Donut Shop Website Information System Project

System Requirements Documentation

Jonah Reardon

Table of Contents

Customer Problem Statements and System Requirements

Customer Problem Statement
Problem Statement
Donut shop customers are trying to find donut shops near them while on the go but it takes a long time because there is no organized website or mobile version which makes them feel frustrated. Donut shop owners and managers find it difficult to keep track of all of the newly opened, existing, and closed stores. The donut shop managers find it hard to manage all of the store information because there is no centralized website which makes them feel overwhelmed. The donut shop customers expect that a website software system would help by allowing them to view store information digitally around the world and while on the go with a mobile device. The donut shop owners and managers expect that the website software system would help by providing a centralized and accessible website to keep store information up to date. Plus, the shop owners believe that the system provides an opportunity for the business to grow, expand, and possibly get more customers by being discoverable on the Internet.

Glossary of Terms
Donut shop: A donut shop is a physical building or place that sells donuts and possibly other food or beverages. The donut shop has a location which includes a street address, city, state, country, zip code. Donut shops also have store hours which includes when the store opens and closes. The store also has a store ID, name, and phone number. A graphic of a donut shop is below.

Donut shop customers: Donut shop customers are people who buy food or other goods from the donut shop.

Donut shop owners: Donut shop owners are people who own the donut shops.

Donut shop managers: Donut shop managers are people who manage the donut shops and its employees. They are people who are in charge of coordinating the work of other donut shop employees. Some managers can supervise or oversee other employees. Other managers can help manage the allocation of donut shop resources.

Donut shop store information: Donut shop store information is organized, meaningful data or a collection of facts about the donut shops. The store information contains information about the locations such as street address, city, state, country, and zip code that are connected to each specific donut shop. Other donut shop information include the store hours, store ID, name, and phone number that is made meaningful by connecting the data to each donut shop.

System Requirements

Functional Requirements

No.	**Priority Weight**	**Description**
REQ-1	High	The system should have the capacity to contain at least 10 donut shop records.
REQ-2	High	Four different types of users (guests, donut shop customers, donut shop owners, and donut shop managers) should be able to view donut shop information.
REQ-3	High	The three different types of donut shops are gourmet, vegan, and variety.
REQ-4	Medium	Logged in donut shop customers should be able to favorite stores and view a list of their favorite stores on the home page.
REQ-5	High	Only 2 different types of users (donut shop owners and donut shop managers) should be allowed to add, edit, or delete donut shops in the system.
REQ-6	High	The system should not allow duplicates of stores. They cannot have the same store ID or store name.
REQ-7	High	A donut shop owner or manager must input donut shop data into the website system.
REQ-8	High	Response time for loading webpages must not exceed 5 seconds.
REQ-9	High	The system must be operational 24 hours, seven days a week (24/7).
REQ-10	Medium	The system must filter for shops based on the city or postal code in less than 4 seconds.
REQ-11	High	A donut shop record must be added, deleted, or changed only by a donut shop owner or manager.
REQ-12	High	The system must provide logon security for the application.

REQ-13	High	The system must allow users to log into their account by entering their username and password.
REQ-14	High	The system must allow users to log into their accounts on mobile by entering their username and password.
REQ-15	High	The system must allow users to reset their passwords by clicking on "I forgot my password".
REQ-16	Low	The system should automatically put stores into viewable categories based on their city location.
REQ-17	High	The database records need to be updated in real time

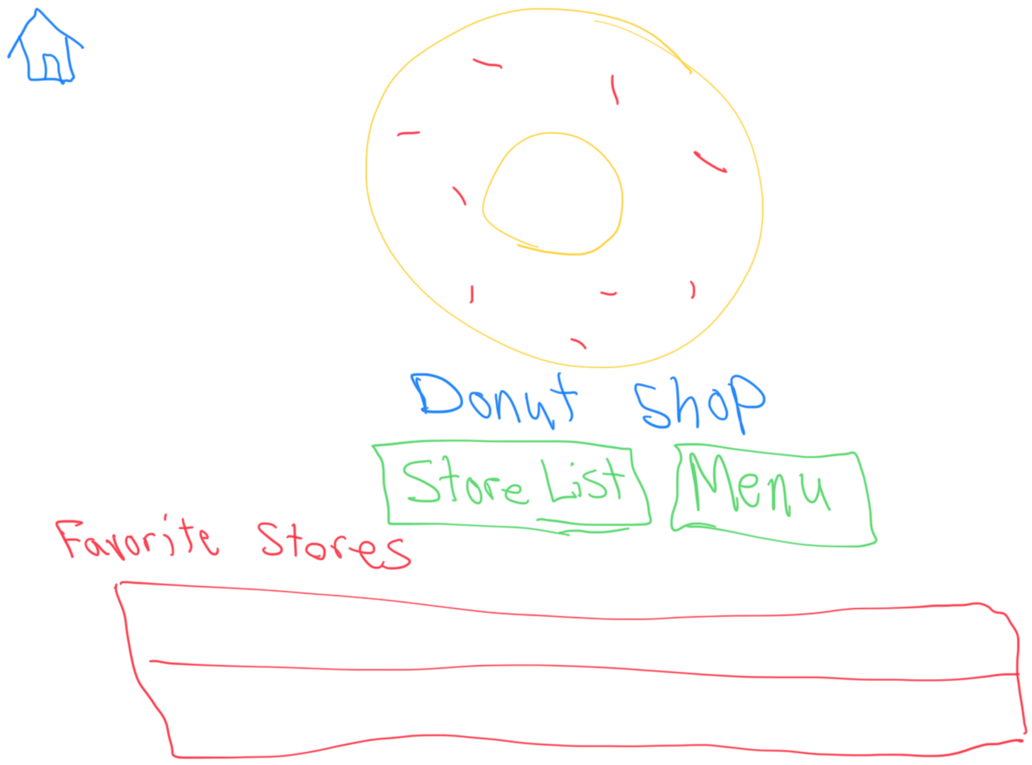
Nonfunctional Requirements
Some of the requirements listed in the table above are non-functional requirements such as REQ-8, REQ-9, REQ-10, and REQ-17. These non-functional requirements are criteria about how the system should perform within categories such as the performance or speed of the system. Other categories include functionality, usability, reliability, and supportability.

User Interface Requirements

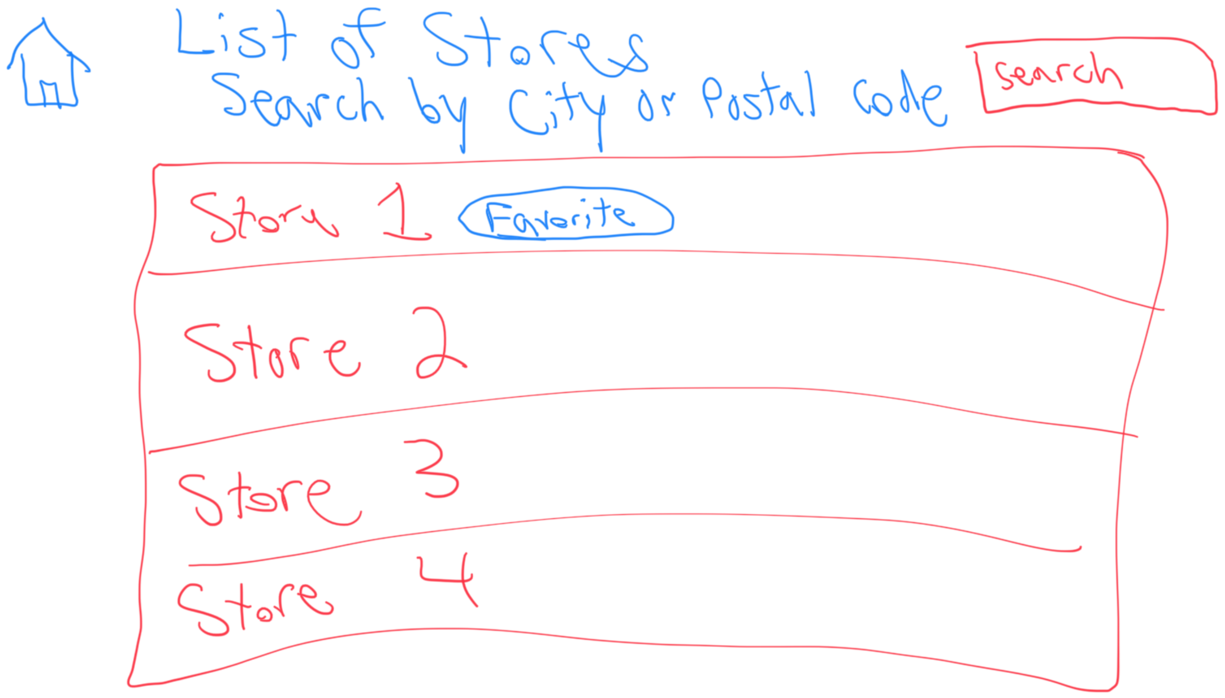
UI REQ No.	**Priority Weight**	**Description**
UIREQ-1	Medium	If the donut shop customer has favorited a store, the system and user interface should display a list of their favorite stores on the homepage.
UIREQ-2	Medium	If the donut shop customer has favorited a store, the system and user interface should display a favorite store tag in the store list page.
UIREQ-3	High	The user interface should allow a donut shop customer to favorite their store. This can be done by using a switch that the user can toggle.
UIREQ-4	High	If the user inputs the wrong login information, the user interface should display a notification and highlight the input boxes that need to be fixed.
UIREQ-5	High	If the user does not input information where they need to or inputs information that is in the wrong format, the user interface should display a notification and highlight the input boxes that need to be fixed.

UIREQ-6	High	The user interface should provide easy navigation and be quick to respond.
UIREQ-7	High	The user interface should have a homepage button (house icon) to quickly get back to the main page for faster navigation and for if the user ever gets lost.

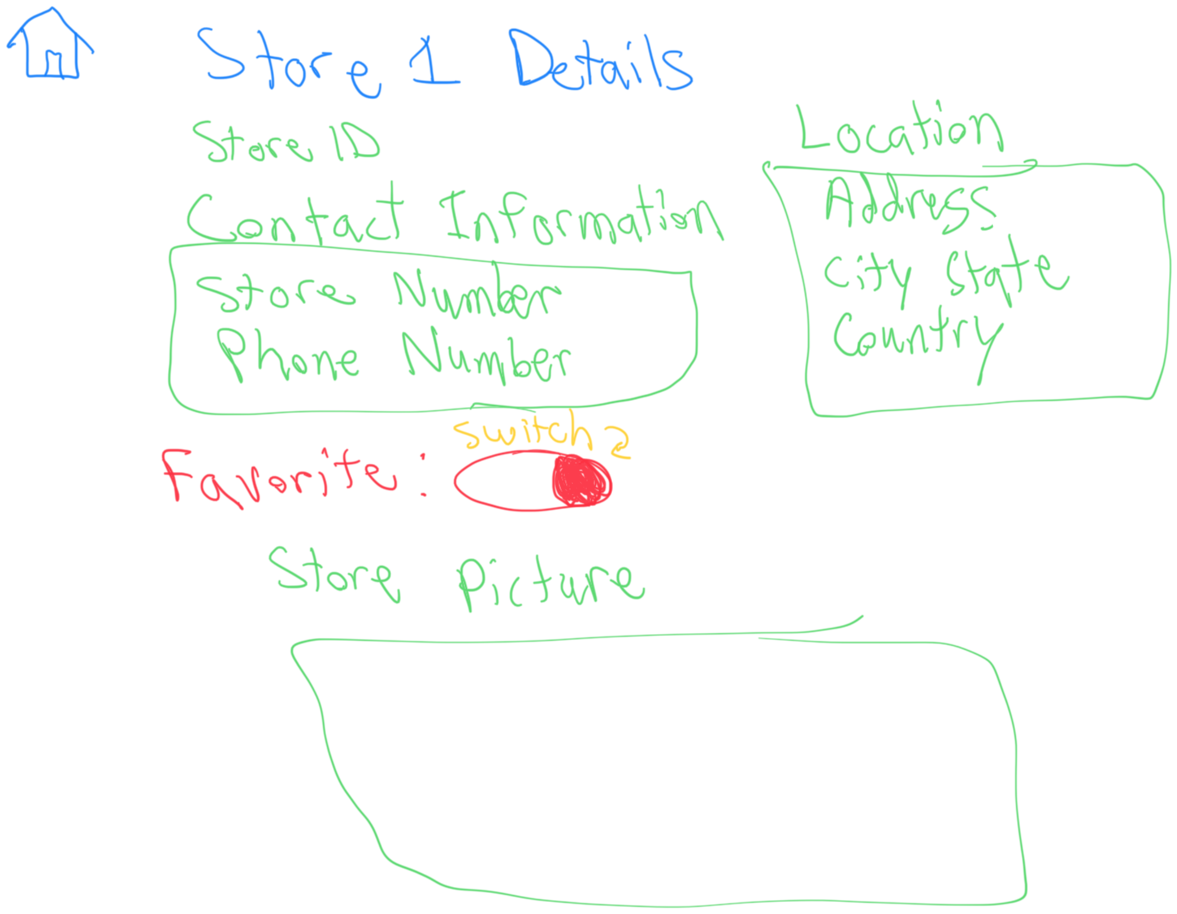
Favorite stores on Homepage:



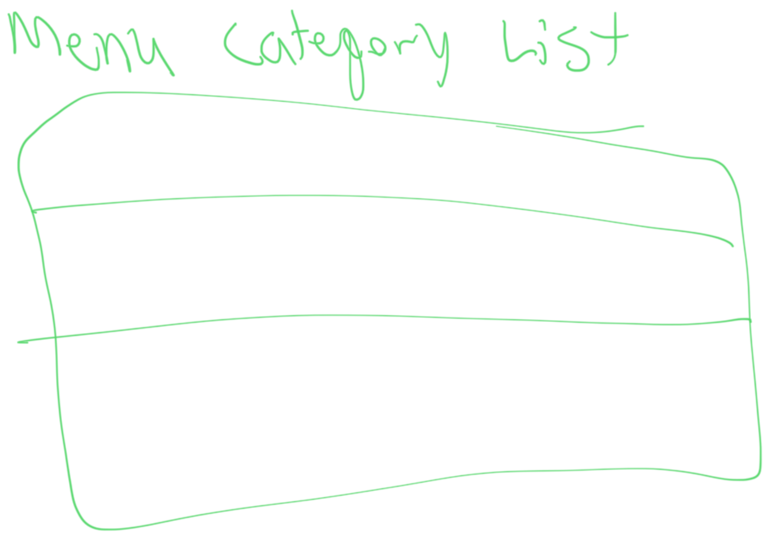
Favorite Stores Tag in List of Stores Page:



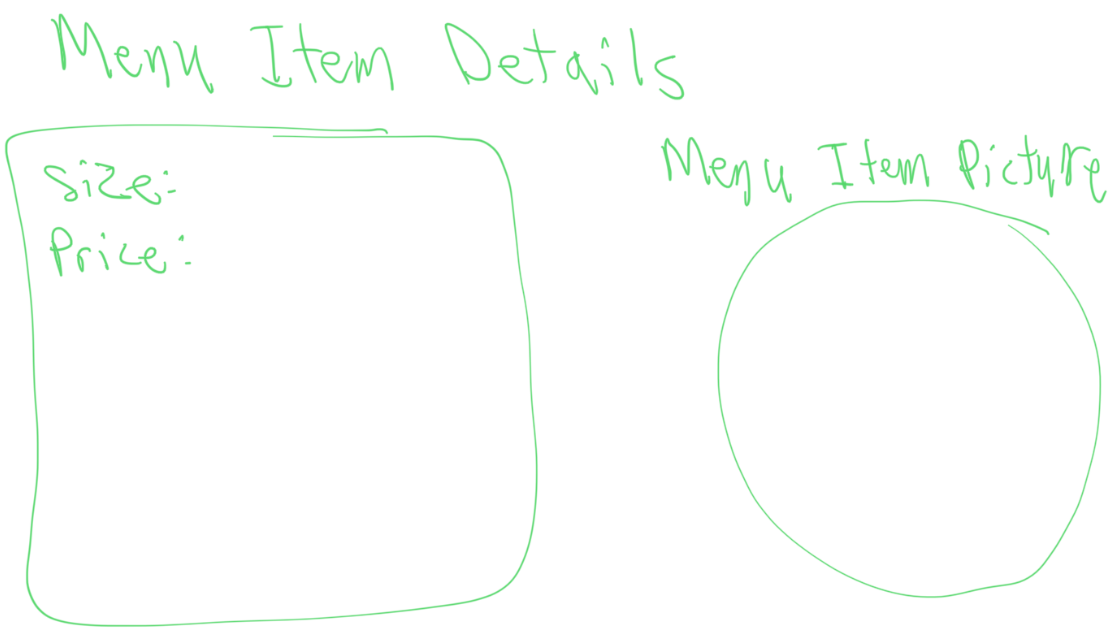
Store Details Page with Store Information and Favorite Switch:



Donut Shop Menu Category List Page:



Donut Shop Menu Items Details:



Plan of Work

✅Week 1: Solidify what front-end, back-end, and database programs I am going to use.
✅Week 2: Set up a project development environment by connecting front-end, back-end, and database software together, figure out the structure and framework for the system.
✅Week 3: Create the login for the system so customers and shop owners can login.
✅Week 4: Create the registration for new customers and shop owners to make a login or account.
Week 5: Design the features for shop owners to add, modify, or remove stores, store details (store name, phone number, street address, city, state, country, zip code), food categories, menu items, and menu item details.
Week 6: Implement the features for shop owners to add, modify, or remove stores, store details (store name, phone number, street address, city, state, country, zip code), food categories, menu items, and menu item details.
Week 7: Continue to implement the features for shop owners to add, modify, or remove stores, store details (store name, phone number, street address, city, state, country, zip code), food categories, menu items, and menu item details.
Week 8: Verify the features work, record, and turn in current progress for the midterm.
Week 9: Iterate and improve current features for the shop owners.
Week 10: Design the features for customers to be able to view the stores and filter for stores based on their city or postal code. Design the features for customers to view basic store, menu category, and menu item information
Week 11: Implement the features for customers to be able to view the stores and filter for stores based on their city or postal code. Implement the features for customers to view basic store, menu category, and menu item information
Week 12: Design the features for customers to be able to favorite stores and menu items.

Week 13: Implement the features for customers to be able to favorite stores and menu items.
Week 14: Debug and test to make sure all features work as intended.
Week 15: Test and record the demonstration and progress made for the final presentation.

Choosing and solidifying what front-end, back-end, and database programs I was going to use was harder than I thought. I had to do a lot of research to figure out the best technology stack I think I should use for this project. It took a while to learn the differences between all of the software programs.
Setting up the development environment took longer than I anticipated to try to connect the front-end, back-end, and database software together. Creating a login for the system and creating the different types of users so they have access to different parts of the system was more straightforward and similar to the way other software programs handle logging in. I started designing some of the features and figuring out the user interface. I will continue to work on designing, programming, and adding features to the system throughout the rest of the semester.

Functional Requirement Specification

Stakeholders

- Donut Shop customers
- Donut Shop owners
- Donut Shop managers
- Donut Shop employees
- Donut Shop sponsor
- Donut Shop CEO
- Donut Shop president
- Donut Shop board of directors

Actors and Goals

Primary Actors

- Customer: This actor can filter to view stores and their details based on zip code. They can also favorite their stores and menu items to make it faster to find their donut shop and get the products they want.

Secondary Actors

- Donut Shop Manager: This actor can add, remove, or modify stores, their store details, store locations, food categories, menu items, and menu item details.
- System: This is responsible for providing details of favorited or filtered stores and menu items.

Use Cases

Donut Shop Manager (total: 30)

- Add store: to add a new store (2)
- Add food category: to add a new food category (2)
- Add menu item: to add a new menu item (2)
- Add menu item details: to add details about a menu item (2)
- Modify store: to modify information about a store (2)
- Modify food category: to modify a food category (2)
- Modify menu item: to modify a menu item (2)
- Modify menu item details: to modify details about a menu item (2)
- Remove store: to remove a store (2)
- Remove food category: to remove a food category (2)
- Remove menu item: to remove a menu item (2)
- Remove menu item details: to remove details about a menu item (2)
- View account: to view account details (2)
- Login/Logout: to login or logout from donut shop manager account (4)

Customer (total: 30)

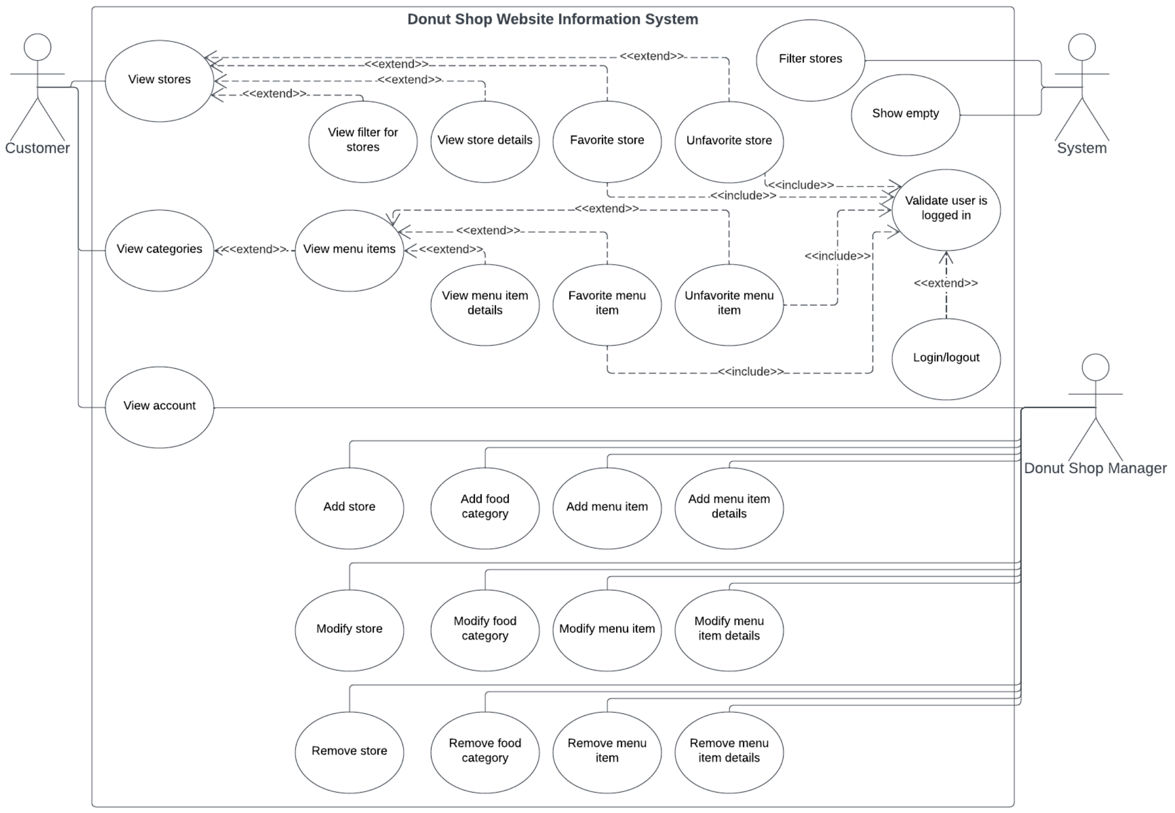
- Favorite store: to take a store and designate it as a favorite (3)
- Unfavorite store: to take a store and designate it as not a favorite anymore (3)
- Favorite menu item: to take a menu item and designate it as a favorite (3)
- Unfavorite menu item: to take a store and designate it as not a favorite anymore (2)
- View stores: to view basic store information (2)
- View store details: to view the store detailed information (2)
- View categories: to view menu categories (2)
- View menu items: to view menu items (2)
- View menu item details: to view menu item details (2)
- View filter for stores: to view the filter for stores based on zip code (3)
- View account: to view account details (2)
- Login/Logout: to login or logout from donut shop customer account (4)

System (total: 7)

- Filter stores: to get the stores filtered (3)
- Show empty: to display the status that there are no shops (2)
- Validate user is logged in: to make sure the user is logged in to their account (2)

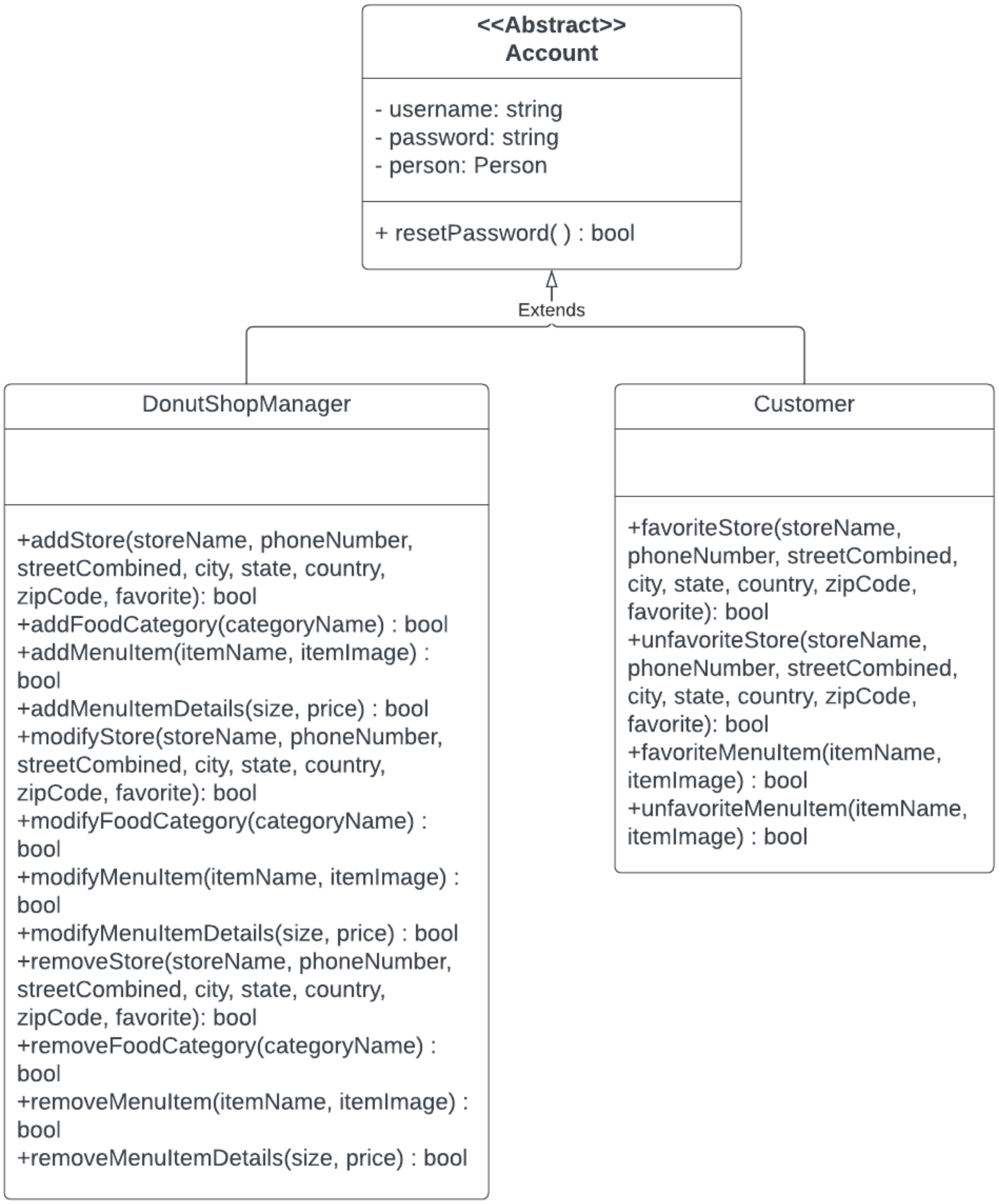
Use Case Diagram

The <<extend>> arrows indicate what the actor can do after another specific use case. However, they do not have to do these use cases.
The <<include>> arrows indicate the use case that has to be done after another use case such as validating that a user is logged in to be able to favorite something.



Class Diagram

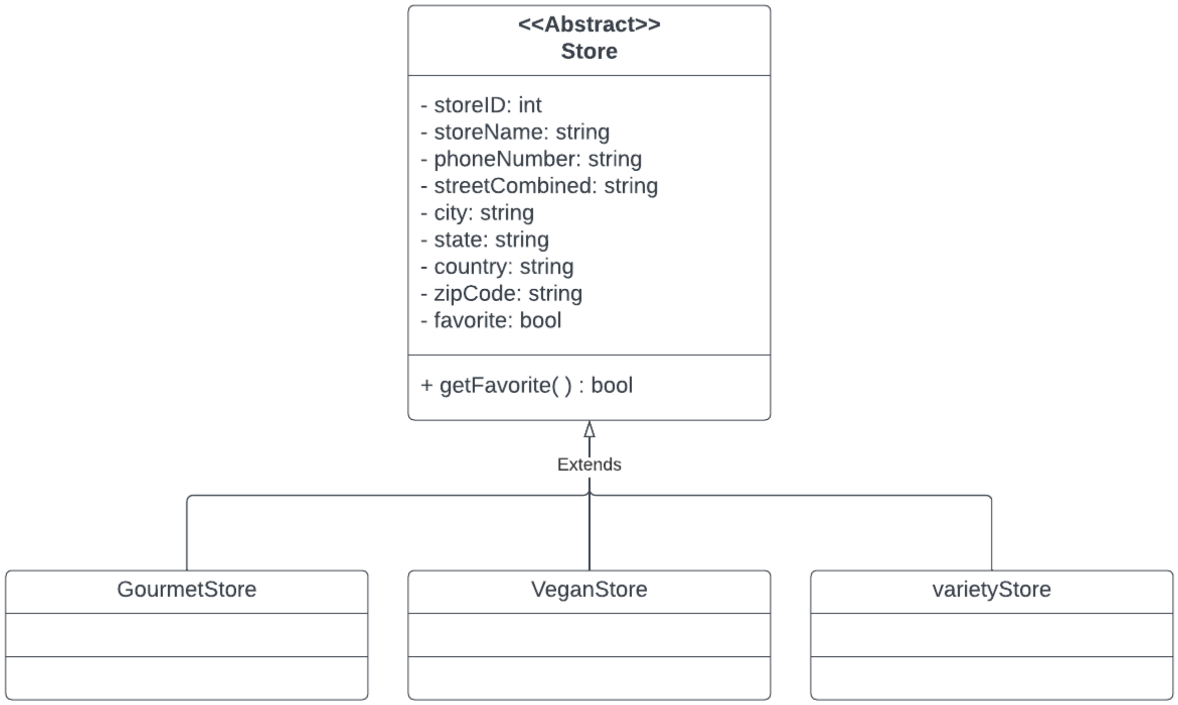
Account
Based on our requirements, there should be two types of accounts. One type of account is a Donut Shop Manager Account and another type of account is a Customer Account. We can define Account as an abstract class and both Donut Shop Manager and Customer as extended classes. Abstraction is more scalable and allows us to add more types of accounts later on.



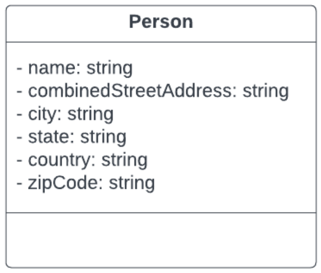
<<Abstract>>
Account
- username: string
- password: string
- person: Person
+ resetPassword( ) : bool
Extends
DonutShopManager
+addStore(storeName, phoneNumber, streetCombined, city, state, country, zipCode, favorite): bool
+addFoodCategory(categoryName) : bool
+addMenuItem(itemName, itemImage) : bool
+addMenuItemDetails(size, price) : bool
+modifyStore(storeName, phoneNumber, streetCombined, city, state, country, zipCode, favorite): bool
+modifyFoodCategory(categoryName) : bool
+modifyMenuItem(itemName, itemImage) : bool
+modifyMenuItemDetails(size, price) : bool
+removeStore(storeName, phoneNumber, streetCombined, city, state, country, zipCode, favorite): bool
+removeFoodCategory(categoryName) : bool
+removeMenuItem(itemName, itemImage) : bool
+removeMenuItemDetails(size, price) : bool
Customer
+favoriteStore(storeName, phoneNumber, streetCombined, city, state, country, zipCode, favorite): bool
+unfavoriteStore(storeName, phoneNumber, streetCombined, city, state, country, zipCode, favorite): bool
+favoriteMenuItem(itemName, itemImage) : bool
+unfavoriteMenuItem(itemName, itemImage) : bool

Store

Based on our requirements, there should be three types of stores. The three types of stores are Gourmet Store, Vegan Store, and Variety Store. We can define Store as an abstract class and have Gourmet Store, Vegan Store, and Variety Store as extended classes. Abstraction is more scalable and allows us to add more types of stores later on.



An example of the customed data type for Person is below:

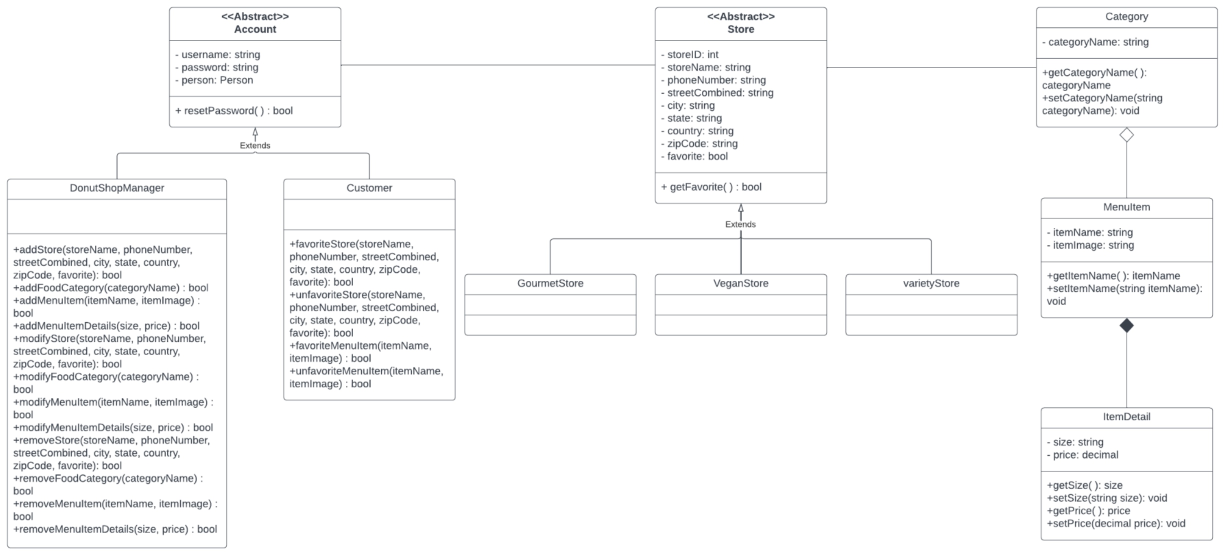


The class Diagram is below:
The relationships between the classes consist of:
Association
- Account as a two way association with Store
- Store has a two way association with Category

Composition
- The Menu Item includes or is composed of item Details

Aggregation
- This is a specific type of association where the Menu Items can exist independently from the Category

Inheritance
- As discussed earlier Donut Shop Manager and Customer inherit Accounts
- Gourmet Store, Vegan Store, and Variety Store inherit the Store class



<<Abstract>>
Account
- username: string
- password: string
- person: Person
+ resetPassword( ) : bool
Extends
DonutShopManager
+addStore(storeName, phoneNumber, streetCombined, city, state, country, zipCode, favorite): bool
+addFoodCategory(categoryName) : bool
+addMenuItem(itemName, itemImage) : bool
+addMenuItemDetails(size, price) : bool
+modifyStore(storeName, phoneNumber, streetCombined, city, state, country, zipCode, favorite): bool
+modifyFoodCategory(categoryName) : bool
+modifyMenuItem(itemName, itemImage) : bool
+modifyMenuItemDetails(size, price) : bool
+removeStore(storeName, phoneNumber, streetCombined, city, state, country, zipCode, favorite): bool
+removeFoodCategory(categoryName) : bool
+removeMenuItem(itemName, itemImage) : bool
+removeMenuItemDetails(size, price) : bool
Customer
+favoriteStore(storeName, phoneNumber, streetCombined, city, state, country, zipCode, favorite): bool
+unfavoriteStore(storeName, phoneNumber, streetCombined, city, state, country, zipCode, favorite): bool
+favoriteMenuItem(itemName, itemImage) : bool
+unfavoriteMenuItem(itemName, itemImage) : bool
<<Abstract>>
Store
- storeID: int
- storeName: string
- phoneNumber: string
- streetCombined: string
- city: string
- state: string
- country: string
- zipCode: string
- favorite: bool
+ getFavorite( ) : bool
Extends
GourmetStore
VeganStore
varietyStore
Category
- categoryName: string
+getCategoryName( ): categoryName
+setCategoryName(string categoryName): void
MenuItem
- itemName: string
- itemImage: string
+getItemName( ): itemName
+setItemName(string itemName): void
ItemDetail
- size: string
- price: decimal
+getSize( ): size
+setSize(string size): void
+getPrice( ): price
+setPrice(decimal price): void

System Sequence Diagrams

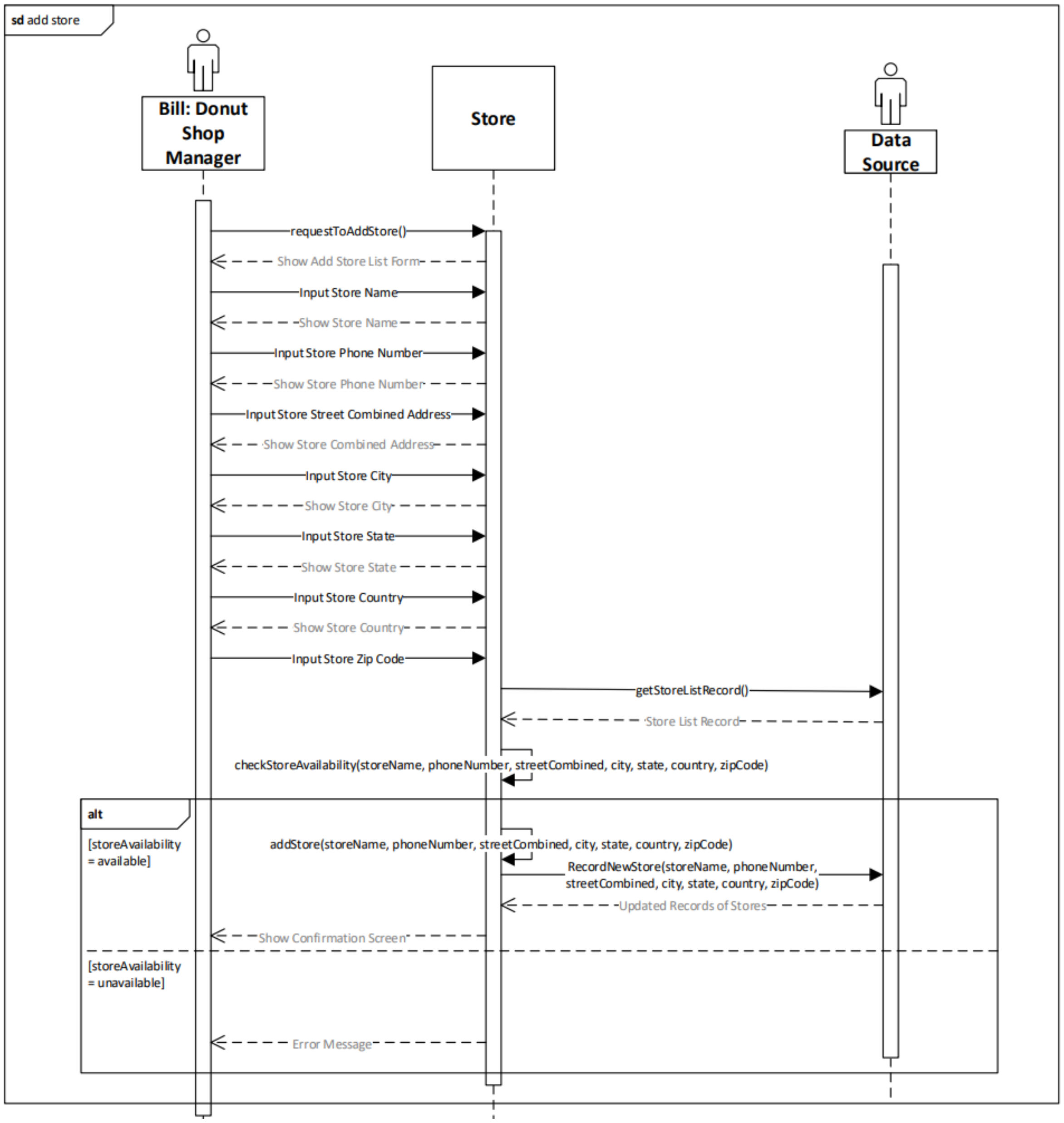
Sequence Diagram for Adding a Store
Actor: Donut Shop Manager (Bill)
Object: Store
Interacts with the Data Source or Database

Steps of adding a store:

1. The Donut Shop Manager requests to add a store
2. The Donut Shop Manager Inputs Store Name, phone number, street combined address, city, state, country, and zip code information needed to make a store
3. The Store gets a list of stores from the datasource or database
4. The Store checks the store availability
5. If the store is available to add: 1. The store is added and recorded in the datasource or database 2. A confirmation screen is shown
6. If the store is not available to add and is unsuccessful: 1. The Donut Shop Manager sees an error message for the unsuccessful adding of the store and that the store is not available to be added.

The sequence diagram is shown below:

Sequence Diagram to Add a Store


sd add store
Bill: Donut Shop Manager
Store
Data Source
requestToAddStore()
Show Add Store List Form
Input Store Name
Show Store Name
Input Store Phone Number
Show Store Phone Number
Input Store Street Combined Address
Show Store Combined Address
Input Store City
Show Store City
Input Store State
Show Store State
Input Store Country
Show Store Country
Input Store Zip Code
getStoreListRecord()
Store List Record
checkStoreAvailability(storeName, phoneNumber, streetCombined, city, state, country, zipCode)
alt
[storeAvailability = available]
addStore(storeName, phoneNumber, streetCombined, city, state, country, zipCode)
RecordNewStore(storeName, phoneNumber, streetCombined, city, state, country, zipCode)
Updated Records of Stores
Show Confirmation Screen
[storeAvailability = unavailable]
Error Message

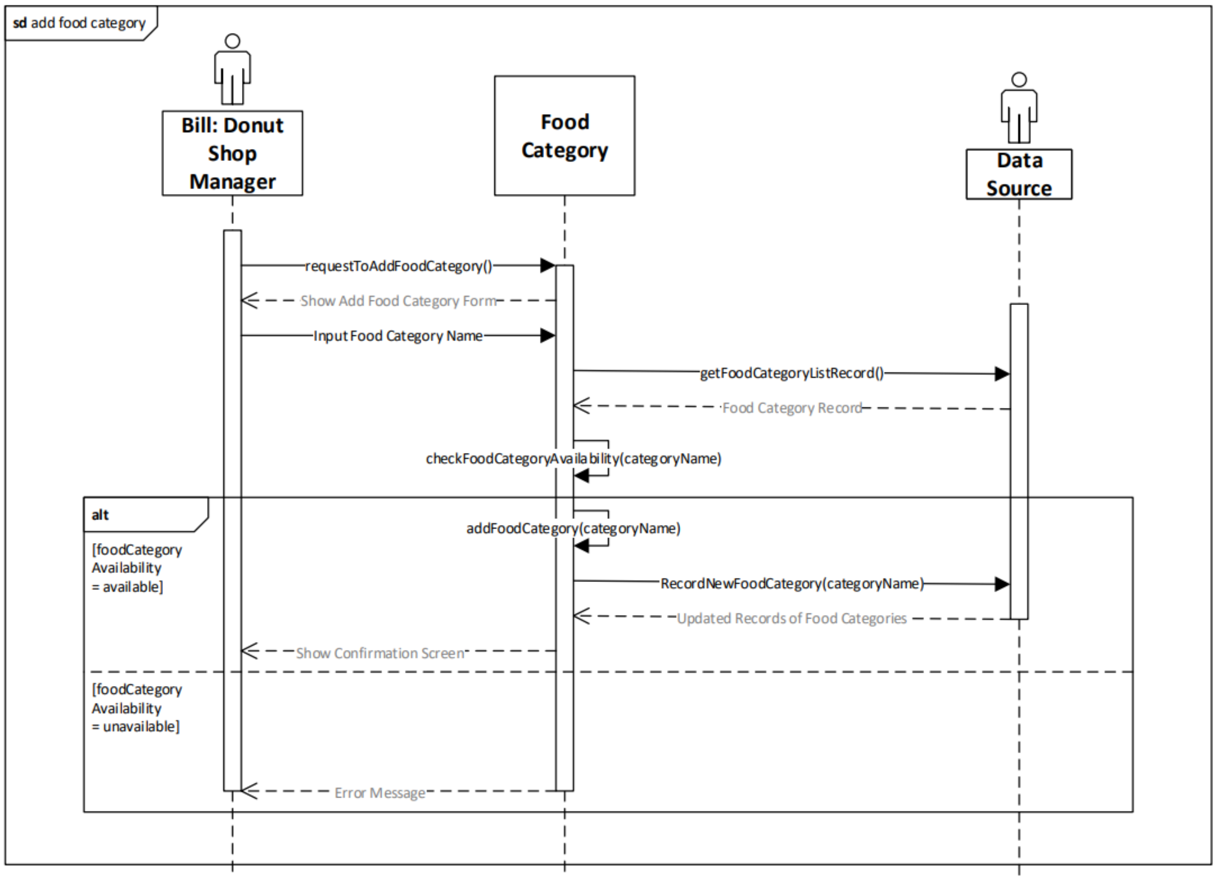
Sequence Diagram for Adding a Food Category
Actor: Donut Shop Manager (Bill)
Object: Food Category
Interacts with the Data Source or Database

Steps of adding a Food Category:
1. The Donut Shop Manager requests to add a food category
2. The Donut Shop Manager Inputs the food category name

3. The Food Category gets a list of records of food categories from the datasource or database
4. The Food Category checks the food category availability
5. If the Food Category is available to add: 1. The food category is added and recorded in the datasource or database 2. A confirmation screen is shown
6. If the Food Category is not available to add and is unsuccessful: 1. The Donut Shop Manager sees an error message for the unsuccessful adding of the food category and that the food category is not available to be added.

The sequence diagram is shown below:

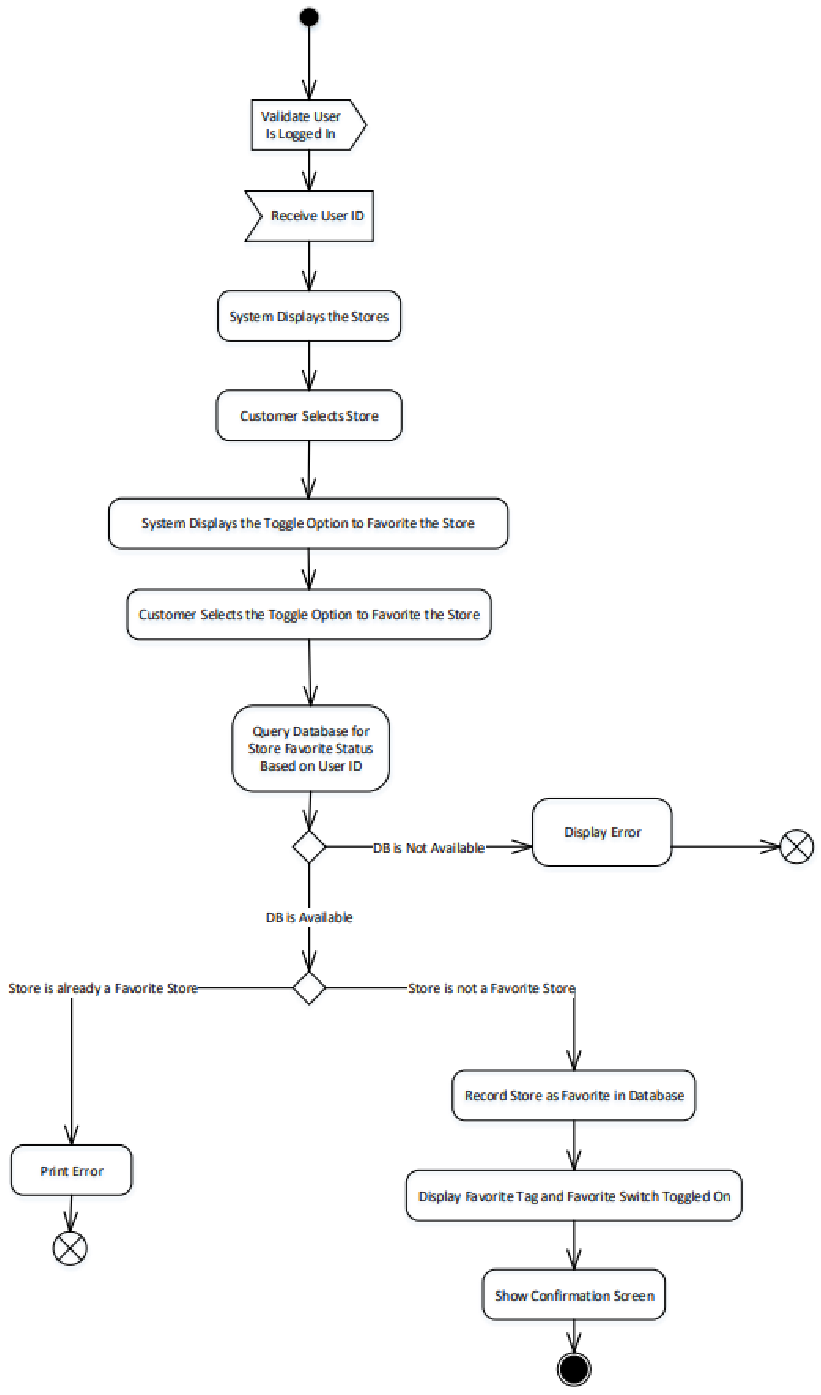
Sequence Diagram to Add a Food Category


sd add food category
Bill: Donut Shop Manager
Food Category
Data Source
requestToAddFoodCategory()
Show Add Food Category Form
Input Food Category Name
getFoodCategoryListRecord()
Food Category Record
checkFoodCategoryAvailability(categoryName)
alt
[foodCategory Availability = available]
addFoodCategory(categoryName)
RecordNewFoodCategory(categoryName)
Updated Records of Food Categories
Show Confirmation Screen
[foodCategory Availability = unavailable]
Error Message

Activity Diagrams

Activity Diagram for Favoriting a Store Use Case
States:
-Initial State: Make sure that the User is Logged In and then Show the Stores
-Final State: 1.The Customer sees a confirmation screen that the Store was favorited successfully and they can see the favorite tag and favorite switch toggled on. 2. The Store is already a favorite so the system prints an error for the customer to see.
Actions:
The Customer is logged in and views the stores that are displayed from the system. The customer selects the store they want to favorite and clicks the toggle option to favorite the store. The database is queried to figure out what stores the customer has favorited. The system then updates the store as a favorite if it is not already a favorite.

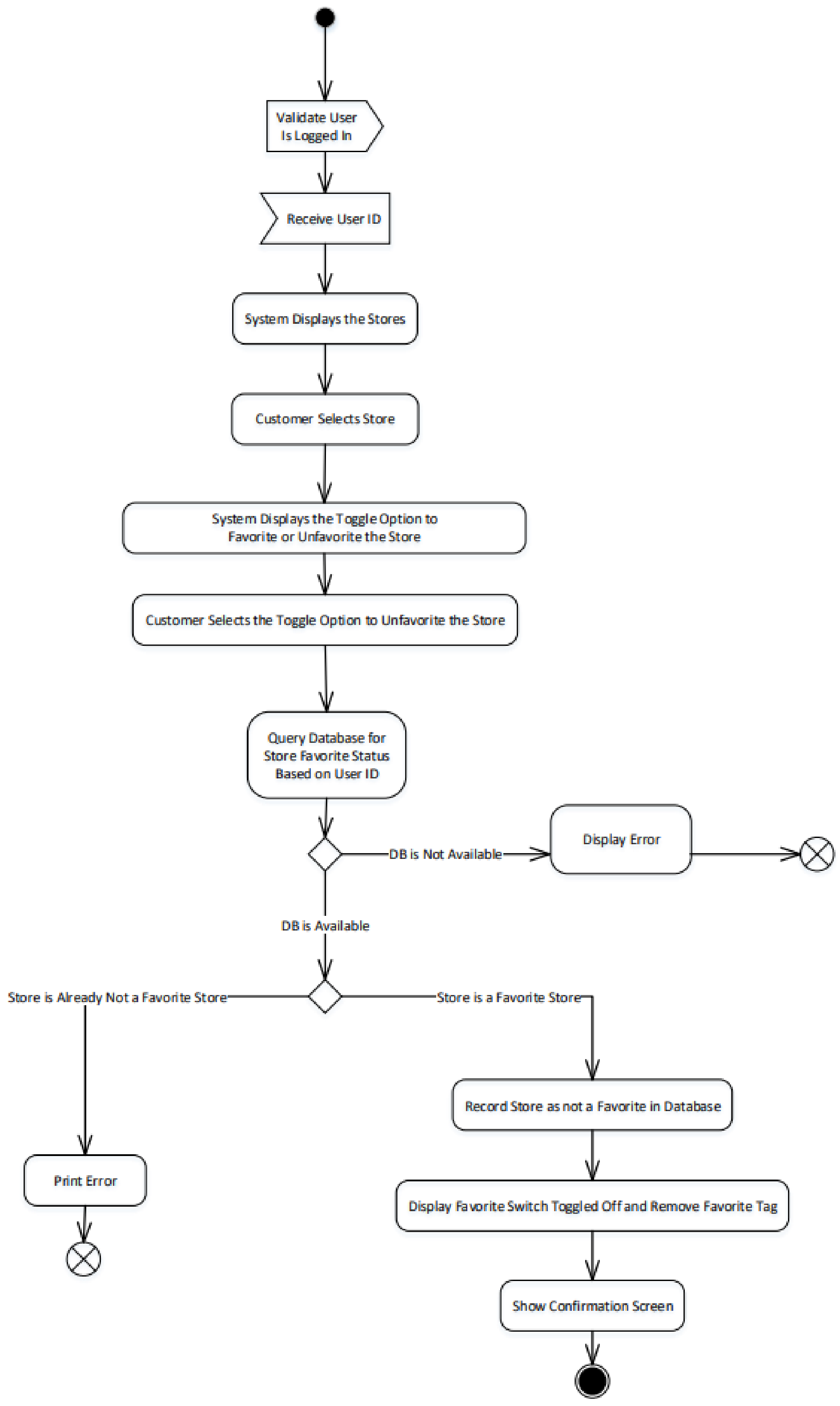
Activity Diagram for Favorite Store Use Case


Validate User Is Logged In
Receive User ID
System Displays the Stores
Customer Selects Store
System Displays the Toggle Option to Favorite the Store
Customer Selects the Toggle Option to Favorite the Store
Query Database for Store Favorite Status Based on User ID
DB is Not Available
Display Error
DB is Available
Store is already a Favorite Store
Store is not a Favorite Store
Record Store as Favorite in Database
Print Error
Display Favorite Tag and Favorite Switch Toggled On
Show Confirmation Screen

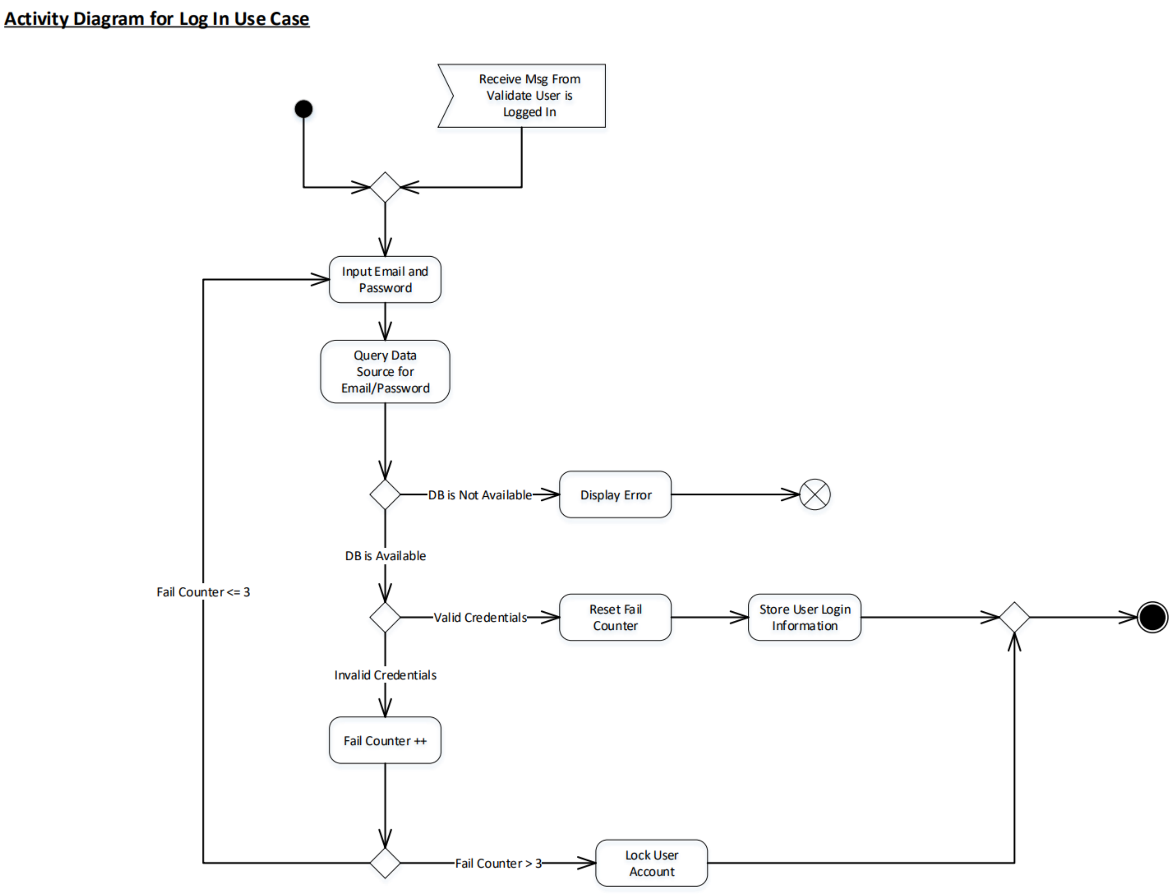
Activity Diagram for Unfavoriting a Store Use Case
States:
-Initial State: Make sure that the User is Logged In and then Show the Stores
-Final State: 1.The Customer sees a confirmation screen that the Store was unfavorited successfully and they can see no favorite tag and the favorite switch toggled off. 2. The Store is already not a favorite so the system prints an error for the customer to see.

Actions:
The Customer is logged in and views the stores that are displayed from the system. The customer selects the store they want to unfavorite and clicks the toggle option to unfavorite the store. The database is queried to figure out what stores the customer has favorited. The system then updates the store as not a favorite if it is a favorite.
The Activity diagram for unfavoriting a store is shown below:

Activity Diagram for Unfavorite Store Use Case

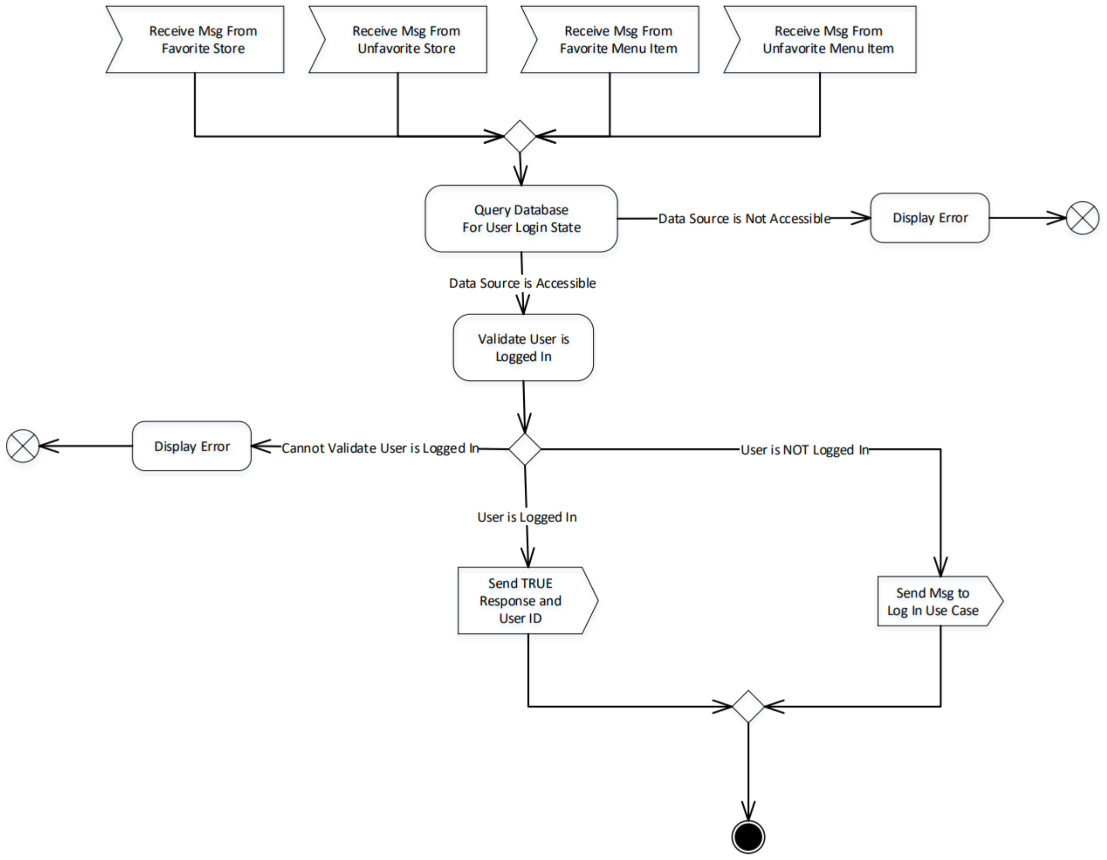

Validate User Is Logged In
Receive User ID
System Displays the Stores
Customer Selects Store
System Displays the Toggle Option to Favorite or Unfavorite the Store
Customer Selects the Toggle Option to Unfavorite the Store
Query Database for Store Favorite Status Based on User ID
DB is Not Available
Display Error
DB is Available
Store is Already Not a Favorite Store
Store is a Favorite Store
Record Store as not a Favorite in Database
Print Error
Display Favorite Switch Toggled Off and Remove Favorite Tag
Show Confirmation Screen

The states and actions for the LogIn Use Case is displayed using symbols within the activity diagram below:



The states and actions for the Validate User is Logged In Use Case is displayed using symbols within the activity diagram below:

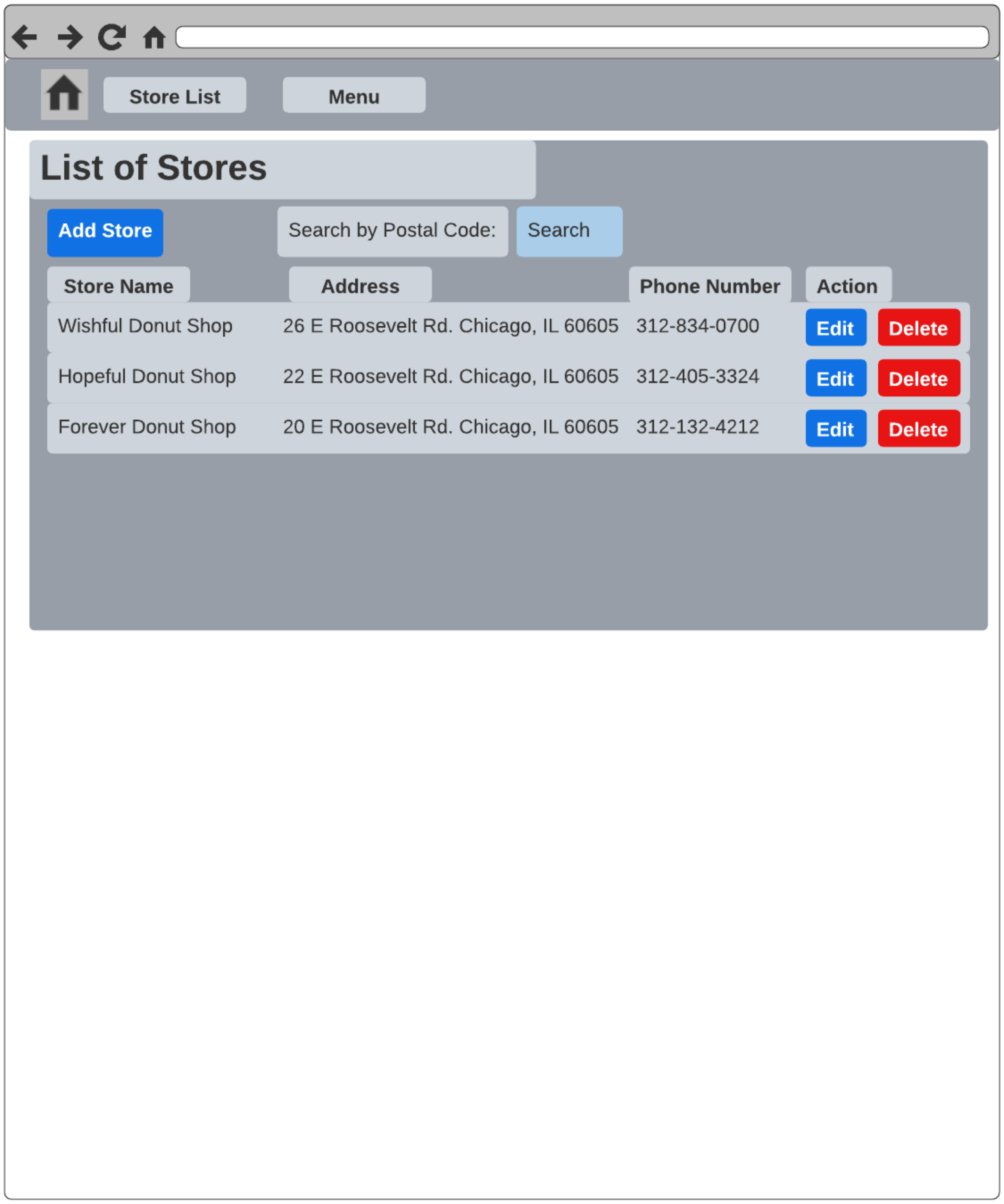
Activity Diagram for Validate User is Logged In Use Case


Receive Msg From Favorite Store
Receive Msg From Unfavorite Store
Receive Msg From Favorite Menu Item
Receive Msg From Unfavorite Menu Item
Query Database For User Login State
Data Source is Not Accessible
Display Error
Data Source is Accessible
Validate User is Logged In
Cannot Validate User is Logged In
Display Error
User is NOT Logged In
User is Logged In
Send TRUE Response and User ID
Send Msg to Log In Use Case

User Interface Specification

Preliminary Design

The User Interface for the Donut Shop Manager to view the list of stores, filter for stores use case, and click buttons to navigate to the add store use case, modify store use case, and delete store use case is shown below. The navigation bar on the top also helps navigate to the menu use cases such as the add food category use case.
List of Stores Page below:

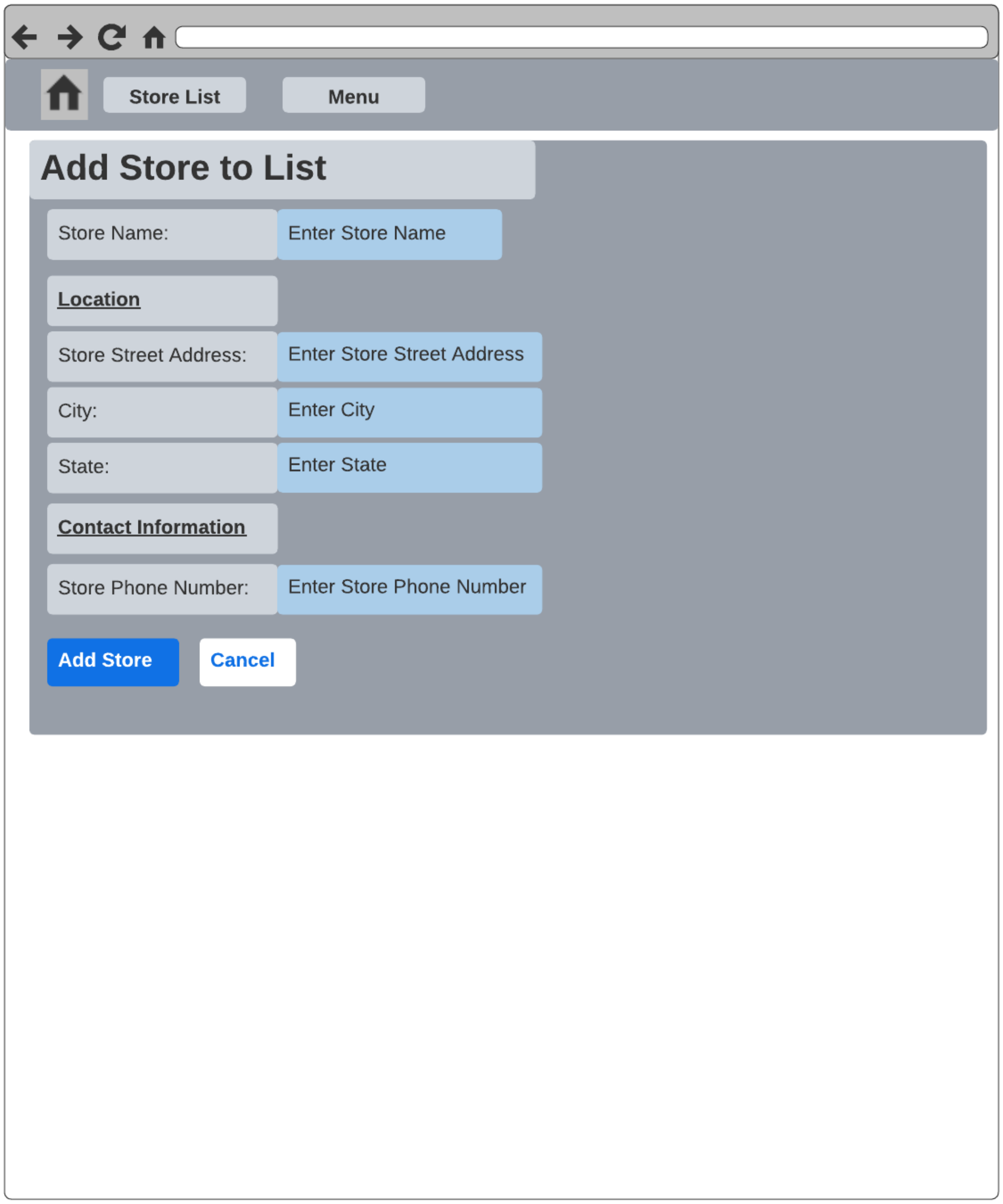


List of Stores

Store Name	Address	Phone Number	Action
Wishful Donut Shop	26 E Roosevelt Rd. Chicago, IL 60605	312-834-0700	Edit Delete
Hopeful Donut Shop	22 E Roosevelt Rd. Chicago, IL 60605	312-405-3324	Edit Delete
Forever Donut Shop	20 E Roosevelt Rd. Chicago, IL 60605	312-132-4212	Edit Delete

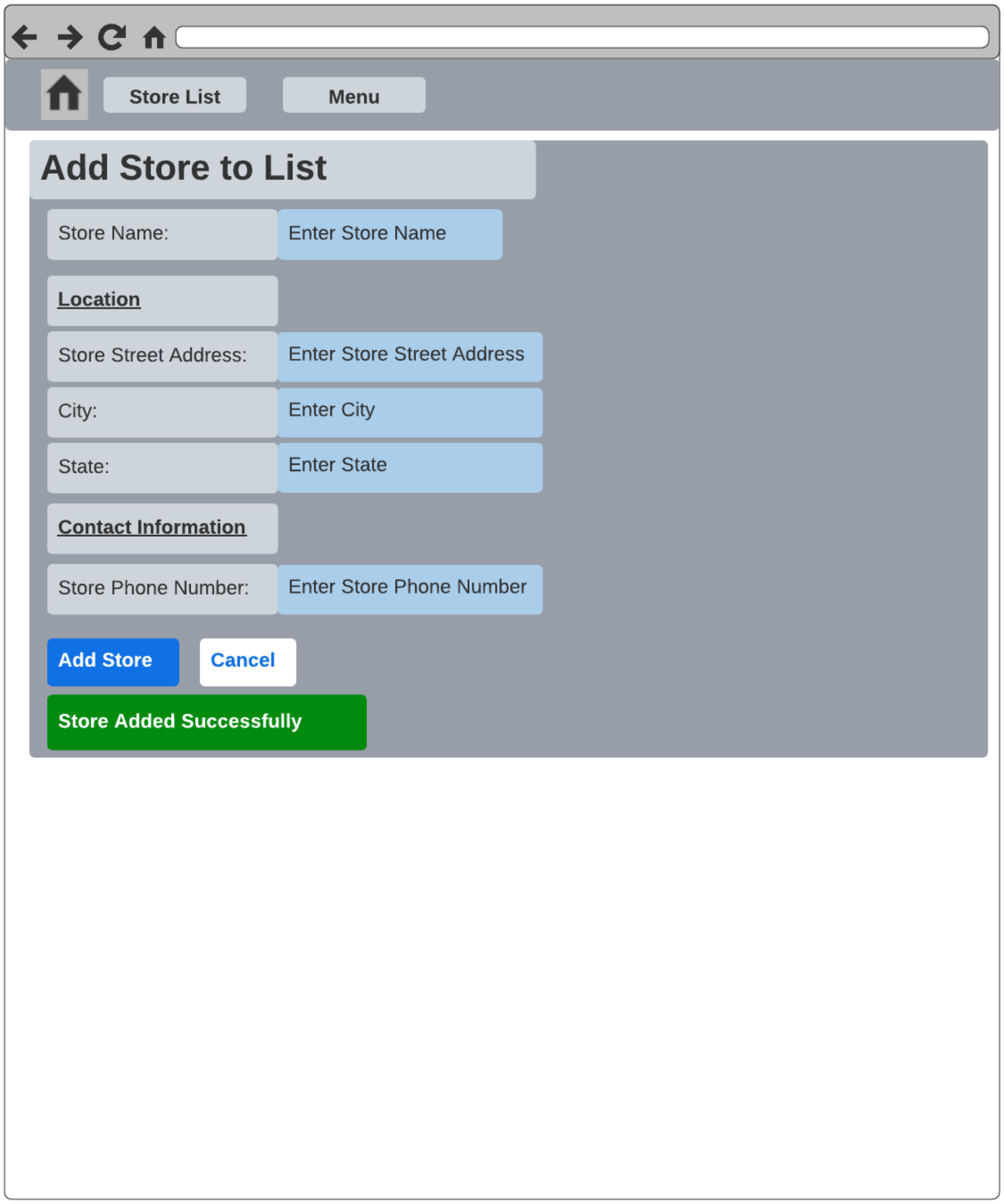
User Interface for Add Store Use Case

How Users Enter Information: The Donut Shop Manager enters information by entering store information into the relevant text fields that are a light blue color in the add store user interface image below. The text fields have initial directions within the text fields to tell the Donut Shop Manager what content to put within the text fields such as “Enter Store Name”. This provides directions so that the user is less likely to get confused. Once the Donut Shop Manager enters all of the store information into the relevant text fields, they can click the blue “Add Store” button to add the store to the list. If the user wants to cancel adding the store, they can click the white “Cancel” button at the bottom of the page. The buttons, text fields, and layout is shown in the mockup image below.



How the Results are Displayed for Users: The Results of adding the store are displayed to the Donut Shop Manager by showing success or error messages depending on if the store was added successfully or not. Image examples of stores successfully added and error messages are shown below. The results of the store being added to the list are also displayed in the Store List page as a new row.

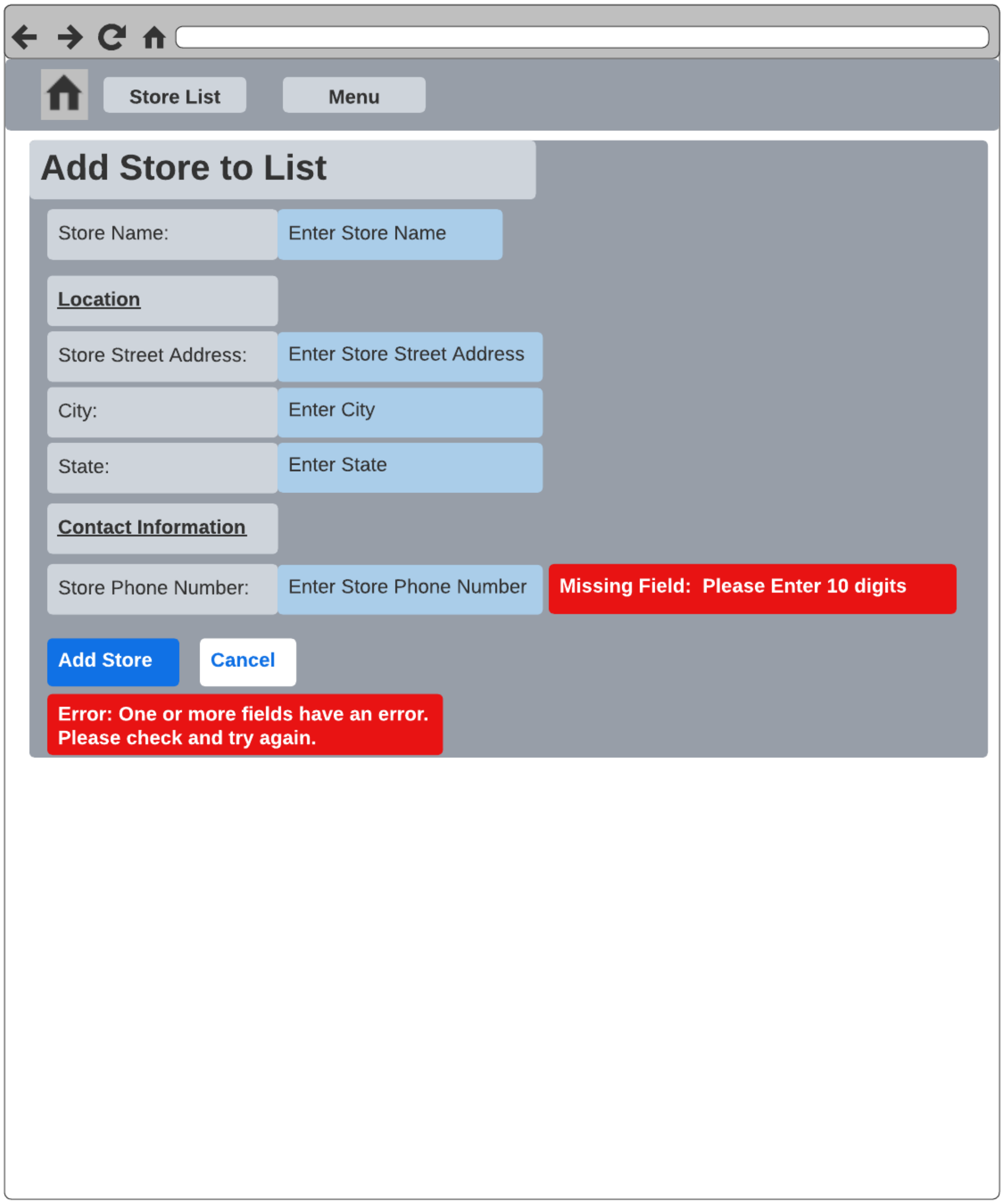
Navigation Paths: The navigation path that the Donut Shop Manager follows to add a store includes that the user needs to log in and enter their username and password into the text fields. Then, the user clicks/navigates to the “Store List” button in the navigation bar at the top of the page. Then, they are taken to the “List of Stores” page where they are shown a list of stores. After that they click the blue “Add Store” button near the top of the screen which navigates them to the “Add Store to list” page. This is the page where they enter the store information and click the blue “Add Store” button to add the store to the list.
This is the navigation path: Login > Home Page > Store List > Add Store



Add Store to List

Store Name:	Enter Store Name
Location	
Store Street Address:	Enter Store Street Address
City:	Enter City
State:	Enter State
Contact Information	
Store Phone Number:	Enter Store Phone Number

Store Added Successfully

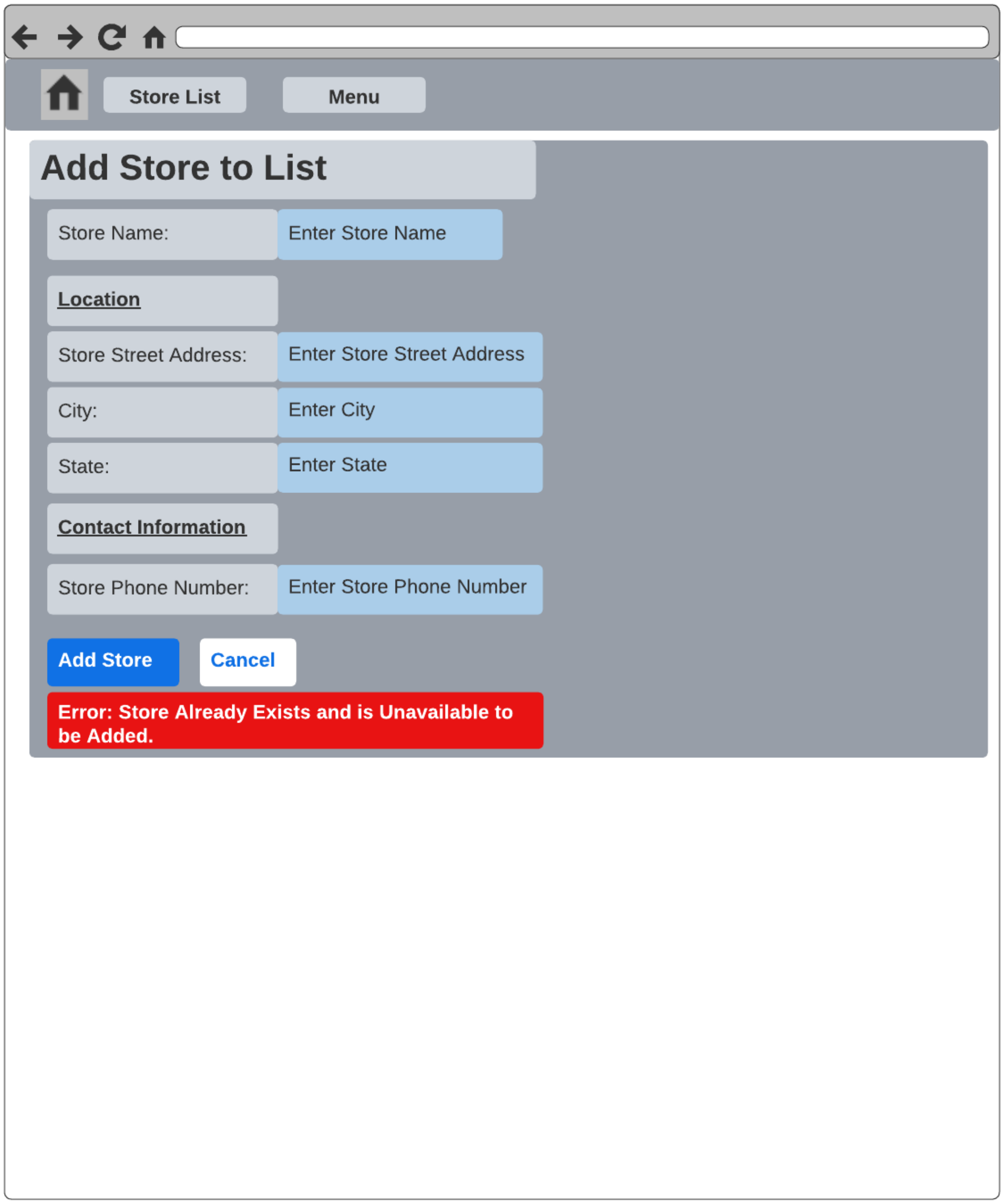


Add Store to List

Store Name:	Enter Store Name
Location	
Store Street Address:	Enter Store Street Address
City:	Enter City
State:	Enter State
Contact Information	
Store Phone Number:	Enter Store Phone Number

Missing Field: Please Enter 10 digits

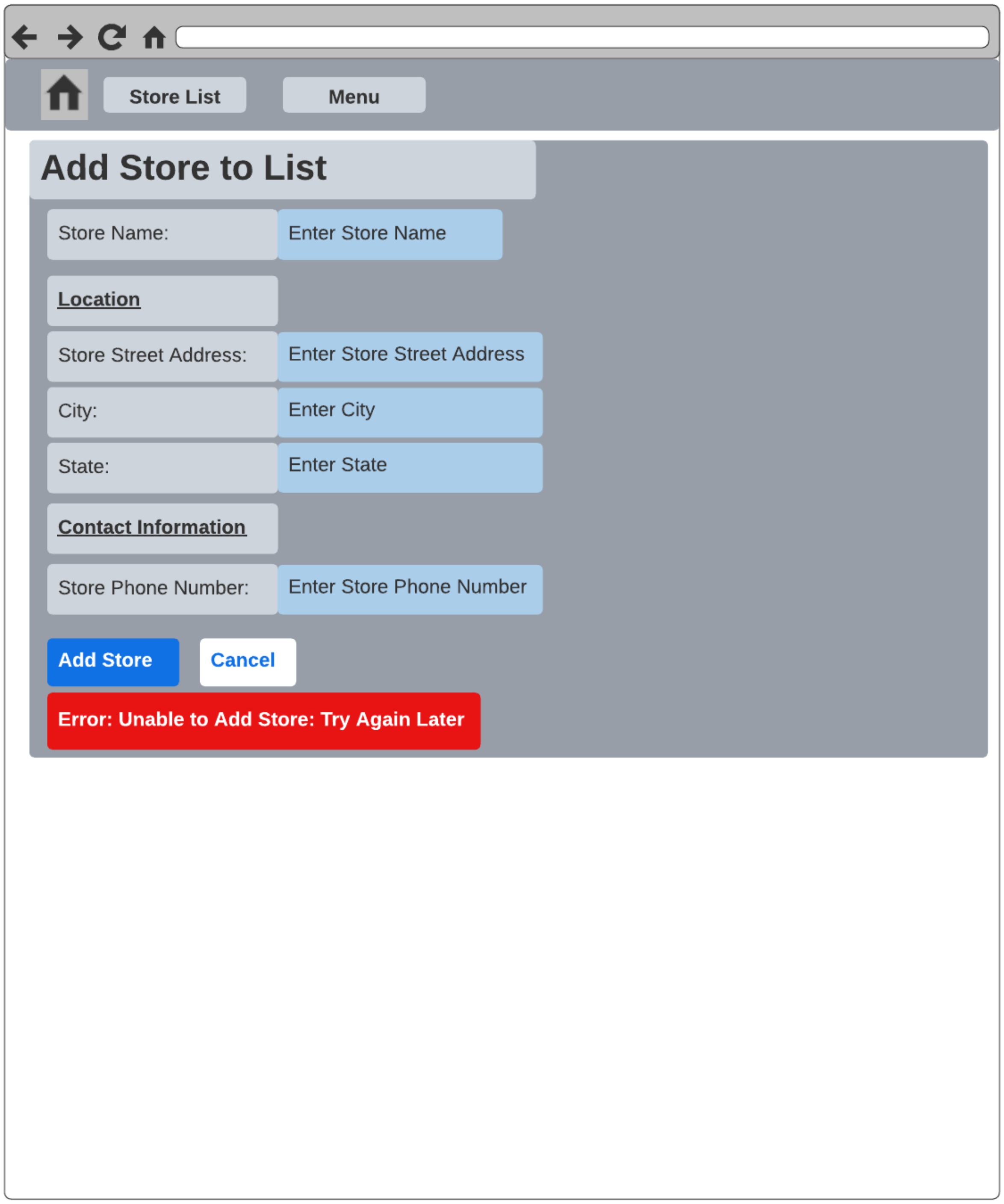
Error: One or more fields have an error. Please check and try again.



Add Store to List

Store Name:	Enter Store Name
Location	
Store Street Address:	Enter Store Street Address
City:	Enter City
State:	Enter State
Contact Information	
Store Phone Number:	Enter Store Phone Number

Error: Store Already Exists and is Unavailable to be Added.



Store List

Menu

Add Store to List

Store Name:	Enter Store Name
Location	
Store Street Address:	Enter Store Street Address
City:	Enter City
State:	Enter State
Contact Information	
Store Phone Number:	Enter Store Phone Number

Add Store

Cancel

Error: Unable to Add Store: Try Again Later

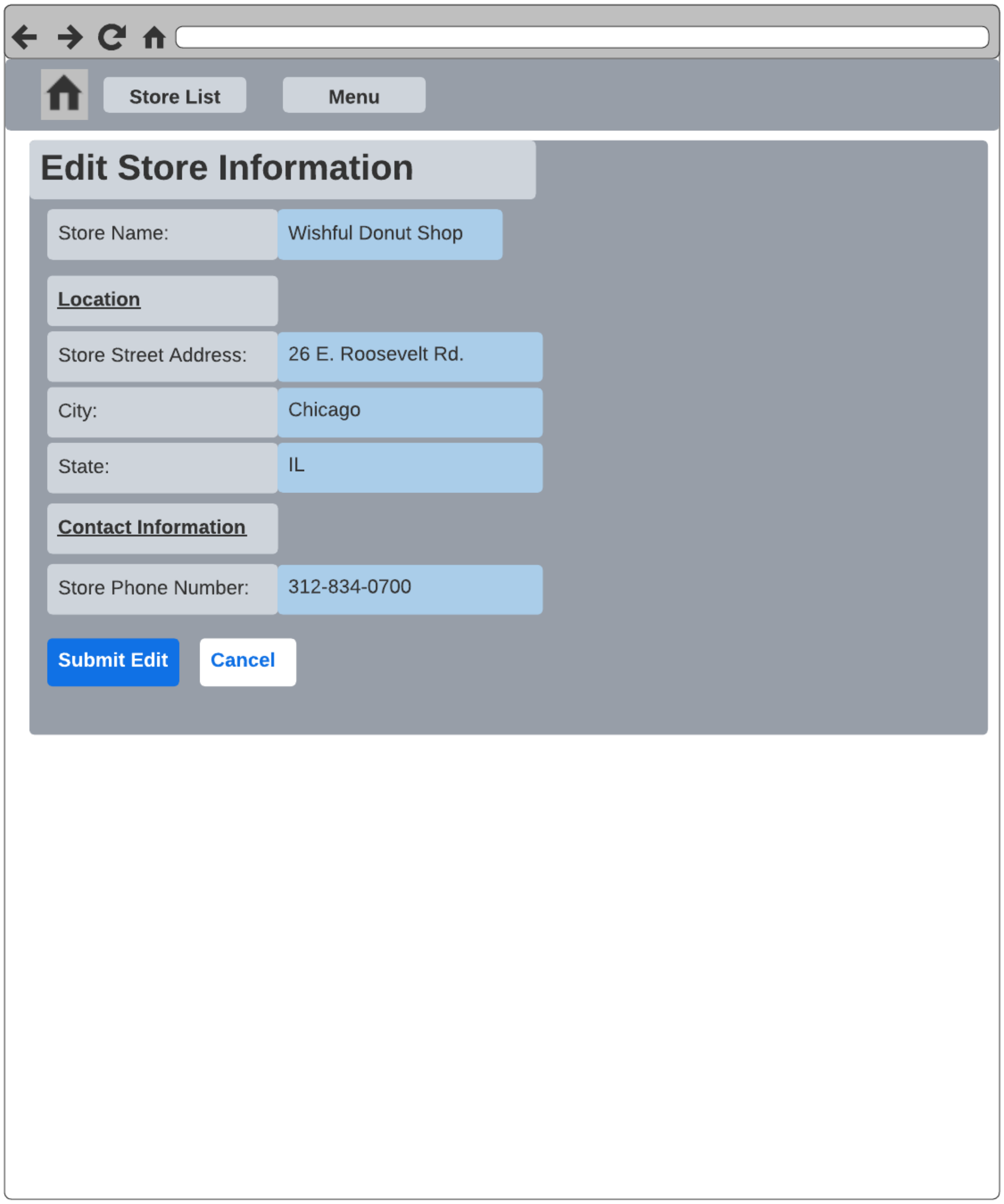
User Interface for Modify/Edit Store Use Case

How Users Enter Information: The Donut Shop Manager enters information by entering store information into the relevant text fields that are a light blue color in the Edit Store Information store user interface image below. The text fields are already filled in with the store information based on which store's edit button they clicked in the "List of Stores" page. The Donut Shop

Manager would then modify the text such as “Wishful Donut Shop” to change to a different store name. Next to the text fields there is text to indicate what store information to edit within the light blue text field. Once the Donut Shop Manager edits all of the store information in the relevant text fields, they can click the blue “Submit Edit” button to Edit the store in the list. If the user wants to cancel Editing the store, they can click the white “Cancel” button at the bottom of the page. The buttons, text fields, and layout is shown in the mockup image below.

How the Results are Displayed for Users: The Results of editing/modifying the store are displayed to the Donut Shop Manager by showing success or error messages depending on if the store was edited successfully or not. These success and error messages would be very similar to the images I showed previously except it would say “store edited successfully. The results of the store being edited/modified in the list are also displayed in the Store List page as an edited row.

Navigation Paths: The navigation path that the Donut Shop Manager follows to modify/edit a store includes that the user needs to log in and enter their username and password into the text fields. Then, the user clicks/navigates to the “Store List” button in the navigation bar at the top of the page. Then, they are taken to the “List of Stores” page where they are shown a list of stores. After that they click the blue “Edit” button in the row that corresponds to the store information that they want to edit which navigates them to the “Edit Store Information” page. This is the page where they edit the store information and click the blue “Submit Edit” button to edit the store in the list.
This is the navigation path: Login > Home Page > Store List > Edit



Edit Store Information

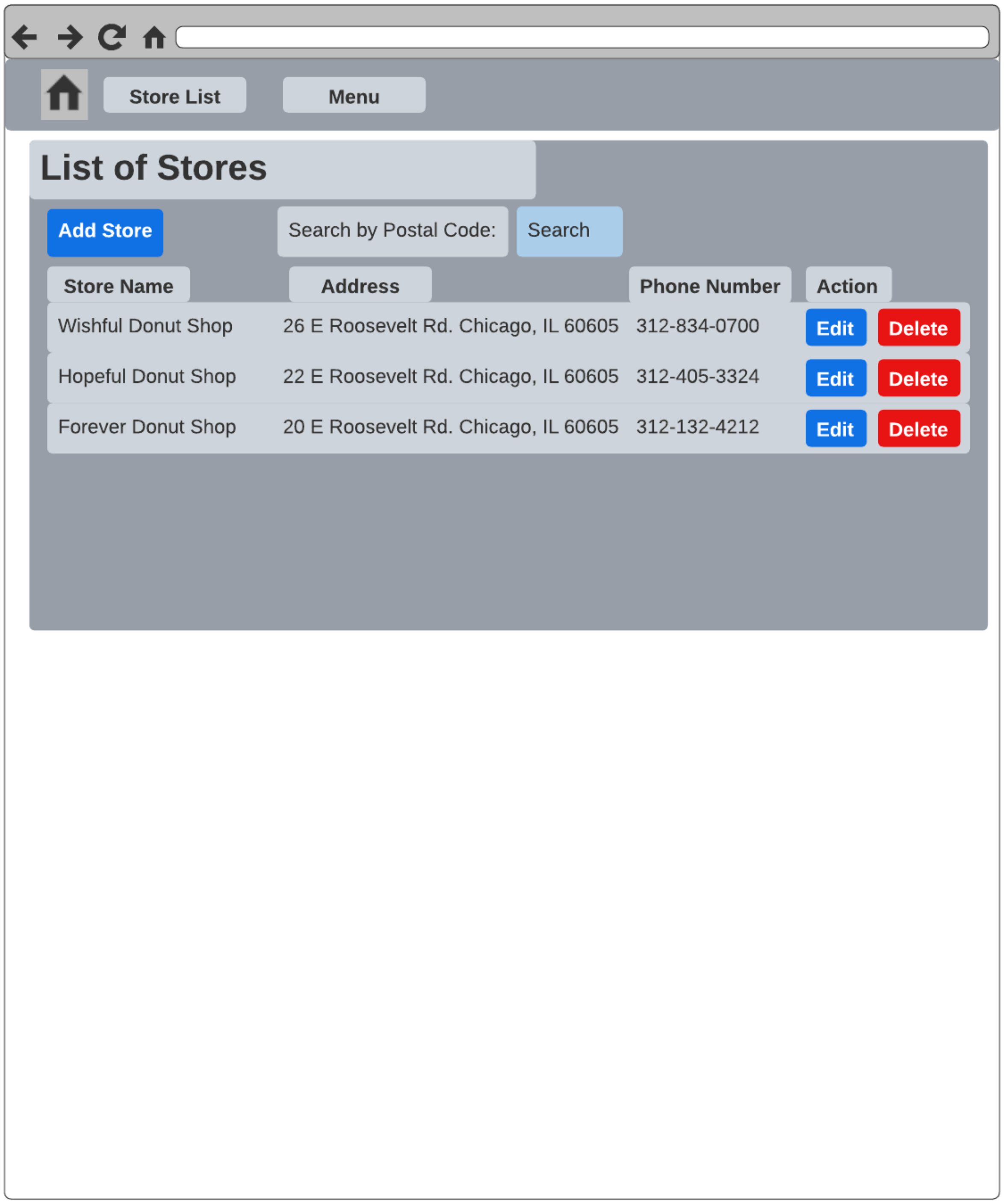
Store Name:	Wishful Donut Shop
Location	
Store Street Address:	26 E. Roosevelt Rd.
City:	Chicago
State:	IL
Contact Information	
Store Phone Number:	312-834-0700

User Interface for Remove/Delete Store Use Case

How Users Enter Information: The Donut Shop Manager does not really need to enter information to delete stores. However, they do need to click on the red “Delete” button that corresponds and is in the same row as the store information that they want to delete in the “List of Stores” page. Then, the user is shown a confirmation screen to make sure that the user wants to delete the store. This page is called “Delete Store”. There is a red “Delete Store” button that the user needs to click to confirm the delete. If the user wants to cancel deleting the store, they can click the white “Cancel” button at the bottom of the page. The buttons, text fields, and layout is shown in the mockup image below.

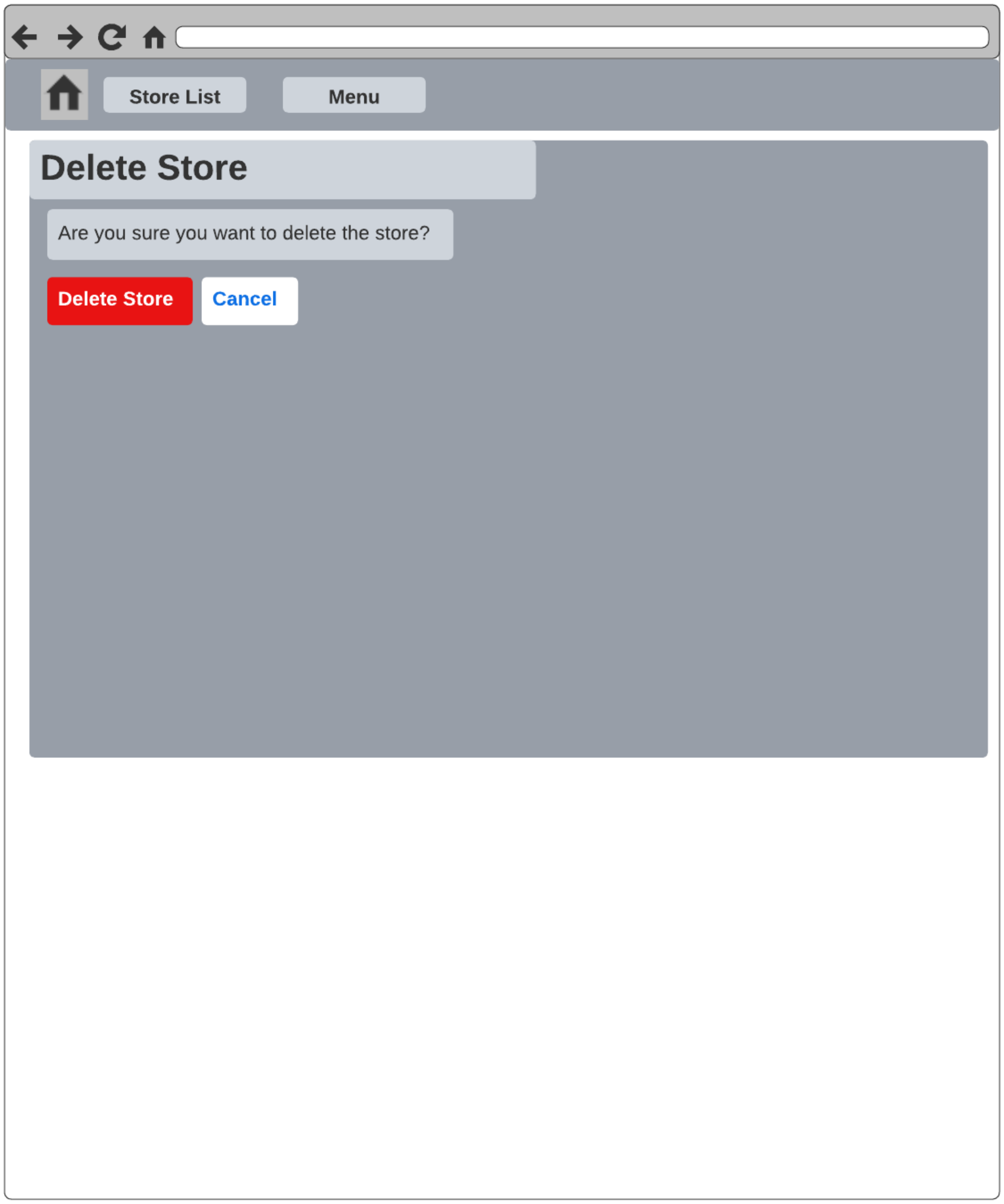
How the Results are Displayed for Users: The Results of deleting the store are displayed to the Donut Shop Manager by showing success or error messages depending on if the store was deleted successfully or not. These success and error messages would be very similar to the images I showed previously except it would say “store deleted successfully”. The results of the store being deleted in the list are also displayed in the Store List page as a row that is gone/deleted.

Navigation Paths: The navigation path that the Donut Shop Manager follows to delete a store includes that the user needs to log in and enter their username and password into the text fields. Then, the user clicks/navigates to the “Store List” button in the navigation bar at the top of the page. Then, they are taken to the “List of Stores” page where they are shown a list of stores. After that they click the red “Delete” button in the row that corresponds to the store information that they want to delete which navigates them to the “Delete Store” page. This is the page where they delete the store information and click the red “Delete Store” button to delete the store in the list.
This is the navigation path: Login > Home Page > Store List > Delete



List of Stores

Store Name	Address	Phone Number	Action
Wishful Donut Shop	26 E Roosevelt Rd. Chicago, IL 60605	312-834-0700	Edit Delete
Hopeful Donut Shop	22 E Roosevelt Rd. Chicago, IL 60605	312-405-3324	Edit Delete
Forever Donut Shop	20 E Roosevelt Rd. Chicago, IL 60605	312-132-4212	Edit Delete

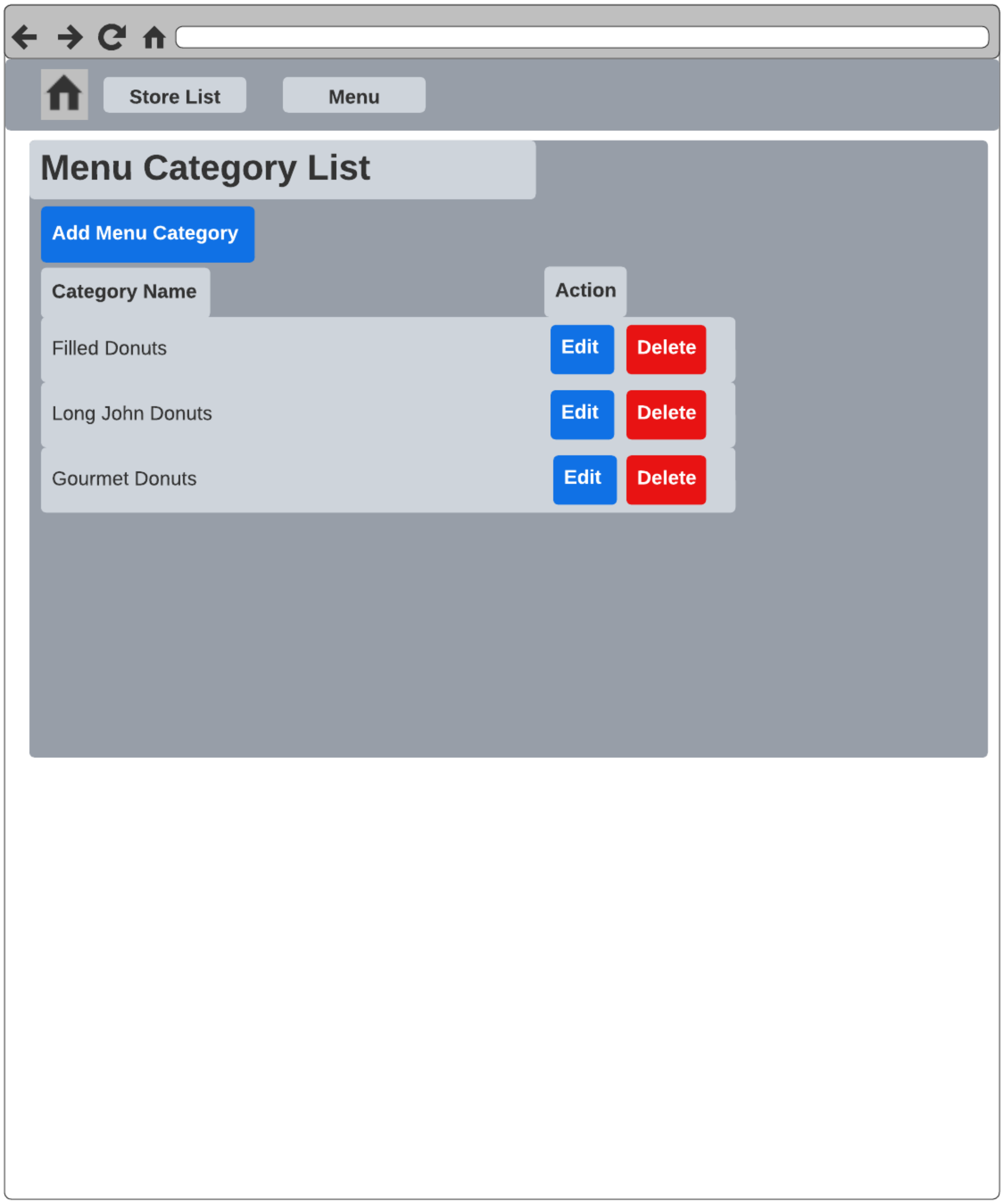


User Interface for Remove/Delete Food Category Use Case

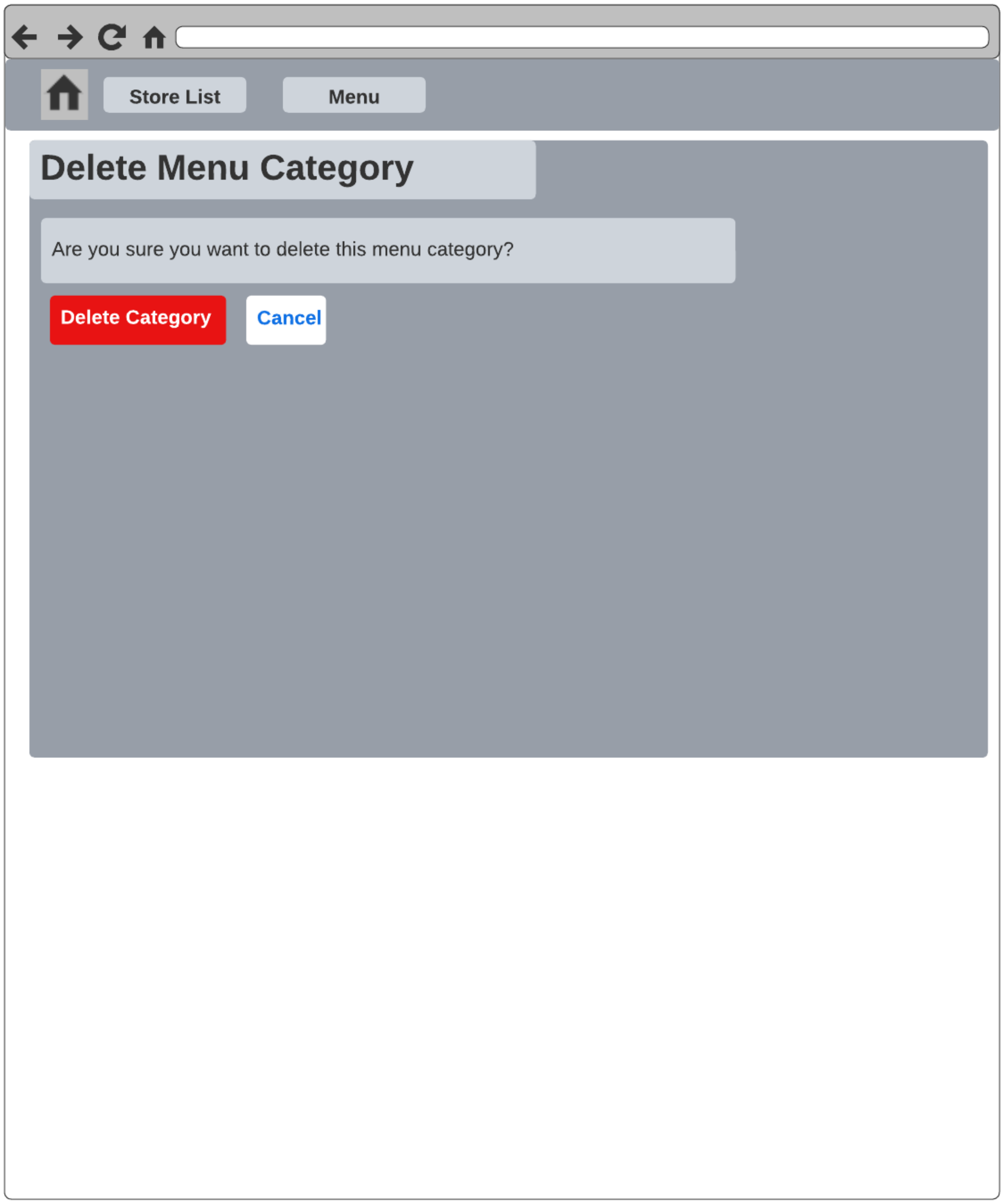
How Users Enter Information: The Donut Shop Manager does not really need to enter information to delete stores. However, they do need to click on the red “Delete” button that corresponds and is in the same row as the menu category information that they want to delete in the “Menu Category List” page. Then, the user is shown a confirmation screen to make sure that the user wants to delete the menu category. This page is called “Delete Menu Category”. There is a red “Delete Category” button that the user needs to click to confirm the delete. If the user wants to cancel deleting the menu category, they can click the white “Cancel” button at the bottom of the page. The buttons, text fields, and layout is shown in the mockup image below.

How the Results are Displayed for Users: The Results of deleting the menu category are displayed to the Donut Shop Manager by showing success or error messages depending on if the store was deleted successfully or not. These success and error messages would be very similar to the images I showed previously except it would say “menu category deleted successfully”. The results of the menu category being deleted in the list are also displayed in the Menu Category List page as a row that is gone/deleted.

Navigation Paths: The navigation path that the Donut Shop Manager follows to delete a menu category includes that the user needs to log in and enter their username and password into the text fields. Then, the user clicks/navigates to the “Menu” button in the navigation bar at the top of the page. Then, they are taken to the “Menu Category List” page where they are shown a list of menu categories. After that they click the red “Delete” button in the row that corresponds to the menu category information that they want to delete which navigates them to the “Delete menu category” page. This is the page where they delete the menu category information and click the red “Delete menu category” button to delete the menu category in the list.
This is the navigation path: Login > Home Page > Menu > Delete



Category Name	Action
Filled Donuts	Edit Delete
Long John Donuts	Edit Delete
Gourmet Donuts	Edit Delete



Delete Menu Category

Are you sure you want to delete this menu category?

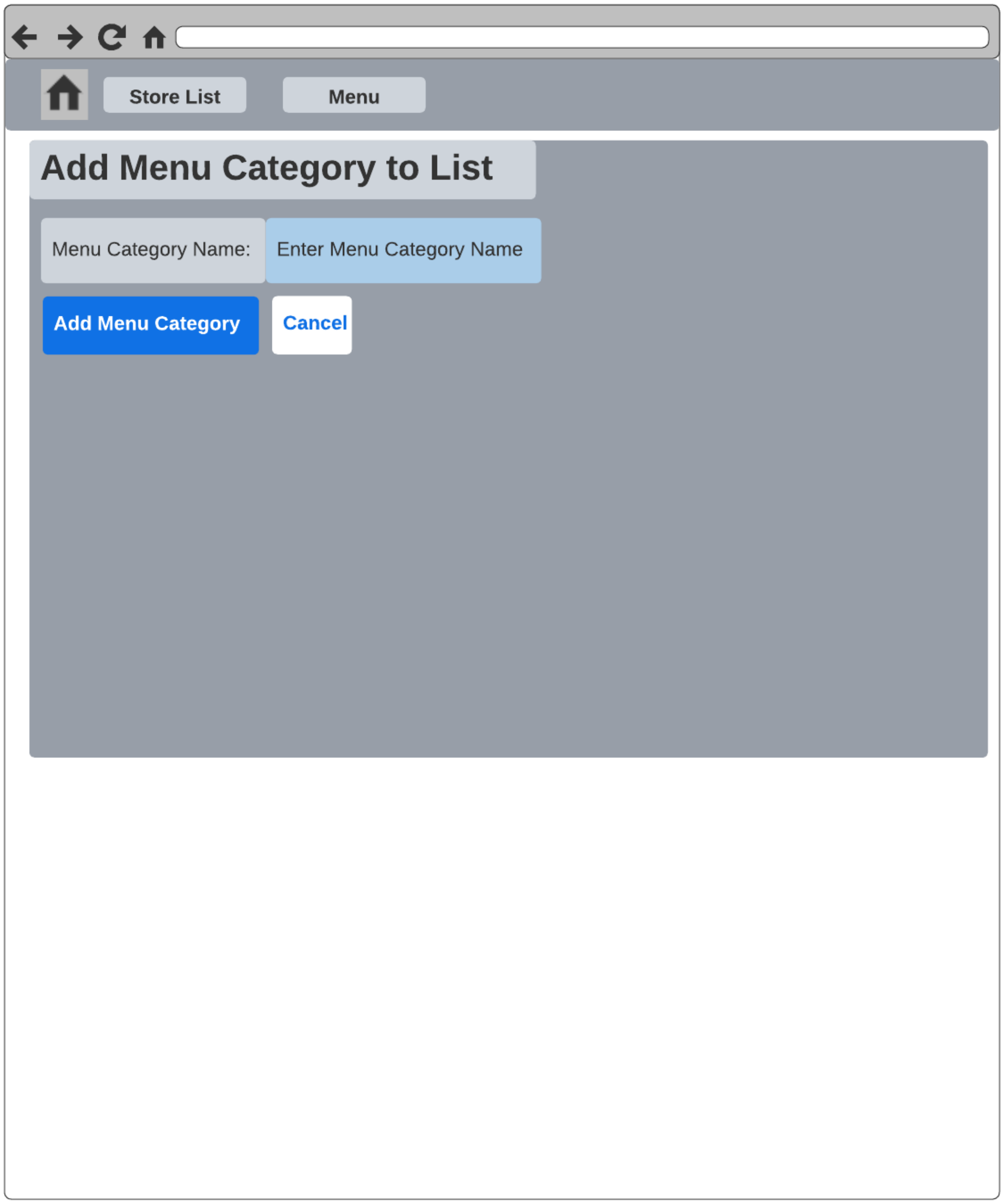
User Interface for Add Menu Category Use Case

How Users Enter Information: The Donut Shop Manager enters information by entering menu category information (category name) into the relevant text field that is a light blue color in the add menu category user interface image below. The text field has initial directions within the text field to tell the Donut Shop Manager what content to put within the text fields such as “Enter

Menu Category Name". This provides directions so that the user is less likely to get confused. Once the Donut Shop Manager enters all of the menu category information into the relevant text field, they can click the blue "Add Menu Category" button to add the menu category to the list. If the user wants to cancel adding the menu category, they can click the white "Cancel" button at the bottom of the page. The buttons, text fields, and layout is shown in the mockup image below.

How the Results are Displayed for Users: The Results of adding the menu category are displayed to the Donut Shop Manager by showing success or error messages depending on if the menu category was added successfully or not. These success and error messages would be very similar to the images I showed previously except it would say "menu category added successfully". The results of the menu category being added to the list are also displayed in the Menu Category List page as a new row.

Navigation Paths: The navigation path that the Donut Shop Manager follows to add a menu category includes that the user needs to log in and enter their username and password into the text fields. Then, the user clicks/navigates to the "Menu" button in the navigation bar at the top of the page. Then, they are taken to the "Menu Category List" page where they are shown a list of menu categories. After that they click the blue "Add Menu Category" button near the top of the screen which navigates them to the "Add Menu Category" page. This is the page where they enter the menu category information (category name) and click the blue "Add Menu Category" button to add the Menu Category to the list.
This is the navigation path: Login > Home Page > Menu > Add Menu Category



User Effort Estimation

Usage Scenario of Adding a new Store (called Wishful Donut Shop with an address of 26 E Roosevelt Rd. Chicago, IL 60605 and a phone number of 312-834-0700):

Sequence of Events:
The Donut Shop Manager will need to click to log in to their account. They will then need to enter their username and password into the text fields. Then, the user clicks the “Store List” button in the navigation bar at the top of the page. Then, they are taken to the “List of Stores” page where they are shown a list of stores. After that they click the blue “Add Store” button near the top of the screen which navigates them to the “Add Store to list” page. This is the page where they enter the store information. They will need to click each of the five text fields corresponding to each store information section and then input the store information by typing on the keyboard. In this scenario, they click on the “enter store name” text field and enter “Wishful Donut Shop”. After that they click on the “enter store address” text field and enter “26 E Roosevelt Rd.” After that they click on the “enter city” text field and enter “Chicago”. After that they click on the “enter state” text field and enter “IL”. After that they click on the “enter store phone number” text field and enter “312-834-0700” Then they click the blue “Add Store” button to add the store to the list.

This is the navigation path: Login > Home Page > Store List > Add Store

Number of Clicks and Keystrokes:
Click “LogIn” + click “enter Username” text field + 2 keystrokes for username of “hi” + click “enter Password” text field + 2 keystrokes for password of “hi” + click “LogIn Button” + Click “Store List” button in Navigation bar at the top of the screen + click “Add Store” + 5 clicks (one for clicking each store information text fields) + 60 keystrokes to type all of the store information into each text field (Wishful Donut Shop 26 E Roosevelt Rd. Chicago, IL 60605 312-834-0700 + click “Add Store”
1+ 1 + 2 + 1+ 2+ 1+ 1 + 1 + 5 + 60 + 1 = 76 clicks and keystrokes total

Number of Clicks:
Click “LogIn” + click “enter Username” text field + click “enter Password” text field + click “LogIn Button” + Click “Store List” button in Navigation bar at the top of the screen + click “Add Store” + 5 clicks (one for clicking each store information text fields) + click “Add Store”
1+ 1 + 1+ 1+ 1 + 1 + 5 + 1 = 12 clicks total

Number of Clicks if Already Logged In:
Click “Store List” button in Navigation bar at the top of the screen + click “Add Store” + 5 clicks (one for clicking each store information text fields) + click “Add Store”
1 + 1 + 5 + 1 = 8 clicks total

Number of Clicks if Already Logged In and just want to get to the Add Store Page:
Click “Store List” button in Navigation bar at the top of the screen + click “Add Store”
1 + 1 = 2 clicks total

Usage Scenario of Removing/Deleting a Store (called Wishful Donut Shop with an address of 26 E Roosevelt Rd. Chicago, IL 60605 and a phone number of 312-834-0700):

The Donut Shop Manager will need to click to log in to their account. They will then need to enter their username and password into the text fields. Then, the user clicks the “Store List” button in the navigation bar at the top of the page. Then, they are taken to the “List of Stores” page where they are shown a list of stores. After that they click the red “Delete” button in the row that corresponds to the store information (Wishful Donut Shop) that they want to delete which navigates them to the “Delete Store” page. This is the page where they delete the store information and click the red “Delete Store” button to delete the store in the list.
This is the navigation path: Login > Home Page > Store List > Delete

Number of Clicks:
Click “LogIn” + click “enter Username” text field + click “enter Password” text field + click “LogIn Button” + Click “Store List” button in Navigation bar at the top of the screen + click “Delete” button in the row with the store they want to delete + click “Delete Store” button
1+ 1 + 1+ 1+ 1 + 1 + 1 = 7 clicks total

Number of Clicks if Already Logged In:
Click “Store List” button in Navigation bar at the top of the screen + click “Delete” button in the row with the store they want to delete + click “Delete Store” button
1 + 1+ 1 = 3 clicks total

Number of Clicks if Already Logged In and just want to get to the Delete Store Page:
Click “Store List” button in Navigation bar at the top of the screen + click “Delete” button in the row with the store they want to delete
1 + 1 = 2 clicks total

<u>Usage Scenario of Removing/Deleting a Menu/Food Category </u>(called Filled Donuts):

The Donut Shop Manager will need to click to log in to their account. They will then need to enter their username and password into the text fields. Then, the user clicks the “Menu” button in the navigation bar at the top of the page. Then, they are taken to the “Menu Category List” page where they are shown a list of menu categories. After that they click the red “Delete” button in the row that corresponds to the menu category information (Filled Donuts) that they want to delete which navigates them to the “Delete Menu Category” page. This is the page where they delete the menu category information and click the red “Delete Category” button to delete the menu category in the list.
This is the navigation path: Login > Home Page > Menu > Delete

Number of Clicks:
Click “LogIn” + click “enter Username” text field + click “enter Password” text field + click “LogIn Button” + Click “Menu” button in Navigation bar at the top of the screen + click “Delete” button in the row with the menu category they want to delete + click “Delete Category” button
1+ 1 + 1+ 1+ 1 + 1 + 1 = 7 clicks total

Number of Clicks if Already Logged In:

Click “Menu” button in Navigation bar at the top of the screen + click “Delete” button in the row with the menu category they want to delete + click “Delete Category” button
1 + 1+ 1 = 3 clicks total

Number of Clicks if Already Logged In and just want to get to the Delete Store Page:
Click “Menu” button in Navigation bar at the top of the screen + click “Delete” button in the row with the menu category they want to delete
1 + 1 = 2 clicks total

Traceability Matrix

System Requirements

No.	Priority Weight (1-5: 1: lowest, 5: highest)	Description
REQ1	5	Donut Shop Manager or Donut Shop Owners can log in and log out from the system
REQ2	5	Only Donut Shop Owners or Donut Shop Managers should be allowed to add donut shops in the system
REQ3	2	The system should filter for shops based on the city or postal code in less than 4 seconds
REQ4	5	Only Donut Shop Owners or Donut Shop Managers should be allowed to edit donut shops and their information in the system
REQ5	5	Donut Shop Owners and Donut Shop Managers should be able to view donut shop information, food categories, and menu items
REQ6	5	Only Donut Shop Owners or Donut Shop Managers should be allowed to delete donut shops in the system
REQ7	4	Logged in Donut Shop Owners or Donut Shop Managers should be able to add newly offered food categories for the donut shops
REQ8	4	Logged in Donut Shop Owners or Donut Shop Managers should be able to add newly offered menu items for the donut shops
REQ9	3	Only Donut Shop Owners or

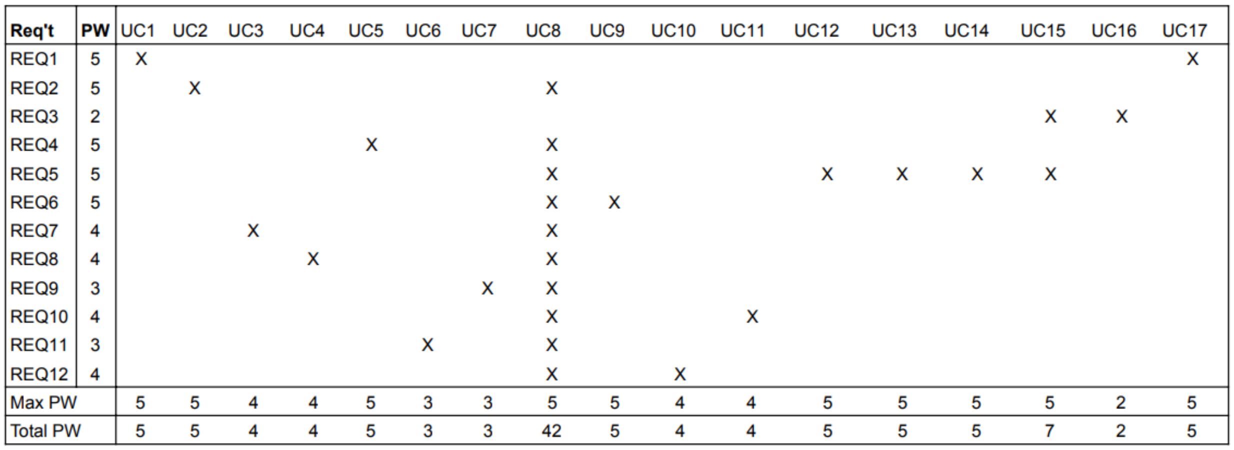
		Donut Shop Managers should be able to edit menu items for the donut shops
REQ10	4	Only Donut Shop Owners or Donut Shop Managers should be able to remove menu items that are no longer offered by the donut shops
REQ11	3	Only Donut Shop Owners or Donut Shop Managers should be able to edit food categories for the donut shops
REQ12	4	Only Donut Shop Owners or Donut Shop Managers should be able to remove food categories that are no longer offered by the donut shops

Use Cases

No.	Description
UC1	Login: to login to a donut shop manager account
UC2	Add store: to add a new store to the database and the list on the webpage
UC3	Add food category: to add a new food category to the database and the list on the webpage
UC4	Add menu item: to add a new menu item to the database and the list on the webpage
UC5	Modify store: to modify information about a store in the database and on the webpage
UC6	Modify food category: to modify a food category in the database and on the webpage
UC7	Modify menu item: to modify a menu item in the database and on the webpage
UC8	Validate user is logged in: to make sure the user is logged in to their account
UC9	Remove store: to remove a store from the database and from the list on the webpage
UC10	Remove food category: to remove a food category from the database and from the list on the webpage
UC11	Remove menu item: to remove a menu item from the database and from the list on the webpage
UC12	View stores: to view store information on the webpage

UC13	View categories: to view menu categories on the webpage
UC14	View menu items: to view menu items on the webpage
UC15	View filter for stores: to view the filter for stores based on zip code on the webpage
UC16	Filter stores: to get the stores filtered based on the zip code the user plugged in
UC17	Logout: to logout from a donut shop manager account

Traceability Matrix



Req't	**PW**	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13	UC14	UC15	UC16	UC17
REQ1	5	X																X
REQ2	5		X						X									
REQ3	2															X	X	
REQ4	5					X			X									
REQ5	5								X				X	X	X	X		
REQ6	5								X	X								
REQ7	4			X					X									
REQ8	4				X				X									
REQ9	3							X	X									
REQ10	4								X			X						
REQ11	3						X		X									
REQ12	4								X		X							
Max PW		5	5	4	4	5	3	3	5	5	4	4	5	5	5	5	2	5
Total PW		5	5	4	4	5	3	3	42	5	4	4	5	5	5	7	2	5

System Architecture and System Design

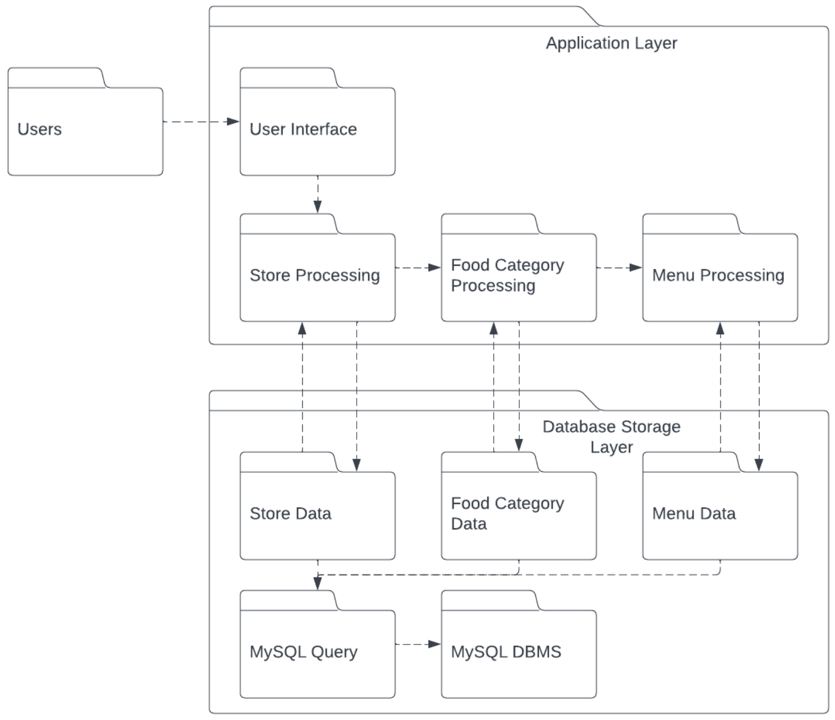
Architectural Styles

The Donut Shop system is a client-server architectural style. The client is one computer that communicates with the server by sending requests and waiting for the server to send a response. The client does not need to know how the server works to get the request or response. Instead, the client can request for a webpage and the web server is a server that serves the web pages. The server is a computer that provides services to the client. I am using XAMPP (Cross-platform, Apache, MySQL, PHP, and Perl) to make a Donut Shop website information system on a local webserver on my computer. The Apache is the server to show the web pages in the localhost and it is responsible for dealing with requests from the client and handing off the request on the thread to the PHP processor. Then, I am using MySQL as a database to store persistent data (Store Information, Menu Item Information) in tables to be used in requests. The PHP is a programming language that I am using to store requests and get access to the persistent data stored in the MySQL database. The user interface runs on the client and the data is stored on the server. The system will run on one machine. The Donut Shop system is a local webserver on my computer and so there is only one client computer.

Identifying Subsystems

The UML package diagram shown below portrays the subsystems of the donut shop system. The Donut Shop system is a client-server architectural style which is why there is an application layer or client layer and then the database storage layer which is the server layer. The application has a Users package which interacts with the User Interface package that indicates what the user sees on the display. This is based on HTML code and CSS styling with buttons and forms on web pages to show the user information and allow the user to input information. This accesses information for the store processing, food category processing, and menu processing packages which handle adding, modifying, and removing stores, food categories, and menu items. These packages then retrieve or send store, food category, and menu items inputted by the user to and from the database storage layer. It does this by getting the data inputted and sending a MySQL Query which is then sent to the MySQL DBMS or database. The database is then updated or responds to the query with requested data to then be displayed to the user on the web page.

UML Package Diagram:



Persistent Data Storage

The Donut Shop system requires to save data outliving a single execution of the system. The persistent objects are the Stores, the Users (username, password), the Food Categories, and the Menu Items. The storage management strategy for storing these objects and data is to store them in a relational database. I am using MySQL as a database management system and a database.

Global Control Flow

Execution Orders

The Donut Shop system is an event-driven system that waits in a loop for events, and every user can generate actions in a different order. The system has web pages with buttons that the user clicks with a mouse. These events trigger actions such as going to another web page or running an sql query to insert information that was filled out by the user in a form. Some of these buttons are displayed on the same webpage which give the user the choice to do different actions such as adding a store, editing store information, or deleting a store which allows users to generate actions in a different order.

Time Dependency

The Donut Shop system does not have timers. The Donut Shop system is of event-response type, with no concern for real-time. The system is periodic but there are no time constraints for each period. Instead, whenever the user is on the homepage, the system queries the sql database and then displays the table or data on the webpage. Whenever, the user wants to add, edit, or delete data from the database, the user goes to a different page and then loads back to the homepage where the system queries the sql database again to display any updates to the database. The time dependency is able to be of event-response type with no concern for real-time because only one computer uses the system.

Hardware Requirements

The Donut Shop system depends on system resources such as a screen display. The system will also need some hard drive disk storage to hold files and XAMPP software.

- Color Display: Minimum resolution: 640 x 480 pixels
- Computer: Desktop/PC or Mobile Device (cellphone/tablet)
- Memory: 1 Gb RAM
- Hard Drive: Minimum 1 Gb hard drive disk space

User Interface Design and Implementation, Design of Tests

User Interface Design and Implementation

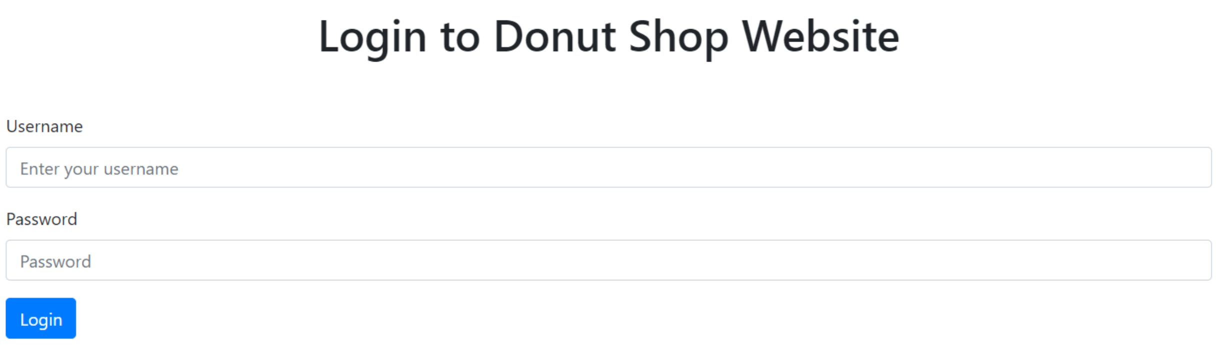
Most of the user interface that I implemented is similar to the initial screen mock-ups I created in the User Interface Specification document. However, I did make some revisions to the initial screen mock-ups. One part I revised or added is a logout button in the navigation bar at the top right of the web pages. The logout button being in a navigation bar at the top of every page makes it so that the user can log out of the application at any time with one click of a button. This requires little user effort. Another revision I made to the user interface is that I added a welcome to the specific user at the top right of the web pages so that the user knows that they are logged in with their own credentials. A third revision I made to the user interface is that I added a title “Donut Shop” at the top, middle of all of the web pages. This makes all of the users know what application or system they are using. A fourth revision I made is that an error message shows up at the top of the screen if a user inputs the wrong credentials while logging in. This helps the user know what they did wrong and what they should do.

One significant change I made to the user interface that reduces the user effort is that there is no delete store confirmation page after the user clicks a delete button on one of the stores. This reduces the user effort because the user does not have to click delete a second time. However, I think I will add the delete confirmation page because accidentally deleting something when you didn’t mean to can feel terrible or cause more problems. They might not remember what they accidentally deleted and potentially lose that data.

Another change is that there are no cancel buttons in the “add store” page and “edit store” page. However, the user can still click the “store list” navigation bar button to navigate back to the store list page when they do not want to add or edit a store. The “store list” navigation bar button requires the same user effort (one click) as the “cancel” button. I think I will still add the cancel button as another way to cancel what the user is doing and go back to the “store list” page. This makes it more clear to the user how to exit or cancel what they are doing.

The other changes I made is that I added a zip code to the stores and instead of listing “address” on the store list page, I listed the detailed sections of the address. In addition, I put input fields on their own line. I also got rid of the home button on the navigation bar because the “store list” is the homepage. The navigation bar “menu” button, menu pages, and the search field are planned to be added for the second demo. I implemented the user interface design and initial screen mock-ups by using a combination of PHP, HTML, and CSS. The CSS was used to help style the HTML created page and the PHP was used to help query, send, and retrieve information from the database.

Screenshots:
User Interface for Login Case



Login to Donut Shop Website

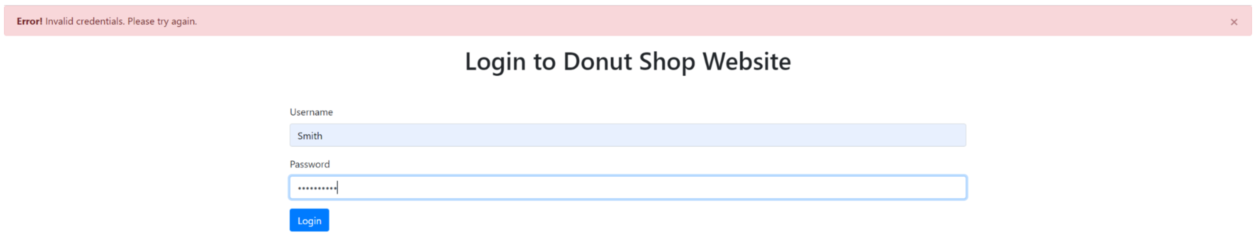
Username

Enter your username

Password

Password

Login



Error! Invalid credentials. Please try again. ×

Login to Donut Shop Website

Username

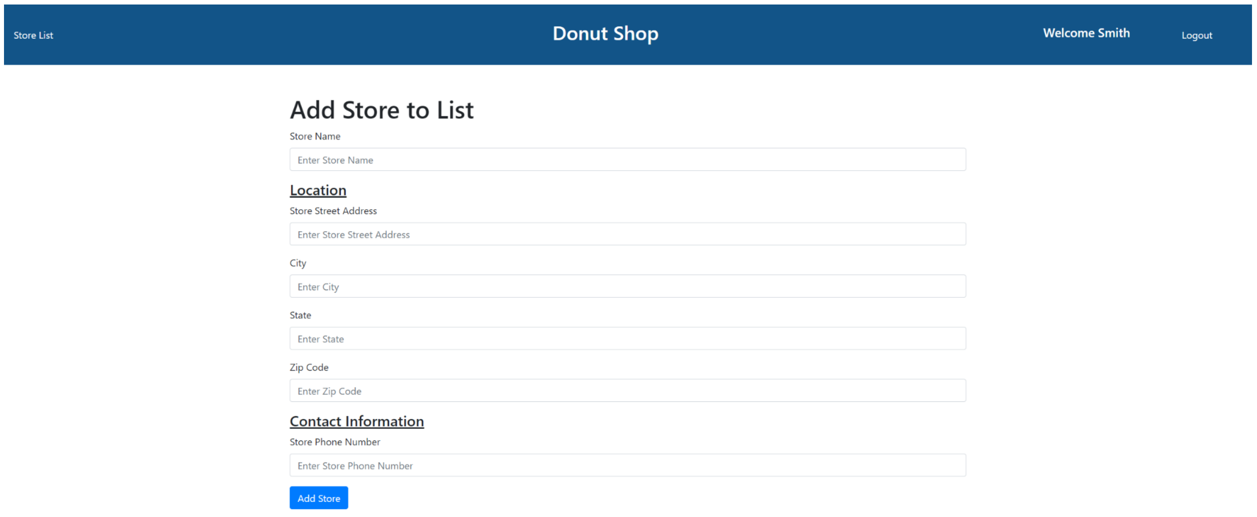
Smith

Password

••••••••••

Login

User Interface for Add Store Use Case



Add Store to List

Store Name

Enter Store Name

Location

Store Street Address

Enter Store Street Address

City

Enter City

State

Enter State

Zip Code

Enter Zip Code

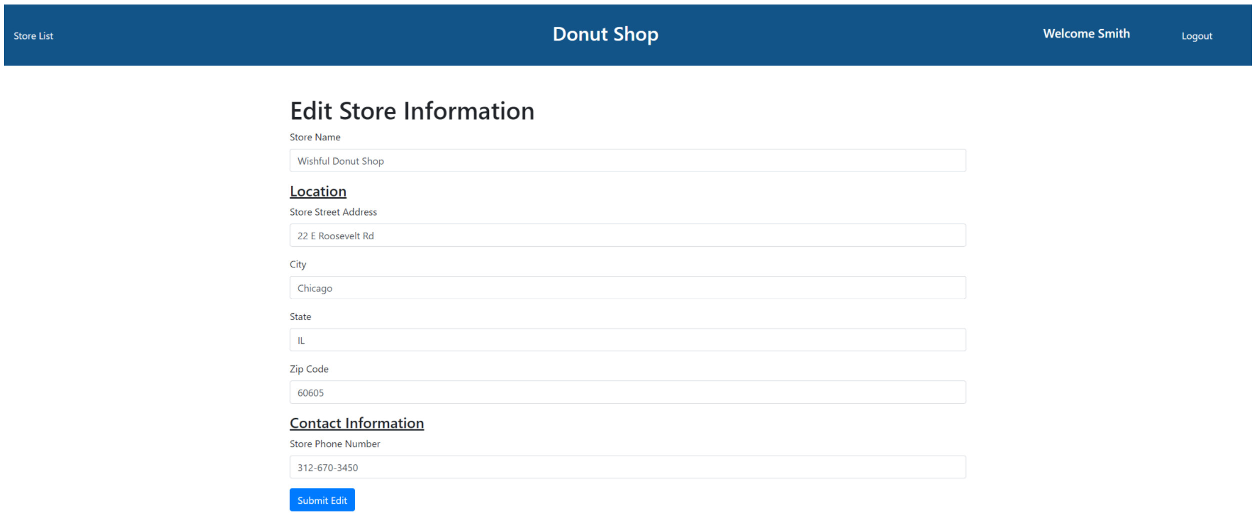
Contact Information

Store Phone Number

Enter Store Phone Number

Add Store

User Interface for Modify/Edit Store Use Case



Store List | Donut Shop | Welcome Smith | Logout

Edit Store Information

Store Name: Wishful Donut Shop

Location

Store Street Address: 22 E Roosevelt Rd

City: Chicago

State: IL

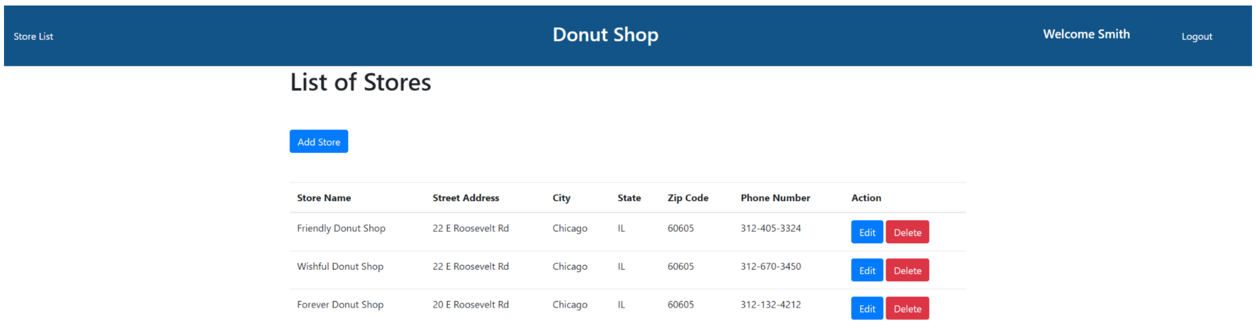
Zip Code: 60605

Contact Information

Store Phone Number: 312-670-3450

Submit Edit

User Interface for View Stores, View Store Details, Add Store, Modify/Edit Store, Remove/Delete Store, Logout Use Cases



Store List | Donut Shop | Welcome Smith | Logout

List of Stores

Add Store

Store Name	Street Address	City	State	Zip Code	Phone Number	Action
Friendly Donut Shop	22 E Roosevelt Rd	Chicago	IL	60605	312-405-3324	Edit Delete
Wishful Donut Shop	22 E Roosevelt Rd	Chicago	IL	60605	312-670-3450	Edit Delete
Forever Donut Shop	20 E Roosevelt Rd	Chicago	IL	60605	312-132-4212	Edit Delete

Design of Tests

I plan to do some unit testing of the Donut Shop system to make sure individual components or modules of code work correctly. One of the test cases that will be programmed and used for unit testing of the Donut Shop system include testing that the invalid credentials notification works properly. This test case includes testing that if the ‘invalid’ variable is true, then the error banner notification is properly printed out. Another test case would test the opposite to show that the invalid credentials notification is not shown if the ‘invalid’ variable is false or login is successful. Instead, the unit test will make sure that a log in success banner is shown. By testing both sides, possible results, and making sure the results match the expected results this increases the test coverage. Another unit test is testing that the redirecting back to the login page works such as if the user is not logged in for the session. The unit test will test if the expected outcome of being redirected back to the login page works as intended. There should also be a unit test that shows that if the user is logged in for the session, the user is not redirected back to the login page unless they log out. This will increase the test coverage of the results. Another test case includes testing the storage of array data from a function and the logic of it. No matter if an array of data is retrieved from a database or retrieved from an already created array, the data should be able to be correctly stored in variables for use such as displaying in a table. This unit test will show that this function and the variables are working correctly by comparing values to the expected results. Other unit tests include testing the post methods to make sure the user input is retrieved correctly from input fields in the web forms. The unit tests would check that the individual inputs match the expected results and that the patch method or mock values have the correct parameters and expected results. More unit tests will be added with any new feature I add to the program. The goal is to have the test coverage of my tests to be 100% or close to that so that I test most of the code of my application.

The Integration Testing strategy that I will use is bottom-up integration testing. I plan on conducting bottom-up integration testing by first testing if lower level modules of code work together. Then, I will continue to test with more and more modules up to the higher level modules to make sure the program works. I plan on conducting the integration testing by testing the interface connection between the Login module and the Store List module. I will test that when correct login credentials are entered, the expected result is for the user to be redirected to the Store List page. I will also test other cases of expecting error messages when the user inputs the wrong login credentials. Also, I will test that the user should not be able access the store list module or any other page if the user is not logged in, and instead should expect the result to be that the user is redirected to the login page. Then, I will test the interface connection and flow of data between the Store List module and the Add Store Module. I expect that inputting data into the Add Store module and clicking “add store” will result in the store data to be shown as a new row in the Store List module. I will test that the modules are successfully communicating or querying the database. These are some examples of how I plan on conducting integration testing.

I do not have any algorithms to test. However, I plan on testing non-functional requirements by timing how long it takes for certain actions to happen. For example, one of my non-functional requirements is that the response time for loading web pages should not exceed 5 seconds. Another non-functional requirement is that the system must filter for shops in less than 4 seconds. These non-functional requirements can be tested by using a timer to see if the load time took shorter or longer than the required maximum amount of time. I plan on testing user interface requirements by purposefully inputting wrong information into the input fields to make sure that the system responds properly by displaying error messages. An example of one of my user interface requirements is that if the user inputs the wrong login information, the user interface should display a notification and/or highlight the input boxes that need to be fixed. I can test this by inputting invalid credentials and then see if the user interface displays the expected result of showing an invalid credentials error message notification.

Project Proposal

Problem Statement
The donut shop website information system is useful to help fix the issue of customers not having access to shop information in an organized, easy to use, fast, and useful manner. This website information system will allow customers to see store information digitally around the world and while on the go with a mobile device. This system will address the issues of customers not knowing the location of stores nearby them, the store hours, and details about menu items. This system provides an opportunity for the business to grow, expand, and possibly get more customers by being discoverable on the Internet. Also, the system will help the donut shop stay competitive with other businesses using websites. This website information system can be extended to be used for any restaurant or business.

Objectives of the System
The objectives of the system focus on trying to help manage and keep basic shop and menu information up to date. This system will help make the shop information easily accessible and organized. This system will help shop owners and managers easily be able to add new shops and remove shops that have closed down. This would increase transparency by making shop hours, locations, and recent shop openings/closings easily findable. Shop customers will be able to favorite stores and find stores nearby.

System Requirements
The requirements for the system are that the system must be able to let customers and shop owners do these activities:
What customers can do with the system:

- Favorite stores
- Favorite categorize menu items
- View stores nearby based on the city or postal code
- Keyword search and filter for stores based on the city or postal code
- View a list of all stores
- View categories, menu items, and their details to find items that the customer wants
- View basic store information

What Shop owners can do with the system:

- Add stores with store details and locations
- Add food categories, menu items, and details
- Modify stores with store details and locations
- Modify food categories, menu items, and details
- Remove stores with store details and locations
- Remove food categories, menu items, and details

Typical Customers
The system will have different access and features tailored to these different groups:

- Customers (people who buy food at the donut shop or like donuts)
- Donut shop owners or managers

Project Planning and Development Approach

The software, hardware, and network requirements for project planning are listed below. The development approach and technical stack is in the software section.

1. Software
    a. Front-end: HTML/ CSS
    b. Back-end: PHP
    c. Database: MySQL
2. Hardware
    a. Desktop/PC
    b. Mobile Devices (cellphone, tablet)
3. Network
    a. Good Internet speed
    b. As little website downtime as possible
    c. Scheduled website maintenance once a week

Project Plan

Development Plan

Week 1: Solidify what front-end, back-end, and database programs I am going to use.
Week 2: Set up a project development environment by connecting front-end, back-end, and database software together, figure out the structure and framework for the system.
Week 3: Create the login for the system so customers and shop owners can login.
Week 4: Create the registration for new customers and shop owners to make a login or account.
Week 5: Design the features for shop owners to add, modify, or remove stores, store details (store name, phone number, street address, city, state, country, zip code), food categories, menu items, and menu item details.
Week 6: Implement the features for shop owners to add, modify, or remove stores, store details (store name, phone number, street address, city, state, country, zip code), food categories, menu items, and menu item details.
Week 7: Continue to implement the features for shop owners to add, modify, or remove stores, store details (store name, phone number, street address, city, state, country, zip code), food categories, menu items, and menu item details.
Week 8: Verify the features work, record, and turn in current progress for the midterm.
Week 9: Iterate and improve current features for the shop owners.
Week 10: Design the features for customers to be able to view the stores and filter for stores based on their city or postal code. Design the features for customers to view basic store, menu category, and menu item information
Week 11: Implement the features for customers to be able to view the stores and filter for stores based on their city or postal code. Implement the features for customers to view basic store, menu category, and menu item information
Week 12: Design the features for customers to be able to favorite stores and menu items.
Week 13: Implement the features for customers to be able to favorite stores and menu items.

Week 14: Debug and test to make sure all features work as intended.
Week 15: Test and record the demonstration and progress made for the final presentation.

Reference

Submitted Documents in Canvas

1. Proposal
2. Customer Problem Statements and System Requirements
3. Functional Requirements Specification
4. System Sequence Diagram and Activity Diagram
5. User Interface Specification