



Team Name:
Team SharkBytes

Team Leader:
Brandon Tran

Team Members:
Mayra Sanchez
Justin Reid
Manuel Beltran

State of California
Computer Science Department
Software Development

RAD

Date Issued:
February 25, 2020

Date Due:
March 10, 2020

Version 1.0

Table of Contents

Introduction	5
Purpose	5
Document Conventions	5
Intended Audience and Reading Suggestions	5
Product Scope	6
References	6
Overall Description	6
Product Perspective	6
Product Functions	7
Operating Environment	8
Design and Implementation Constraints	8
User Documentation	8
Assumptions and Dependencies	9
External Interface Requirements	9
User Interfaces	9
Hardware Interfaces	14
Software Interfaces	14
Communication Interfaces	14
System Features	14
Customer Use Cases	14
Use Case #1:	14
Create Customer Account	14
Use Case #2:	15
Customer Login	15
Use Case #3:	16
Customer Login (Google Authentication)	16
Use Case #4:	17
Edit Customer Account	17
Use Case #5:	18
Change Customer Password	18
Use Case #6:	19
Add Payment Method	19
Use Case #7:	20
Edit Payment Method	20
Use Case #8	20
Open Tab	20

Use Case #9:	21
Checkout	21
Use Case #10:	22
Customer Membership Card	22
Employee Use Cases	23
Use Case #11:	23
Create Restaurant Account	23
Use Case #12:	24
Employee Login	24
Use Case #13:	25
Edit Employee Account	25
Use Case #14:	25
Change Employee Password	25
Use Case #15:	26
Register Employee	26
Use Case #16:	27
Remove Employee	27
Use Case #17:	28
Create Menu Tab	28
Use Case #18:	29
Create Menu Item	29
Use Case #19:	30
Edit Menu Item	30
Use Case #20:	31
New Check	31
Use Case #21:	31
New Check by Proxy	31
Use Case #22:	32
Recall Check	32
Use Case #23:	33
Recall Check by Proxy	33
Use Case #24:	34
Change Party Size	34
Use Case #25:	35
Change Table Number	35
Use Case #26:	35
Seat Number	35
Use Case #27:	36
Balance	36

Use Case #28:	37
Balance Print	37
Use Case #29:	37
Credit Card Transaction	37
Use Case #30:	38
Cash Transaction	38
Use Case #31:	39
Split Check	39
Use Case #32:	40
Finalize Check	40
Use Case #33:	41
Allergy	41
Use Case #34:	41
Special Instruction	41
Use Case #35:	42
Apply Discount	42
Use Case #36:	43
Complimentary Item	43
Use Case #37:	44
Void Item	44
Use Case #38:	45
Employee Links Membership ID	45
Nonfunctional Requirements	46
Other Requirements	47

1. Introduction

1.1. Purpose

We hope to create a seamless and time-efficient POS system that creates a better experience for restaurants and guests. We want millions of restaurants, new and old, to be able to easily integrate and run their businesses on *Dine n' Dash*. We believe everything works smoother when everyone is on the same page, so why not incorporate this methodology into the food industry. We want to give the power of controlling and viewing orders to the customers, then quickly choose the preferred way of payment with available new technologies. As a result, restaurant owners can boost their revenue with software convenience tools to assist them in providing a pleasant experience for the guest.

1.2. Document Conventions

This document follows the MLA format and utilizes bold text to signify the title of each section and subsection headings. Also, it includes a table of contents to refer to each section more precisely when browsing for a specific part in the document.

1.3. Intended Audience and Reading Suggestions

This document's sole purpose is to be read and analyzed by the development team, project managers, marketing staff, testers, and documentation writers. The remaining sections of this SRS contain the overall descriptions of the product, external interface requirements, system features, non-functional requirements, and other requirements. This document is sorted and organized according to the order of ascending specificity. The team needs to be very familiar with the material as each group will be reviewing:

Overall Description - Marketing staff will be extensively describing the products

System features - Testers will be responsible for unit testing each test case/suite and provide feedback for developers' room for improvement.

External Interface Requirements - The development team will know what is necessary to compose the external backbone elements. This information will have to be relayed over to the marketing staff so that they can better advertise the product.

Non-functional and functional Requirements - The development team will be responsible for constructing these requirements within the product.

1.4. Product Scope

This software product is to be used by restaurant owners and diners of the restaurant. We want new restaurants to integrate our software as their primary POS system. Furthermore, we want existing restaurants to upgrade their current POS system with a more advanced sales system. We believe that our software will not only provide the typical POS systems used to communicate in a restaurant but allow customers to engage with the restaurant to enable them to have a more luxurious experience.

1.5. References

For further reference, please see the prior sections of the SRS document regarding the product scope. Also, the use case document can be used to understand further a particular part of the product or project by the team.

2. Overall Description

2.1. Product Perspective

This software system is to replace traditional POS systems that are active in the food industry. This application that stores information from three sources to a database to allow for a seamless transaction.

User details:

- Includes the users' essential information such as their name, email, date of birth, and phone number.

Employee details:

- Includes the employees' essential information, such as their name, access role, location, and authorization level.

Restaurant details:

- Includes the restaurant's essential information, such as their name, type of food, location, and store hours.

API details:

- Stripe: Incorporate Stripe API for credit card fraud detection. Primary system to accept payments from guests. Receive tokens to check for confirmation of payment.
- Firebase: It allows us to sync and store data in real-time easily. Firebase is mainly for Google Authentication for faster registration/login process.
- Future Considerations: Apple Pay, Google Pay, Paypal. A faster way to process a transaction for a guest is using these APIs

2.2. Product Functions

Customer Use Cases

- Use Case #1: Create Customer Account
- Use Case #2: Customer Log In
- Use Case #3: Customer Log In (Google Authentication)
- Use Case #4: Edit Customer Account
- Use Case #5: Change Customer Password
- Use Case #6: Add Payment Method
- Use Case #7: Edit Payment Method
- Use Case #8: Open Tab
- Use Case #9: Checkout
- Use Case #10: Customer Membership Card

Employee Use Cases

- Use Case #11: Create Restaurant Account
- Use Case #12: Employee Log In
- Use Case #13: Edit Employee Account
- Use Case #14: Change Employee Password
- Use Case #15: Register Employee
- Use Case #16: Remove Employee
- Use Case #17: Create Menu Tab
- Use Case #18: Create Menu Item
- Use Case #19: Edit Menu Item
- Use Case #20: New Check
- Use Case #21: New Check By Proxy
- Use Case #22: Recall Check
- Use Case #23: Recall Check by Proxy
- Use Case #24: Change Party Size
- Use Case #25: Change Table Number
- Use Case #26: Seat Number
- Use Case #27: Balance
- Use Case #28: Balance Print
- Use Case #29: Credit Card Transaction
- Use Case #30: Cash Transaction
- Use Case #31: Split Check
- Use Case #32: Finalize Check
- Use Case #33: Allergy
- Use Case #34: Special Instructions

- Use Case #35: Apply Discount
- Use Case #36: Complimentary Item
- Use Case #37: Void Item
- Use Case #38: Employee Links Membership ID

2.3. Operating Environment

The software will operate with the following software components, frameworks, and applications. The software will be running under the following:

Front - End

Flutter: Will be used as a front-end framework for both Android and iOS platforms.

Dart: Will be used to support all the features of Object-Oriented programming paradigm.

Back - End

Node.js: To provide back-end functionality.

Firebase: For authentication purposes.

JSON - For transmitting data in the application.

RESTful API - HTTPs requests to GET, PUT, POST, and DELETE data.

2.4. Design and Implementation Constraints

- Working with new technologies: There's a learning curve for new technologies, and this will impact the amount of progress and implementations of our project.
- Time constraints: With the set amount of time, future features may not be able to be implemented. Essential features will be the priority within the application.
- Applying Object-Oriented Programming Language to a new language will take time to understand precise syntax.
- UML Diagram of this application is extensive to create but will provide a benchmark on how our app is connected.
- Scrum: Our team will need to manage our time precisely to finish specific features in each sprint.

2.5. User Documentation

None.

2.6. Assumptions and Dependencies

- Firebase: Requires servers to be running
- APIs: Requires continued third-party support
- Memory Size: Must be large enough to install the application
- Internet connection: Requires internet connection to function
- Constrained to web or mobile: Assumed will only be running on these platforms

3. External Interface Requirements

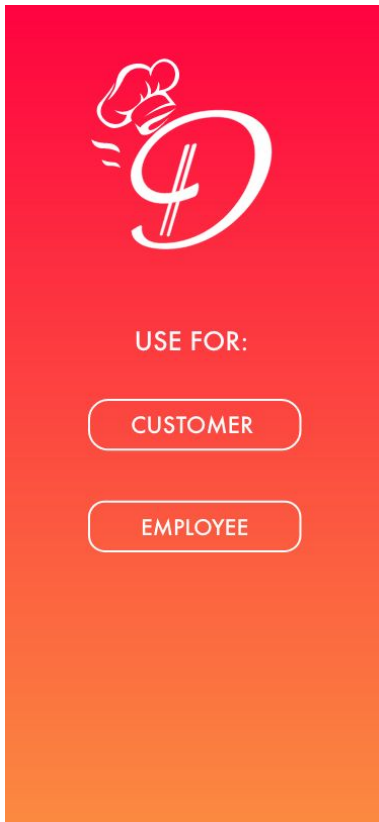
3.1. User Interfaces

The *Dine n' Dash* user interface has been designed for the customer's interest to provide them with a seamless interaction while utilizing the application to create, edit, and log in to account. The sole purpose of this application is to make the user's experience with their restaurant of choice as transparent and efficient as possible to allow them more time to enjoy their experience.

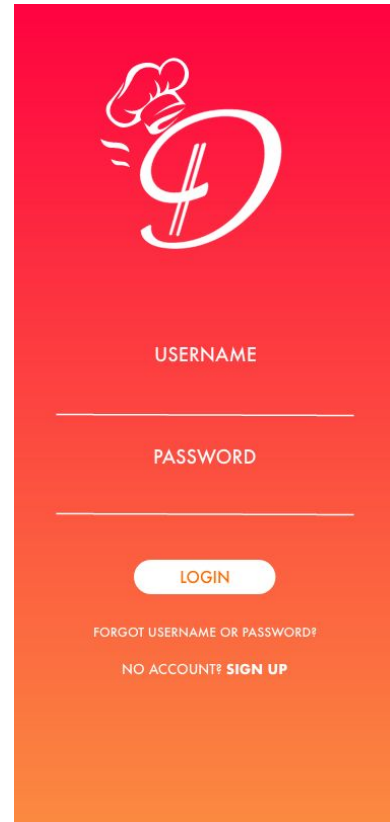
The home screen of the application allows the user to choose between two tabs to view their open tab or history of transactions. Also, the user can scan their QR with the place at the location of the restaurant. The user can click on a profile icon within the home screen to allow them to edit existing account information. As well they can click the payment method in which they would like to choose to transact.

When clicking the edit profile icon, it will display the users' current profile information, payment method, and settings to allow that information to change in the database. By clicking the payment icon will view the user with the options to either select mobile pay or pay in a restaurant.

Below, you will find the current renderings of our application.



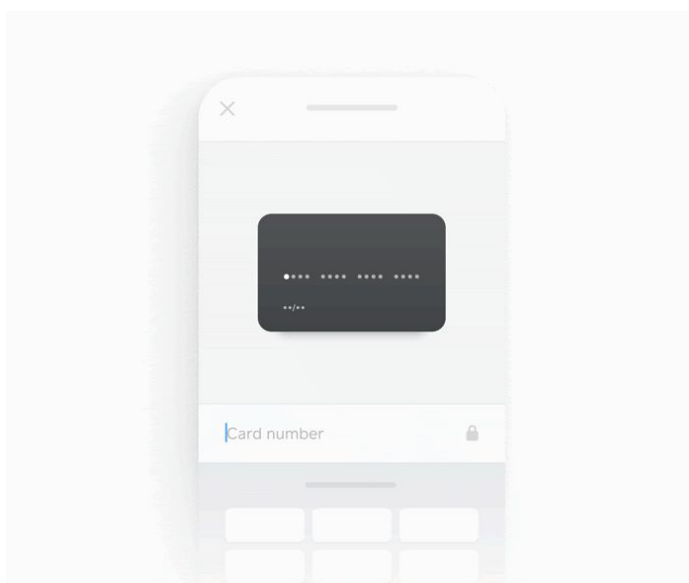
Login as either a customer or an employee. Proceeds to a login page that ask users to enter specific information to enter into the applications.



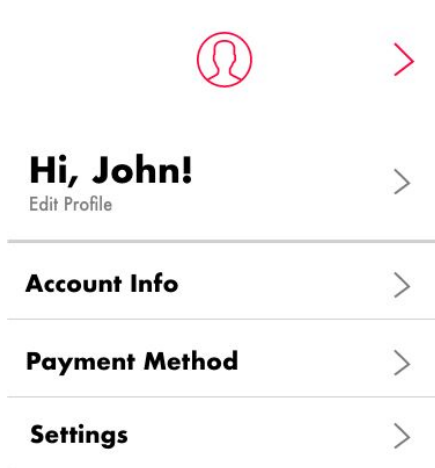
Employee View: This rendering shows the process of viewing a current tab for a table. You are able to discount the tab, add an additional item to the tab, print out the tab, as well as checkout the tab.

Customer View: This rendering shows the Customers tab. They have the option to close out their tab through their mobile device. The tab currently shows the lists of items, modifications, prices subtotal, tax, and total price of the entire tab.

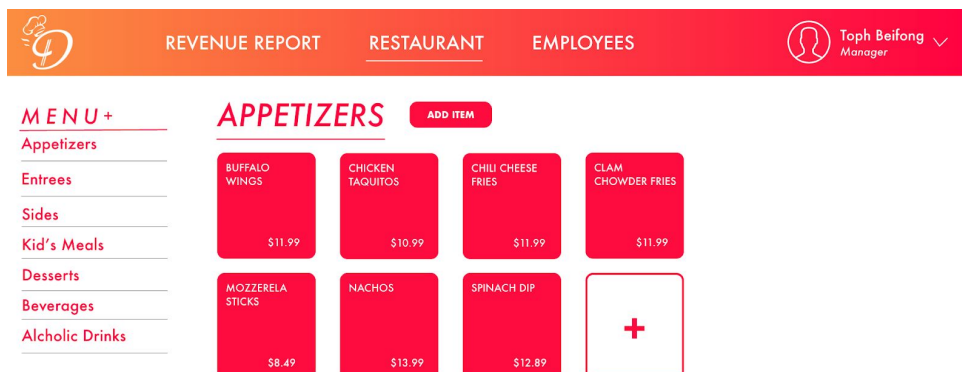
< Receipt	
Raspberry Lem...(2) <i>No modifications</i>	\$7.39
Water(2) <i>+No Ice</i>	\$0.00
Buffalo Wings <i>+ Extra Hot - No drums</i>	\$23.79
Fettucine Chicken... <i>+ Broccoli + Alfredo Sauce on the Side</i>	\$15.50
Barbecue Steak <i>+ Medium - Well - No barbecue sauce</i>	\$23.79
<hr/>	
SUBTOTAL: \$70.47	TAX: \$6.34
TOTAL: \$76.81	
<div>PAY</div>	



Customer View: This rendering shows the process of checking out with Stripe API. Inputting credit card information to check within its system for fraud detection.





Customer View: This rendering shows the profile tab in the application. Users of the application is able to edit their information, add/modify their payment methods, and other settings.



Employee View: This rendering shows the owner of the restaurant is able to add/modify menu items to display in the POS system. This is being shown as our web layout.

Employee View: This rendering shows the button to add an item to the menu. It asks for the item name, item price, item description, and optional picture for the food item.


REVENUE REPORT
RESTAURANT
EMPLOYEES


Toph Beifong
Manager

MENU +

Appetizers

Entrees

Sides

Kid's Meals

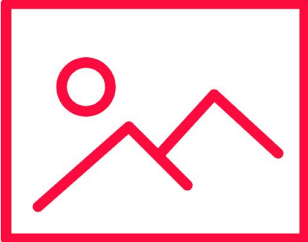
Desserts

Beverages


Alcoholic Drinks


APPETIZERS

Item Name:
Item Price:
Item Description:



ADD ITEM



REVENUE REPORT
RESTAURANT
EMPLOYEES


Toph Beifong
Manager

SEARCH

ADD EMPLOYEE

First Name	Last Name	Employee ID	Title	Date
Toph	Beifong	01665499	Manager	12/2/19
Justin	Reid	01665499	Lead	12/22/19
Mario	Super	01665499	Chef	12/27/19
Cassie	Food	01665499	Server	12/27/19
Luigi	Bros	01665499	Bartender	12/27/19
Brandon	Tran	01665499	Bartender	12/27/19



Brandon Tran
01665499


EDIT INFO


REMOVE


REVENUE REPORT

Employee View: This rendering shows the Employees tab to view revenue reports of all employees of the restaurant. Logged in as Manager.

Employee View: This rendering shows the Profile Tab to view Employee profiles as well as view revenue report for specific employee.


REVENUE REPORT
RESTAURANT
PROFILE


Cassie Food
Bartender



Cassie Food
01665499

Title: Bartender
Phone: 714-123-4567

REVENUE REPORT

3.2. Hardware Interfaces

The *Dine n' Dash* application will contain no hardware interfaces as it will be installed on a smartphone to utilize the application.

3.3. Software Interfaces

The *Dine n' Dash* application will be solely run on Android and iOS platforms. The application will feature touch screen capabilities, camera accessibility for QR Code, and geofencing system to assist with determining geographical boundaries. The app will also utilize JSON to send and receive data from the client-side server. All of the features will be made possible through the use of the following framework Flutter to deliver high-quality functionality for iOS, Android, and Web.

3.4. Communication Interfaces

The application will allow the users' to communicate back and forth with the host through the use of front-end and back-end connections using Flutter and Node.js. We are using asynchronous communication to combine both ends together.

4. System Features

Customer Use Cases

Use Case #1:	Create Customer Account
Actors:	Customer
Pre-Conditions:	<ul style="list-style-type: none">• Device is installed• Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none">• Customer Taps "Customer Account" Button<ul style="list-style-type: none">◦ App loads Sign In Page• Customer Taps "Create Account"• Customer Enters required Information• Customer Taps "Create Account"<ul style="list-style-type: none">◦ App verifies information◦ Information is added to

	database <ul style="list-style-type: none"> Account is Created
Post Conditions:	<ul style="list-style-type: none"> Unique Tag (QR code) is assigned to Customer Account is added to database
Error Conditions:	<ul style="list-style-type: none"> Blank input on required fields <ul style="list-style-type: none"> Error message shows Confirm won't continue Incorrect values inputted in fields <ul style="list-style-type: none"> Error message shows Confirm won't continue Email is already in database <ul style="list-style-type: none"> Error message shows Confirm won't continue Connection is lost <ul style="list-style-type: none"> Error message shows Attempts to reconnect
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> Usability: <ul style="list-style-type: none"> Clear directions are displayed for input account info and error messages Straightforward and intuitive steps Performance <ul style="list-style-type: none"> After info is entered, account creation takes less than 2 seconds.

Use Case #2:	Customer Login
Actors:	Customer
Pre-Conditions:	<ul style="list-style-type: none"> Device is installed Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> Customer Taps "Username" field <ul style="list-style-type: none"> Type Username Customer Taps "Password" field <ul style="list-style-type: none"> Type Password Customer Clicks "Log In"

	<ul style="list-style-type: none"> ○ App verifies information ○ Information is checked with the database.
Post Conditions:	<ul style="list-style-type: none"> ● Customer is Logged In to Application <ul style="list-style-type: none"> ○ Able to view Features
Error Conditions:	<ul style="list-style-type: none"> ● Blank input on required fields <ul style="list-style-type: none"> ○ Error message shows ○ Confirm won't continue ● Incorrect values inputted in fields <ul style="list-style-type: none"> ○ Error message shows ○ Confirm won't continue ● Connection is lost <ul style="list-style-type: none"> ○ Error message shows ○ Attempts to reconnect
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> ● Usability: <ul style="list-style-type: none"> ○ Clear directions are displayed for input account info and error messages ○ Straightforward and intuitive steps ● Performance <ul style="list-style-type: none"> ○ After info is entered, account log in takes less than 2 seconds.

Use Case #3:	Customer Login (Google Authentication)
Actors:	Customer
Pre-Conditions:	<ul style="list-style-type: none"> ● Device is installed ● Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> ● Customer Taps "Google Sign In" ● Customer Taps "Email" field <ul style="list-style-type: none"> ○ Type Google Email ● Customer Taps "Password" field <ul style="list-style-type: none"> ○ Type Google Password ● Customer Clicks "Log In" <ul style="list-style-type: none"> ○ App verifies information ○ Information is checked with the

	database.
Post Conditions:	<ul style="list-style-type: none"> • Customer is Logged In to Application <ul style="list-style-type: none"> ◦ Able to view Features
Error Conditions:	<ul style="list-style-type: none"> • Blank input on required fields <ul style="list-style-type: none"> ◦ Error message shows ◦ Confirm won't continue • Incorrect values inputted in fields <ul style="list-style-type: none"> ◦ Error message shows ◦ Confirm won't continue • Connection is lost <ul style="list-style-type: none"> ◦ Error message shows ◦ Attempts to reconnect
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Clear directions are displayed for input account info and error messages ◦ Straightforward and intuitive steps • Performance <ul style="list-style-type: none"> ◦ After info is entered, account log in takes less than 2 seconds.

Use Case #4:	Edit Customer Account
Actors:	Customer
Pre-Conditions:	<ul style="list-style-type: none"> • Customer is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Customer Taps Settings Tab • Customer Taps Account Management <ul style="list-style-type: none"> ◦ System brings up account Management Options • Customer Taps Edit Account • Customer edits information • Customer taps "Confirm Change" <ul style="list-style-type: none"> ◦ Information is updated to server
Post Conditions:	<ul style="list-style-type: none"> • Information is updated to server

Error Conditions:	<ul style="list-style-type: none"> • Invalid input <ul style="list-style-type: none"> ◦ Error message shows • Blank input on required fields <ul style="list-style-type: none"> ◦ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Clear directions are displayed for input account info and error messages ◦ Easy to find this option • Performance <ul style="list-style-type: none"> ◦ Changes are made within 30 seconds

Use Case #5:	Change Customer Password
Actors:	Customer
Pre-Conditions:	<ul style="list-style-type: none"> • Customer is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Customer Taps Settings • Customer Taps Account Management <ul style="list-style-type: none"> ◦ System brings up account Management Options • Customer Taps Change Password • Customer Enters old password • Customer Enters new password • Confirms new password • Customer Confirms <ul style="list-style-type: none"> ◦ Information is verified ◦ Information is added to database
Post Conditions:	<ul style="list-style-type: none"> • Edited Information is added to server
Error Conditions:	<ul style="list-style-type: none"> • Invalid input <ul style="list-style-type: none"> ◦ Error message shows • Blank input on required fields <ul style="list-style-type: none"> ◦ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Clear directions are displayed for input and error messages ◦ Easy to find this option

	<ul style="list-style-type: none"> • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second
--	---

Use Case #6:	Add Payment Method
Actors:	Customer
Pre-Conditions:	<ul style="list-style-type: none"> • Customer is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Customer Taps Settings • Customer Taps Account Management <ul style="list-style-type: none"> ◦ System brings up account Management Options • Customer Taps Manage Payment Method • Customer Taps Add a Card • Customer Enters Card Information • Customer Confirms <ul style="list-style-type: none"> ◦ Information is verified ◦ Information is added to database
Post Conditions:	<ul style="list-style-type: none"> • Information is added to server
Error Conditions:	<ul style="list-style-type: none"> • Invalid input <ul style="list-style-type: none"> ◦ Error message shows • Blank input on required fields <ul style="list-style-type: none"> ◦ Error message shows • Invalid Card <ul style="list-style-type: none"> ◦ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Clear directions are displayed for input account info and error messages ◦ Easy to find this option • Performance <ul style="list-style-type: none"> ◦ Additions are made within 2 seconds

Use Case #7:	Edit Payment Method
Actors:	Customer
Pre-Conditions:	<ul style="list-style-type: none"> • Customer is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Customer Taps Settings • Customer Taps Account Management <ul style="list-style-type: none"> ◦ System brings up account Management Options • Customer Taps Manage Payment Method • Customer Taps Edit • Customer Selects Card • Customer edits info • Customer Confirms <ul style="list-style-type: none"> ◦ Information is verified ◦ Information is added to database
Post Conditions:	<ul style="list-style-type: none"> • Edited Information is added to server
Error Conditions:	<ul style="list-style-type: none"> • Invalid input <ul style="list-style-type: none"> ◦ Error message shows • Blank input on required fields <ul style="list-style-type: none"> ◦ Error message shows • Invalid Card <ul style="list-style-type: none"> ◦ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Clear directions are displayed for input account info and error messages ◦ Easy to find this option • Performance <ul style="list-style-type: none"> ◦ Additions are made within 2 seconds

Use Case #8	Open Tab
Actors:	Customer
Pre-Conditions:	<ul style="list-style-type: none"> • Customer is logged in

	<ul style="list-style-type: none"> • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Customer taps “Mobile Pay” tab • Customers clicks on available “Current Tab” <ul style="list-style-type: none"> ○ System brings up any check that is opened to the associated profile.
Post Conditions:	<ul style="list-style-type: none"> • Tab is added to Customer profiles in the system
Error Conditions:	<ul style="list-style-type: none"> • No Open Checks <ul style="list-style-type: none"> ○ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ○ Clear directions are displayed for input and error messages ○ Easy to find this option • Performance <ul style="list-style-type: none"> ○ Creation of tab is made within 3 second

Use Case #9:	Checkout
Actors:	Customer
Pre-Conditions:	<ul style="list-style-type: none"> • Customer is logged in • Device is connected to internet • Tab is opened in Database
Flow Of Control:	<ul style="list-style-type: none"> • Customer Taps “Mobile Pay” • Customer Selects Current Tab <ul style="list-style-type: none"> ○ Tab opens • Customer Taps Checkout <ul style="list-style-type: none"> ○ Review Screen Opens • Customer Selects Payments Option • Customer Confirms • Options: <ol style="list-style-type: none"> 1. Customer Pays by CC <ol style="list-style-type: none"> a. Next screen will show credit card transaction page. <ol style="list-style-type: none"> i. Credit Card Input b. Customer Confirms Payment Method

	<ul style="list-style-type: none"> c. Credit Card Charged <p>2. Customer Pays by Cash</p> <ul style="list-style-type: none"> a. Server arrives to table with check. • Customer Enters Gratuity Amount • Customer Confirms • Customer can Leave messages for Employee. • Customer Clicks "Payment Submit"
Post Conditions:	<ul style="list-style-type: none"> • Tab is closed
Error Conditions:	<ul style="list-style-type: none"> • No Open Checks <ul style="list-style-type: none"> ◦ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Clean Easy to find Payment options • Performance <ul style="list-style-type: none"> ◦ Payment Checkout Option Screen will process within 1 second

Use Case #10:	Customer Membership Card
Actors:	Customer
Pre-Conditions:	<ul style="list-style-type: none"> • Customer is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Customer Taps "Scan" • Customer Views "Member Card" <ul style="list-style-type: none"> ◦ QR Code is shown and linked to Membership ID for quicker access ◦ Shows Profile Name and Membership ID as well. • Customer Clicks "Exit"
Post Conditions:	<ul style="list-style-type: none"> • Customer profile information will be linked to the tab. <ul style="list-style-type: none"> ◦ Linked by the Employee
Error Conditions:	<ul style="list-style-type: none"> • QR Code not shown <ul style="list-style-type: none"> ◦ Error message shows

Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Clean Easy to find Scan • Performance <ul style="list-style-type: none"> ◦ Scan option Screen will be shown within 1 second
--	--

Employee Use Cases

Use Case #11:	Create Restaurant Account
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Device is installed • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps “Restaurant Account” Button <ul style="list-style-type: none"> ◦ App loads Sign In Page • Employee Taps “Create Account” • Employee Enters necessary Info • Employee Taps “Create Account” <ul style="list-style-type: none"> ◦ App verifies information ◦ Information is added to database ◦ Account is Created
Post Conditions:	<ul style="list-style-type: none"> • Unique Tag (Employee ID) is assigned to Employee • Account is added to database
Error Conditions:	<ul style="list-style-type: none"> • Blank input on required fields <ul style="list-style-type: none"> ◦ Error message shows ◦ Confirm won’t continue • Incorrect values inputted in fields <ul style="list-style-type: none"> ◦ Error message shows ◦ Confirm won’t continue • Email is already in database <ul style="list-style-type: none"> ◦ Error message shows ◦ Confirm won’t continue • Connection is lost <ul style="list-style-type: none"> ◦ Error message shows ◦ Attempts to reconnect
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability:

	<ul style="list-style-type: none"> ○ Clear directions are displayed for input account info and error messages ○ Straightforward and intuitive steps ● Performance <ul style="list-style-type: none"> ○ After info is entered, account creation takes less than 2 seconds.
--	--

Use Case #12:	Employee Login
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> ● Device is installed ● Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> ● Employee Taps “Username” field <ul style="list-style-type: none"> ○ Type Username ● Employee Taps “Password” field <ul style="list-style-type: none"> ○ Type Password ● Employee Clicks “Log In” <ul style="list-style-type: none"> ○ App verifies information ○ Information is checked with the database.
Post Conditions:	<ul style="list-style-type: none"> ● Employee is Logged In to Application <ul style="list-style-type: none"> ○ Able to view Features
Error Conditions:	<ul style="list-style-type: none"> ● Blank input on required fields <ul style="list-style-type: none"> ○ Error message shows ○ Confirm won’t continue ● Incorrect values inputted in fields <ul style="list-style-type: none"> ○ Error message shows ○ Confirm won’t continue ● Connection is lost <ul style="list-style-type: none"> ○ Error message shows ○ Attempts to reconnect
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> ● Usability: <ul style="list-style-type: none"> ○ Clear directions are displayed for input account info and error messages ○ Straightforward and intuitive

	steps <ul style="list-style-type: none"> • Performance <ul style="list-style-type: none"> ○ After info is entered, account log in takes less than 2 seconds.
--	---

Use Case #13:	Edit Employee Account
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps Settings Tab • Employee Taps Account Management <ul style="list-style-type: none"> ○ System brings up account Management Options • Employee Taps Edit Account • Employee reedits information • Employee taps “Confirm Change” <ul style="list-style-type: none"> ○ Information is updated to server
Post Conditions:	<ul style="list-style-type: none"> • Information is updated to server
Error Conditions:	<ul style="list-style-type: none"> • Invalid input <ul style="list-style-type: none"> ○ Error message shows • Blank input on required fields <ul style="list-style-type: none"> ○ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ○ Clear directions are displayed for input account info and error messages ○ Easy to find this option • Performance <ul style="list-style-type: none"> ○ Changes are made within 30 seconds

Use Case #14:	Change Employee Password
----------------------	---------------------------------

Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps Settings • Employee Taps Account Management <ul style="list-style-type: none"> ◦ System brings up account Management Options • Employee Taps Change Password • Employee Enters old password • Employee Enters new password • Employee Confirms new password • Employee Confirms change <ul style="list-style-type: none"> ◦ Information is verified ◦ Information is added to database
Post Conditions:	<ul style="list-style-type: none"> • Edited Information is added to server
Error Conditions:	<ul style="list-style-type: none"> • Invalid input <ul style="list-style-type: none"> ◦ Error message shows • Blank input on required fields <ul style="list-style-type: none"> ◦ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Clear directions are displayed for input and error messages ◦ Easy to find this option • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second

Use Case #15:	Register Employee
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps Restaurant Management <ul style="list-style-type: none"> ◦ System brings up account Management Options • Employee Taps Manage Employee

	<ul style="list-style-type: none"> • Employee Taps Register Employee <ul style="list-style-type: none"> ◦ System request admin credentials • Enter Employee Info • Enter Employee authorization level • Employee Confirms <ul style="list-style-type: none"> ◦ Information is added to database
Post Conditions:	<ul style="list-style-type: none"> • Edited Information is added to server
Error Conditions:	<ul style="list-style-type: none"> • Invalid input <ul style="list-style-type: none"> ◦ Error message shows • Blank input on required fields <ul style="list-style-type: none"> ◦ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Clear directions are displayed for input and error messages ◦ Easy to find this option • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second

Use Case #16:	Remove Employee
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps Restaurant Management <ul style="list-style-type: none"> ◦ System brings up account Management Options • Employee Taps Manage Employee • Employee Taps Remove Employee <ul style="list-style-type: none"> ◦ System request admin credentials • Employee Selects Employee to remove • Employee Confirms <ul style="list-style-type: none"> ◦ Information is added to database

Post Conditions:	<ul style="list-style-type: none"> • Edited Information is added to server
Error Conditions:	<ul style="list-style-type: none"> • Invalid input <ul style="list-style-type: none"> ◦ Error message shows • Blank input on required fields <ul style="list-style-type: none"> ◦ Error message shows • Invalid Authorization <ul style="list-style-type: none"> ◦ Error Message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Clear directions are displayed for input and error messages ◦ Easy to find this option • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second

Use Case #17:	Create Menu Tab
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps “Restaurant Management” <ul style="list-style-type: none"> ◦ System brings up account Management Options • Employee Taps Manage Menu • Employee Taps Create Menu Tab <ul style="list-style-type: none"> ◦ System request admin credentials • Enter Menu Tab Name • Employee Confirms <ul style="list-style-type: none"> ◦ Information is verified ◦ Information is added to database
Post Conditions:	<ul style="list-style-type: none"> • Edited Information is added to server

Error Conditions:	<ul style="list-style-type: none"> • Blank input on required fields <ul style="list-style-type: none"> ◦ Error message shows • Invalid Authorization <ul style="list-style-type: none"> ◦ Error Message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Clear directions are displayed for input and error messages ◦ Easy to find this option • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second

Use Case #18:	Create Menu Item
Actors:	<ul style="list-style-type: none"> • Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps Restaurant Management <ul style="list-style-type: none"> ◦ System brings up account Management Options • Employee Taps on Manage Menu • Employee Taps Add Menu Item <ul style="list-style-type: none"> ◦ System request admin credentials • Enter Item Info such as Name, Price, and Picture(optional). • Enter Tab Selection • Employee Confirms <ul style="list-style-type: none"> ◦ Employee reviews info before submitting ◦ Information is added to database
Post Conditions:	<ul style="list-style-type: none"> • Edited Information is added to server
Error Conditions:	<ul style="list-style-type: none"> • Blank input on required fields <ul style="list-style-type: none"> ◦ Error message shows • Invalid Authorization <ul style="list-style-type: none"> ◦ Error Message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability:

	<ul style="list-style-type: none"> ○ Clear directions are displayed for input and error messages ○ Easy to find this option ● Performance <ul style="list-style-type: none"> ○ Additions are made within 2 second
--	--

Use Case #19:	Edit Menu Item
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> ● Employee is logged in ● Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> ● Employee Taps Restaurant Management <ul style="list-style-type: none"> ○ System brings up account Management Options ● Employee Taps on Manage Menu ● Employee Taps Edit Menu Item <ul style="list-style-type: none"> ○ System request admin credentials ● Employee Selects menu item to modify ● Edit Item Info Such as Name, Price, Picture, and Tab Selection ● Employee Confirms <ul style="list-style-type: none"> ○ Information is added to database
Post Conditions:	<ul style="list-style-type: none"> ● Edited Information is added to server
Error Conditions:	<ul style="list-style-type: none"> ● Blank input on required fields <ul style="list-style-type: none"> ○ Error message shows ● Invalid Authorization <ul style="list-style-type: none"> ○ Error Message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> ● Usability: <ul style="list-style-type: none"> ○ Clear directions are displayed for input and error messages ○ Easy to find this option ● Performance <ul style="list-style-type: none"> ○ Additions are made within 2 second

Use Case #20:	New Check
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps Orders tab • Employee Clicks “New Check” <ul style="list-style-type: none"> ○ Only use if wanting to add a new check. • Employee Input Table Number • Employee Clicks “Enter” • Employee Input Party Size • Employee Clicks “Enter” • Employee Confirms New Check <ul style="list-style-type: none"> ○ Tab is opened ○ System will show all menu buttons in the POS that will be punched in the system as well as to the kitchen.
Post Conditions:	<ul style="list-style-type: none"> • Tab added to current Tabs of Employee’s database
Error Conditions:	<ul style="list-style-type: none"> • Invalid input for Party Size <ul style="list-style-type: none"> ○ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ○ Quick option to do • Performance <ul style="list-style-type: none"> ○ Additions are made within 1 second

Use Case #21:	New Check by Proxy
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in

	<ul style="list-style-type: none"> • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps Orders tab • Employee Clicks “New Check by Proxy” <ul style="list-style-type: none"> ○ Employee List Opens • Employee scrolls to find other Employee’s name in Database • Employee Confirms Choice • Employee Inputs Table Number • Employee Clicks “Enter” • Employee Inputs Party Size • Employee Clicks “Enter” • Employee Confirms New Check by Proxy <ul style="list-style-type: none"> ○ Tab is opened ○ System will show all menu buttons in the POS that will be punched in the system as well as to the kitchen. ○
Post Conditions:	<ul style="list-style-type: none"> • Tab added to current Tabs of Employees Database
Error Conditions:	<ul style="list-style-type: none"> • Invalid input for Party Size <ul style="list-style-type: none"> ○ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ○ Quick option to do • Performance <ul style="list-style-type: none"> ○ Additions are made within 1 second

Use Case #22:	Recall Check
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps Orders tab • Employee Clicks “Recall Check”

	<ul style="list-style-type: none"> ○ Only use if wanting to modify open tabs. ● Employee Choose what Tab to modify ● Employee Clicks “Enter” <ul style="list-style-type: none"> ○ System will show all menu buttons in the POS that will be punched in the system as well as to the kitchen. ● Employee Confirms Balance
Post Conditions:	<ul style="list-style-type: none"> ● Modified Tab added to current Tabs of Employees Database
Error Conditions:	<ul style="list-style-type: none"> ● No Tabs Opened <ul style="list-style-type: none"> ○ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> ● Usability: <ul style="list-style-type: none"> ○ Quick option to do ● Performance <ul style="list-style-type: none"> ○ Additions are made within 1 second

Use Case #23:	Recall Check by Proxy
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> ● Employee is logged in ● Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> ● Employee Taps Orders tab ● Employee Clicks “Recall Check by Proxy” <ul style="list-style-type: none"> ○ Only use if wanting to modify open tabs for other Employees. ● Employees Finds Employee Name ● Employee Confirms choice by clicking “Enter” ● Employee Choose what Tab to modify ● Employee Clicks “Enter” <ul style="list-style-type: none"> ○ System will show all menu buttons in the POS that will be punched in the system as well as to the kitchen.

	<ul style="list-style-type: none"> • Employee Confirms Balance
Post Conditions:	<ul style="list-style-type: none"> • Edited Information will be modified in the current tab of the database of the Employee.
Error Conditions:	<ul style="list-style-type: none"> • No Tabs Opened <ul style="list-style-type: none"> ◦ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Quick option to do • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second

Use Case #24:	Change Party Size
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ◦ Opens POS buttons • Employee Clicks on “Party Size” • Employee Confirms updated party size with new value. • Employee Confirms Balance
Post Conditions:	<ul style="list-style-type: none"> • Edited Information will be modified in the current tab of the database.
Error Conditions:	<ul style="list-style-type: none"> • Invalid input for Party Size <ul style="list-style-type: none"> ◦ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Easy navigation for button ◦ Easy to understand how to change party size. • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second

Use Case #25:	Change Table Number
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ◦ Opens POS buttons • Employee Clicks on "Table Number" • Employee Confirms updated Table number with new value. • Employee Confirms Balance
Post Conditions:	<ul style="list-style-type: none"> • Edited Information will be modified in the current tab of the database.
Error Conditions:	<ul style="list-style-type: none"> • None
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Easy navigation for button ◦ Easy to understand how to change table number. • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second

Use Case #26:	Seat Number
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ◦ Opens POS buttons

	<ul style="list-style-type: none"> • Employee Clicks on “Seat Number” • Employee Confirms Seat Number value • Employee Click on Item <ul style="list-style-type: none"> ◦ Items will be associated with Seat Number on the tab. • Employee Confirms Balance
Post Conditions:	<ul style="list-style-type: none"> • Edited Information will be modified in the current tab of the database.
Error Conditions:	<ul style="list-style-type: none"> • None
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Easy navigation for button ◦ Easy to understand how to assign items to seat number . • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second

Use Case #27:	Balance
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ◦ Opens POS buttons • Employee Clicks on “Balance” <ul style="list-style-type: none"> ◦ Takes you back to home screen. ◦ Updates the total amount of the tab in the Database.
Post Conditions:	<ul style="list-style-type: none"> • Edited Information will be modified in the current tab of the database.
Error Conditions:	<ul style="list-style-type: none"> • None

Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ○ Easy navigation for button ○ Easy to understand how to void item. • Performance <ul style="list-style-type: none"> ○ Additions are made within 1 second
--	---

Use Case #28:	Balance Print
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ○ Opens POS buttons • Employee Clicks on “Balance” <ul style="list-style-type: none"> ○ Takes you back to home screen. ○ Updates the total amount of the tab in the Database. • Employee Receives print out of current tab.
Post Conditions:	<ul style="list-style-type: none"> • Edited Information will be modified in the current tab of the database.
Error Conditions:	<ul style="list-style-type: none"> • None
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ○ Easy navigation for button ○ Easy to understand how to void item. • Performance <ul style="list-style-type: none"> ○ Additions are made within 1 second

Use Case #29:	Credit Card Transaction
----------------------	--------------------------------

Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ◦ Opens POS buttons • Employee Clicks "Cash Out" button <ul style="list-style-type: none"> ◦ Payment Method buttons will appear. • Employee Selects "Charge" • Employee Chooses "Credit Card" • Employee Charges credit card. • Employee Receives payment authorization checks
Post Conditions:	<ul style="list-style-type: none"> • Tab is charged
Error Conditions:	<ul style="list-style-type: none"> • Invalid Credit Card <ul style="list-style-type: none"> ◦ Error Message will show • Insufficient Fund <ul style="list-style-type: none"> ◦ Error Message will show
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Easy navigation for button ◦ Easy to understand how to deal with charge tender • Performance <ul style="list-style-type: none"> ◦ Additions are made within 3 second

Use Case #30:	Cash Transaction
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ◦ Opens POS buttons

	<ul style="list-style-type: none"> • Employee Clicks “Cash Out” button <ul style="list-style-type: none"> ◦ Payment Method buttons will appear. • Employee Selects value of cash • Employee Chooses “Cash” <ul style="list-style-type: none"> ◦ Remaining balance will need to processed by credit card. • Employee Receives Total Amount of Change Receipt.
Post Conditions:	<ul style="list-style-type: none"> • Tab is charged
Error Conditions:	<ul style="list-style-type: none"> • Invalid Credit Card <ul style="list-style-type: none"> ◦ Error Message will show • Insufficient Fund <ul style="list-style-type: none"> ◦ Error Message will show
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Easy navigation for button ◦ Easy to understand how to deal with charge tender • Performance <ul style="list-style-type: none"> ◦ Additions are made within 3 second

Use Case #31:	Split Check
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ◦ Opens POS buttons • Employee Clicks on “Split Check” • Employee Enters how many checks • Employee Moves Items to each Check • Employee Confirms choice • Employee Receives the number of Checks that was split.
Post Conditions:	<ul style="list-style-type: none"> • Tab is split into multiple tabs

Error Conditions:	<ul style="list-style-type: none"> • None
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ○ Easy navigation for button ○ Easy to understand how to deal with charge tender • Performance <ul style="list-style-type: none"> ○ Additions are made within 3 second

Use Case #32:	Finalize Check
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ○ Opens POS buttons • Employee Clicks on Credit Card that was charged. • Employee Finalize Gratuity amount. • Employee Confirms amount by clicking enter. • Employee Receives confirmation papers.
Post Conditions:	<ul style="list-style-type: none"> • Tab is charged with gratuity
Error Conditions:	<ul style="list-style-type: none"> • None
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ○ Easy navigation for button ○ Easy to understand how to deal with charge tender • Performance <ul style="list-style-type: none"> ○ Additions are made within 3 second

Use Case #33:	Allergy
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ◦ Opens POS buttons • Employee Clicks on “Allergy” • Employee Clicks on specific allergy • Employee Click on Item <ul style="list-style-type: none"> ◦ Allergy will be associated with Food Item on the tab. • Employee Confirms Balance
Post Conditions:	<ul style="list-style-type: none"> • Allergy Information will be modified in the current tab of the database.
Error Conditions:	<ul style="list-style-type: none"> • None
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Easy navigation for button ◦ Easy to understand how to include allergies. • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second

Use Case #34:	Special Instruction
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ◦ Opens POS buttons • Employee Clicks on “Special Instruction”

	<ul style="list-style-type: none"> • Employee Types in specific modifications for and item. • Employee Click on Item <ul style="list-style-type: none"> ◦ Special Instruction will be associated with Food Item on the tab. • Employee Confirms Balance
Post Conditions:	<ul style="list-style-type: none"> • Special Instruction Information will be modified in the current tab of the database.
Error Conditions:	<ul style="list-style-type: none"> • None
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Easy navigation for button ◦ Easy to understand how to include special instruction. • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second

Use Case #35:	Apply Discount
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ◦ Opens POS buttons • Employee Clicks on “Discount” under manager authorization button. <ul style="list-style-type: none"> ◦ Database will check for Employee’s authorization level. ◦ Only allow for Full Authorization/Partial Authorization Employees • Employee Selects types of Discount percentage • Employee Confirms Balance

Post Conditions:	<ul style="list-style-type: none"> • Edited Information will be modified in the current tab of the database.
Error Conditions:	<ul style="list-style-type: none"> • Invalid Authorization <ul style="list-style-type: none"> ◦ Error Message shows • Invalid Comp <ul style="list-style-type: none"> ◦ Error Message shows ◦ Deals with repeated comps on an individual item.
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Easy navigation for button ◦ Easy to understand how to discount check. • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second

Use Case #36:	Complimentary Item
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ◦ Opens POS buttons • Employee Clicks on “Comp Item” under manager authorization button. <ul style="list-style-type: none"> ◦ Database will check for Employee’s authorization level. ◦ Only allow for Full Authorization/Partial Authorization Employees • Employee Selects item to Comp • Employee Confirms Balance
Post Conditions:	<ul style="list-style-type: none"> • Edited Information will be modified in the current tab of the database.

Error Conditions:	<ul style="list-style-type: none"> • Invalid Authorization <ul style="list-style-type: none"> ◦ Error Message shows • Invalid Comp <ul style="list-style-type: none"> ◦ Error Message shows ◦ Deals with repeated comps on an individual item.
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> • Usability: <ul style="list-style-type: none"> ◦ Easy navigation for button ◦ Easy to understand how to comp item. • Performance <ul style="list-style-type: none"> ◦ Additions are made within 1 second

Use Case #37:	Void Item
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> • Employee is logged in • Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> • Employee Taps on Order • Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ◦ Opens POS buttons • Employee Clicks on "Void Item" under manager authorization button. <ul style="list-style-type: none"> ◦ Database will check for Employee's authorization level. ◦ Only allow for Full Authorization/Partial Authorization Employees • Employee Selects item to Void • Employee Confirms Balance
Post Conditions:	<ul style="list-style-type: none"> • Edited Information will be modified in the current tab of the database.
Error Conditions:	<ul style="list-style-type: none"> • Invalid Authorization <ul style="list-style-type: none"> ◦ Error Message shows • Invalid Comp

	<ul style="list-style-type: none"> ○ Error Message shows ○ Deals with repeated comps on an individual item.
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> ● Usability: <ul style="list-style-type: none"> ○ Easy navigation for button ○ Easy to understand how to void item. ● Performance <ul style="list-style-type: none"> ○ Additions are made within 1 second

Use Case #38:	Employee Links Membership ID
Actors:	Employee
Pre-Conditions:	<ul style="list-style-type: none"> ● Employee is logged in ● Device is connected to internet
Flow Of Control:	<ul style="list-style-type: none"> ● Employee Taps on Order ● Employee Clicks on Recall Check/Recall Check by Proxy <ul style="list-style-type: none"> ○ Opens POS buttons ● Employee Clicks on “Membership” ● Employee Inputs Membership ID <ul style="list-style-type: none"> ○ Ability to scan QR code for faster convenience. ● Employee Confirms Balance
Post Conditions:	<ul style="list-style-type: none"> ● Edited Information will be modified in the current tab of the database. ● Customer Membership ID is linked to the Current Tab.
Error Conditions:	<ul style="list-style-type: none"> ● Invalid Membership ID <ul style="list-style-type: none"> ○ Error message shows ● Invalid QR Code <ul style="list-style-type: none"> ○ Error message shows
Non-Functionality/Quality Requirement:	<ul style="list-style-type: none"> ● Usability: <ul style="list-style-type: none"> ○ Easy navigation for button ○ Easy to understand how to input membership ID. ● Performance

- | | |
|--|---|
| | <ul style="list-style-type: none">○ Inputs are made within 1 second |
|--|---|

5. Nonfunctional Requirements

5.1. Performance

5.1.1. Performance under 1 second

- Use Case #5: Change Customer Password
- Use Case #9: Checkout
- Use Case #10: Customer Membership Card
- Use Case #14: Change Employee Password
- Use Case #15: Register Employee
- Use Case #16: Remove Employee
- Use Case #17: Create Menu Tab
- Use Case #20: New Check
- Use Case #21: New Check By Proxy
- Use Case #22: Recall Check
- Use Case #23: Recall Check by Proxy
- Use Case #24: Change Party Size
- Use Case #25: Change Table Number
- Use Case #26: Seat Number
- Use Case #27: Balance
- Use Case #28: Balance Print
- Use Case #33: Allergy
- Use Case #34: Special Instructions
- Use Case #35: Apply Discount
- Use Case #36: Complimentary Item
- Use Case #37: Void Item
- Use Case #38: Employee Links Membership ID

5.1.2. Performance under 2 seconds

- Use Case #1: Create Customer Account
- Use Case #2: Customer Log In
- Use Case #3: Customer Log In (Google Authentication)
- Use Case #6: Add Payment Method
- Use Case #7: Edit Payment Method
- Use Case #11: Create Restaurant Account
- Use Case #12: Employee Log In
- Use Case #18: Create Menu Item
- Use Case #19: Edit Menu Item

-
- 5.1.3. Performance under 3 seconds
 - Use Case #8: Open Tab
 - Use Case #29: Credit Card Transaction
 - Use Case #30: Cash Transaction
 - Use Case #31: Split Check
 - Use Case #32: Finalize Check
- 5.1.4. Performance under 30 seconds
 - Use Case #4: Edit Customer Account
 - Use Case #13: Edit Employee Account

6. Other Requirements

None.