|  |
| --- |
|  |
| Zach Nafziger |
| Seth Loew |
| Aaron Rosenberger |
| Michael Kytka |
| Sam Kibler |
| 12/9/2015 |
| Editor: Aaron Rosenberger |
| COMP 451 Senior Project I |

**Table of Contents**

[1 Introduction 1](#_Toc437428775)

[1.1 Purpose of system 1](#_Toc437428776)

[1.2 Target user group 1](#_Toc437428777)

[1.3 Target hardware and software platform 1](#_Toc437428778)

[1.4 Risks 1](#_Toc437428779)

[2 User Interface 2](#_Toc437428780)

[2.1 Workflow for entire application 2](#_Toc437428781)

[2.2 Design rationale and context 2](#_Toc437428782)

[2.3 Mockups and Screenshots 3](#_Toc437428783)

[2.4 Cognitive walkthrough for each task 21](#_Toc437428784)

[2.4.1 Create an account 21](#_Toc437428785)

[2.4.2 Login 21](#_Toc437428786)

[2.4.3 Make a new order 21](#_Toc437428787)

[2.4.4 Accept available order request 22](#_Toc437428788)

[2.4.5 Update status to available 22](#_Toc437428789)

[2.4.6 Update status to unavailable 23](#_Toc437428790)

[2.5 Improvements made through iterative design process 23](#_Toc437428791)

[3. Testing Plan 27](#_Toc437428792)

[3.1 Overview 27](#_Toc437428793)

[3.2 Acceptance Tests 27](#_Toc437428794)

[3.2.1 Login View 27](#_Toc437428795)

[3.2.2 Create Account view 28](#_Toc437428796)

[3.2.3 Terms and Conditions view 30](#_Toc437428797)

[3.2.4 Home view 31](#_Toc437428798)

[3.2.5 I want food view 31](#_Toc437428799)

[3.2.6 Select Restaurant view 33](#_Toc437428800)

[3.2.7 Food Item view 34](#_Toc437428801)

[3.2.8 Available Drivers view 34](#_Toc437428802)

[3.2.9 Delivery Location view 35](#_Toc437428803)

[3.2.10 Expires In view 36](#_Toc437428804)

[3.2.11 Current Order view 38](#_Toc437428805)

[3.2.12 My Orders view 39](#_Toc437428806)

[3.2.13 Restaurants Tab view 39](#_Toc437428807)

[3.2.14 Orders Tab view 42](#_Toc437428808)

[3.2.15 Order Pickup view 42](#_Toc437428809)

[3.2.16 Paid for Food view 43](#_Toc437428810)

[3.2.17 Order Confirmation view 44](#_Toc437428811)

[3.2.18 Settings view 44](#_Toc437428812)

[4 Ethics and Intellectual Property 47](#_Toc437428813)

[4.1 Intellectual Property Rights 47](#_Toc437428814)

[4.2 Plans for After College 47](#_Toc437428815)

[4.3 Trade Secrets 48](#_Toc437428816)

[4.4 Trademarks 48](#_Toc437428817)

[4.5 Copyrights 48](#_Toc437428818)

[4.6 Ethical Issues 48](#_Toc437428819)

[4.7 Societal Implications 48](#_Toc437428820)

[4.8 Legal and Regulatory Issues 49](#_Toc437428821)

[5 Summary 49](#_Toc437428822)

[6 Bibliography 50](#_Toc437428823)

# 1 Introduction

This brief overview of Foodini provides some context for and general description of the application’s purpose.

## 1.1 Purpose of system

Foodini is an application that connects people who want food with people who are willing to pick up food from local restaurants. Users can submit their food requests and drivers can select orders to accept from a list of orders. Drivers can also specify certain restaurants and receive users’ requests for that restaurant.

## 1.2 Target user group

The system targets people who are hungry and want food. It also targets people with cars who are willing to pick up food. Although the application can serve a wide range of locations and people, we will initially launch it on college campuses.

## 1.3 Target hardware and software platform

The system will install and run on Apple iPhone running iOS 8 or later, and follow Apple’s iOS Human Interface Guidelines [1]. It will use Apple Maps [2] for location data.

## 1.4 Risks

The greatest risk for the success of our software is the very small user base when the app is initially released. Another risk is that the scope of the project is too large for the group to finish in time. There is the potential that some Apple Maps data may not be completely up to date. There is always the risk that we will discover an obstacle we did not consider while developing Foodini.

# 2 User Interface

The following section details the plan for Foodini’s user interface.

## 2.1 Workflow for entire application

The figure below shows the workflows for the application. The left workflow is for customers, and the right workflow is for drivers.

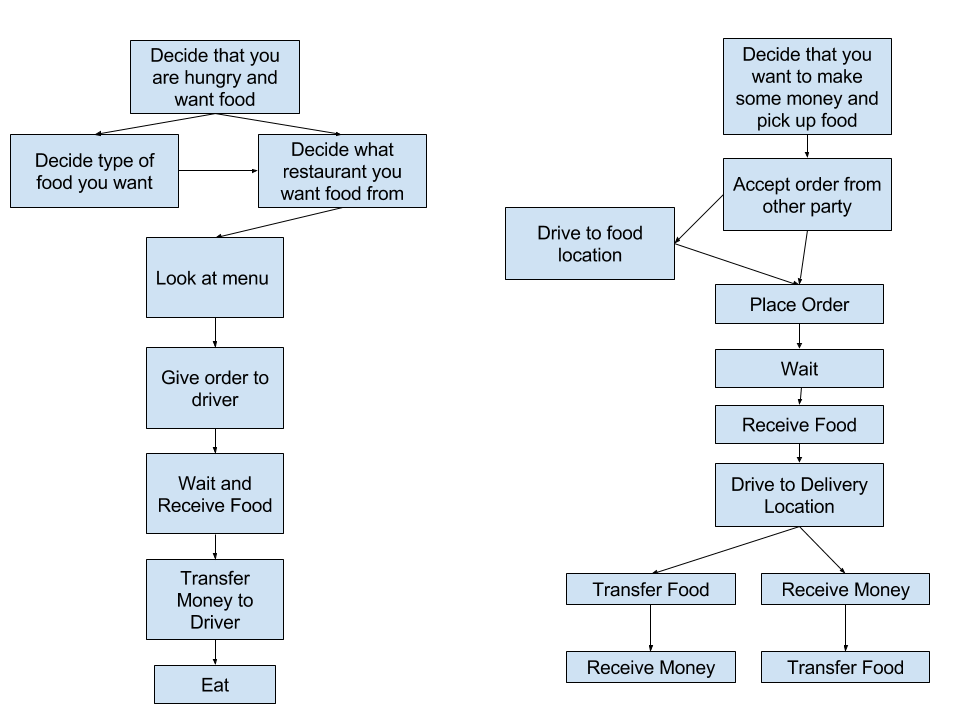


Figure 1 - Workflow

## 2.2 Design rationale and context

We consulted several sources during the design process, including Apple’s iOS Human Interface Guidelines [1], Apple’s Design Principles [3], and Lewis and Rieman’s Task-Centered User Interface Design [4]. The project was inspired by the rise of people working as private contractors [5]. With this in mind, we took some cues from applications such as Uber [6], including the concepts and iOS interfaces, along with several other applications including GrubHub [7] [8] [9], Domino’s Pizza [10], Uber Eats [11], Starbucks [12], and Seamless [13]; and Yik Yak’s location-based functionality [14]. We strive to enter the market along with possible well-known competitors such as DoorDash [15], Caviar (to a lesser degree) [16], and PostMates [17] as well in certain areas, notably if we expand our idea to certain metropolitan areas. Vishrut Milay’s article “The Bits and Bytes of Food: a Study of Emerging Internet-based Food Businesses” shed some light on the successful businesses and concepts that the team should be aware of throughout the beginning stages of the product planning and implementation stages [18].

After much discussion and research, we decided to forego payment within the system. Rather than using services such as Apple Pay [19], Stripe [20], or PayPal [21], we will allow users of the system to complete payment through an external application. Although it would be possible to incorporate these services, we cannot feasibly do so within the scope of the development period, largely because onerous legal considerations would accompany the use of these services [22].

One of the goals is to maximize user trust with the application. We must design the application with user trust in mind because if the average users does not trust Foodini, few people will use the application and it will fail economically. Positive feelings of reliability and security are essential to build and maintain users' faith in the software [23] [24] [25].

## 2.3 Mockups and Screenshots

Figures 2 – 18 demonstrate how we plan to implement the interface, as well as use cases for the application. The screens are described below.

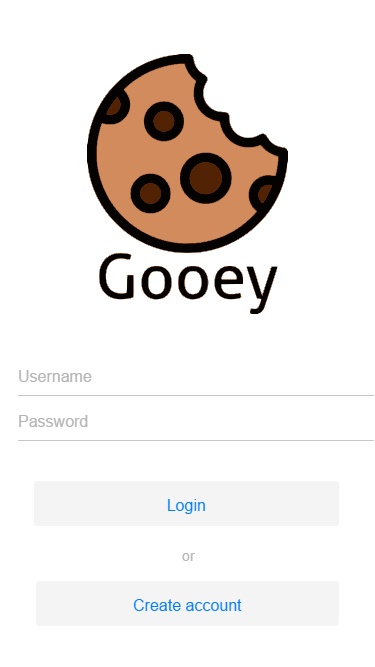


Figure 2 - Login view

The "Login" view is what the user will see the first time that the application is opened. From this page, the user can enter his username and password and then press "Login" to move to the “Home” view (Figure 5), or press "Create account" to move to the "Create Account" view (Figure 3).

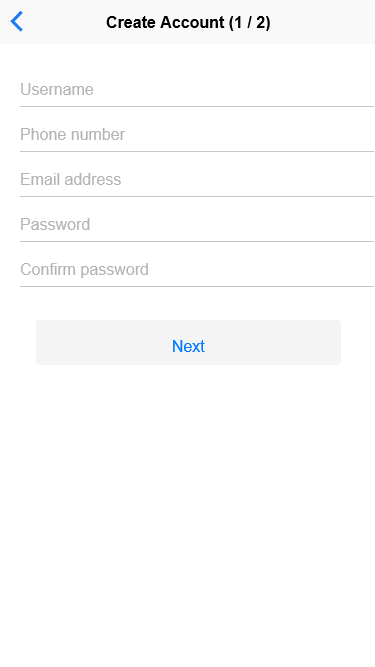


Figure 3 - Create Account view

On the “Create Account” view, the user must enter his username, phone number, email address, and a confirmed password in order to proceed with creating an account. The top left arrow button takes the user back to the “Login” view, and the “Next” button takes the user to the “Terms and Conditions” view (Figure 4).

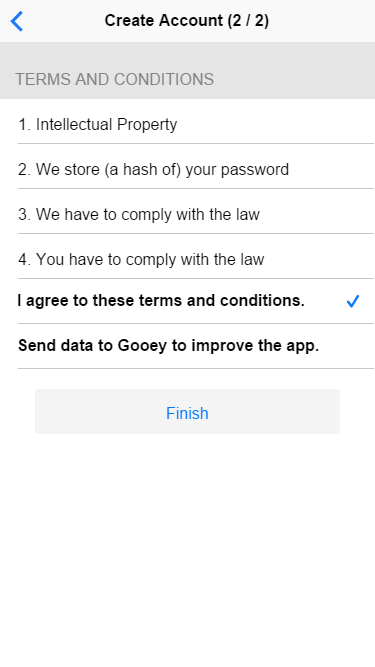


Figure 4 - Terms and Conditions view

On the "Terms and Conditions" view, the user must agree to the terms and conditions in order to finish the registration process. He can also check a box to send interaction data to Foodini improving future releases of the application. The user can press the “Finish” button to move to the "Home" view (Figure 5) or press the top left arrow button to return to the "Create Account" view (Figure 3).

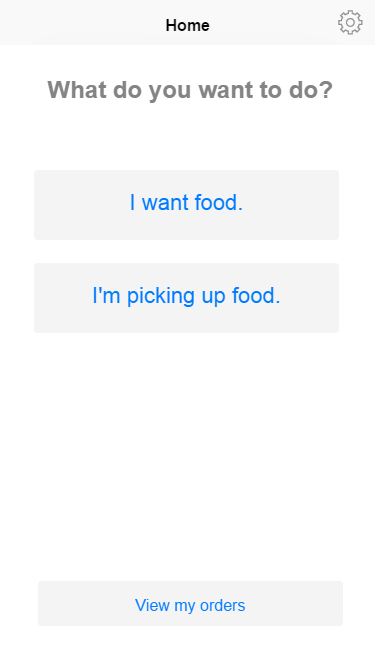


Figure 5 - Home view

On the "Home" view, the user can navigate to the various main views within the system. The user can tap on the "I want food." button to move to the "New Order" view (Figure 6), or he can tap on the "I’m picking up food." button to move to the "Restaurants Tab" view (Figure 14). The user can also tap on the settings cog in the top right corner to move to the “Settings” view (Figure 18).

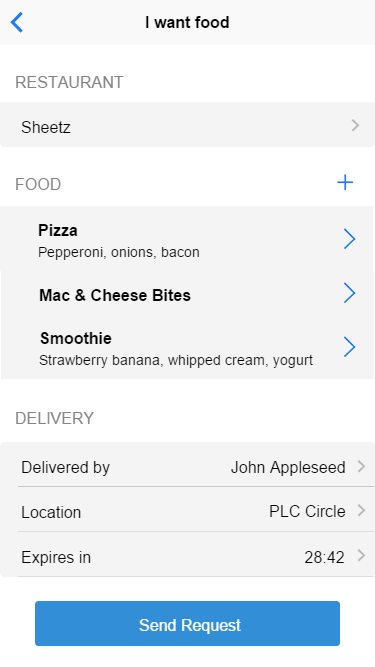


Figure 6 - New Order view

In the "New Order" view, a user can select a restaurant, add food items, choose a driver, and send his request. To return to the "Home" view (Figure 5), he can tap the top left arrow button. The user can tap the on the restaurant field to move to the "Select Restaurant" view (Figure 7) to select his desired restaurant. The user can tap the plus button next to the food header to move to the "Food Item" view (Figure 8), with a blank item. He can tap one of the existing food items to move to the "Food Item" view (Figure 8) but with the proper fields filled according to his selection. If he taps the driver, the system takes him to the "Select Driver" view (Figure 9). If he taps the delivery location, the system takes him to the "Delivery Location" view (Figure 10). If he taps the "Send Request" button, the application sends his order to the selected driver and takes him to the "Paid for Food" view (Figure 17).

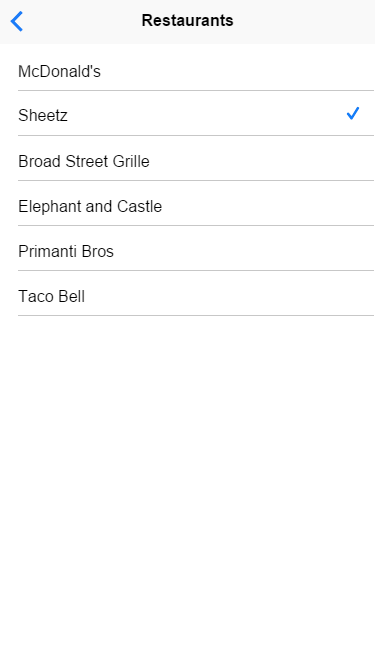


Figure 7 - Select Restaurant view

In the "Select Restaurant" view, the user can see a list of the restaurants within five miles of the user’s location. If a user taps one, a check appears next to it, indicating that he has selected it, and the check from the previously selected restaurant disappears to indicate that in doing so, he has unselected his prior restaurant choice. If the user taps the top left arrow button, he moves back to the "New Order" view (Figure 6).



Figure 8 - Food Item view

In the "Food Item" view, the user can create or edit food items for his order. He can type the order name in the top text bar, as well as any additional details about the order in the text area beneath. If he taps the top left arrow button, he returns to the "New Order" view (Figure 6).

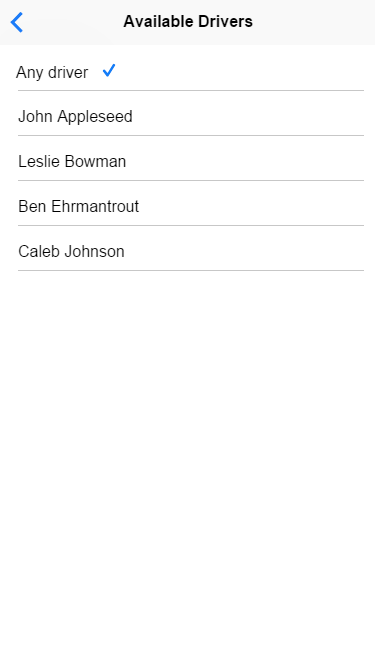


Figure 9 - Select Driver view

In the "Select Driver" view, the user can view the drivers who have marked themselves as available. If a user taps the “Any driver” list item, any driver can pick up his order. If the user taps the top left arrow button, he returns to the "New Order" view (Figure 6).

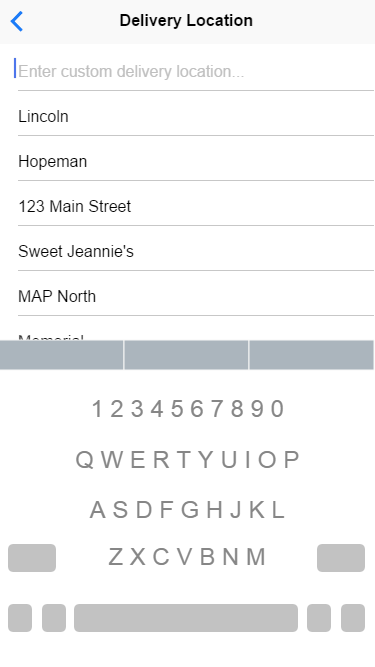


Figure 10 - Delivery Location view

In the "Delivery Location" view, the user can select a location from the pre-populated list, or he can create a custom location. If he taps the name of a location, a check mark appears beside it to indicate the selection. If the user taps the text field above the list, a keyboard appears for typing a custom location. If he taps the top left arrow button, he returns to the "New Order" view (Figure 6).

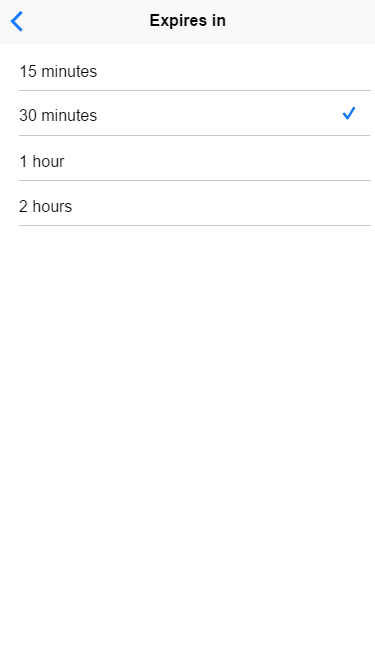


Figure 11 - Expires In view

In the “Expires In” view, the user can tap on the desired time after which his order will expire. A checkmark appears beside the selected time. If the user taps the top left arrow button, he will return to the “New Order” view (Figure 6).

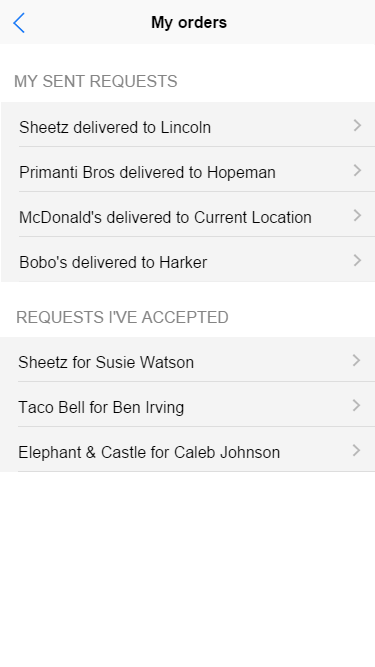


Figure 12 - My Orders view

In the "My Orders" view, the user can see sent requests and accepted requests. The "My sent requests" section shows a list of orders that the user has sent for a driver to pick up. The "Requests I’ve Accepted" section shows a list of orders that the user has accepted and will deliver food to that specific person. If the user taps the top left arrow button, he will return to the “Home” view (Figure 5).

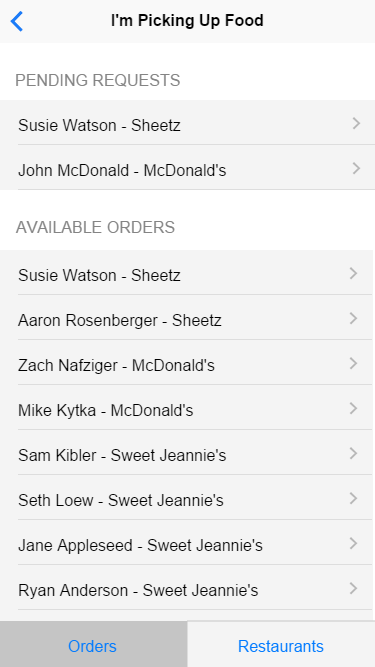


Figure 13 - Orders Tab view

The "Available Orders" view appears when the user taps on the "I’m picking up food." button from the "Home" view (Figure 5). In this view, the driver can see requests sent specifically to him, and requests in which the user specified “Any driver.” If the driver selects any of these list items, he will move to the “Order Pickup” view (Figure 15). If the driver taps the top left arrow button, he returns to the "Home" view (Figure 5).

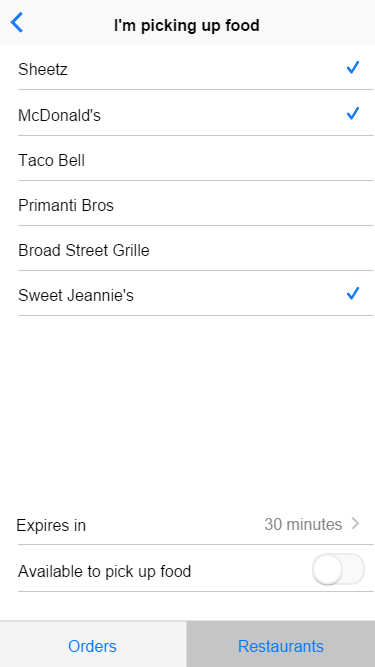


Figure 14 - Restaurants Tab view

Another option for the driver is to select "Restaurants" in the bottom of the screen, which brings him to the “Restaurants Tab” view (Figure 14). This screen allows the driver to select to which restaurants he is willing to drive. The driver can also choose for how long he is willing to get food, and he can also easily toggle his availability using the “Available to pick up food” switch. If the user taps on the top left arrow button, he returns to the “Home” view (Figure 5).

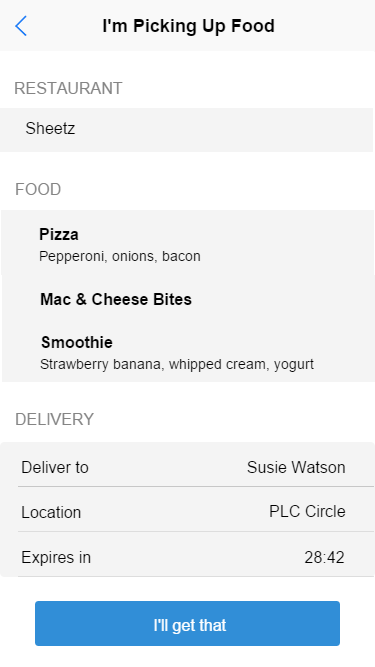


Figure 15 - Order Pickup view

After the driver has selected an order from the "Orders Tab" view (Figure 13), he will see the details of that order on the “Order Pickup” view (Figure 15). The driver can see the restaurant, specific food items, and the delivery information. If the driver wants to get the food for the customer, he will tap on the "I’ll get that" button. If the driver taps the top left arrow button, he will return to the “Orders Tab” view (Figure 13).

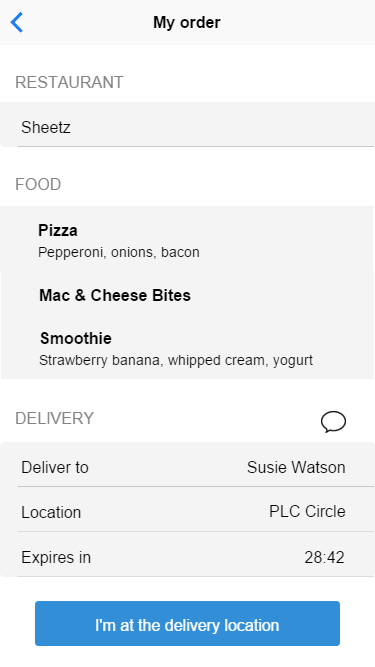


Figure 16 - Order Confirmation view

In the "Order Confirmation" view, the driver can see the summary of the order he is picking up. When the driver is at the delivery location, he will tap "I’m at the delivery location." The driver can tap the chat bubble icon next to the “Delivery” heading if he wants to open a new iMessage to the user. If the user taps the top left arrow button, he returns to the “My Orders” view (Figure 12).

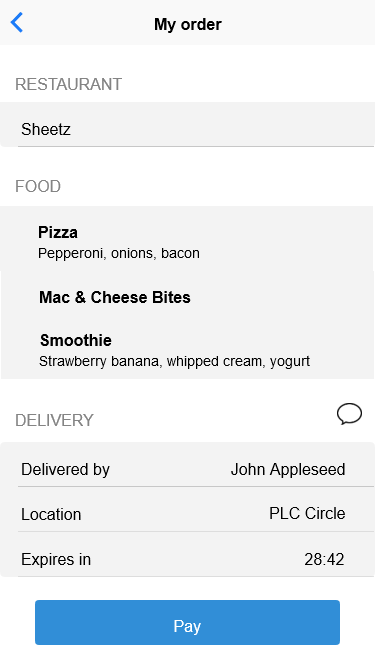


Figure 17 - Paid for Food view

On the “Paid for Food” view, the user’s order has been accepted. The user can open a new iMessage to the driver by tapping the chat bubble button. When the user is ready to pay for the order, he will tap the “Pay” button to move to an external application for payment. If the user taps on the top left arrow button, he returns to the “My Orders” view (Figure 12).

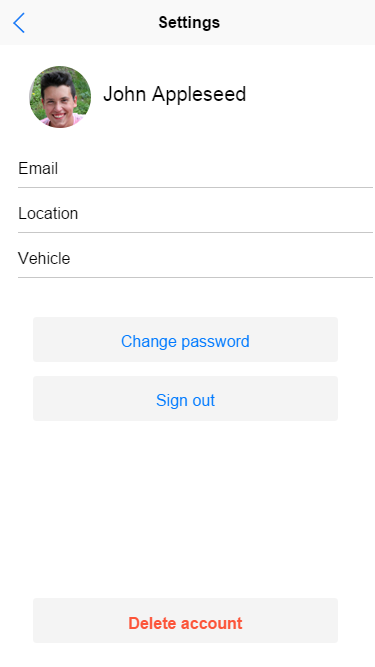


Figure 18 - Settings view

In the “Settings” view, the user can update his email address, add his location, or add his vehicle information. He can also change his profile picture by tapping his current picture, change his password, log out, or delete his account. If the user taps the top left arrow button, he returns to the “Home” view (Figure 5).

## 2.4 Cognitive walkthrough for each task

The user can complete several tasks including creating an account, logging in, making a new order, accepting an order, updating status to available, and updating status to unavailable.

### 2.4.1 Create an account

The user goes into the application for the first time and notices the "Create account" button. Since it is standard to create an account before using an application, he taps the “Create account” button. He then sees the registration screen and fills out all the fields according to their description. After completing the fields, the user notices a checkmark beside the "Password," "Confirm," and "Email" fields, so the user knows that the email and password he typed is valid.

The user sees the "Next" button and knows there must be more to the process. He taps the button and sees the terms and conditions. The user realizes he wants to use a different email address, remembers that tapping the top left arrow takes him back, so he taps the top left arrow, changes his address and taps to go back to the terms and conditions screen. He taps the box to agree to the terms and conditions and the check box appears next to it. He does the same thing for button immediately beneath. This enables the “Finish” button, so the user knows he is done with the create account process. After tapping on "Finish," a notification appears for a short duration saying his account was successfully created, so the user knows he can now login to the app.

### 2.4.2 Login

Having already created an account, the user opens the app after having logged out the last time he used it. He sees the login screen and notices the typical username and password form fields. He taps the username field and types his username. He then moves to the password field and types in his password. He taps the “Enter” button on the keyboard, but noticed a “Login” button that he could have tapped to do the same thing. He sees the "Home" view, which confirms that he entered his information correctly.

### 2.4.3 Make a new order

While on the "Home" view, the user notices two large buttons at his eye level. He is hungry and wants some food, but does not want to drive to a restaurant. The user understands that the “I want food” button will enable him to order some food, so he taps it and the "New order" screen appears.

The user notices three main groups of controls, separated logically to read “From a restaurant, I want food delivered by a driver to this delivery location.” He wants Sheetz, so he taps on the "Restaurant" control and is sees a list of restaurants on a different page. He instinctively scrolls until he finds Sheetz, taps on it to select it, and is taken back to the New Order screen where he finds Sheetz now listed as the "Restaurant."

Scanning down the page to the next control, the user notices the "Food" group and sees no items there. He is familiar with the plus sign meaning "Add a new item," so he taps it and is taken to a page to fill out a new food item. On this page he notices a one-line field with the default text “Title” and a multiline field with the default text “Description” and is reminded of the iOS style email with subject and body. He types in his desired food item and taps the back arrow, because he remembers it takes him back to the previous screen. The user is then taken back to the New Order screen and notices his new food item appears in the list with the title in bold and the description below it.

Next, the user notices a field that says “Driver.” He knows someone is going to have to pick up his food for him, because he definitely is not leaving the library. Therefore, the user taps this field and is presented with a new page containing a list of available drivers. After scanning the options on the list, the user realizes he does not want to choose a particular person to do this for him. He notices a list item removed from the others, located at the top of the page, entitled “Any driver.” Because he is willing to have anyone pick up his order, he taps that item and is taken back to the "New order" screen with "Any Driver" displayed in the "Driver" field.

The user notices a field below the driver field and taps on it because he knows all form fields must be filled out to complete an order. The temporary text reads “Delivery Location,” and from this the user knows he must specify to what location he wants his food delivered. Tapping on the field, the application brings him to a page with a list and textbox. The user realizes that the list contains a number of possible locations sorted by distance to him. However, he does not see the location he wants in the list, and can tell that the textbox is meant for custom locations. The user enters a custom location and taps the upper left arrow button, which he knows means he will be taken back to the previous screen.

The user notices a field below the delivery location field. The text in the field reads “Expires in,” and from this the user knows he must specify for how long his order should remain active. He notices a time already shown and assumes it is the default time, but he wants his expiration time to be longer, so he knows to tap the field. The application takes the user to a page with a list of elapsed times and the default time already selected. Similar to Apple’s auto-lock timer in iOS settings, the user knows that he can select only one time. He selects an elapsed time and taps the back button, knowing it will return him to the "New order" screen with his newly chosen expiration time shown in the appropriate field.

The user notices a button at the bottom that reads "Send Request." He assumes that tapping the button will send his request, so he glances at all of his previously entered information to ensure correctness. Satisfied, the user taps the "Send request" button and immediately notices a pop up box informing him of his successfully sent request. He now knows his listing is visible to drivers, so he taps "OK."

### 2.4.4 Accept available order request

The driver wants to list himself as available so that people who want food will be able to send him requests. He knows that he is going to end up driving to a restaurant at some point, and he observes the button “I’ll pick up food,” so he selects it. He notices a list of restaurants and a switch at the bottom of the page that reads “Available to pick up food.” From this he determines that he should select the restaurants from which he is willing to pick up food. After the driver finishes the selections, he observes an expiration option right between the restaurants and the switch. He likes the one-hour default expiration so he does not change the field. Remembering the “Available to pick up food” switch, he flips it on and notices his expiration time starts counting down. From this, the driver realizes that his listing will expire in that amount of time.

### 2.4.5 Update status to available

The driver goes to the “Restaurants” screen because he previously went to this page in order to list himself as available. He sees that the “available to pick up food” switch is on, so he realizes that he must switch it off to become unavailable and switches it off. The list of restaurants becomes enabled and the expiration countdown resets, assuring him that he is now listed as unavailable.

### 2.4.6 Update status to unavailable

The driver goes to the “Restaurants” screen because he previously went to this page in order to list himself as available. He sees that the “available to pick up food” switch is on, so he realizes that he must switch it off to become unavailable and switches it off. This enables the list of restaurants and the expiration countdown resets, assuring him that he is now listed as unavailable.

## 2.5 Improvements made through iterative design process

Throughout the design process, we changed the application significantly. Initially, the application was going to include everything that the driver and user could need, including a built-in payment system and a built-in messaging system. However, we removed both from the final design after a number of considerations.

If Gooey held user money, the entity and we as programmers would be legally responsible both for the protection of those funds and for paying drivers, which could potentially result in financial penalties if any source compromised the data or accounts. The initial business model would be difficult to manage because the team members have no former experience with starting a business, along with a busy class and homework schedule. Gooey could also look toward establishing a crowdsourcing business model in the future similar to what is described in “Crowdsourcing-Based Business Models: How to Create and Capture Value” [26]. The business plan is still in flux and resources such as “Creating Value through Business Model Innovation” [27] have been consulted for guidance. The team considered troublesome users of the system, such as those not picking up or delivering an order and still expecting payment. The team ultimately decided that outsourcing payment to a third party like PayPal would satisfy the requirement for user payment. Figures 19 through 21 illustrate some changes made during the iterative design process.

Before: After:

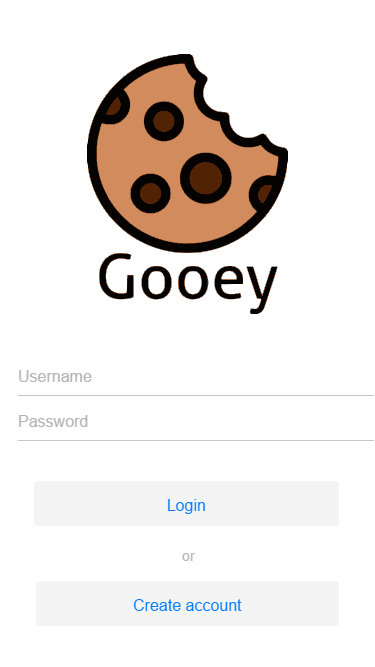
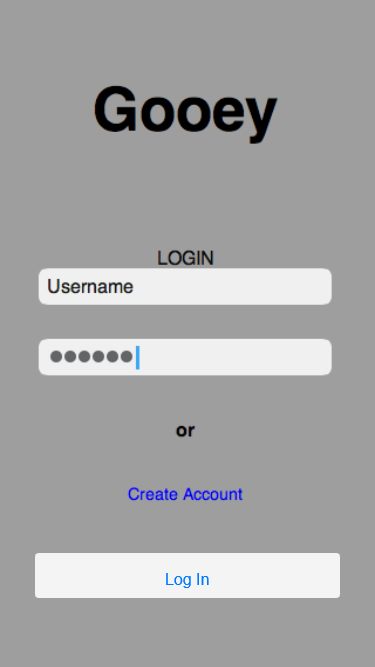


Figure 19 - Login view progression

The first draft of our "Login" view did not make sense. It was not designed correctly because the “Log in” button was beneath the “Create account” link. “Log in” should be the first option that a user clicks on. “Create account” was made as a link instead of a button. We changed the screen to have two buttons on the bottom. This makes it aesthetically pleasing and it gives the screen symmetry. We changed the “Create account” link to a button because that was not iOS standard.

Before: After:

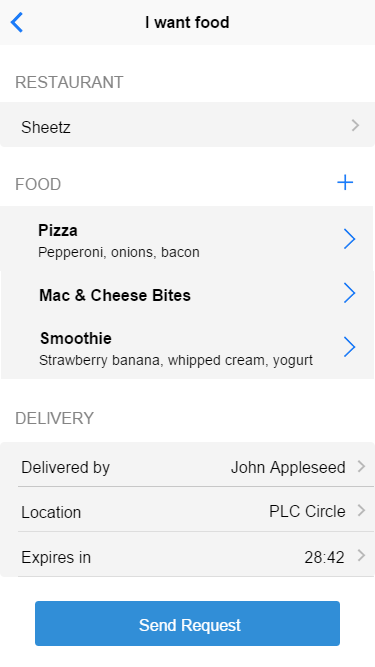
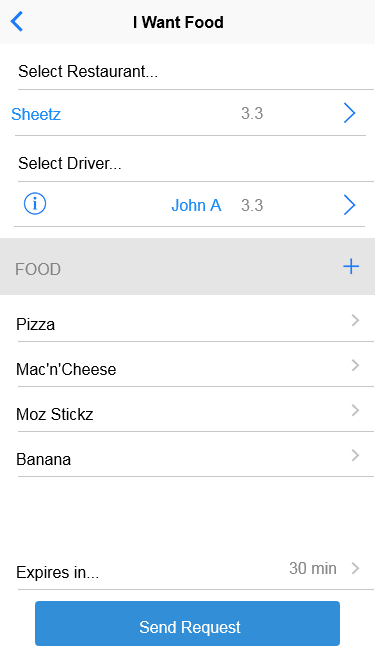


Figure 20 - New Order view progression

The initial attempt of the "New Order" view was confusing and inconsistent because the order of inputs does not flow logically and frustrates the user. The new design has a logical flow and reads well. The controls are grouped by category and shaded gray to make them stand out.

Before: After:

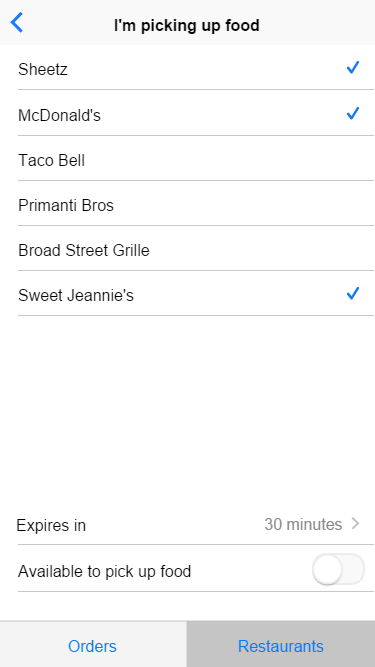
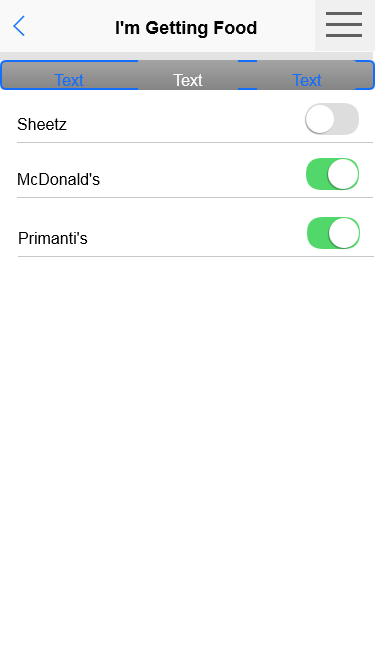


Figure 21 - Restaurant Tab view progression

The "Restaurant Tab" view initially contained switches for the driver to select restaurants for which he would be listed as available. Using checkboxes along with an availability switch and expiration time is the best option for driver convenience. This is because of the scenario where the driver forgets to switch his availability for some restaurants. The change also prevents frustration: previously, the driver would have to toggle a switch for each restaurant whenever he wanted to change his availability. With the change, the driver can toggle his availability for a remembered list of restaurants with a single switch.

# 3. Testing Plan

This section contains all of the information relating to our testing plan.

## 3.1 Overview

Every element of the application will be thoroughly tested to ensure that the team implemented all functions and interfaces correctly.

## 3.2 Acceptance Tests

The following list of test cases will ensure that our applications meets the standards specified for our application.

### 3.2.1 Login View

Test name: Signing in (correct)

Test number: 1

Element being tested: Login button

Preconditions: The user is on the "Login" view and has filled out the username and password fields correctly.

Test procedure: The user will tap the "Login" button.

Expected outcome: The user will be signed in and taken to the home view.

Results:

Test name: Signing in (incorrect)

Test number: 2

Element being tested: Login button

Preconditions: The user is on the "login" view and has filled out the username and password fields incorrectly.

Test procedure: The user will tap the "login" button.

Expected outcome: The user will not be signed in and the username and password fields will be outlined in red.

Results:

Test name: Enter correct password & tap enter

Test number: 3

Element being tested: Password field on "login" view

Preconditions: The user is on the "login" view and has filled out a username.

Test procedure: The user will enter a password correctly and tap the enter keyboard button.

Expected outcome: See test for "signing in (correct)".

Results:

Test name: Enter incorrect password & tap enter

Test number: 4

Element being tested: Password field on "login" view

Preconditions: The user is on the "login" view and has filled out a username.

Test procedure: The user will enter a password incorrectly and tap the enter keyboard button.

Expected outcome: See test for "signing in (incorrect)".

Results:

Test name: No zombie accounts

Test number: 5

Element being tested: Login button

Preconditions: The user is on the "login" view. The user previously selected "delete account" from the "Settings" view, typed his proper password, and confirmed "delete account" in the modal popup, returning him to the "login" view.

Test procedure: Press "login"

Expected outcome: The user will not be signed in and the username and password fields will be outlined in red. This verifies that auto-login will fail after account deletion.

Results:

Test name: Create account

Test number: 6

Element being tested: Create account button

Preconditions: The user is on the "login" view.

Test procedure: The user will tap the "create account" button.

Expected outcome: The user will be brought to the "Create account" page.

Results:

### 3.2.2 Create Account view

Test name: Enter username

Test number: 7

Element being tested: Username text field

Preconditions: The user is on the "Create account" view.

Test procedure: The user will tap the "Username" text field.

Expected outcome: A keyboard will appear allowing the user to type his username.

Results:

Test name: Enter phone number

Test number: 8

Element being tested: Phone number text field

Preconditions: The user is on the "Create account" view.

Test procedure: The user will tap the "Phone number" text field.

Expected outcome: A keyboard will appear allowing the user to type his phone number.

Results:

Test name: Enter email address

Test number: 9

Element being tested: Email address text field

Preconditions: The user is on the "Create account" view.

Test procedure: The user will tap the "Email address" text field.

Expected outcome: A keyboard will appear allowing the user to type his email address.

Results:

Test name: Enter password

Test number: 10

Element being tested: Password text field

Preconditions: The user is on the "Create account" view.

Test procedure: The user will tap the "Password" text field.

Expected outcome: A keyboard will appear allowing the user to type his password.

Results:

Test name: Confirm password

Test number: 11

Element being tested: Confirm password text field

Preconditions: The user is on the "Create account" view and the "Password" text field is filled in.

Test procedure: The user will tap the "Confirm password" text field.

Expected outcome: A keyboard will appear allowing the user to type his password again to confirm his password.

Results:

Test name: Confirm password

Test number: 12

Element being tested: Confirm password text field

Preconditions: The user is on the "Create account" view and the "Password" text field is not filled in.

Test procedure: The user will tap the "Confirm password" text field.

Expected outcome: A pop up will appear stating that the user cannot confirm his password because he did not type in his password yet.

Results:

Test name: Create account next screen

Test number: 13

Element being tested: Next button

Preconditions: The user is on the "Create account" view and everything is okay and the password and confirm password texts are the same.

Test procedure: The user will tap the "Next" button

Expected outcome: The user will be brought to the "Terms and Conditions" screen.

Results:

Test name: Create account next screen

Test number: 14

Element being tested: Next button

Preconditions: The user is on the "Create account" view and everything is okay and the password and confirm password texts are not the same.

Test procedure: The user will tap the "Next" button

Expected outcome: A pop up will appear stating that the passwords do not match and he must type it again.

Results:

Test name: Create account next screen (item not filled out)

Test number: 15

Element being tested: Next button

Preconditions: The user is on the "Create account" view and some of the fields are not filled out. Perform this test for every combination.

Test procedure: The user will tap the "Next" button

Expected outcome: A popup appears stating that the page is not completely filled out.

Results:

### 3.2.3 Terms and Conditions view

Test name: I agree checkmark

Test number: 16

Element being tested: Checkmark

Preconditions: The user is on the "Terms and Conditions" view.

Test procedure: The user will tap "I agree to these terms and conditions."

Expected outcome: A checkmark will appear next to "I agree to these terms and conditions."

Results:

Test name: Send data to Google checkmark

Test number: 17

Element being tested: Checkmark

Preconditions: The user is on the "Terms and Conditions" view.

Test procedure: The user will tap "Send data to Gooey to improve the app."

Expected outcome: A checkmark will appear next to "Send data to Gooey to improve the app."

Results:

Test name: Finish

Test number: 18

Element being tested: Finish button

Preconditions: The user is on the "Terms and Conditions" view.

Test procedure: The user will tap the "Finish" button

Expected outcome: The user will be taken to the "Home" view.

Results:

Test name: Finish

Test number: 19

Element being tested: Finish button

Preconditions: The user is on the "Terms and Conditions" view and there is no checkmark on "I agree to these terms and conditions" or "Send data to Gooey to improve the app."

Test procedure: The user will tap the "finish" button.

Expected outcome: A pop up will appear stating that the terms and conditions have not been agreed and you cannot continue.

Results:

Test name: Finish

Test number: 20

Element being tested: Finish button

Preconditions: The user is on the "Terms and Conditions" view and there is a checkmark on "I agree to these terms and conditions", but not for "Send data to Gooey to improve the app."

Test procedure: The user will tap the "Finish" button.

Expected outcome: The user will be taken to the "Home" screen.

Results:

Test name:

Test number: 21

Element being tested: Finish button

Preconditions: The user is on the "Terms and Conditions" view and there is not a checkmark on "I agree to these terms and conditions", but there is a checkmark for "Send data to Gooey to improve the app."

Test procedure: The user will tap the "Finish" button

Expected outcome: A pop up will appear stating that the terms and conditions have not been agreed and you cannot continue.

Results:

### 3.2.4 Home view

Test name: Settings icon

Test number: 22

Element being tested: Settings icon

Preconditions: The user is on the "Home" view.

Test procedure: The user will tap the settings icon.

Expected outcome: The user will be taken to the "Settings” view.

Results:

Test name: I want food

Test number: 23

Element being tested: I want food button

Preconditions: The user is on the "Home" view.

Test procedure: The user will tap the "I want food" button.

Expected outcome: The user will be taken to the "New Order" view.

Results:

Test name: I'm getting food

Test number: 24

Element being tested: I'm getting food button

Preconditions: The user is on the "Home" view.

Test procedure: The user will tap the "I'm getting food" button.

Expected outcome: The user will be taken to the "Select restaurant" view.

Results:

Test name: View my orders

Test number: 25

Element being tested: View my orders button

Preconditions: The user is on the "Home" view.

Test procedure: The user will tap the "View my orders" button

Expected outcome: The user will be taken to the "My Orders" view.

Results:

### 3.2.5 I want food view

Test name: Choose restaurant

Test number: 26

Element being tested: Select restaurant arrow

Preconditions: The user is on the "I want food" view

Test procedure: The user will tap the "Select restaurant" arrow

Expected outcome: The "Select restaurant" view will appear

Results:

Test name: Add new food item

Test number: 27

Element being tested: Plus button

Preconditions: The user is on the "I want food" view

Test procedure: The user will tap the plus button in the food section

Expected outcome: The "New food item" view will appear

Results:

Test name: Select driver

Test number: 28

Element being tested: Select driver field

Preconditions: The user is on the "I want food" view. The user has completed the "Select restaurant" field and has added at least one food item

Test procedure: The user will tap the "Select driver" field

Expected outcome: The "Available drivers" view appears

Results:

Test name: Select location

Test number: 29

Element being tested: Select location field

Preconditions: The user is on the "I want food" view. The user has completed the "Select restaurant" field and has added at least one food item and has selected a driver

Test procedure: The user will tap the "Select location" field

Expected outcome: The "Select delivery location" view appears

Results:

Test name: Select expiration

Test number: 30

Element being tested: Expiration time field

Preconditions: The user is on the "I want food" view. The user has filled out all other fields

Test procedure: The user will tap the "Expiration time" field

Expected outcome: The "Expires in" view appears

Results:

Test name: Send request (filled out)

Test number: 31

Element being tested: Send request button

Preconditions: All fields on the "I want food" view are filled out

Test procedure: The user will tap the "Send request" button

Expected outcome: A popup will appear confirming that the request was sent, and the user is taken to the request summary view.

Results:

Test name: Send request (nothing filled out)

Test number: 32

Element being tested: Send request button

Preconditions: Nothing is filled out on the "I want food" view

Test procedure: The user will tap the "Send request" button

Expected outcome: The request is not sent and the user is informed of fields that need to be filled out

Results:

Test name: Send request (partially filled out)

Test number: 33

Element being tested: Send request button

Preconditions: Some items are filled out on the "I want food" view. Perform test with all combinations.

Test procedure: The user will tap the "Send request" button

Expected outcome: The request is not sent and the user is informed of fields that need to be filled out

Results:

Test name: Add new food item, part ii

Test number: 34

Element being tested: Food items listed in "I want food" view of consumer

Preconditions: User is on the "I want food" view. The user has completed the "Add new food item" test already (pressed the + button and gone to the "add new food item" view)

Test procedure: User types in a new food item. Tap back when finished.

Expected outcome: Returns to the "I want food" view, now listing the new food item under the "Food" heading of the view

Results:

Test name: Choose destination

Test number: 35

Element being tested: Delivery location field

Preconditions: The user is on the "I want food" view and has taped the select location button.

Test procedure: The user will tap the field and enter his location using his keyboard.

Expected outcome: Known locations should be auto-completed as the user types.

Results:

### 3.2.6 Select Restaurant view

Test name: Top left arrow button

Test number: 36

Element being tested: Top left arrow button

Preconditions: The user is on the "Select restaurant" view and no restaurant is selected.

Test procedure: The user will tap the top left arrow button.

Expected outcome: The user will be brought to the "New order" screen with no change to the select restaurant section.

Results:

Test name: Select a restaurant

Test number: 37

Element being tested: Checkmark

Preconditions: The user is on the "Select restaurant" view.

Test procedure: The user will tap a restaurant.

Expected outcome: A checkmark will appear next to the restaurant selected.

Results:

Test name: Top left arrow button

Test number: 38

Element being tested: Top left arrow button

Preconditions: The user is on the "Select restaurant" view.

Test procedure: The user will tap the top left arrow button.

Expected outcome: The user will be brought to the "New order" screen and the restaurant with the checkmark next to it will be in the restaurant section.

Results:

### 3.2.7 Food Item view

Test name: Top left arrow button

Test number: 39

Element being tested: Top left arrow button

Preconditions: The user is on the "food item" view and nothing has been put into the food item field.

Test procedure: The user will tap the top left arrow button.

Expected outcome: The user will be brought to the "New order" screen and there will be no changes to the food section.

Results:

Test name: Top left arrow button

Test number: 40

Element being tested: Top left arrow button

Preconditions: The user is on the "food item" view and there is information in the food item field.

Test procedure: The user will tap the top left arrow button.

Expected outcome: The user will be brought to the "New order" screen and the food item he typed will be in the food section.

Results:

### 3.2.8 Available Drivers view

Test name: Select any driver

Test number: 41

Element being tested: Any driver list item

Preconditions: The user is on the "Available drivers" view. An element other than "any driver" is selected

Test procedure: The user will tap the "any driver" list item

Expected outcome: The previously selected field will be deselected, and the "any driver" field will be selected.

Results:

Test name: Select any driver (already selected)

Test number: 42

Element being tested: Any driver list item

Preconditions: The user is on the "Available drivers" view. The "any driver" list item is already selected.

Test procedure: The user will tap the "any driver" list item.

Expected outcome: The selection does not change.

Results:

Test name: Available drivers

Test number: 43

Element being tested: Specific drivers' list items

Preconditions: The user is on the "Available drivers" view

Test procedure: The user will tap a specific driver's list item.

Expected outcome: If another list item is selected, it is deselected. The tapped-on list item is selected.

Results:

Test name: Available drivers top left arrow button

Test number: 44

Element being tested: Top left arrow button on the "Available drivers" view

Preconditions: The user is on the "Available drivers" view and a driver is selected.

Test procedure: The user will tap the top left arrow button.

Expected outcome: The user will be taken back to the "New order" view and the driver selected will appear in the delivered by field.

Results:

Test name: Available drivers top left arrow button

Test number: 45

Element being tested: Top left arrow button on the "Available drivers" view

Preconditions: The user is on the "Available drivers" view and a driver is not selected.

Test procedure: The user will tap the top left arrow button.

Expected outcome: The user will be taken back to the "New order" view and nothing will have changed in the delivered by field.

Results:

### 3.2.9 Delivery Location view

Test name: Custom delivery location.

Test number: 46

Element being tested: Custom delivery location text field.

Preconditions: The user is on the "Delivery location" view.

Test procedure: The user will tap the "enter custom delivery location" text field.

Expected outcome: A keyboard will pop up and the user will type the delivery location he would like.

Results:

Test name: Select delivery location.

Test number: 47

Element being tested: Delivery location checkmark

Preconditions: The user is on the "Delivery location" view.

Test procedure: The user will tap a previously selected location that will appear underneath the "enter custom delivery location" text field.

Expected outcome: A checkmark will appear next to that delivery location.

Results:

Test name: Top left arrow button

Test number: 48

Element being tested: Top left arrow button

Preconditions: The user is on the "Delivery location" view and a location is selected.

Test procedure: The user will tap the top left arrow button.

Expected outcome: The user will be taken back to the "New order" screen and the location will be the location that was selected.

Results:

Test name: Top left arrow button

Test number: 49

Element being tested: Top left arrow button

Preconditions: The user is on the "Delivery location" view and a location is not selected.

Test procedure: The user will tap the top left arrow button.

Expected outcome: The user will be taken back to the "New order" screen and the location will be the location will not have changed.

Results:

### 3.2.10 Expires In view

Test name: Make expiration time 30 minutes (already selected)

Test number: 50

Element being tested: 30 minutes list item

Preconditions: User is on the "Expires in" view. The "30 minutes" element is selected.

Test procedure: The user will tap the "30 minutes" list item.

Expected outcome: Nothing changes.

Results:

Test name: Make expiration time 30 minutes (not already selected)

Test number: 51

Element being tested: 30 minutes list item

Preconditions: User is on the "Expires in" view. An element other than "30 minutes" is selected

Test procedure: The user will tap the "30 minutes" list item.

Expected outcome: The previously selected item is deselected and the "30 minutes" item is selected.

Results:

Test name: Make expiration time 1 hour (already selected)

Test number: 52

Element being tested: 1 hour list item

Preconditions: The user is on the "Expires in" view. The "1 hour" is selected.

Test procedure: The user will tap the "1 hour" list item.

Expected outcome: Nothing changes.

Results:

Test name: Make expiration time 1 hour (not already selected)

Test number: 53

Element being tested: 1 hour list item

Preconditions: The user is on the "Expires in" view. An element other than the "1 hour" list item is selected.

Test procedure: The user will tap the "1 hour" list item.

Expected outcome: The previously selected item is deselected and the "1 hour" item is selected.

Results:

Test name: Make expiration time 2 hours (already selected)

Test number: 54

Element being tested: 2 hours list item

Preconditions: The user is on the "Expires in" view. The "2 hours" list item is selected.

Test procedure: The user will tap the "2 hours" list item.

Expected outcome: Nothing changes.

Results:

Test name: Make expiration time 2 hours (not already selected)

Test number: 55

Element being tested: 2 hours list item

Preconditions: The user is on the "Expires in" view. An element other than "2 hours" is selected.

Test procedure: The user will tap the "2 hours" list item.

Expected outcome: The previously selected item is deselected and the "2 hours" item is selected.

Results:

Test name: Expiration time countdown 30 minutes

Test number: 56

Element being tested: Expires in time listing

Preconditions: The user is on the "Expires in" view. Selected item is the 30 minutes option for the order

Test procedure: Wait 1 minute. Press back. Return to the "I want food" view.

Expected outcome: The "Expires in" item should now approximately say "29:00" or equivalent

Results:

Test name: Expiration time countdown 15 minutes

Test number: 57

Element being tested: Expires in time listing

Preconditions: The user is on the "Expires in" view. Selected item is the 15 minutes option for the order

Test procedure: Wait 1 minute. Press back. Return to the "I want food" view.

Expected outcome: The "Expires in" item should now approximately say "14:00" or equivalent

Results:

Test name: Expiration time countdown 1 hour

Test number: 58

Element being tested: Expires in time listing

Preconditions: The user is on the "Expires in" view. Selected item is the 1 hour option for the order

Test procedure: Wait 1 minute. Press back. Return to the "I want food" view.

Expected outcome: The "Expires in" item should now approximately say "59:00" or equivalent

Results:

Test name: Expiration time countdown 2 hour

Test number: 59

Element being tested: Expires in time listing

Preconditions: The user is on the "Expires in" view. Selected item is the 2 hours option for the order.

Test procedure: Wait 2 minutes. Press back. Return to the "I want food" view.

Expected outcome: The "Expires in" item should now approximately say "1:58:00" or equivalent

Results:

Test name: Return to "I want food" view from "Expires in" view

Test number: 60

Element being tested: Top left arrow button on "Expires in" view

Preconditions: The user is on the "Expires in" view

Test procedure: The user will tap the top left arrow button.

Expected outcome: The user is taken back to the "I want food" view. The selected expiration time is displayed in the "expiration time" field.

Results:

Test name: Make expiration time 15 minutes (already selected)

Test number: 61

Element being tested: 15 minutes list item

Preconditions: On the "Expires in" view. The "15 minutes" element is selected.

Test procedure: The user will tap the "15 minutes" list item.

Expected outcome: Nothing changes.

Results:

Test name: Make expiration time 15 minutes (not already selected)

Test number: 62

Element being tested: 15 minutes list item

Preconditions: User is on the "Expires in" view. An element other than "15 minutes" is selected

Test procedure: The user will tap the "15 minutes" list item.

Expected outcome: The previously selected item is deselected and the "15 minutes" item is selected.

Results:

### 3.2.11 Current Order view

Test name: Pay

Test number: 63

Element being tested: Pay button

Preconditions: The user is on the "Current order" view.

Test procedure: The user will tap the "Pay" button.

Expected outcome: The "Pay" button will change to say "order completed."

Results:

Test name: Top left arrow button

Test number: 64

Element being tested: Top left arrow button

Preconditions: The user is on the "Current order" view.

Test procedure: The user will tap the top left arrow button.

Expected outcome: The user will be brought to the "My Orders" screen.

Results:

Test name: Chat

Test number: 65

Element being tested: Chat icon

Preconditions: The user is on the "Current order" view.

Test procedure: The user will tap the chat icon.

Expected outcome: The driver's phone number will appear and the user can text with the driver.

Results:

### 3.2.12 My Orders view

Test name: Select my sent requests

Test number: 66

Element being tested: My sent requests selection

Preconditions: The user is on the "My Orders" view.

Test procedure: The user will tap one of the options under "My sent requests."

Expected outcome: The user will be brought to the "current order" screen.

Results:

Test name: Select requests I'm picking up

Test number: 67

Element being tested: Requests I'm picking up selection

Preconditions: The user is on the "My Orders" view.

Test procedure: The user will tap one of the options under "Requests I'm picking up."

Expected outcome: The user will be brought to the "I paid for food" screen.

Results:

Test name: Top left arrow button

Test number: 68

Element being tested: Top left arrow button

Preconditions: The user is on the "My Orders" view.

Test procedure: The user will tap the top left arrow button.

Expected outcome: The user will be brought to the "Home" screen.

Results:

### 3.2.13 Restaurants Tab view

Test name: Select a restaurant

Test number: 69

Element being tested: Restaurant checklist item

Preconditions: User is on the "Restaurants" tab of the "I'm picking up food" view. The restaurant checklist is populated with restaurants

Test procedure: Tap a restaurant list item

Expected outcome: The list item becomes checked

Results:

Test name: Pickup food expiration decrement

Test number: 70

Element being tested: Time next to "Expires in" component of "I'm picking up food" view

Preconditions: User is on the "Restaurants" tab of the "I'm picking up food" view. Available to pick up food" option is selected

Test procedure: Wait 1 minute. Press top left arrow button. Press "I'm picking up food" button

Expected outcome: The time listed next to "Expires in" has dropped by approximately one minute.

Results:

Test name: Pickup food availability remains

Test number: 71

Element being tested: Available to pick up food option in "I'm picking up food" view

Preconditions: User is on the "Restaurants" tab of the "I'm picking up food" view. "Available to pick up food" is enabled

Test procedure: Press back. Press "I'm picking up food"

Expected outcome: Available to pick up food is (still) enabled.

Results:

Test name: Pickup food availability constant

Test number: 72

Element being tested: Available to pick up food option in "I'm picking up food" view (negative)

Preconditions: User is on the "Restaurants" tab of the "I'm picking up food" view. "Available to pick up food" is not enabled

Test procedure: Press back. Press "I'm picking up food"

Expected outcome: Available to pick up food option is (still) not enabled

Results:

Test name: Driver availability switch on

Test number: 73

Element being tested: Available to pick up food switch in driver view

Preconditions: User is on the "Restaurants" tab of the "I'm picking up food" view. "Available to pick up food" switch is currently off

Test procedure: Tap the switch

Expected outcome: Available to pick up food switch turns on, and the restaurants are grayed out

Results:

Test name: Driver availability switch off

Test number: 74

Element being tested: Available to pick up food switch in driver view

Preconditions: User is on the "Restaurants" tab of the "I'm picking up food" view. "Available to pick up food" switch is currently on

Test procedure: Tap the switch

Expected outcome: Available to pick up food switch turns off, "Expires in" time resets, and the restaurants are available to select (i.e. no longer grayed out).

Results:

Test name: Pickup food gray out

Test number: 75

Element being tested: Restaurant selection checkboxes in "I'm picking up food" section

Preconditions: User is on the "Restaurants" tab of the "I'm picking up food" view. "Available to pick up food" option is selected

Test procedure: Attempt to check or uncheck a restaurant selection

Expected outcome: Nothing happens

Results:

Test name: View available orders

Test number: 76

Element being tested: Orders nav button

Preconditions: User is on the "Restaurants" tab of the "I'm picking up food" view.

Test procedure: Tap the "orders" button

Expected outcome: User sees the "Available Orders" view.

Results:

Test name: Restaurant selection

Test number: 77

Element being tested: tap restaurant

Preconditions: The user is a customer and is on the "Restaurants" tab of the "I'm picking up food" view

Test procedure: The user will tap a restaurant

Expected outcome: A checkmark will appear next to the restaurant

Results:

Test name: Restaurant selection items checkmarked

Test number: 78

Element being tested: Top left arrow button

Preconditions: The user is on the "Restaurants" tab of the "I'm picking up food" view and a restaurant is selected

Test procedure: The user will tap the top left arrow button in the upper left corner

Expected outcome: The "My Orders" view will appear and the restaurant name that is checked will appear under "Restaurant"

Results:

Test name: Restaurant selection not checked

Test number: 79

Element being tested: Top left arrow button

Preconditions: The user is on the "Restaurants" tab of the "I'm picking up food" view and a restaurant is not selected

Test procedure: The user will tap the top left arrow button in the upper left corner

Expected outcome: The "My Orders" view will appear and select restaurant will be under "Restaurant"

Results:

### 3.2.14 Orders Tab view

Test name: View orders driver side

Test number: 80

Element being tested: Tap customer

Preconditions: User is on the "Orders" tab of the "I'm picking up food" view

Test procedure: The user will tap one of the orders in the list

Expected outcome: The "I'm getting food" view will appear

Results:

Test name: View orders driver side

Test number: 81

Element being tested: Top left arrow button

Preconditions: User is on the "Orders" tab of the "I'm picking up food" view

Test procedure: The user will tap the top left arrow button in the upper left corner

Expected outcome: The "Available Orders" view will appear

Results:

Test name: View an available order

Test number: 82

Element being tested: List item

Preconditions: User is on the "Orders" tab of the "I'm picking up food" view

Test procedure: Tap a list item.

Expected outcome: User is taken to the "order pickup" view for that item.

Results:

Test name: Reject an available order

Test number: 83

Element being tested: Requests for anyone order list from "orders" tab of "I'm picking up food" section

Preconditions: User is on the "Orders" tab of the "I'm picking up food" view

Test procedure: The user swipes left on the order

Expected outcome: The order is removed from the user's list of available orders.

Results:

Test name: Reject a pending request

Test number: 84

Element being tested: Requests for me order list from "Orders" tab of "I'm picking up food" section

Preconditions: User is on the "Orders" tab of the "I'm picking up food" view

Test procedure: The user swipes left on the order to be removed

Expected outcome: The order is removed from the list of pending requests.

Results:

### 3.2.15 Order Pickup view

Test name: Top left arrow button

Test number: 85

Element being tested: Top left arrow button

Preconditions: User is on the "Order pickup" view.

Test procedure: The user taps on the top left arrow button.

Expected outcome: The user is taken to the "Available Orders" screen.

Results:

Test name: I'll get that

Test number: 86

Element being tested: I'll get that button

Preconditions: User is on the "Order pickup" view.

Test procedure: The user taps on the "I’ll get that" button.

Expected outcome: The user is taken to the "I paid for food" screen.

Results:

### 3.2.16 Paid for Food view

Test name: Top left arrow button

Test number: 87

Element being tested: Top left arrow button

Preconditions: User is on the "I paid for food" view.

Test procedure: The user will tap the top left arrow button

Expected outcome: The user will be brought to the "My Orders" screen.

Results:

Test name: I paid for food

Test number: 88

Element being tested: I paid for food button

Preconditions: User is on the "I paid for food" view.

Test procedure: The user will tap the "I paid for food" button

Expected outcome: A pop up will appear with how much you spent on the item.

Results:

Test name: Chat with customer

Test number: 89

Element being tested: Chat icon

Preconditions: User is on the "I paid for food" view.

Test procedure: The user will tap the chat icon.

Expected outcome: The user will be given the customer's phone number and they will be able to text each other.

Results:

### 3.2.17 Order Confirmation view

Test name: Top left arrow button

Test number: 90

Element being tested: Top left arrow button

Preconditions: User is on the "Order Confirmation" view.

Test procedure: The user will tap the top left arrow button.

Expected outcome: The user will be brought to the "My Orders" screen.

Results:

Test name: I'm at the delivery location

Test number: 91

Element being tested: I'm at the delivery location button

Preconditions: User is on the "Order Confirmation" view.

Test procedure: The user will tap the "I'm at the delivery location" button.

Expected outcome: Popup will appear to confirm the driver is at the delivery location.

Results:

Test name: Chat with customer

Test number: 92

Element being tested: Chat icon

Preconditions: User is on the "Order Confirmation" view.

Test procedure: The user will tap the chat icon.

Expected outcome: The user will be given the customer's phone number and they will be able to text each other.

Results:

### 3.2.18 Settings view

Test name: Change password

Test number: 93

Element being tested: Change password button

Preconditions: The user is on the "Settings" view

Test procedure: The user will tap "Change password" button

Expected outcome: A pop up will appear allowing the user to change the password

Results:

Test name: Change password (password not changed)

Test number: 94

Element being tested: Ok button

Preconditions: The user has taped on "Change password" in "Settings" view and he has not typed in a new password

Test procedure: The user will tap the ok button

Expected outcome: Text will appear saying that the password has not been changed, type in a different password

Results:

Test name: Cancel "Change password"

Test number: 95

Element being tested: Cancel button

Preconditions: The user has taped on "Change password" in "Settings" view

Test procedure: The user will tap the cancel button

Expected outcome: The user will be brought back to the "Settings" view as if nothing happened

Results:

Test name: Change password

Test number: 96

Element being tested: Ok button

Preconditions: The user has taped on "Change password" in "Settings" view and has typed in a new password

Test procedure: The user will tap the ok button

Expected outcome: The user will be brought back to the "Settings" view and text will appear saying that the password was changed.

Results:

Test name: Signing out

Test number: 97

Element being tested: Sign out button

Preconditions: The user is on the "Settings" view and is signed in.

Test procedure: The user taps on the "Sign out" button.

Expected outcome: The user will be signed in and taken back to the "login" view.

Results:

Test name: Add user picture

Test number: 98

Element being tested: User "Settings" view picture display

Preconditions: User is on the "Settings" view.

Test procedure: Tap the picture icon. Select a picture from the device's pictures menu or camera. Press "ok".

Expected outcome: New picture is displayed as the user's picture in the "Settings" view

Results:

Test name: Delete account button

Test number: 99

Element being tested: Delete account button

Preconditions: User is on the "Settings" view.

Test procedure: User taps the "delete account" button.

Expected outcome: Delete account confirmation modal pops up.

Results:

Test name: Confirm deleting account (wrong password)

Test number: 100

Element being tested: Delete account confirmation button

Preconditions: User is on the "Settings" view and has tapped "delete account", with the confirmation modal now visible.

Test procedure: User types in an incorrect password and taps "delete account".

Expected outcome: A pop up notifies the user that the password has been incorrectly typed. The account is not deleted.

Results:

Test name: Confirm deleting account (correct password)

Test number: 101

Element being tested: Delete account confirmation button

Preconditions: User is on the "Settings" view and has tapped "Delete account", with the confirmation modal now visible.

Test procedure: User types in his correct password and taps "Delete account".

Expected outcome: The user is signed out and taken to the "login" view with no pre-filled username or password. His account is deleted.

Results:

Test name: Cancel deleting account confirmation

Test number: 102

Element being tested: Cancel deleting account confirmation button

Preconditions: User is on the "Settings" view and has tapped "Delete account", with the confirmation modal now visible.

Test procedure: User taps the "cancel" button.

Expected outcome: The "Delete account" modal disappears.

Results:

Test name: Email focus in on "Settings view"

Test number: 103

Element being tested: Email field

Preconditions: User is on the "Settings" view.

Test procedure: Tap email field

Expected outcome: Email field is "focused in"

Results:

Test name: Location focus in on "Settings view"

Test number: 104

Element being tested: Location field

Preconditions: User is on the "Settings" view.

Test procedure: Tap location field

Expected outcome: Location field is "focused in"

Results:

Test name: Vehicle focus in on "Settings view"

Test number: 105

Element being tested: Vehicle model field

Preconditions: User is on the "Settings" view.

Test procedure: Tap vehicle field

Expected outcome: Vehicle field is "focused in"

Results:

Test name: Save email

Test number: 106

Element being tested: Done button

Preconditions: User is on the "Settings" view. Email field is filled in

Test procedure: Tap "Done" button on keyboard

Expected outcome: Email is saved

Results:

Test name: Save location

Test number: 107

Element being tested: Done button

Preconditions: User is on the "Settings" view. Location field is filled in

Test procedure: Tap "Done" button on keyboard

Expected outcome: Location is saved

Results:

Test name: Save vehicle model

Test number: 108

Element being tested: Done button

Preconditions: User is on the "Settings" view. Vehicle model information is filled in

Test procedure: Tap "Done" button

Expected outcome: Vehicle model information is saved

Results:

# 4 Ethics and Intellectual Property

The following contains the project’s ethical issues and intellectual property claims.

## 4.1 Intellectual Property Rights

Gooey is participating in the Grove City College Entrepreneurship department’s VentureLab program. This releases the team from the College’s original intellectual property ownership on any work done on this project while on campus or using College resources [28]. The team completely owns all intellectual property rights for all code, designs, and ideas.

## 4.2 Plans for After College

Gooey may not remain the same after graduation from Grove City College. Some team members have expressed intentions of leaving the team after graduation, due to full-time jobs. Zachary Nafziger has no plans to continue with future development of the project outside of Grove City College and will no longer be involved with the design or implementation of future versions of the application. Seth Loew has a full-time job but is interested in pursuing the project and being part of establishing a business with Gooey. Aaron Rosenberger intends to pursue working on the project and establishing a business. Michael Kytka plans to have a full-time job but wants to continue working on Gooey as he is able. Sam Kibler plans to do full-time work as well, but would like to continue the project. The members agree that they must implement additional features before officially establishing a business in order to remain viable.

## 4.3 Trade Secrets

The project makes use of customized sorting and matching algorithms, as well as data analysis algorithms. The application uses these trade secrets in management and creation of lists of drivers, customers, restaurants, and delivery locations.

For example, the service may use an algorithm similar to SHARK search, which manages and queries a network of nodes arranged by a hierarchy of keywords [29] [30]. Additionally, the relatively new industry of data mining is beginning to consider customer preferences, “rather than assumed general characteristics” [31].

## 4.4 Trademarks

The Gooey cookie logo and all variations are trademarked. Variations include the lone image, the word “Gooey” in the font Aller, and the image and word together.

## 4.5 Copyrights

The entire project and all designs created by team members are copyrighted by Gooey © 2015.

## 4.6 Ethical Issues

As with most socially-powered applications, there is the potential for users to use the system with nefarious intentions. Examples of such abuse include predators looking for potential victims and finding them through the system, and users selling illegal goods and services, such as illicit drugs and prostitution [4]. Software like ours cannot easily avoid this problem, but our application collects data about how users are interacting with the system so that proper legal action can be taken afterwards, with presentable evidence. This data collection is potentially an ethical issue, especially if a malicious user was to gain access to private information, such as names, email addresses, phone numbers, location data, password hashes, and application usage information. In order to protect users of the application and the members of Gooey, we will have several measures in place.

First of all, users must agree to a license agreement in order to sign up for an account with Foodini. This agreement allows us to store necessary information about them, such as an email address and phone number. In the registration process, users may choose to send usage information to Gooey, in which case they would allow us to gather data such as screen taps, buttons, form data, screen views, and crash statistics. Second, all information collected through this application will be encrypted in a reasonable manner to prevent theft of the data.

Some of the data collected through this application, including location data, restaurant data, and food items is very valuable to restaurants and corporations. In the future, we plan to sell anonymized data to interested third parties who we determine do not have a malicious intent. When registering for an account, the users will be made aware of this through the license agreement.

## 4.7 Societal Implications

If this application gains a large enough database, it could significantly affect both restaurants that do and do not offer delivery. For restaurants that do not offer delivery, the service would likely increase business, since customers would no longer need to travel to the restaurant in person to buy food, increasing the likelihood of impulsive purchases. However, the application would likely negatively affect restaurants that offer delivery. In many cases, delivery can give restaurants an edge above others in the area, since customers do not need to physically go to the restaurant location. Essentially, Foodini offers delivery for every restaurant, and challenges existing in-house delivery systems.

We are aware of these societal implications, and believe that the positive outcomes of offering delivery outweigh the negative outcomes. Additionally, the service can encourage use of restaurants’ in-house delivery, where it exists, above the application’s delivery.

## 4.8 Legal and Regulatory Issues

The Gooey team spent a significant amount of time researching patents to ensure that our application did not infringe upon them. We found a number of patents, and determined that our application will be unique, and will not infringe upon any current patents [8] [9] [32] [33] [34] [35] [36] [37] [38].

Our application will store user information, including name, phone number, location, email address, and a password hash, and user interactions with the system. This data will be reasonably secured and encrypted to protect user privacy, and no payment information will be stored on the application, simplifying the application and preventing data compromises [39]. We will use HTTPS connections wherever possible to ensure secure communication. Before releasing HTTPS-containing software to the public, we will obtain all necessary federal munitions exemptions for internet encryption protocols in order to ensure safe exportation of our app to other countries, per federal law and Apple Store policy.

# 5 Summary

Foodini is a system that connects users who want food with users who are willing to deliver food for a small profit. Our primary target audience is college students in an urban environment who have iPhones with iOS 8 or newer. We carefully constructed and revised the application interface resulting in a simple and clear user experience. The robustness is complemented by security and privacy, which is paramount. Data is encrypted and users can rest easy knowing their information is protected. Happy users are returning users, and we believe the application will make lives easier for students across the country.

# 6 Bibliography

|  |  |
| --- | --- |
| [1] | Apple Inc., "Designing for iOS," 2015. [Online]. Available: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/.  Apple’s official guide for application design on iOS. This resource was carefully considered as we designed our application to fit within Apple’s standards. |
| [2] | Apple Inc., "Maps for Developers," 2015. [Online]. Available: https://developer.apple.com/maps/.  A portal for Apple’s official documentation for using Maps in development. This was important to consider as we researched the capabilities of Apple Maps, and will be crucial in developing the application. |
| [3] | Apple Inc., "Design Principles," 2015. [Online]. Available: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Principles.html.  Apple’s official user interface design guidebook. This was considered during the application’s design in order to comply with Apple’s iOS paradigm. |
| [4] | C. Lewis and J. Rieman, Task-Centered User Interface Design, Boulder CO: Clayton Lewis and John Rieman, 1994.  Lewis and Rieman’s textbook about Task-Centered User Interface Design, and the textbook for our course. Many of the design concepts discussed in this resource were considered in the design of our application. |
| [5] | A. Aloisi, "Commoditized Workers: The Rising of On-Demand Work, A Case Study Research on a Set of Online Platforms and Apps," Bocconi University, Milano Italy, 2015.  This publication provided insight on the viable, growing market of people working as private contractors. In the future, Foodini will have to declare a system similar to those described in this publication. |
| [6] | Uber Technologies Inc., "Uber," 2015. [Online]. Available: https://www.uber.com/.  A mobile application that connects drivers and users who need rides. This was one of the critical influences in the initial application idea. |
| [7] | GrubHub Inc., "GrubHub Food Delivery & Takeout," 2014. [Online]. Available: https://www.grubhub.com/.  A mobile and web application that finds nearby restaurants who deliver. It was researched as a potential competitor. |
| [8] | C. J. Webb, "Method of and System for Group Meal Ordering via Mobile Devices". USA Patent 20120036028, 4 Aug 2012.  A food ordering method for mobile devices that includes an organizer selecting a restaurant and creating a single group meal order. This is very similar to what Foodini will do, but it includes menus of the restaurants, thereby limiting the restaurant and food selection. |
| [9] | D. Scifo, "Online Food Ordering System and Method". USA Patent 20090204492, 13 Aug 2009.  This patent involves restaurants registering for an online food ordering system and entering their menus into the online system, with a registration fee. Foodini requires no effort on the part of the restaurants, and there is no registration fee. |
| [10] | Domino's Pizza Inc., "Domino's Pizza," 2015. [Online]. Available: https://www.dominos.com/en/about-pizza/online-delivery/.  Domino’s Pizza online ordering gave us some great ideas about how to make creating an order easy for the consumer. |
| [11] | Uber Technologies Inc., "Uber Eats," 2015. [Online]. Available: http://ubereats.com/eats/.  Uber Eats is a small experiment by Uber in food delivery. It is more of a food ordering app, but we took it into consideration when designing the application. |
| [12] | Starbucks Corporation, "Starbucks," 2015. [Online]. Available: http://www.starbucks.com/coffeehouse/mobile-apps.  The Starbuck’s system of payment was an inspiration for how the system was going to handle payment before the payment feature was removed from the application. |
| [13] | Seamless North America LLC, "Seamless," 2015. [Online]. Available: https://itunes.apple.com/us/app/seamless-free-food-delivery/id381840917?mt=8.  Seamless is an app by GrubHub that allows “seamless” food ordering. We analyzed their system along with other similar apps to see what we should and (more importantly) should not do. |
| [14] | Yik Yak LLC, "Yik Yak - Your Local Feed," 2015.  An anonymous, digital, public message board application that shows posts from within a five mile radius. This was a strong contributor in the idea to create a highly localized mobile application. |
| [15] | "DoorDash," DoorDash Food Delivery, 2015. [Online]. Available: https://www.doordash.com.  Doordash provides food with a promise of delivery within 60 minutes. DoorDash serves over 100 cities and areas around the U.S., employing some partnership relationships with restaurants and some more independent arrangements (i.e. to-go pick up at the window). Its menus are defined, and it hires its drivers. However, after starting in 2013, the company now is valued at almost $1B, with $35M backing (including from YCombinator). This is one company to look out for as competition if Foodini becomes a business. |
| [16] | "Caviar," Caviar, 2015. [Online]. Available: https://www.trycaviar.com.  Now owned by Square, Caviar operates in roughly 18 large U.S. cities, hires and pays drivers, and establishes partnerships with high-end merchants for food delivery in a similar but more structured manner than DoorDash. Foodini plans to be much more independent than this (from both restaurants and drivers). Also, Foodini’s initial target market is not as high-end as Caviar’s. |
| [17] | "Postmates - On-Demand, 24/7," Postmates, 2015. [Online]. Available: https://postmates.com.  Postmates provides food and other delivery services in over a hundred cities around the U.S. from participating restaurants and merchants. The business hires ‘postmates’ (much like Uber does) who are responsible for the timely delivery of goods to customers near their location. This is less crowd-sourced, as drivers are hired by the company. Also, Postmates merchants partner with the business to participate. Foodini will follow a more independent approach to delivery. |
| [18] | V. Milay, "The Bits and Bytes of Food: Study of Emerging Internet-based Food Businesses," *DSpace@MIT (Massachusetts Institute of Technology),* 2014.  This MIT paper describes the effects to the market, success, and challenges of internet-based food delivery businesses in the past several years, with a special eye to startups. It gave us a few more names of applications to be aware of, such as Postmates, as well as profitability considerations for various business models. The information here could prove invaluable if we implement this as a business. |
| [19] | Apple Inc., "Apple Pay," 2015. [Online]. Available: http://www.apple.com/apple-pay/.  Apple’s payment system that can be used by itself or within applications. This was researched before it was decided to eliminate payment native to our system. |
| [20] | Stripe, "Stripe," 2015. [Online]. Available: https://stripe.com.  A suite of APIs that can be used to integrate payment within applications. This was researched before it was decided to eliminate payment native to our system. |
| [21] | Paypal Holdings Inc., "PayPal," 2015. [Online]. Available: https://www.paypal.com/home.  An online application for paying individuals and businesses. This was researched before it was decided to eliminate payment native to our system. |
| [22] | J. Selvadurai, "Legal And Ethical Responsibilities In Mobile Payment Privacy," *International Journal of Scientific & Technology Research,* vol. 2, no. 6, June 2013.  This publication provided insight on how to handle payment within the app, and was still useful, despite the decision to not handle payment natively. |
| [23] | K. Siau and Z. Shen, "Building Customer Trust in Mobile Commerce," *Communications of the ACM,* vol. 46, no. 4, 2003.  Document detailing how to design systems that appeal to the user’s trust. This was consulted when determining how much information is shared with the application and with other users. |
| [24] | P. Tarasewich, "Designing Mobile Commerce Applications," *Communications of the ACM,* vol. 46, no. 12, pp. 57-60, 2003.  This paper discusses the design of mobile commerce applications and some of the challenges that arise with it. |
| [25] | V. Venkatesh, V. Ramesh and A. P. Massey, "Understanding Usability in Mobile Commerce," *Communications of the ACM,* vol. 46, no. 12, pp. 53-56, 2003.  This paper discusses the design of mobile commerce applications and how usability plays a factor. |
| [26] | T. Kohler, "Crowdsourcing-Based Business Models: How to Create and Capture Value," *California Management Review,* pp. 63-84, 2015.  This article addresses how to produce value from a crowdsourcing based business model. It demonstrates how existing companies have found success through a crowdsourcing model. It provides valuable advice and insight in this area. |
| [27] | R. Amit and C. Zott, "Creating Value Through Business Model Innovation," *MIT Sloan Management Review,* 20 Mar 2012.  This article seeks to show the reader how an innovative business model can be extremely effective and profitable. |
| [28] | Student Life and Learning, "VentureLab Intellectual Property Release," Grove City College, 2015.  Grove City’s Intellectual Property, which talks about mobile applications that are made on Grove City College’s campus. |
| [29] | M. Hersovici, M. Jacovi, Y. Maarek, D. Pelleg, M. Shtalhaim and S. Ur, "The Shark-Search Algorithm, an Application: Tailored Web Site Mapping," *Computer Networks and ISDN Systems,* vol. 30, no. 1-7, pp. 317-326, 1998.  The Shark Search Algorithm was discussed when talking about trade secrets that the team could have. |
| [30] | J. Mischke and B. Stiller, "Rich and Scalable Peer-to-Peer Search with SHARK," *Fifth Annual International Workshop on Active Middleware Services,* June 2003.  SHARK is a search algorithm that uses a hierarchy of keywords. We can use something similar to this to serve lists of restaurants, delivery locations, and food items. |
| [31] | M. Shaw, C. Subramaniam, G. Tan and M. Welge, "Knowledge management and data mining for marketing," *Decision Support Systems,* vol. 31, no. 1, pp. 127-137, 2001.  This article describes the rising demand for metadata and the use of that data to adjust customer preferences. Our application could help companies better understand what consumers want and when they want it. |
| [32] | Hothand Inc., "Mobile Commerce Framework". USA Patent 7693752, 6 Apr 2010.  A subscription-based system for providing commerce information for mobile devices between merchants and customers. This is not an issue for us, since Foodini is not subscription based. |
| [33] | J. F. Paulucci, "Food Order/Delivery System". USA Patent 4797818, 10 Jan 1989.  A computerized order/delivery system for use in the food service industry includes a central computer that accepts customer food orders and, based on a customer identifier, automatically selects the store that is to prepare the food and transmits the required information to that location. This is not an issue for us, since Foodini does not automatically select a store, and no information is transmitted to any restaurants. |
| [34] | J. M. Garrett, "Mobile Restaurant Ordering System". USA Patent 20120209730, 16 Aug 2012.  A mobile ordering system designed to process an order made when a customer enters a geographic region determined to be the restaurant. Our application does not rely on our customer being in the restaurant to process the order. |
| [35] | Sears Brands LLC, "Systems and Methods for Mobile Integrated Ordering". USA Patent 20130317949, 18 Nov 2013.  This patent is for methods and apparatus regarding an e-commerce environment with mobile ordering capabilities. It is somewhat related to our application, but Foodini will be much more encompassing. |
| [36] | Ameranth Wireless, Inc., "Information Management and Synchronous Communications System with Menu Generation, and Handwriting and Voice Modification of Orders". USA Patent 6982733, 3 Jan 2006.  This patent is for talking to your phone and making your order or changing an order. At this time, it does not relate to our project, but if we continue the project after graduation, it is something we need to be aware of. |
| [37] | T. Wong and T. Shen, "Mobile Based Voiceless Drive Through Ordering System and Method". USA Patent 20130024299, 24 Jan 2013.  This patent is for ordering food from a restaurant’s Wireless network. While this does not relate to our project, one of the embodiments is to allow ordering from a remote location, which is similar to what our application does. |
| [38] | K. Hernblad, "Customer-based Wireless Ordering and Payment System for Food Service Establishments Using Terminals and Mobile Devices". USA Patent 20040054592, 18 Mar 2004.  This patent discusses payment from a mobile device for a food item from a menu or a menu number from a restaurant. In the beginning stages of the application, we discussed having menus of restaurants for the user to pick from. Since then, we have decided against it due to legal issues and complexity of coding. |
| [39] | A. Jain and D. Shanbhag, "Addressing Security and Privacy Risks in Mobile Applications," *IT Professional,* vol. 14, no. 5, pp. 28-33, 2012.  Jan and Shanbhag’s paper talks about the rise of mobile applications and the security issues that are not being addressed. This was discussed as a team when considering ethical issues as well as risks for our application. |
| [40] | A. Jain and D. Shanbhag, "Addressing Security and Privacy Risks in Mobile Applications," *IT Professional,* vol. 14, no. 5, pp. 28-33, 2012.  Jan and Shanbhag’s paper talks about the rise of mobile applications and the security issues that are not being addressed. This was discussed as a team when considering ethical issues as well as risks for our application. |
| [41] | S. A. Scotto, "Utility for creating heatmaps for the study of competitive advantage in the restaurant marketplace". United States Patent 8595050, 4 November 2014.  These GrubHub patents showed us what is possible with geolocation services and data collection, and better informed us on our choice for what to do with data. |
| [42] | S. A. Scotto and R. S. Zivin, "Utility for determining competitive restaurants". United States Patent 8595050, 26 November 2013.  These GrubHub patents showed us what is possible with geolocation services and data collection, and better informed us on our choice for what to do with data. |
| [43] | S. A. Scotto, J. L. Battle, J. Evnin, Z. Danielson and K. Adams, "System, method and apparatus for managing made-to-order food tickets for a restaurant service". United States Patent 9009067, 14 April 2015.  These GrubHub patents showed us what is possible with geolocation services and data collection, and better informed us on our choice for what to do with data. |

|  |  |
| --- | --- |
| Writing Quality (15%) (Supports performance indicator f-3)   * Proper spelling and grammar * Writing is clear and to the point * Follows proper idiom for CS technical writing * Cogent and concise | Writing quality is good: clear, cogent and mostly uses the proper idiom.  15/15 |
| Depth and completeness of report (50%) (Supports performance indicator f-2)   * All sections are in the report (missing section is a deduction of 20%). * Discussion is complete, clear, and concise. (Confusing, tangential or superficial information or explanation is a deduction of 10% per section.) * All sections have sufficient depth and required data. (Failure to supply data when required results in a deduction of 20 %.) | The report is complete. The designs and analysis are well explained, as are the tests. However, I found the ordering of the tests confusing, as it doesn’t follow the layout of the app’s views. Also, there’re a few inconsistencies in view or control names.  48/50 |
| Document appearance and organization (20%) (Supports performance indicator f-1)   * Looks professional. * Chapters or sections start on a new page, all other items properly formatted. * All pages are numbered. * All figures and tables are captioned, numbered, and referred to in the text. * Document contains all required sections. | Formatting is good, with all figures and tables properly referenced in the text. I’m not a fan of the font, as it looks “bold” and too chunky. But, it is a reasonable design choice.  The right margin is slim. I recommend one inch margins on all sides.  20/20 |
| References (10%)   * Appropriate number and type of references * All references are cited in the text. * Summary of all bibliographic items | References are good, particularly the patents.  10/10 |
| **Total for the report** | 98 |