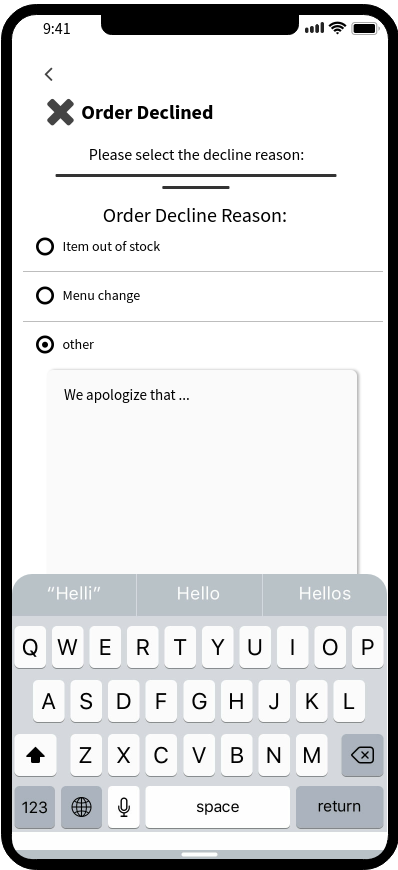
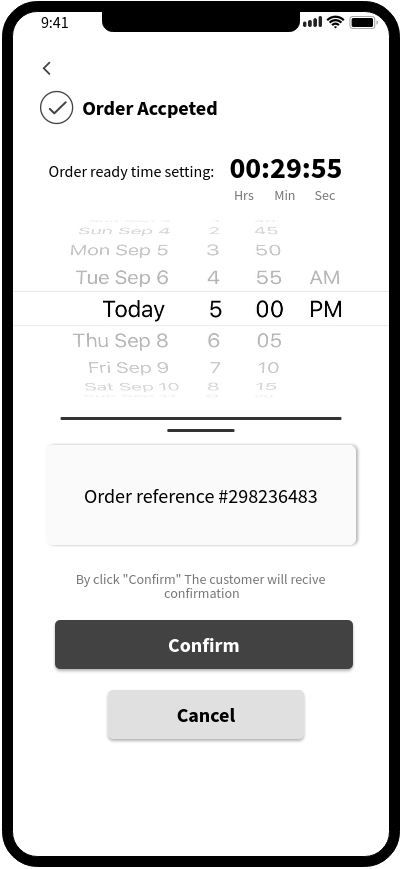
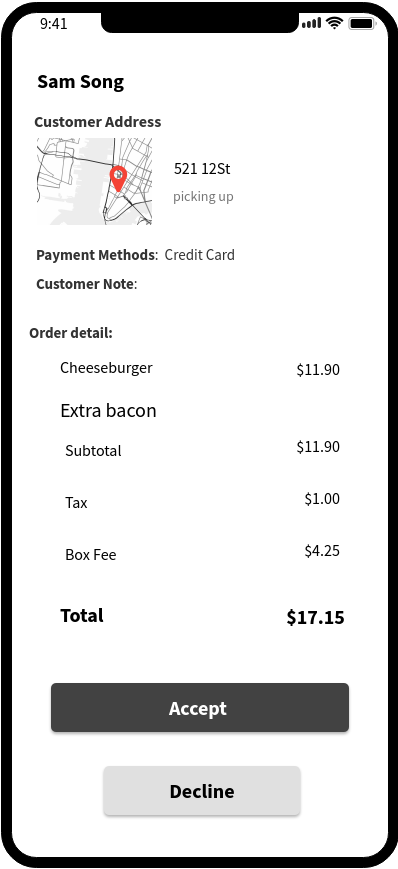


## **Figure 3.2 -** This storyboard showcases the user’s journey as they open our app for the first time and place their first order.

# User Interface Design - Local Restaurant

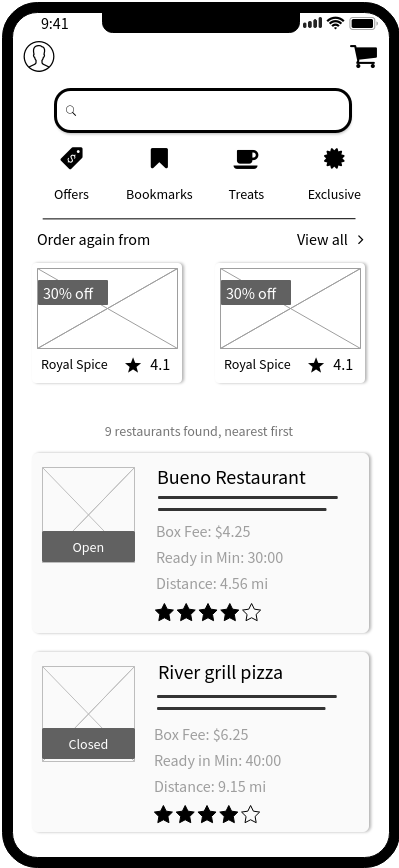
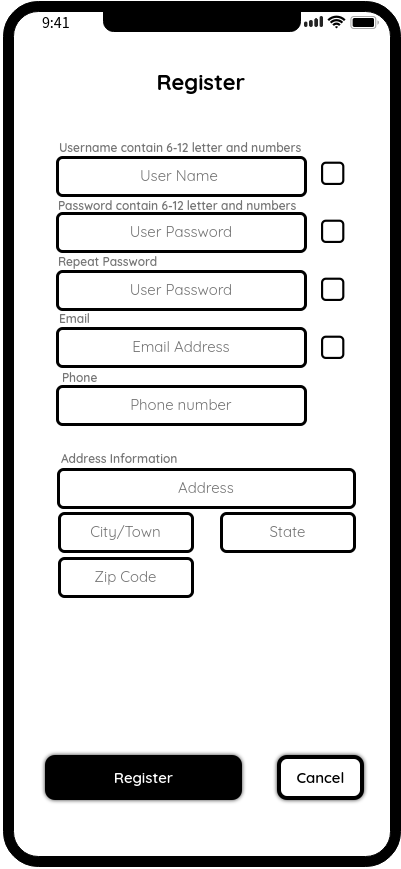
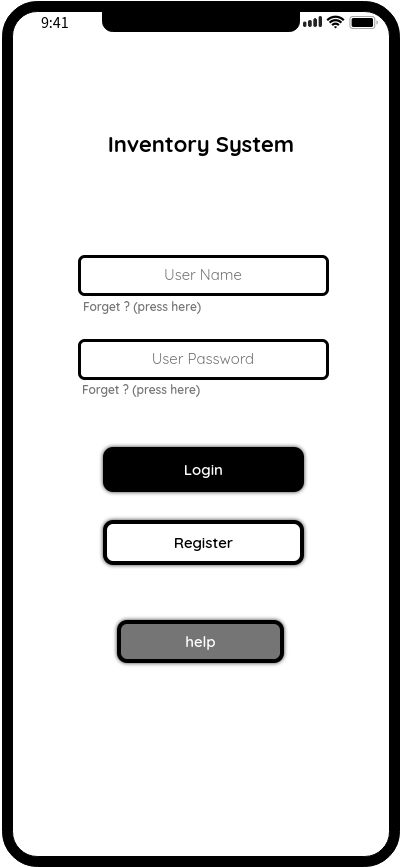
****

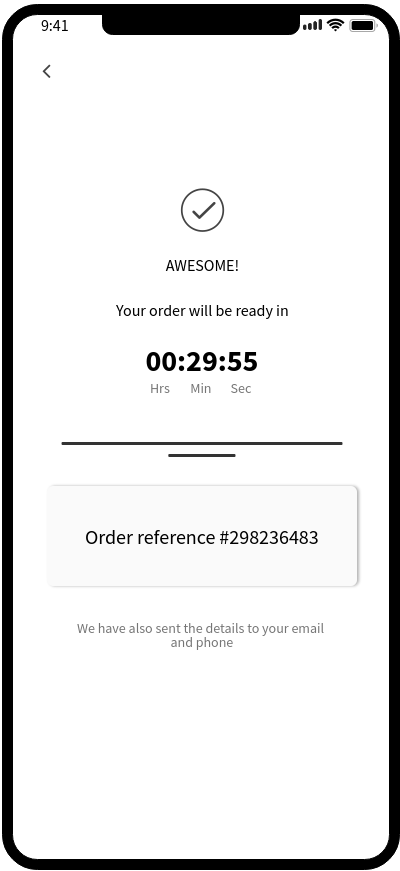
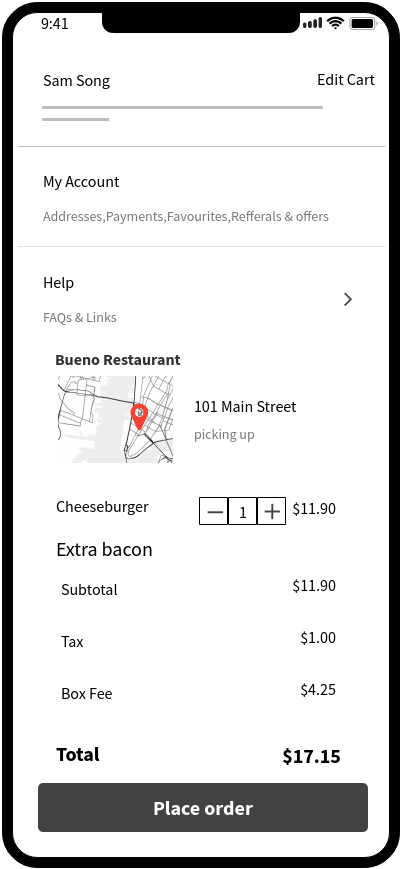
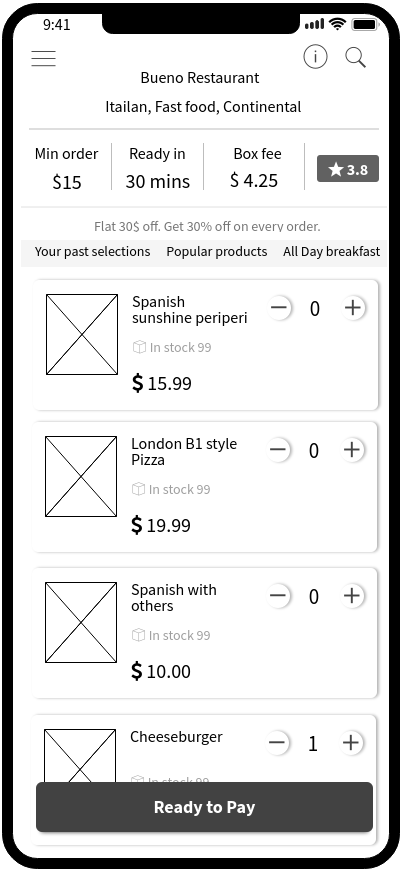
****

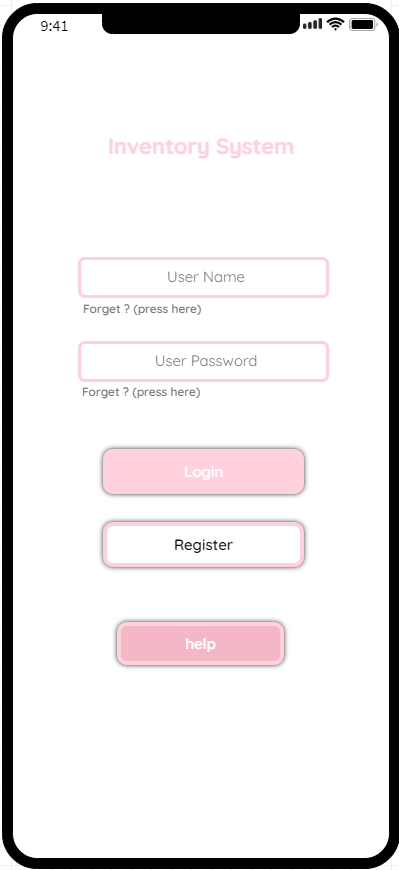
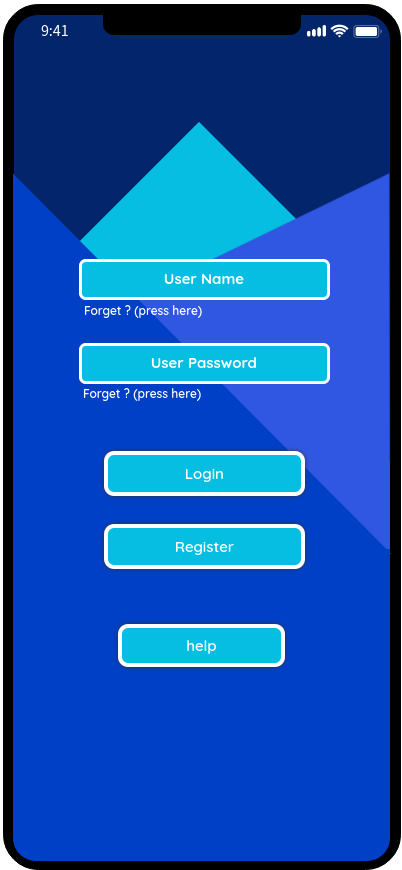
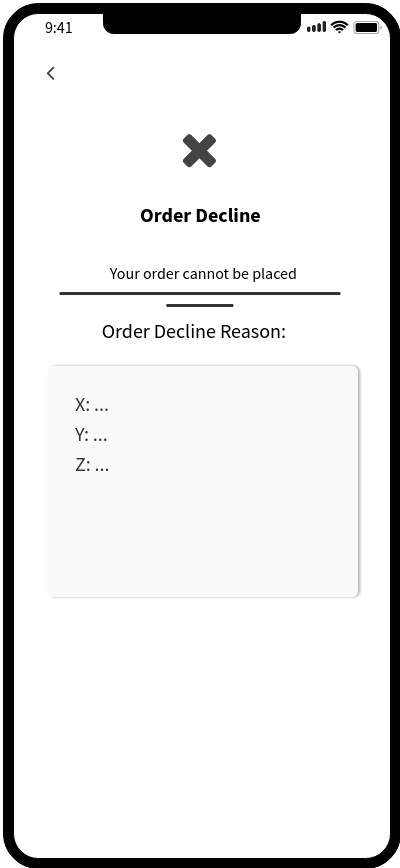
****

**Figure 3.3 -** This showcases the UI from a local restaurant’s perspective. Managers and Admin have access to real-time updates on inventory levels. Changes to menus and prices can be executed via the Menu Edit tab and are instantaneous.

# User Interface Design - Customer/Supplier

****

****

****

**Figure 3.4 -** This showcases the UI from a customer/supplier’s perspective. Customers have access to a restaurant’s most up-to-date menu, prices, and location. Suppliers may login to the inventory management system to complete order fulfillment requests. Users are given customizable coloring options to best reflect their aesthetic preferences.