Project Group A2 - DESIGNZ Team User Guide

Zane Shepherd, Anastasia Tyutyunnik, Matthew Simpson, Diego Hernandez, and Orel Benkarmona

December 3rd, 2020

Abstract

The final deliverable, the user manual, at last a long journey has come to an end. Inside you will find all of our thoughts, tasks, and resources we used to build our food delivery app.The first steps were based in the design process we learn during our semester. Emphasize , Define, Ideate, Prototype, and test. They may not be laid out that way in the document but we made sure to cover them all. Also included within are all the resources we used to code the app and how we went about using them down to the letter. We even managed to outline some next steps so as to add more functionality and included photos to aid in the recreation of the app. May you find this 3xoerience as fulfilling as it was for us.

Table of Contents

Introduction

Through this project our group has created an app from scratch, and through this learned many things about the design process. From the beginning with the empathize stage, and until the end with the test stage, we have worked with the JAMZ team to create the best possible solution for them. This included multiple meetings, surveys and trials with errors to come to a refined final product, which was composed of the best elements created throughout this process. Furthermore, this was investigated in the empathize, define and ideate stages. All of these stages, while long, were very important as they allowed us to thoroughly develop an app that suits not only the needs of the clients, but also desirable features to make our app more attractive. Which led to the final product, which was refined in the prototype and test stages. All in all, this progress has led us to create an innovative solution in the form of the app. To continue, the usage, maintenance, and implementation with future considerations will be explored in this User Guide.

Our Design Process

Empathize

For the design process, we have defined all elements as “needs” and categorised them as “required, highly desired, optimal, not necessary and unwanted”, tested out many different layouts and designs and voted for the best one according to our design criteria which we looked for the easiest and most fascinating look. Furthermore we created an app from scratch using React Native Library. Thus, finished with a fully functional prototype. Before the design process, we empathize with what our client wants.

First of all, we have come up with a problem statement, which states our main goal during the design which was “The client JAMZ is in need of a simple, sleek, and easily navigable mobile app as a user interface that rural residents can use to connect to urban city restaurants to place and track food orders using drone delivery.” In order to create our problem statement; first of all, we have acknowledged the conflict, then understood the situation and the main problems and noted down during the meeting with clients and reached an agreement as a group and decided our main goal by setting our problem statement.

Secondly, as the client has made it very clear that there are many characteristics the mobile app must have.Which are;

1. Account login and information saving. So that customer information, such as address and payment method, may be stored and reused for consequent visits to the app without any delays.
2. Designing a restaurant page which lists restaurants around the 10km radius of the current location and categorised by their total pricing.
3. Cart functionality in order to add and remove items.
4. Tracking Feature which shows the current location of the order to the user with the help of a map.
5. Order progress notifications that show the clients the current progress of their order. For instance: confirming, in progress, picked up and arrived.

Then we categorised the needs of the project into sub groups which states the level of the importance and priority. It has been categorized accordingly to what we empathized during client meetings. It has categorized as required, highly desired, optimal, not necessary and unwanted. As an example, ratings page, accessibility features, grouped categories, FAQ page and settings page for the prototype considered as a highly desired level.

Finally, we have decided our constraints criteria using metrics in order to start the design process.

Problem Statement

We derived the problem statement from the client requirement and from our own benchmark of competing products that will be talked about shortly. In the end we came up with the statement as follows:

"The client JAMZ is in need of a simple, sleek, and easily navigable mobile app as a user interface that rural residents can use to connect to urban city restaurants to place and track food orders using drone delivery."

Benchmarking

We as a group did both User and Technical benchmarking. For the user benchmarking we compared the different features between them such as layout and colour pallette. We also benchmark the individual features we thought would be applicable to the app based on the client requirements and the results from the user benchmarking.

Technical Benchmarking

App Specifications	**UberEats**	**SkipTheDishes**	**Doordash**	**Instacart (Grocery Delivery)**
Subscriptions	Eats Pass ($9.99 a month, one month free trial)	None	Subscription for free deliveries (DashPass)$9.99 per month	Subscription Plan(PC Insiders) $199 per year
Price of delivery	15% of order plus added fee of 2$ or $ dollars if less than 15$ purchased	$2.99-$4.99 depending or distance between restaurant	$1.99-$4.99 depending on distance	$3.99 for orders over $35, varies if order under $35
Range	Depends in restaurant, typically around 5 miles from restaurant	Within respective city limits	Typically around 5 miles from restaurant	Delivers based on chosen supermarket
Offers	-Invite another user offer -Promotions for certain restaurants	-Has a points program section -Invite another user offer	Promotions for certain restaurants	-Invite another user offer
Layout	Homepage with promotions and nearby restaurants	Homepage with promotions and nearby restaurants	Homepage with nearby restaurants -Pickup tab to	Homepage with items on sale. And divided by section

	-Search tab with premade categories -Offers tab -Cart tab -User account tab -Shows current delivery address	-Shows current delivery address -Search tab with categories -Tab for past orders	track orders -Offers tab to track newest offers -Search tab with restaurant categories -Recent Orders tab	-Search feature with categories -Cart tab -Integration with PC Optimum
Rewards program	*Uber Rewards* -Earn by using Uber app[or Uber eats -get discounts on rides and deliveries	*Skip Rewards ©* -Earn 5 points per dollar spent -$2.50 dollar off per 2’500 points accumulated	*Dasher rewards* for drivers	*PC Optimum* -Partner with Loblaws, Esso, and Shoppers Drug Mart

As you can see from the user benchmark many of the benchmarked apps shared similar features such as a Homepage and rewards programs. While many of those features were not applicable the design they used were quite useful for our final design.

Design Specifications	**Relation (=, < or >)**	**Value**	**Units**	**Verification Method**	**Priority (1-5)**
		Constraints			
Price Of Delivery	==	15% of order price + added fee of 2$ + additional cost if more drones required	$	Estimate, final check	3
Restaurant display radius	>=	10	Km	Analysis	4
Size	adaptive	The size of the screen of the phone.	mm	Test	3
Platforms	==	Multiple Smartphone Platforms and website	N/A	Test	4
Users	==	Suburb/Rural citizens	N/A	N/A	2
Accessible	==	Yes	N/A	Analysis	3
Order Confirmation Time	==	1	minute	Test	2
App size	<=	300	MB	N/A	1
Resolution	adaptive	According To Phone Resolution	DPI	Test	2
		Functional Criteria			
New user account creation view	==	Yes	N/A	Test	5
Created accounts stored via communication with back-end	==	Yes	N/A	Test	5
Login authentication via communication with backend for existing users	==	Yes	N/A	Test	5

Choice to remain logged in (phone app) or remember your login credentials	==	Yes	N/A	Test	3
Create a home address associated with your account	==	Yes	N/A	Test	4
Ability to use google maps APIs to assess location via choice of saved home address, input address, or current location from google maps API.	==	Yes	N/A	Test	5
Show restaurants within a (10km?) radius of the specified location	==	Yes	N/A	Test	5
Incorporates views for each restaurant to show their menu items	==	Yes	N/A	Test	5
You may navigate to these views via individual search of a restaurant or	==	Yes	N/A	Test	3
Create a "cart" data structure which can add or remove objects associated to menu items. This means each menu item will need an associated object	==	Yes	N/A	Test	5
Menu item objects will have properties of price and weight	==	Yes	N/A	Test	5
Total cart weight will be used to determine how many drones are needed for delivery	==	Yes	N/A	Test	5
Required amount of drones will be taken into account for the delivery price	==	Yes	N/A	Test	5
Communicate with 3rd party purchasing app	==	Yes	N/A	Test	5
Display order progress	==	Yes	N/A	Test	5
Display current location of drone on Map	==	Yes	N/A	Test	4
Display confirmation	==	Yes	N/A	Test	5
Non-functional criteria					
Simple, sleek and modern aesthetic	==	Yes	N/A	Test	5
Very easy to navigate	==	Yes	N/A	Test	4

between sections of the app, and each section is clearly labelled					
Sections of the app are properly organized and categorized	==	Yes	N/A	Test	4
App can be used and opened everyday at all times	==	Yes	N/A	Test	4
Available to be used by multiple customers at a time (not within the same app)	==	Yes	N/A	Test	5
Operates smoothly with no bugs	==	Yes	N/A	Test	4
Communication is easily facilitated between customers and businesses, and both parties are notified immediately of any messages	==	Yes	N/A	Test	3
Updates are made every half hour about the availability/condition of the drones' service during harsh weather conditions, when the customer is attempting or waiting for an order	==	Yes	N/A	Test	2
Frequency of notifications can be altered	==	Yes	N/A	Test	1
A report services for missing or damaged drones accessible to users	==	Yes	N/A	Test	2
Visual and audio tutorials when the app is first launched, on how to use the app and interact with the drone	==	Yes	N/A	Test	2
Tutorials can be kept to explain how features of the app work	==	Yes	N/A	Test	2
The users' information will be kept secure	==	Yes	N/A	Test	5
Multiple items can be added to a "cart"	==	Yes	N/A	Test	5
Visual organization of categories of food and restaurants	==	Yes	N/A	Test	4
Items can be reviewed before submitting the order	==	Yes	N/A	Test	3
Live updates on location of the drone	==	Yes	N/A	Test	3
Customers and businesses can confirm the arrival of the drone and food	==	Yes	N/A	Test	4
Frequently Asked Questions (FAQ) section is available on the app for the customers to view and post questions	==	Yes	N/A	Test	2
Items can be customized to the customer's liking, depending on the restrictions of the business	==	Yes	N/A	Test	3

The chart above includes all the criteria that we thought applicable to app development plus as well as our rankings of them, in a scale of one to five. We thought to establish a hierarchy early on to ensure the=at the most important features would be implemented first in accordance with the design process.

Design Criteria

Design criteria began with the requirement set out by our client JAMZ. THis included among other things; account creation, cart page, tracking etc. All these features and more had to have full functionality in order to have a full working prototype. Examples of the full functionality include having the cart page having the functionality to save the items that were chosen in the item page, that meant that data had to be transferable from page to another which meant having a unified data structure. Notification updates were also an important aspect as we needed to relay weather updates as the drones cannot fly in adverse weather conditions

We also had to consider the non-functional aspect of the app. Aesthetics mere of most importance as leaving a lasting impression is of the utmost importance. We needed to make everything as visual as possible which meant having large readable buttons and pictures of the restaurant and food items. We also had to come with a catchy logo that immediately drew attention to. We also wanted the app to be as accessible as possible, which meant that it had to be compatible with both Android and Apple.
In addition, as drone delivery is a new concept we decided that adding a FAQ(Frequently ASked QUestions) section would help new users feel at ease as they could easily get the answers to their questions as easily as possible.

Design Concepts

When creating design concepts, it was important to generate as many ideas as possible. This meant quantity was better than quality. For this reason, we had every group member design a concept for what the app should look like. Each member created either a flow chart, sketch, or digital media mockup.

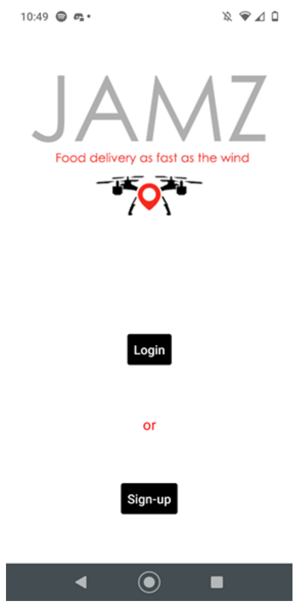
Based on these, each group member voted on each other's design out of five, and through this we chose a winner, which ended up being a flowchart design. The primary reason for choosing this design was because of how intuitive the design was, and how explicitly stated the user experience was in the flowchart. In addition to this, we created a high fidelity mockup of the flowchart using photoshop, so as to grasp a better visual understanding as to how the final product will look like.

Prototyping and Testing

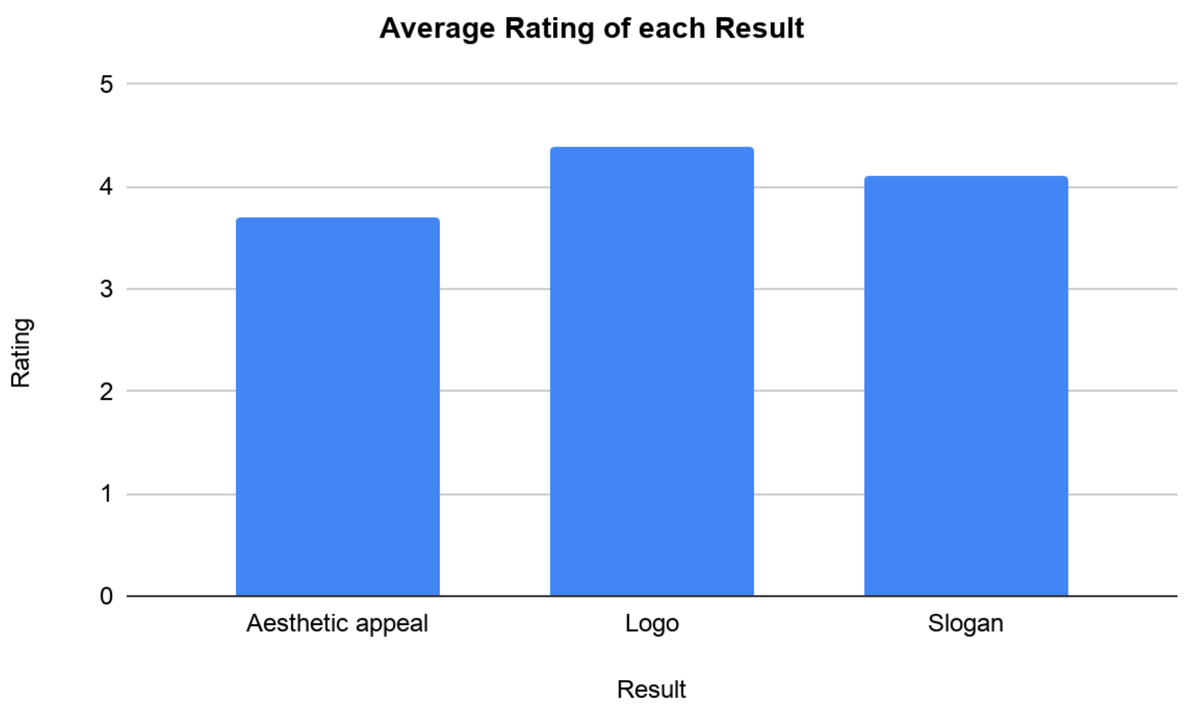
The current stage of the interface is the third prototype of the application. Prototypes one and two were very similar to the current prototype, but with fewer screens and more bugs. After completing each prototype we generated feedback by surveying potential users, and converted this feedback into data that could be analysed and used. Our sample sizes were limited in numbers, and more extensive data collection should be performed before publishing the app. However, this information is still useful in understanding the strengths and weaknesses of the interface.

Prototype 1:
Prototype 1 was used to generate feedback about aesthetics and functionality. Functionality issues have been fixed in recent prototypes, so the relevant feedback relates to the design aesthetics.

Sample screen:



Relevant survey data:

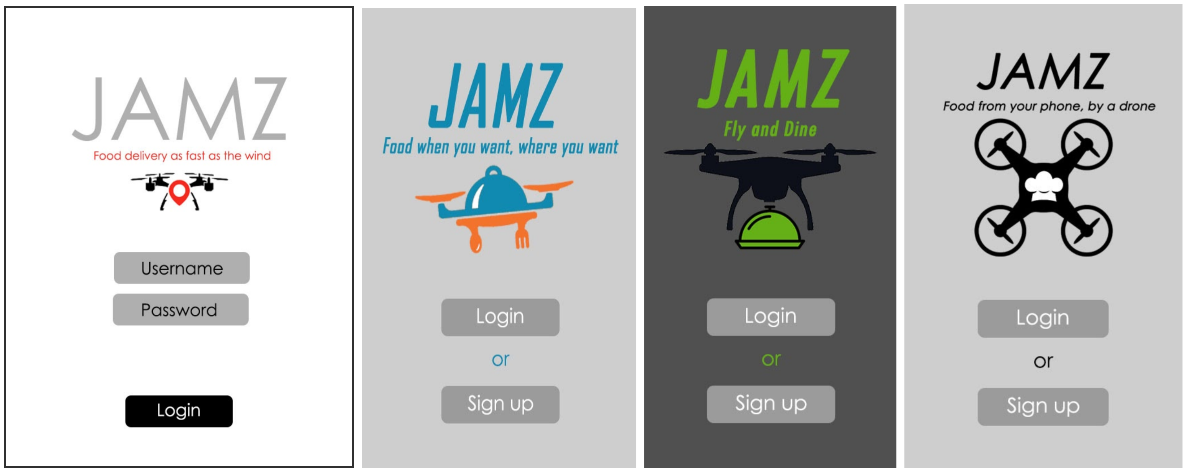


Any comments of complaints related to the aesthetic appeal?
“It's simple and clean. Although, I don't like that the name JAMZ gets covered by the icons on the top of the screen (signal, bettery, etc.)”
“There is too much white...I feel something is missing.”
“I don't know what the ovoid symbol is supposed to mean/represent. I get that the symbol overall is a drone. I would suggest simplifying/streamlining the rotors of the logo slightly.”
“ok”
“Very neutral and it ain’t “spicy” (easy to follow though)”
“Colours work well with one another create appealing look”
“It feels a little empty. I'm not sure if I'm expecting a banner at the header or footer but there a little bit too much whitespace imo.”
“I think you should pull the login, or, and signup buttons up about 10 -15 percent. Also not sure if the "or" text is necessary.”

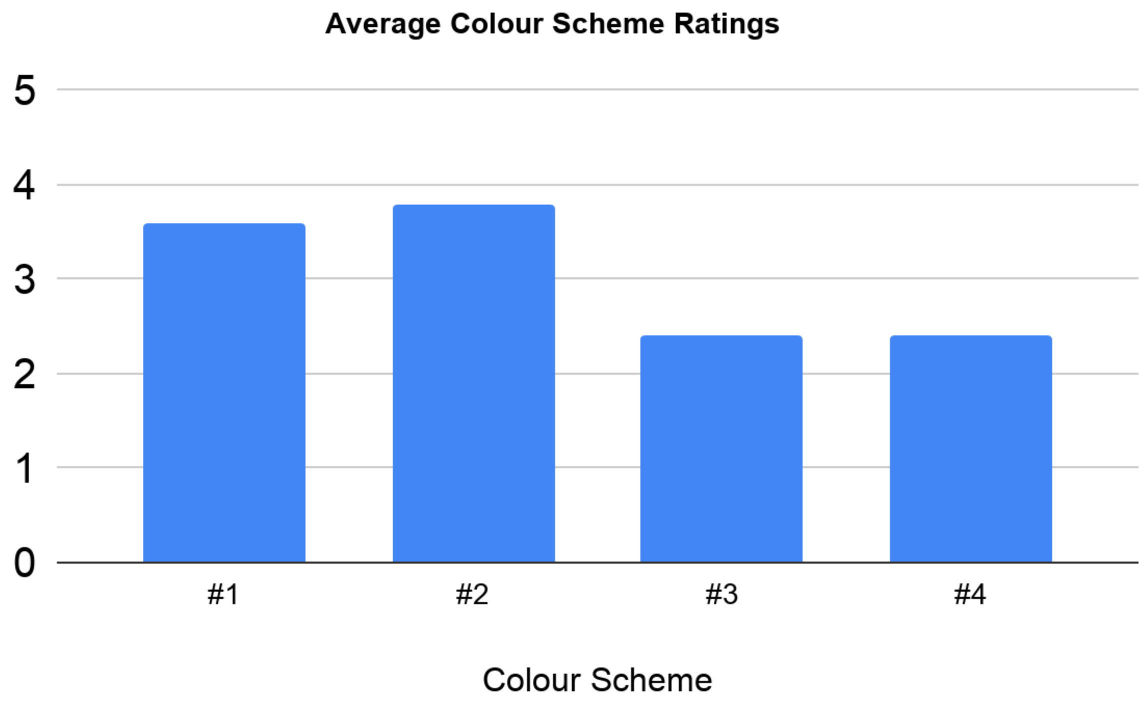
Prototype 2:

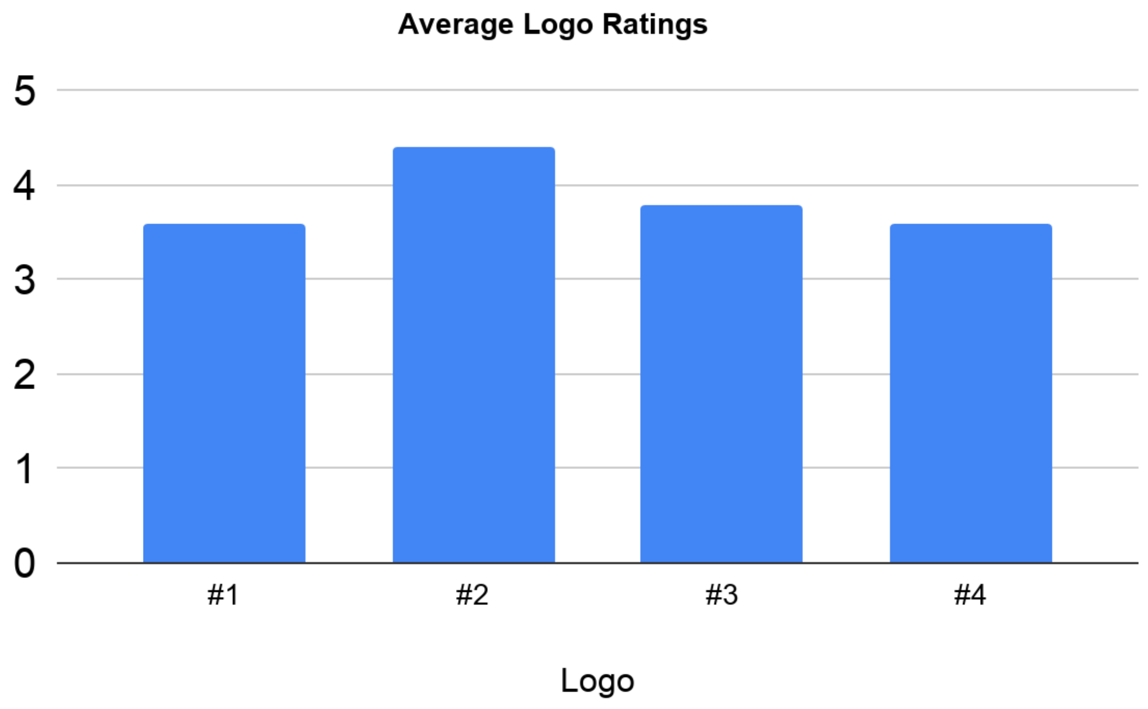
Prototype 2 focused on generating design alternatives and adding more screens to the interface. Relevant data about this prototype focuses on alternate designs.

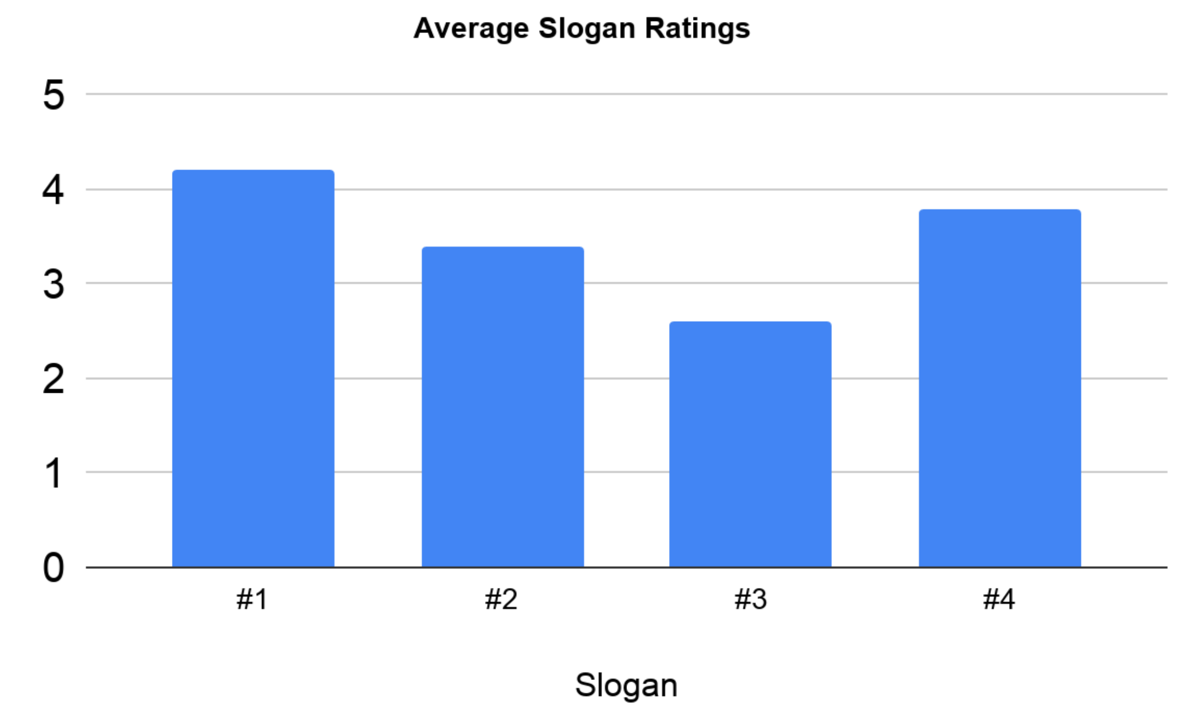
Designs:


JAMZ
Food delivery as fast as the wind
Username
Password
Login
JAMZ
Food when you want, where you want
Login
or
Sign up
JAMZ
Fly and Dine
Login
or
Sign up
JAMZ
Food from your phone, by a drone
Login
or
Sign up

Relevant data:

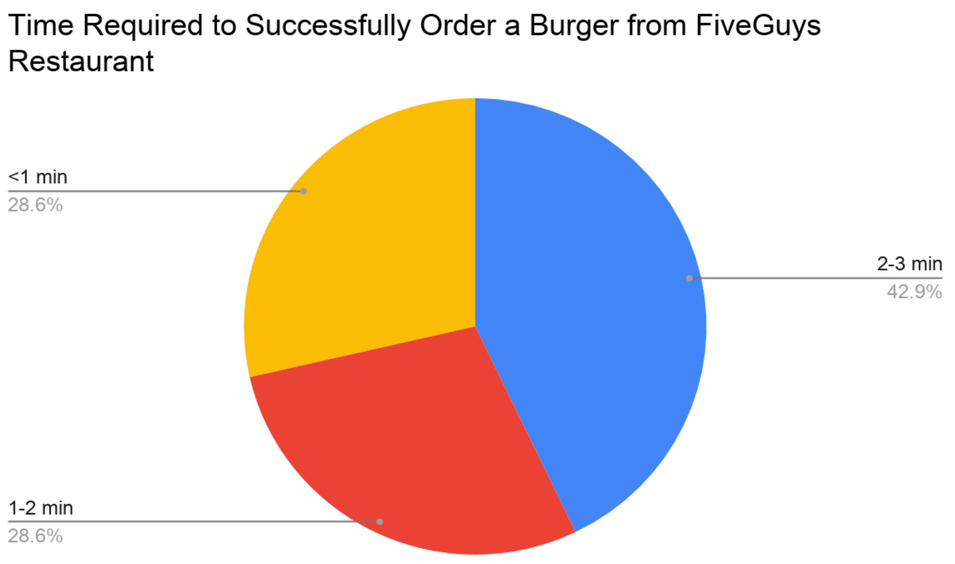

Average Colour Scheme Ratings
5
4
3
2
1
0
#1
#2
#3
#4
Colour Scheme

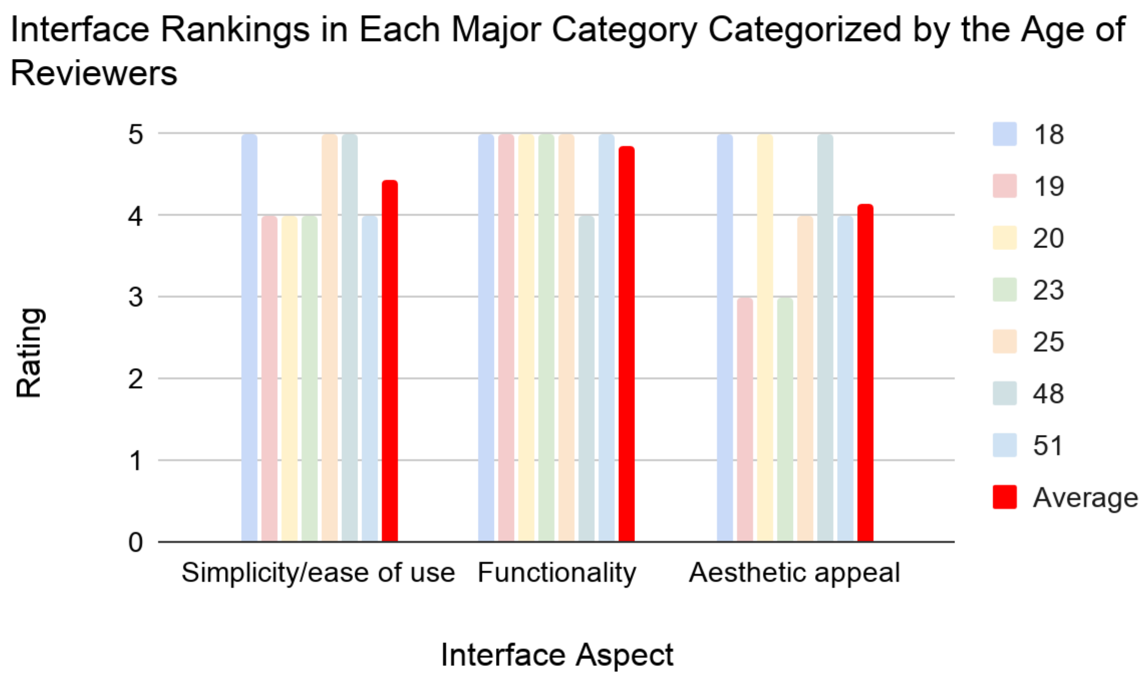

Average Logo Ratings
5
4
3
2
1
0
#1
#2
#3
#4
Logo



Prototype 3:
To view this prototype, go to the section of this document titled “Running our App Using Expo.” This is a comprehensive prototype and it allowed us to gather feedback pertaining to how a user would actually use the interface. All users who were surveyed were asked to use the app to order food, and then provide feedbacK about the user experience.

Relevant Data:





Any comments or issues encountered relating to ease of use?
nah man this app is dope
Not all the buttons are fully functional yet, but the interface is very intuitive
Don't use on a non touch display
no issues, although five guys had already showed up, I didn't need to search for it
What functionalities would you add to the interface?
option for tipping
nothing to add
scrolling
Website for non mobile users
payment info!
Any comments of complaints related to the aesthetic appeal?
nope

I like the visuals of the food items
Text is cut out in some sections
not a fan of white on grey, also too much white imo
Any other comments or concerns?
nope
Looks good
It's Good

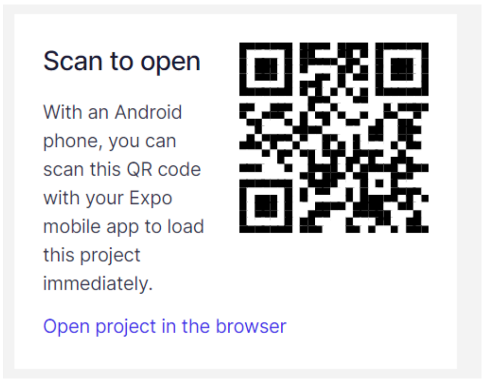
The most relevant feedback that we received was the comment highlighted in yellow stating that we should add a payment info screen. We chose to not add a payment screen in this prototype because we did not have a back-end yet, but this comment is important to take note of, because the app will not be publishable without such a screen.

Cost

To date the project has cost nothing but time, but if it were to be published commercially the client would have to purchase a google maps API key, which is priced in accordance to site/app traffic. Furthermore they would have to purchase access to all png files that were used within the design, which would cost approximately $100. The final cost is work hours. Taking into account our lack of experience, we would estimate that it would take a professional about 100 hours, or three work weeks to complete the project, seeing as our group spent about 300 hours on it. Seeing as software developers are paid about $45 per hour on average, the cost of labour for the project will be in the range of $4500.

Running Our App using Expo:

An interactive version of our app is available at:
https://expo.io/@matts1121/projects/jamzv7
To open it go to the link then scan the QR code from the Expo-Client app on your phone or select “Open project in the browser”, as seen below.

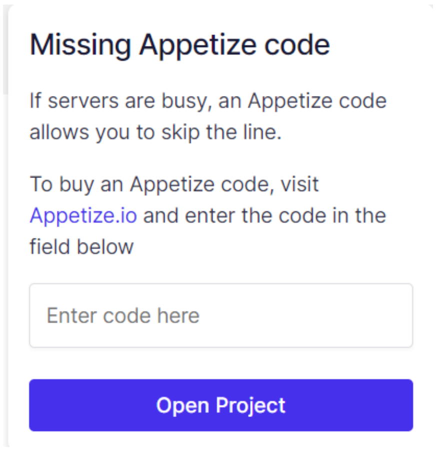


Scan to open

With an Android phone, you can scan this QR code with your Expo mobile app to load this project immediately.

Open project in the browser

If you select to open it in the browser it will ask you for an Appetize code but you can just click "Open Project" instead, as seen below.

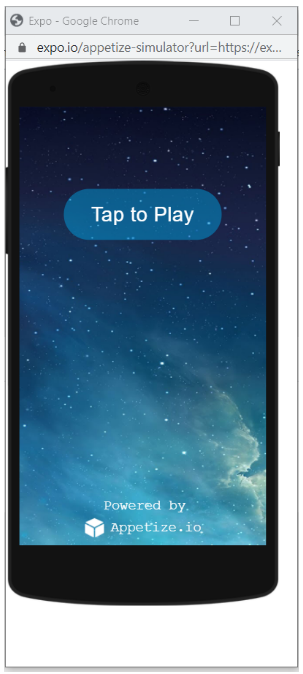


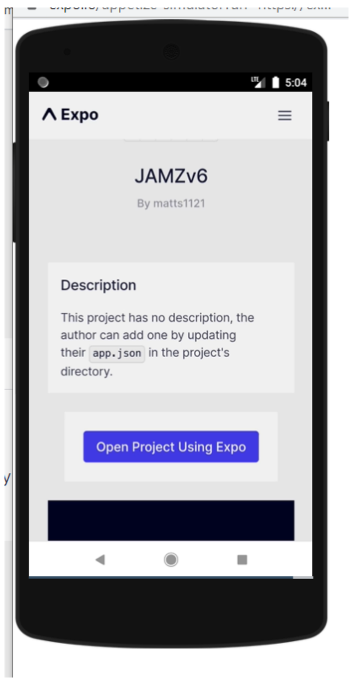
Missing Appetize code

If servers are busy, an Appetize code allows you to skip the line.

To buy an Appetize code, visit Appetize.io and enter the code in the field below

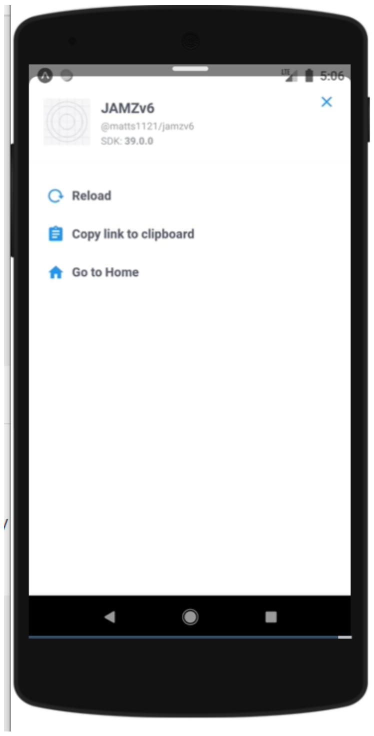
It will open an android emulator in your browser and then you click "Tap to Play". This then brings you to the link on the emulator. Scroll down by clicking and dragging (as if it were a phone) and click "Open Project Using Expo".



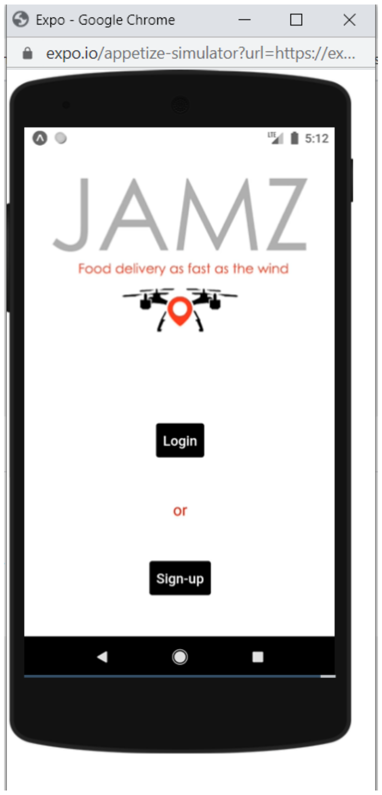


This project has no description, the author can add one by updating their `app.json` in the project's directory.

If it brings up a page that looks like the one below, that's just the developer tools menu. You can close this with the "x" in the top right corner.



You should then see our app start screen like below. From here you can interact with the app as if it were running on an android phone.


Expo - Google Chrome
expo.io/appetize-simulator?url=https://ex...
5:12
JAMZ
Food delivery as fast as the wind
Login
or
Sign-up

How to Use the App

Start Screen/ Login/ Signup:

This is the entry page which will the user see when entering the app. In this screen they have to login to their active account or have to create a new account with the help of pressing the signup button. In this screen there are two sections which ask the user ID and password beneath it. The user can login to their current account with their username and password. Else, they will click on the signup button which asks the user to put their personal details like phone number, full name and email address. Also the user has to create a password and confirm it after all.

Address Page:

In the address page users have to choose their location with the help of a map and location services which every phone has. The order will be sent to the selected location.

Home Page:

In this screen the user can select the category of food they want to be served by selecting “fast food mexian,etc” and they will be linked to a page which the restaurants around 10 km radius will be listed due to the chosen category. Else, beneath the categories, there will be some restaurants displayed as “nearby restaurants". Every listed restaurant has a rating out of 5 stars and the category of the restaurant is being shown next to the name of the restaurant. Moreover in this screen the user can sort the nearby restaurants as they wish to. For instance, relevance, price, arrival time and in an alphabetical order.

Categories/ Sorting/ Searching:

In this screen the restaurants that are being stored to a specific category will appear due to which category the user selects. In addition to that next to every restaurant name the average price will appear which states the average cost of the food to that specific restaurant.

Restaurant Page

If a user selects a restaurant, the menu of that specific restaurant will be displayed on this page. Every restaurant has a different menu so the content differ depending on the restaurant which the user will choose.

Items Page

On this page, you are given an image, name and a short description of the item - which is provided by the restaurants - as well as the total price of the food. Below, you will see options for toppings, spice levels, etc. and then be able to select either a limited amount or any amount - depending on the restaurant's allowance for this. After, if you have any notes or dietary restrictions, you will be able to add it in the text box at the bottom of the page. Finally, at the bottom, you can change the quantity of the item, see the price of the item, and then add it to your cart.

Cart Page

Once items have been added to the cart, the items will be listed with their names, and their prices and quantities will be shown beside the name. You can also directly adjust the quantity of the food items from this page as well. From here you can then see your receipt before you finalize the order. This includes total, taxes, fees, and tipping. Then, you will be able to press the confirmation button to finalize the order.

Progress Page

The progress page is meant to update the user periodically on the status of their order. Statuses that may be shown include: order received, order confirmed, order made, order picked up, order on its way, and order arrived. Below the status bar is the live map. This app would be updated based on the drone’s whereabouts and relay it back to the user. Once the drone has arrived and that status bar displays “Order Arrived,” the user will be able to click this as a button, and then be taken to the next page.

Ratings Page

Finally, the last stage of the ordering process is the Ratings Page. This includes dynamic 5 star icons, which allow you to click on them to display the desired amount of stars. Users can input their ratings for the 5 star rating by tapping on them. Below this, there is a textbox to input any comments that the user may have for the client or the restaurant. Finally, at the bottom, there is a button labelled “Ok” which will take the user back to the home screen, finishing the ordering process.

How the App Would Run with Backend Information

Since the general idea of creating this app was to concentrate more on form than functionality, and the clients had not set up a backend database, the app could not be seen in its full potential. This being said, if the app were to receive backend information, a few things in its current form would change.

For starters, on the Home Page, the restaurants listed would be automatically updated to show nearby restaurants with the address that the users enter in the address page beforehand. In addition, the search bar would be able to autocomplete names of restaurants, once the restaurants are registered for use with the app.

This would therefore mean that upon reaching the individual Restaurant Page, the ETA would then be able to be automatically calculated and shown to the user. In addition to this, there is a rating for each restaurant, which would be updated based on users' input for restaurants on the ratings page. In addition to this, businesses would be able to update the price of items, and the description of their restaurants, as well as sub-categories within this page and names of items.

On the individual Items Page, the name will be updated based on what the restaurant's inputs are, as well as the description and total price, and available customization - i.e. toppings, sides, spice level, etc.

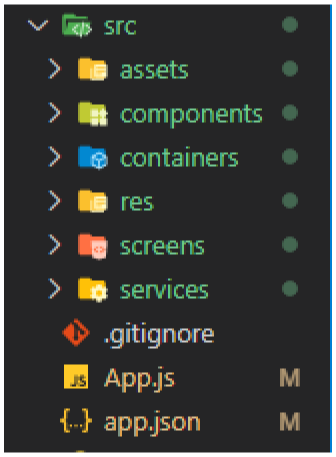
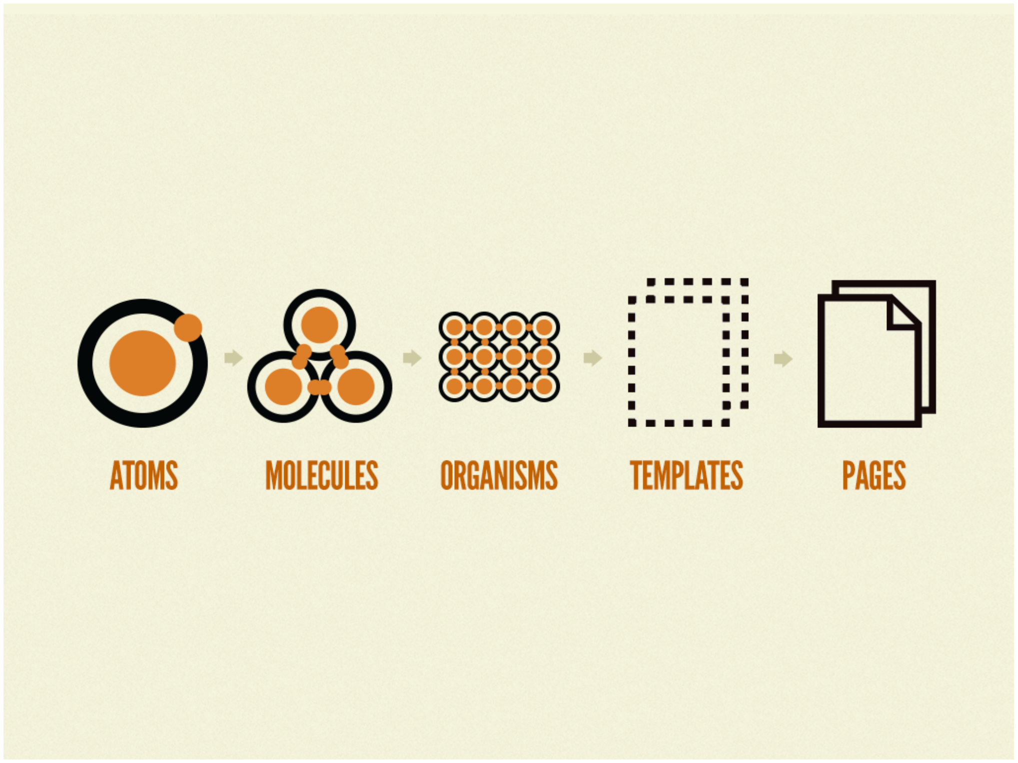
In continuation of this, when users add items to their cart, they would then be able to actually store the items with the backend server allocating memory for this. As of right now, our code itself is unable to do this, and therefore it currently only shows what it would look like if an item is added to the cart. In addition to this, all other calculations for the total, taxes, etc. would be able to update automatically. As well as if the quantity of the item is increased, the total price appearing next to the item would be able to update as well.

The most notable difference to the app would evidently be the Progress Page. Since this page relies heavily on the information provided from the drone and the restaurants, all details on the page currently are placeholders. Primarily, the progress status bar would update each stage backend connection is enabled. For example, once the customer sends in their order, the restaurant will confirm they have received it, and this will therefore be showcased on the user's end. Similar statuses will be shown with the drones updates of the delivery, such as when it has picked up the food, and when it has arrived. Additionally, the live map will then be able to show a live updating of the drones whereabouts since it will be able to track the drone.

Finally, the last notable feature that would be updated is the ratings page. The five star ratings that the users enter would be stored with the information of the restaurant, and therefore would update the ratings of said restaurant periodically - as displayed on the home and restaurant page. In addition, the comments about the food would be sent back to the restaurant and the clients.

Understanding Our Code:

Structure of our code:

- 
```
src
  assets
  components
  containers
  res
  screens
  services
  .gitignore
  App.js        M
  app.json      M
```
- Throughout our coding process we implemented Brad Frost's Atomic Design Methodology.
- 

In this analogy you can think of a basic react element as the atoms. The smallest building block for our app. These would come from either built-in react elements or from component libraries we decided to import to the project. In the analogy, the atoms come together to form molecules which form more complex organisms. This is the same as our react elements being used together to form custom components or full systems like our map. We then make the format structure of our pages as the templates. Finally, we populate the templates with the components and systems we made as well as data about the restaurants.

The main navigation of the app is declared within the App.js file. For in depth details about how the navigation works you can refer to the React Navigation Documentation. The src folder is where we put all the files of code we wrote except the App.js file. The src folder contains folders named:

- assets: Stores all of the pictures to be used in the app.
- components: Files of code we wrote for custom made components.
- Containers: Files of code for more complex component systems or templates
- res: Files that contain data like our color variables or restaurant information
- screens: Files for all of our app screens
- services: Files of code written to interact with data such as the location services

Elements Used from Packages:

Our code was created using node.js, and in this environment users can import packages from external modules to perform specific tasks. These packages contain multiple functions that allow the user to achieve a specific functionality, without having to write out every line of code themselves.

We have included a list of all the modules that we used in our code, as well as each function that we imported from the module and the purpose of these functions

Node Modules; Components; functionalities:

- expo-status-bar
    - StatusBar
        - Configures the status bar at the top of the phone screen (battery, carrier service, etc.)
- expo-constants
    - Constants
        - Allows the user to use functions that remain constant across all devices (we used Constants.statusBarHeights to configure padding above the pages)
- @expo/vector-icons
    - FontAwesome
        - A specific database used to import icons (used for star icons)
    - AntDesign
        - A separate icon database (used for other icons)
- react
    - React
        - Allows the code to be written with JSX elements
    - useState
        - Used to return a stateful (changeable) deat. Will help develop future search functionality in the search bar

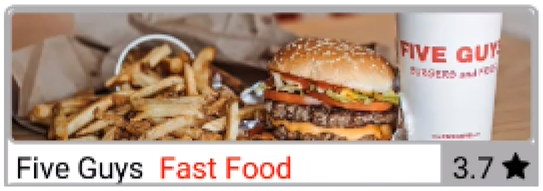
- react-native
    - StyleSheet
        - Used to style objects (similar to CSS styling)
    - Text
        - Used to input a text object
    - View
        - Used as a container to position objects
    - Image
        - Used to input an image
    - TextInput
        - Used to input an object in which the user can type text
    - TouchableOpacity
        - Adds opacity effects to objects upon being clicked. This lets users know when something has been clicked successfully
    - Dimensions
        - Used to get the dimensions (width and height) of an application. This is useful for ensuring the app looks the same on all screens.
    - SafeAreaView
        - Ensures that the content of a screen is not hidden by other aspects of a device (navigation bars, tab bars, etc.)
    - ScrollView
        - Generates scrolling functionality when not all of the content can fit within the screen
- react-native-elements
    - Button
        - Used to create buttons that have specific functionalities when pressed
    - Input
        - Similarly to TextInput, this creates a container in which the user can enter text
- react-native-gesture-handler
    - TouchableOpacity
        - Equivalent to TouchableOpacity in ‘react-native’
- react-native-numeric-input
    - NumericInput
        - Inputs a container that contains a number and a way to increase/ decrease that number
- react-native-geocoding
    - Geocoder
        - *Requires a google maps API key* this function transforms a location description into geographic coordinates
- react-native-google-places-autocomplete
    - GooglePlacesAutocomplete
        - Inputs a search bar with geographical autocomplete results
- react-native-maps

        - MapView
            - Provides a view of Google Maps or Apple maps within the app
        - Marker
            - Creates a marker to pinpoint the location that you are selecting on the map
- @react-native-community/picker
    - Picker
        - Allows for the creation of a selection menu upon clicking a prompt
- @react-navigation/native
    - NavigationContainer
        - Imports functionality which allows the app to navigate between pages
- @react-navigation/stack
    - createStackNavigator
        - Configures the navigation to take the form of a stack

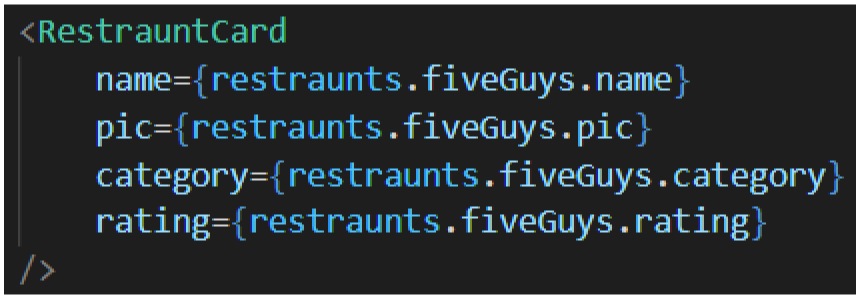
Custom-Designed Components:

Restaurant Card:

Below you can see an image of our restaurant card component:



This is a component which you can simply pass the restaurant properties to and it will render. An example of this is seen below:

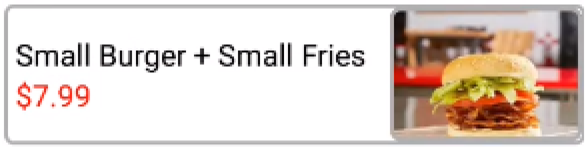


```
<RestrauntCard
    name={restraunts.fiveGuys.name}
    pic={restraunts.fiveGuys.pic}
    category={restraunts.fiveGuys.category}
    rating={restraunts.fiveGuys.rating}
/>
```

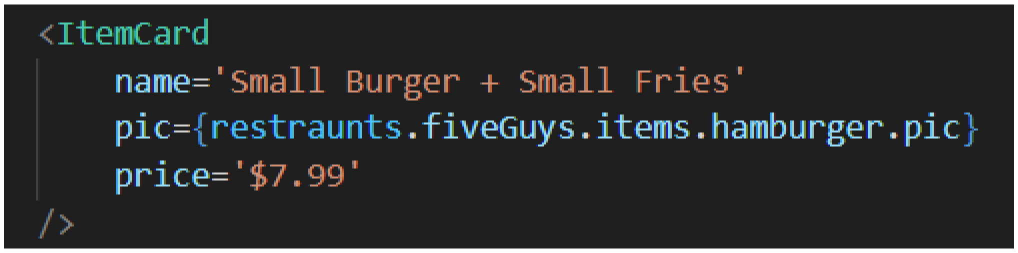
This makes it easy to implement the component for multiple restaurants on the homepage.

Item Card:

Below you can see an image of our item card component:


Small Burger + Small Fries
$7.99

This component functions similarly to the restaurant card so that you can just pass it properties for it to render like below:


```
<ItemCard
    name='Small Burger + Small Fries'
    pic={restraunts.fiveGuys.items.hamburger.pic}
    price='$7.99'
/>
```

This makes it easy to implement the component for multiple menu items on the restaurant page.

Implementation:

Installing Package Dependencies:

Commands for installing packages to the project:

- First, open the command prompt or terminal and run the following command to navigate to the project directory:
  - cd C:\(insert project directory here)
- Once you're in the project directory you can run the following commands to install the package dependencies of the project:
- React Native Elements:
  - npm install react-native-elements
- React Navigation:
  - expo install @react-navigation/native
  - expo install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context @react-native-community/masked-view
  - expo install @react-navigation/stack
- Native Base
  - npm install native-base --save
- React Native Numeric Input
  - npm install react-native-numeric-input --save
- React Native Geocoding
  - npm install react-native-geocoding --save
- React Native Google Places Autocomplete
  - npm install react-native-google-places-autocomplete --save
- React Native Maps
  - npm install react-native-maps --save
- Picker
  - npm install @react-native-community/picker --save

Implementation of Google Maps API:

Before you begin one must create a google maps account to be able to create an API key.
Steps:

- Go to Google Cloud Platform Console
- Create an account
- Once an account has been made Click the menu button and select Home.
- Click the project drop-down and select NEW PROJECT.
  - Select the name of the project
  - The project ID which can be manually made
  - The billing account which should be the default one
- To select the desired API

- Go to the Maps API page
- If the button says ENABLE, click the button to enable the API or SDK.
- Once done, return to the Google Cloud Platform Console
- Locate the Google Maps Platform API page:
- Click the menu button .
- Under Home, scroll down the menu to locate and then click Google Maps Platform.
- You should see if the API has been enabled or not
    - If disabled just click on enable to get it working

- To get the API Key
    - Use this link to go to the page to authorize the API key

Installation of API in React Native

- Open command prompt within the project folder
- Type “ npm i @react-google-maps/api” to begin installation

Usage of API

- Create “MapContainer.jsx” is src folder
- Put container in App.jsx to enable rendering
- Type “ import { GoogleMap, LoadScript } from '@react-google-maps/api';”

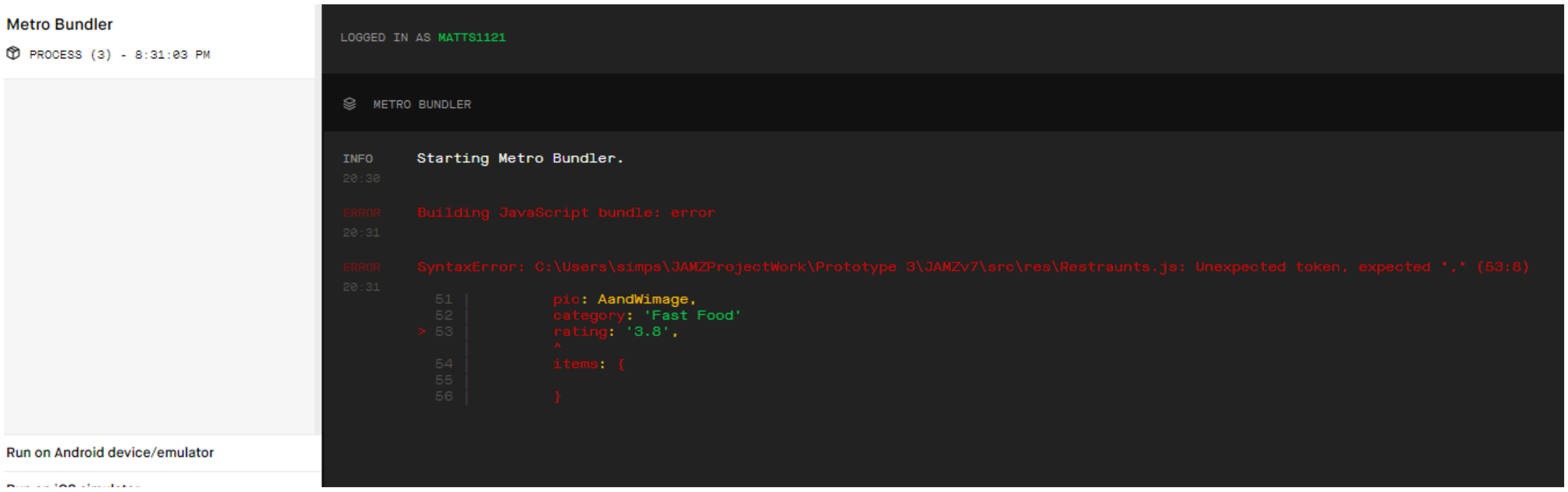
in the desired screen to implement

- (Note: Vastly lower render time as it cause rendering issues if left too high)

Potential Problems and Tips:

Debugging:

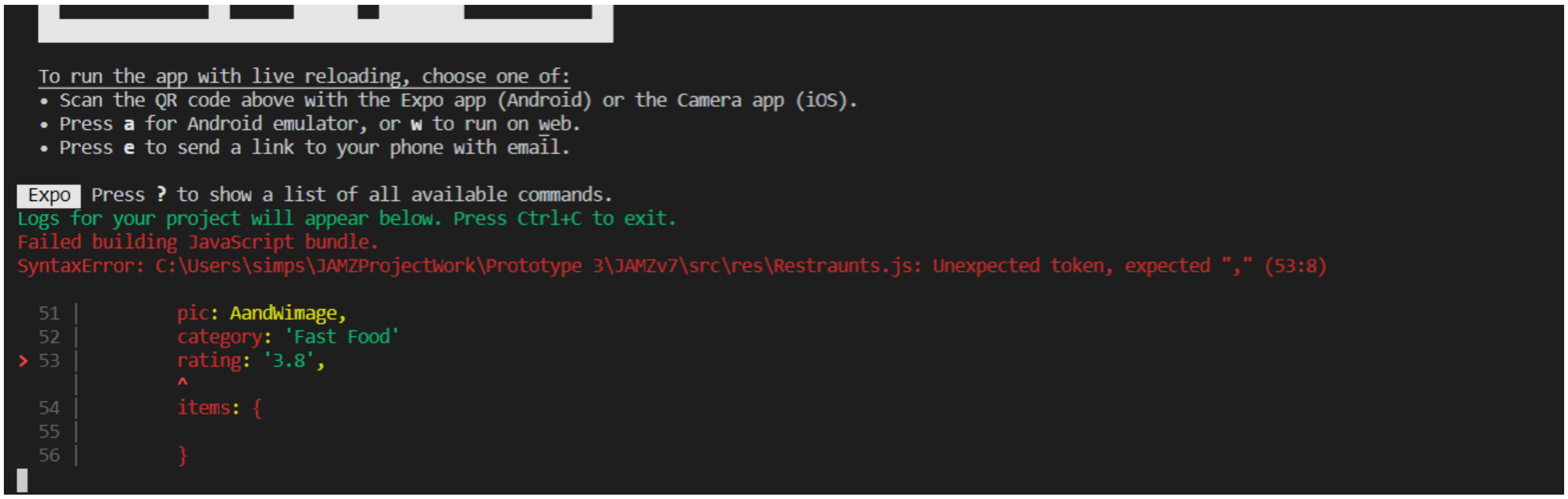
If there is an error rendering your code with the Expo app it will display an error message in red like below:



Metro Bundler
PROCESS (3) - 8:31:03 PM

```
LOGGED IN AS MATTS1121

METRO BUNDLER

INFO     Starting Metro Bundler.
20:30

ERROR    Building JavaScript bundle: error
20:31

ERROR    SyntaxError: C:\Users\simps\JAMZProjectWork\Prototype 3\JAMZv7\src\res\Restraunts.js: Unexpected token, expected "," (53:8)
20:31
           51 |           pic: AandWimage,
           52 |           category: 'Fast Food'
         > 53 |           rating: '3.8',
              |           ^
           54 |           items: {
           55 |
           56 |           }
```

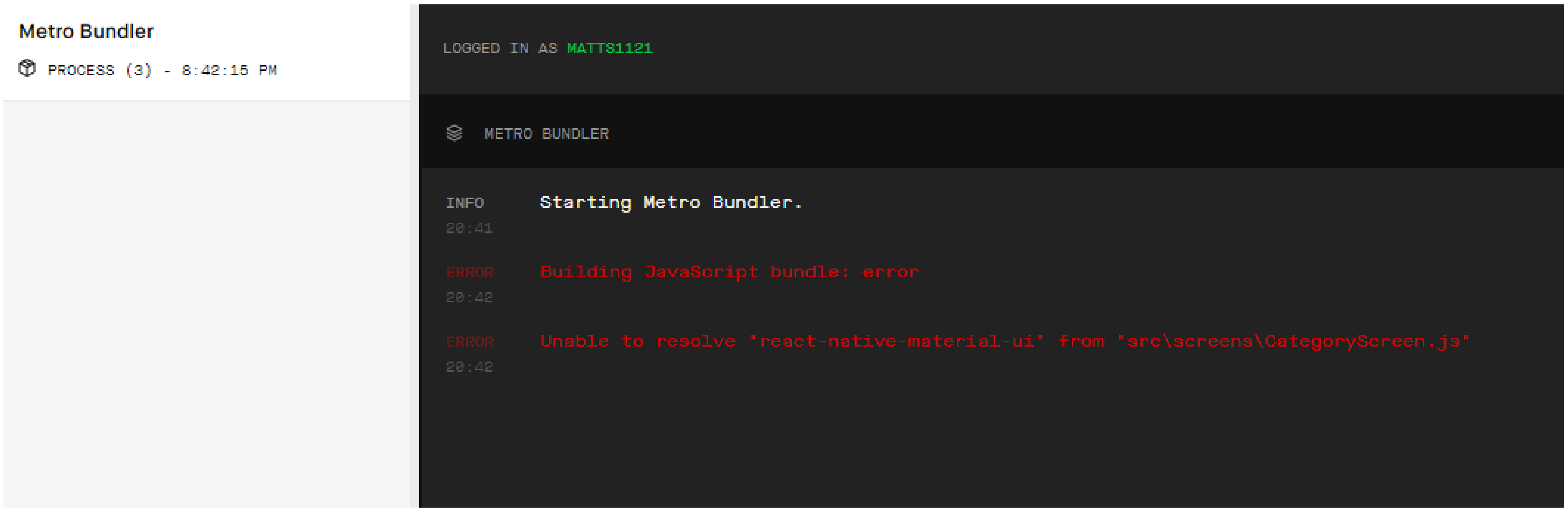
You can also see the error message in the VSCode terminal like below:



```
To run the app with live reloading, choose one of:
 • Scan the QR code above with the Expo app (Android) or the Camera app (iOS).
 • Press a for Android emulator, or w to run on web.
 • Press e to send a link to your phone with email.

 Expo  Press ? to show a list of all available commands.
Logs for your project will appear below. Press Ctrl+C to exit.
Failed building JavaScript bundle.
SyntaxError: C:\Users\simps\JAMZProjectWork\Prototype 3\JAMZv7\src\res\Restraunts.js: Unexpected token, expected "," (53:8)

  51 |         pic: AandWimage,
  52 |         category: 'Fast Food'
> 53 |         rating: '3.8',
     |         ^
  54 |         items: {
  55 |
  56 |         }
```

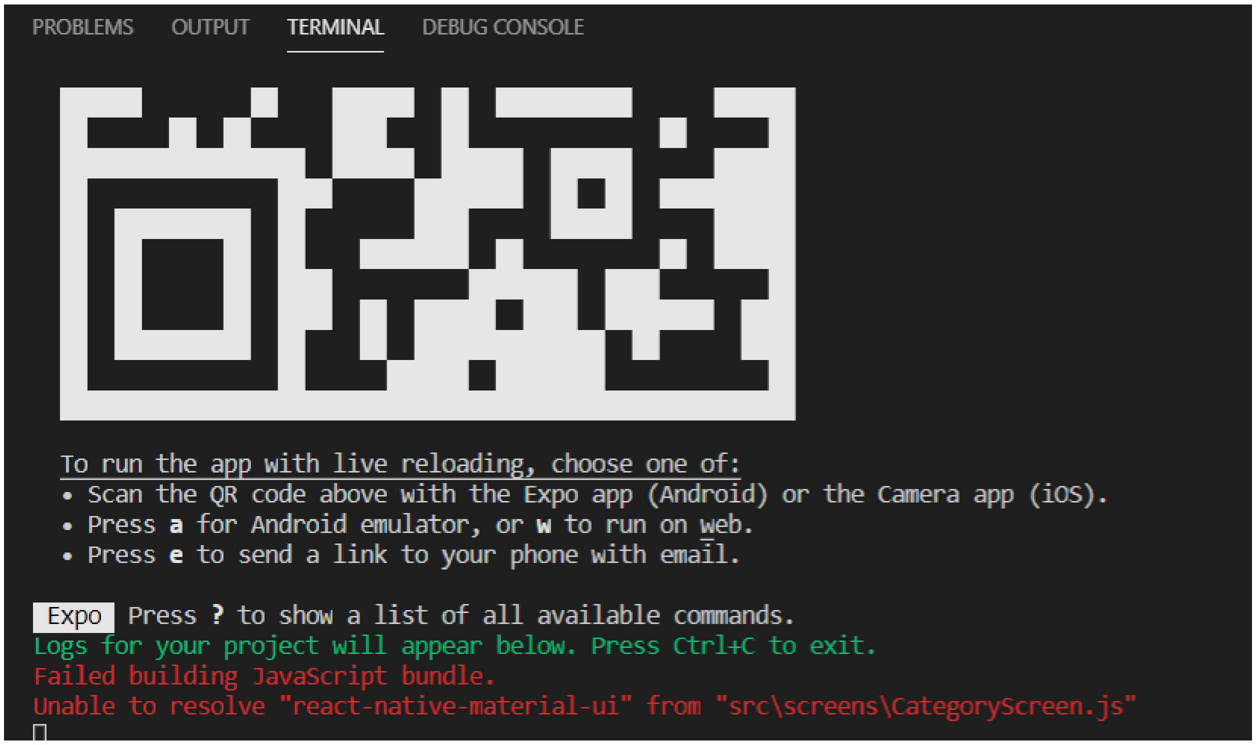
In the above case it is simply a syntax error of a missing “,” and the red arrow indicates where it was expected

Another common error message you might encounter is if there is a missing package dependency. You can see an example of what this message might look like below:



```
Metro Bundler
PROCESS (3) - 8:42:15 PM

LOGGED IN AS MATTS1121

METRO BUNDLER

INFO    Starting Metro Bundler.
20:41

ERROR   Building JavaScript bundle: error
20:42

ERROR   Unable to resolve 'react-native-material-ui' from "src\screens\CategoryScreen.js"
20:42
```

Or within VSCode this would look like:



```
To run the app with live reloading, choose one of:
• Scan the QR code above with the Expo app (Android) or the Camera app (iOS).
• Press a for Android emulator, or w to run on web.
• Press e to send a link to your phone with email.

 Expo  Press ? to show a list of all available commands.
Logs for your project will appear below. Press Ctrl+C to exit.
Failed building JavaScript bundle.
Unable to resolve "react-native-material-ui" from "src\screens\CategoryScreen.js"
```

The way to fix the error in this case would be to install the package using npm. If you need more details on this process refer to the “Installing Package Dependencies” section of this guide or the online documentation for the package you are trying to install.

Future Considerations:

Pricing to implement Google Maps API on a larger scale

Publishing the App:

To publish our code to the Google Play Store or Apple Apps Store you will need to first eject the app from the Expo client. You will also need to eject from Expo if you wish to apply the package dependencies for iOS independently from the Android packages. This was also the reason why our Expo app will not yet render for iOS. We chose to continue coding with Expo and these packages may be

Getting QR Code to render on progress page:

To do this use the component library react-native-qrcode-svg. An example of how to write the code for this component may be seen below:

```
import QRCode from 'react-native-qrcode-svg';

// Simple usage, defaults for all but the value
render() {
  return (
    <QRCode
      value="http://awesome.link.qr"
    />
  );
};
```

In order to make the QR code unique you should pass the value property the stored value of the user's phone number.

Drone Error Cases:

In order to show error messages from backend signals you should use the component library react-native-popup-dialog. To include this package in the project run the command below in the directory of the project from the command prompt or terminal:

```
        npm install --save react-native-popup-dialog
```

An example of how to write the code for this component may be seen below:

```
import Dialog, { DialogContent } from 'react-native-popup-dialog';
import { Button } from 'react-native'

<View style={styles.container}>
  <Button
    title="Show Dialog"
    onPress={() => {
      this.setState({ visible: true });
    }}
  />
  <Dialog
    visible={this.state.visible}
    onTouchOutside={() => {
      this.setState({ visible: false });
    }}
  >
    <DialogContent>
      {...}
    </DialogContent>
  </Dialog>
</View>
```

However, the code above should be changed for the dialog to render from that backend signal rather than a button push.

You might also decide to render the QR code outlined above as a popup dialog component instead of directly on the current page.

Conclusion

Users will often support a product due to its functionality, but what initially draws them in is its aesthetics. In this way, the appearance of an interface is essential to its commercial success. Furthermore, a poorly designed interface can make the service difficult to use and prevent the implementation of a potentially revolutionary product. It is for this reason that our group set out to design a simple, functional, and aesthetically pleasing interface for our client. We have researched, analysed, and conceptualized almost every aspect of a successful front-end interface and have created a functional comprehensive prototype, to be implemented as outlined throughout this document. We have gained a great deal of knowledge and worked to implement it into this final design and we hope that this guide has helped you put all of that work to use.

Appendix:

Project Github Repository:

https://github.com/matt-1121/JAMZ_project_A2

Useful Links for Coding:

- Javascript Basics
- React Tutorial
- React Native Documentation
- React Native Quick Guide
- Expo Documentation
- Javascript's ES6 Features
- Harvard's Course on Mobile App Development with React Native
- React Cheatsheet
- ES6 Cheatsheet
- Node.js Download (choose long term support (LTS) version)
- Table of JS equality comparisons

Links to documentation for packages:

- react-navigation
- react-native-maps
- react-native-elements

- native-base
- Places API

-