

Deliverable K - User Guide

University of Ottawa

GNG1103

Technical Report
Deliverable K: User Manual

Submitted by: GNG1103 - Team A09

Evan Charkowy: 300126081

Lauren Poole: 300168350

Omar Shubair: 300160450

Sofia Ershova: 300114465

Owen Zhang: 300125481

Date: December 3rd, 2020

Abstract:

This is a technical document entailing the steps, materials and knowledge required to reproduce the JAMZ drone delivery user interface shown in the figures listed. It includes a base of necessary knowledge, steps taken to achieve functionality, how the client's needs were met, useful and important resources as well as how to troubleshoot the website. As well, additional sources to help comprehend the process are included under references. The general goal of this user guide is to provide a clear and detailed description of how this team constructed the drone delivery interface and how the desired and necessary features were implemented. Finally, some things to avoid and consider as well as future steps are included at the end to guide someone who wishes to reproduce the product. The former can help avoid making some time-consuming mistakes and overall, reduce confusion as well.

Table of Contents:

Abstract:	2
Table of Contents:	3
List of Figures:	6
List of tables:	7
Introduction:	8
Design Process Steps:	8
Table 1: Customer needs obtained from client meetings	9
Table 2: Comprehensive testing plan of the entire system flow and functionality	10
Table 3: Comprehensive website test results and feedback (one quarter of full testing)	12
Subsystems:	12
Login Page:	15
Figure 1: Login page and code	16
Figure 2: Login page error message	16
Sign Up Page:	16
Figure 3: Sign up page and code	17
Figure 4: Sign up page error handling and code	19
Reset Password Page:	19
Figure 5: Password reset page 1	20
Figure 6: Password reset email	20
Figure 7: Password reset page 2 and code	20
Email Confirmation Page:	20
Figure 8: Email confirmation upon sign up message	21
Home Page:	21
Figure 9: Home page	21
Terms and Conditions Page:	21
Figure 10: Terms and Conditions	22
About Us Page:	22
Figure 11: About us page	22

My Account:	22
Figure 12: Account information page and code	23
Settings page:	23
Figure 13: Settings page	24
Languages page:	24
Figure 14: Languages page and code	25
Order History page:	25
Figure 15: Order history page	26
Customer Feedback page:	26
Figure 16: Customer feedback page	27
Figure 17: Customer feedback submission	28
Personalize your drone page:	28
Figure 18: Personalize your drone page	29
Figure 19: Personalize your drone page and code	29
Dark mode feature:	29
Figure 20: Dark mode feature and code	30
Report page:	30
Figure 21: Report page	31
Phone numbers page:	31
Figure 22: Phone numbers page	31
Order Status:	31
Figure 23: Order status page and code	32
Figure 24: Order verification page	33
Shopping Cart:	33
Figure 25: Restaurant browsing page and code	34
Figure 26: Menu browsing page	34
Figure 27: Order summary/checkout page and code	35
Figure 28: Payment information input page	35
Figure 29: Order confirmation page	36
Order Map:	36
Figure 30: Order Map	37
Materials:	37
Table 4: Bill Of Materials and Respective Prices	38
Lessons learned:	38
What to avoid:	38
Resources/References:	38

Conclusions and Future Trials:

39

List of Figures:

Note: most figures below are composed of two parts. First, the aesthetics is shown which combines the HTML and JavaScript and second, the script which controls the functionality is shown as well afterwards.

[Figure 1: Login page](#)

[Figure 2: Login page error message](#)

[Figure 3: Sign up page and code](#)

[Figure 4: Sign up page error handling and code](#)

[Figure 5: Password reset page 1](#)

[Figure 6: Password reset email](#)

[Figure 7: Password reset page 2 and code](#)

[Figure 8: Email confirmation upon signup message](#)

[Figure 9: Home page](#)

[Figure 10: Terms and conditions](#)

[Figure 11: About us page](#)

[Figure 12: Account information page](#)

[Figure 13: Settings page](#)

[Figure 14: Languages page and code](#)

[Figure 15: Order history page](#)

[Figure 16: Customer feedback page](#)

[Figure 17: Customer feedback page and code](#)

[Figure 18: Personalize your drone page](#)

[Figure 19: Personalize your drone page and code](#)

[Figure 20: Dark mode feature and code](#)

[Figure 21: Report page](#)

[Figure 22: Phone numbers page](#)

[Figure 23: Order status page and code](#)

[Figure 24: Order verification page](#)

[Figure 25: Restaurant browsing page and code](#)

[Figure 26: Menu browsing page](#)

[Figure 27: Order summary/checkout page and code](#)

[Figure 28: Payment information input page](#)

[Figure 29: Order confirmation page](#)

[Figure 30: Order Map and code](#)

[Figure 31: Trello Board schedule](#)

List of tables:

[Table 1: Customer needs obtained from client meeting](#)

[Table 2: Comprehensive testing plan of the entire system flow and functionality](#)

[Table 3: Comprehensive website test results and feedback \(one quarter of full testing\)](#)

[Table 4: Bill of materials and Resources](#)

Introduction:

COVID-19 has brought many changes to all of our lives. With a whopping 300 percent increase of Canadians working from home, new and innovative ideas are coming to light to improve daily life. JAMZ is one such start-up as they wish to deliver delicious food to doorstep via drone. Our task was to create a user interface specified by JAMZ's requirements while optimizing user experience as well. Since many people wish to limit their contact with others due to the pandemic, drone delivery is the perfect solution, albeit it's fairly isolated. Therefore, we have created an interactive and easy to use website to simplify ordering food and to help the user through the process. Throughout this document, we have included our process, details and steps we took. Additionally, things that didn't go well and could have been completed in a more efficient manner should they be done again. Below, pictures of each page as well as important aspects of the code can be found. However, the code is not full or in the correct order. For the complete and correctly stated code, please refer to the project folder submitted on our Maker Repo page.

Design Process Steps:

The first and arguably one of the most important steps to take before beginning a project of this magnitude is defining what the final end goal is and generally, the plan to achieve it in the given time frame.

To discover prioritized needs, the client meetings were very helpful. However, if these hadn't been arranged, passive and active observation could have also been used. The three client meetings and topics are outlined briefly below and as mentioned, were a crucial part to beginning this project in the right direction:

- 1) Presentation of topic from client and responses to open ended questions asked by the designers
- 2) Personal team meetings with clients where constructive feedback (positive and negative) was received about the chosen design. This occurs directly after the ideation

stage of the design process. At this stage, it is very important to take notes on what the client thinks of the design because the further progressed in the project, the harder it is to make major changes without discarding large portions of work.

- 3) General presentation of the current website prototype as well as feedback from the client

From the first meeting especially, the general idea of the project began to take shape in terms of what it should include and be capable of doing. This vision is what makes this empathizing stage so important. **Table 1** outlines the customer’s needs and it’s relative importance on a scale of 1 to 3 (1 being least important and 3 being most important). Note also that the needs themselves are paraphrased quotations from the client meeting.

Table 1: Customer needs obtained from client meetings

Translated and Prioritized Customer Needs		
Number	Need	Importance
1	The software is accessible on the Web, ios and Android for Customers	2
2	The software is accessible on Android and ios for the business. Web is optional	1
3	The software has an Operator	2
4	There is an emergency button that reports vandalism and theft	1
5	The software has a confirmation feature for when the food is delivered	3
6	A login feature to keep transactions secure.	3
7	A main menu with many selections	3
8	Ease of use for the customers and businesses.	2
9	An interactive map is provided	3
10	Time estimation is provided	2
11	Messaging system for help when needed.	2

Once this data is obtained, the project can flow smoothly into the define stage with a problem statement, design criteria and metrics. A couple of important notes follow here: when benchmarking, it is very helpful to pick several competing products and compare them via table format. In this way, viewing which ones rank the highest in the desired categories is easy and then it becomes clear which one can be used to determine features and functionalities. By doing this in the first round of technical benchmarking, it facilitates adding the comparison of another product as well as again, provides clear indication of information rather than large paragraphs of details.

Prototyping:

Prototyping was done in three sections. The first was a basic proof of concept. This is especially useful if the designer has little to no knowledge or experience coding. From our final chosen concept from the ideation stage, the proof of concept was programmed with no functionality. However, it allowed the team to obtain much needed understanding of the level of difficulty as well as the projected time needed to complete tasks and features of difficulty. It also served the purpose of a sanity check. Features that were desired could then be evaluated as being feasible to program or not. It was important to sort the features out of reach before too much time elapsed.

The second and third prototype were very similar. Basically, the team elaborated on features and functionality while performing continuous testing and receiving feedback from the testers (ie: family and friends). This was done until the stopping criteria which was primarily as will be mentioned later, time.

Test plans and Feedback:

Below, the final test plans and tester feedback is provided. Many tests were performed throughout the weeks of prototyping. These tests included rigorous and detailed tests for each subsystem separately. Only the comprehensive tests and results have been shown included however the focused testing followed a similar format with categories that were almost identical.

Table 2: Comprehensive testing plan of the entire system flow and functionality

Test ID	Group Member	Subsystem	Test Objective (why)	Description of Prototype used and of Basic Test Method (what)	Description of Results to be Recorded and how these results will be used (how)	Estimated Test duration and planned start date (when)
1	Everyone	Entire system (all inclusive in regards to the subsystems)	<ul style="list-style-type: none"> -Ensure the full and integrated system is functional -Ensure each feature of the system is documented and clearly available to the user -Ensure the flow of pages and the general ordering procedure is effective and efficient 	Comprehensive analysis of all working components together	<ul style="list-style-type: none"> -Facial expressions while navigating through the website -Time taken for user to successfully created an account and follow set up procedures for their first time to actual checkout -Time taken for a previous user to successfully login and place an order -Time taken for a user to find the QR code for which is needed for the drone to scan and confirm the order -General 'aftertaste' following website usage 	<ul style="list-style-type: none"> -15 to 20 minutes (all inclusive) -Week of November 19th to the 26th, 2020

When testing the entire subsystem, we performed more tests to ensure results were as accurate as possible and that we got feedback from a variety of users (that would potentially use this service in the future too). As outlined below, most of the tests were timed trials. This is because functionality was very important to the client and the efficiency of each subsystem. Its role in the general progression of an order is extremely important.

For each tester, they were asked to perform two tests: the first was without instruction or previous insight as to the full functionality and flow of the program (Test A). The second test

was a day or two later where the user was asked to perform the same task, again without instruction (Test B). After this test, the user was asked about the general feeling they got from the website as well as specific comments/concerns that arose. This was to test how well the general flow of the website provoked recall in the user (goal to provoke recall rather than making the user remember) and evaluate the desired consistency aspect. All testers were friends or family of one of the designers.

Table 3: Comprehensive website test results and feedback (one quarter of full testing)

Test Number	Tester prior experience with website UI	Time taken from sign up to placing order	Time taken to navigate from landing page to QR code	Time taken to find language options	Time taken to find order and discover location of drone	General feeling of layout	Specific comments
1A	little to none	2 mins.	34 secs.	28 secs.	30 secs.	-Smooth	N/A
1B	little to none	1.5 mins.	31 secs.	21 secs.	20 secs.	SAME	-Buttons are big and easy to find -Pages are sometimes too minimalistic
2A	little to none	3 mins.	41 secs. (thought QR code was displayed with map section)	27 secs.	32 secs.	-Good feeling	N/A
2B	little to none	1.5 mins.	15 secs.	19 secs.	29 secs.	SAME	-Include a search bar to find specific settings easier. -Too much purple

Subsystems:

Below, each of the subsystems and it's integrated pages have been shown and explained. The flow of the pages is also documented (subsequent pages in flow are shown).

User Experience:

User experience is a primary focus of the entire project. This is evident in the goal of a user interface: to maximize usability and effectiveness for all users despite age or technological background experience. Most of the testing in fact has been in regards to how the user responds, feels and interacts with the level of efficiency with the website. For this reason, Nielsen's heuristics were attempted to be implemented. For example, during the password reset, there is a progress bar underneath the input box showing the user how many more steps they have before completion. Since this process is two, steps the bar first displays 50% and then 100%. This can be seen in the screenshots of the subsystem below.

Error Handling:

During the second client interview, the client really impressed the need and importance of efficient error handling in terms of interface functionality. One feature that has been added is defining any input requiring a user password as type 'password' so that the characters are keyed out and the password is no longer visible or remembered. For security purposes, this is beneficial to the user and also follows the general convention. Outlined below are the steps taken to prevent user errors during the use of the interface, again specifically for the user login and sign up page. For all error messages to be displayed they appear in small, red text underneath the input boxes. The colour red is used consistently and is also a convention for displaying 'wrong' or 'try again'.

- 1) Proper email format: using a RegEx key, the format of an email entered in the email field under 'sign up' is verified. If the email is not valid, the user cannot proceed past the sign up spot until they have fixed the problem. An error message is printed to the screen upon the attempted click in red letters to notify the user of the issue.
- 2) Password security check: using a RegEx key, the format of any desired password is checked. This is to ensure the customer uses an effective and relatively secure password for their account. Upon accepting this information, Firebase also hashes the password so as the designer, it is not ever available for viewing allowing for optimal security.

- 3) Password matching: as with most 'sign up' forms, the user is asked to enter their password twice as confirmation they have actually entered the password they thought without any typos. If the user's passwords don't match, the interface will throw an error notifying the user and preventing them from proceeding out of the sign up.
- 4) Required fields: this error simply prevents the user from skipping over information in the sign up process. The location is especially important for the map to be able to display available restaurants near the user and the user's location in relation to the drone.
- 5) Duplicate email: Firebase stores the user's information via email and assigns a particular email with a UID. If any changes need to be made to a specific user's profile, the UID is called to access that information. Therefore, there is a verification to ensure that any new member creating an account enters a valid and unique email address in comparison to ones already registered. If this case is not true, the user is notified to try with a different email.
- 6) No database match (login): this error catches a user who has tried to log in without having created an account prior. The interface will notify the user that their email and password do not match what is stored in the Firebase database and that they should try again.
- 7) Forgot my password: if the user forgot their password, they can have an email sent to them via firebase which allows them to reset it. Password reset is only available to users who have already signed in before and have an account.
- 8) Terms and conditions: a terms and conditions page was added in to the sign up process to ensure the user is aware of certain regulations when using the JAMZ services. This can also be modified at a later time by the JAMZ team themselves if they have more specific and detailed criteria. However, the user must check a box to prove they have

indeed read and understood the information and if they haven't, they are prevented from signing in.

Important Notes:

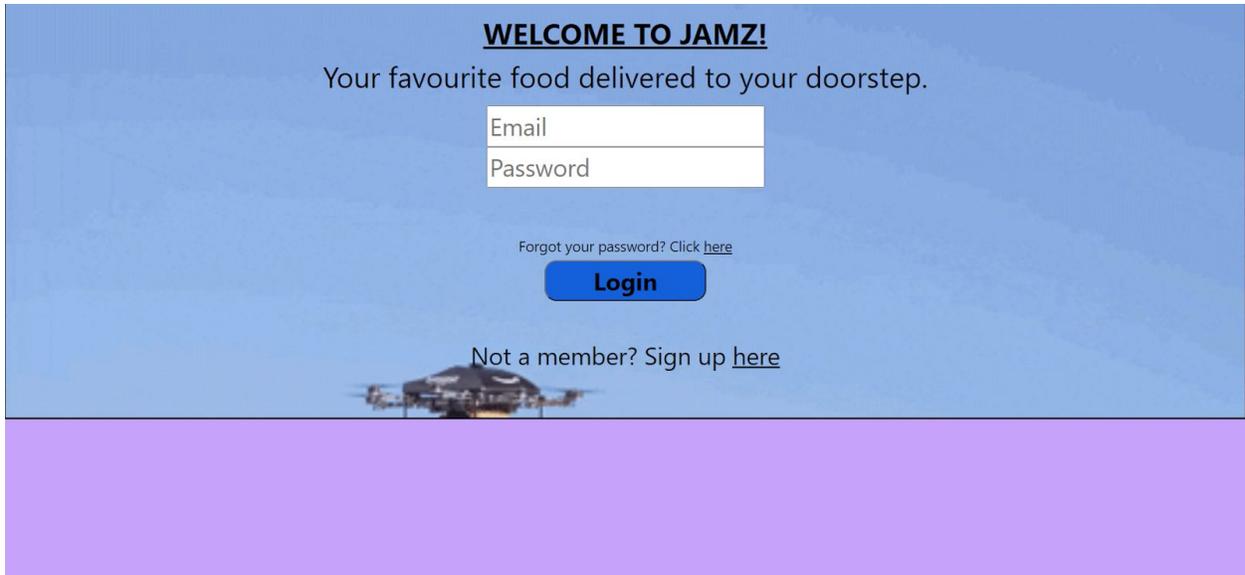
A difficult aspect of implementing Firebase was that it is asynchronous. This means that when a Firebase function is called, the Javascript continues to execute even if the Firebase call isn't finished yet. This leads to some big problems as was discovered in prototyping. First, each Firebase function call has a place and comment for the successful case and the unsuccessful case. However if the function fails, the code continues past the function until Firebase finishes and *then* executes the code for failing. This leads to error messages being displayed out of order as well as the page flipping quickly between desired outcomes. To resolve this, the following approach was used: event listeners were added to each Firebase function call. In this way, they ensured the code ran in the proper order and had all needed information before continuing with execution. This was a major bug in the development of this prototype and slowed progress until the event-listener approach was discovered and implemented.

Login Page:

This is the first page seen by the user when they go onto the webpage, also known as the 'landing page'. It gives the user three options:

- 1) Prompts via placeholder text-boxes indicate the user should enter login information if they have previously used the service
- 2) The reset password 'here' link takes the user to a two-step process walking them through everything (described later in more detail)
- 3) The sign up 'here' link takes the user to another page where they enter more information to enrol in the service

If the user chooses option 1 and logs in, their email is checked against using Firebase Authentication. If the email and password match, they are signed in, if not, the error message displays prompting them to retry or re-evaluate their options. The code is shown below and with it, the database query can be seen along with the successful and failed results.



```
85  function login(){
86
87      var email = document.getElementById("userEmailLogin").value;
88      var password = document.getElementById("userPasswordLogin").value;
89
90      firebase.auth().signInWithEmailAndPassword(email, password).then(function(){
91          //Handle successful login
92          showPage('homePage');
93      }).catch(function(error) {
94          // Handle Errors here.
95          var errorCode = error.code;
96          var errorMessage = error.message;
97          document.getElementById("loginError").innerHTML = "Email and password given
don't match database values.";
98          showPage('loginPage')
99      });
100
101      return;
102  };
```

Figure 1: Login page and code



Figure 2: Login page error message

Sign Up Page:

If the user doesn't have an account already, they are directed to this page to make one. Here, they enter information which is stored using Firebase Realtime Database. The email is given a UID which can be used to pull down this information if it is desired on future pages. The user is given the option to read terms and conditions and sign up.

A screenshot of a sign-up form on a purple background. The form is titled "Sign Up Information:" and includes the instruction "Please fill out all sections." The form fields are: "Last Name", "First Name", "Postal Code", "Email", "Password", and "Confirm Password". There is a small link "Why is this needed?" below the "Postal Code" field. At the bottom, there is a checkbox with the text "I have read and understood the terms and conditions" and a link "Read the terms and conditions: [here](#)". A blue "Sign me up!" button is located at the bottom right of the form.

```
231 ▾ firebase.auth().createUserWithEmailAndPassword(email, password).then(function(){
232     var user = firebase.auth().currentUser;
233
234 ▾     user.updateProfile({
235         displayName: firstName + " " + lastName,
236         photoURL: ""
237     }).then(function() {
238
239         let UID = user.uid;
240         //let orderNumber = user.orderNumber;
241
242 ▾         firebase.database().ref('users/' + UID).set({
243             postalCode: postalCode,
244             lastName: lastName,
245             firstName: firstName,
246             email: email,
247             droneName: droneName,
248
249         });
250 ▾         firebase.database().ref('users/' + UID + "/orderDetails/" + "orderNumber").set({
251             orderNumber: orderNumber,
252             restaurant: restaurant,
253             price: price,
254             date: date,
255             orderLocation: orderLocation,
256
257         });
```

```

257     });
258     /*
259     firebase.database().ref('users/' + UID + "/orderDetails" + "/orderNumber").set({
260       price: price,
261       restaurant: restaurant,
262       date: date,
263     });
264     */
265     showPage('emailConfirmationPage');
266
267     user.sendEmailVerification().then(function() {
268       // Email sent.
269       console.log('emailSent')
270
271     }).catch(function(error) {
272       // An error happened.
273       window.alert('no email')
274       var errorCode = error.code;
275       var errorMessage = error.message;
276       window.alert('no email');
277       console.log('No email sent' + errorCode + " : " + errorMessage);
278     });
279
280     }).catch(function(error) {
281       // An error happened leading to an unsuccessful sign up
282       var errorCode = error.code;
283       var errorMessage = error.message;
284       console.log(errorCode + " : " + errorMessage)
285
286     });
287
288     }).catch(function(error) {
289       // Handle Errors here.
290       var errorCode = error.code;
291       var errorMessage = error.message;
292       document.getElementById("duplicateEmailError").innerHTML = "<br>An account already exists with this email. Please use another one.";
293       showPage('signUpPage');
294       console.log(errorCode + " : " + errorMessage)
295       userValid = false;
296     });
297
298     return userValid;
299   }

```

Figure 3: Sign up page and code

For this page, tons of error handling has been implemented. This is needed as the information is important for emails to be sent for verification, password resetting and emailing the QR code in future pages. Therefore, the website checks to ensure the email follows the proper format, the password follows the proper format, the passwords match, the user has checked that they understand the terms and conditions and they haven't left any fields blank. Error messages are consistently given in red so the user recalls this means they must change something. Some of the javascript is shown below for two of the error cases where the user's email or password is not the right format.

Sign Up Information:

Please fill out all sections.

 * * *

[Why is this needed?](#)

Email must follow proper format.

Password must follow proper format.

```

function validatePassword(){
  var passwordFormat = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,10}/;
  if (passwordFormat.test(document.getElementById("userSignUpPassword").value)) {
    return true;
  }
  else {
    return false;
  }
}

function validateEmail(){
  var emailFormat = /^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$/ig;
  if (emailFormat.test(document.getElementById("userSignUpEmail").value)) {
    return true;
  }
  else
  {
    return false;
  }
}

```



Figure 4: Sign up page error handling and code

Reset Password Page:

In the event that a user has forgotten their password, they can still retrieve their account without having to sign up again. The first step of the password reset process asks for their email. Firebase authentication ensures that there is in fact, an email that is already being used and will send them an email. Next, the user views a confirmation message that the email

has been sent successfully. If not, a window alert pops down from the browser and the user is returned to the login page. The progress bar at the bottom displays how far the user has made it into the process in an effort to maximize user experience with Nielsen's heuristics.

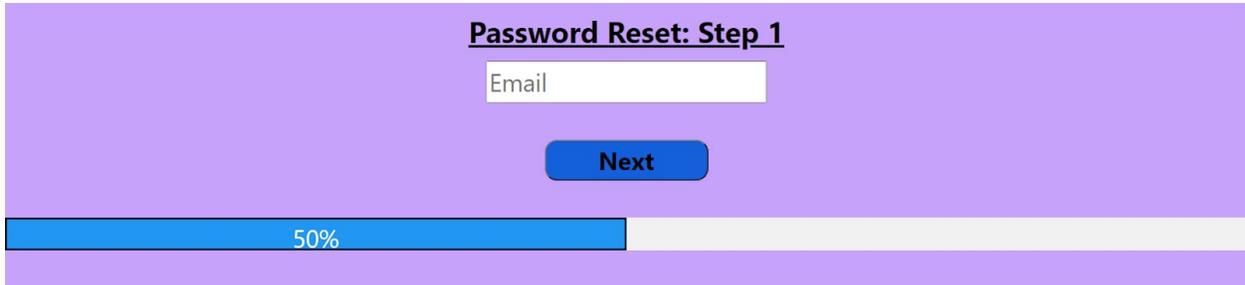


Figure 5: Password reset page 1

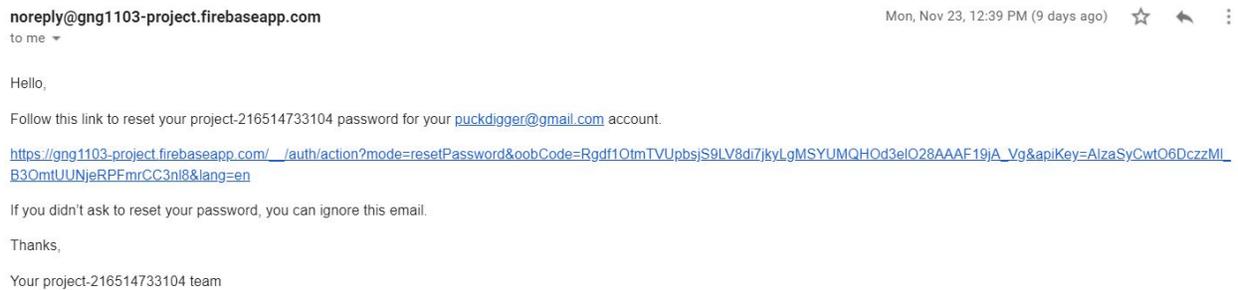


Figure 6: Password reset email

When the user clicks finish, they are taken back to the login page to login with their new credentials that have now been updated in the Firebase Database.



```

function passwordResetPage2 () {
  showPage('resetPasswordPage2')

  var auth = firebase.auth();
  var emailAddress = document.getElementById("userEmailForgotPassword").value;

  auth.sendPasswordResetEmail(emailAddress).then(function() {
    // Email sent.
    console.log('sent email');
  })
  .catch(function(error) {
    //Successful logout code
  });
  firebase.auth().signOut().then(function() {
    //window.alert('Logout was unsuccessful. Please try again.')
  });
  .catch(function(error) {
    // An error happened.
    window.alert('No email found. ');
    showPage('loginPage');
    console.log('no email sent');
    var errorCode = error.code;
    var errorMessage = error.message;
    console.log(errorCode + " : " + errorMessage)
  });
}

```

Figure 7: Password reset page 2 and code

Email Confirmation Page:

Sequentially, this page appears after the user has clicked 'sign me up' on the sign up page. They are prompted to check their email to ensure they were successful in signing up for the service. However, error handling should ensure this process goes very smoothly.

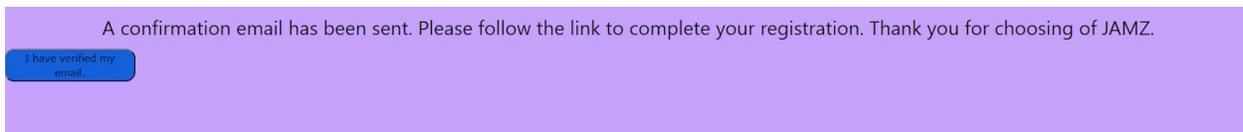


Figure 8: Email confirmation upon sign up message

Home Page:

This is the home page. It was intended to be a sort of 'fall back' page for users to always return to should they become lost or confused in the navigation of the website. Essentially, it serves two main purposes: one, to allow the user a base point to orient themselves inside the interface. In this way, the home page displays all available functionality of the application so it is clear to the user. And two, the home page also facilitates simplified

navigation and connection between all of the subsystems. It allows for flow within individual pages however also allows for variability in the given path a user takes to reach their desired goal. All home buttons on subsequent pages link back to here. It has buttons that lead to all the major features of the service.

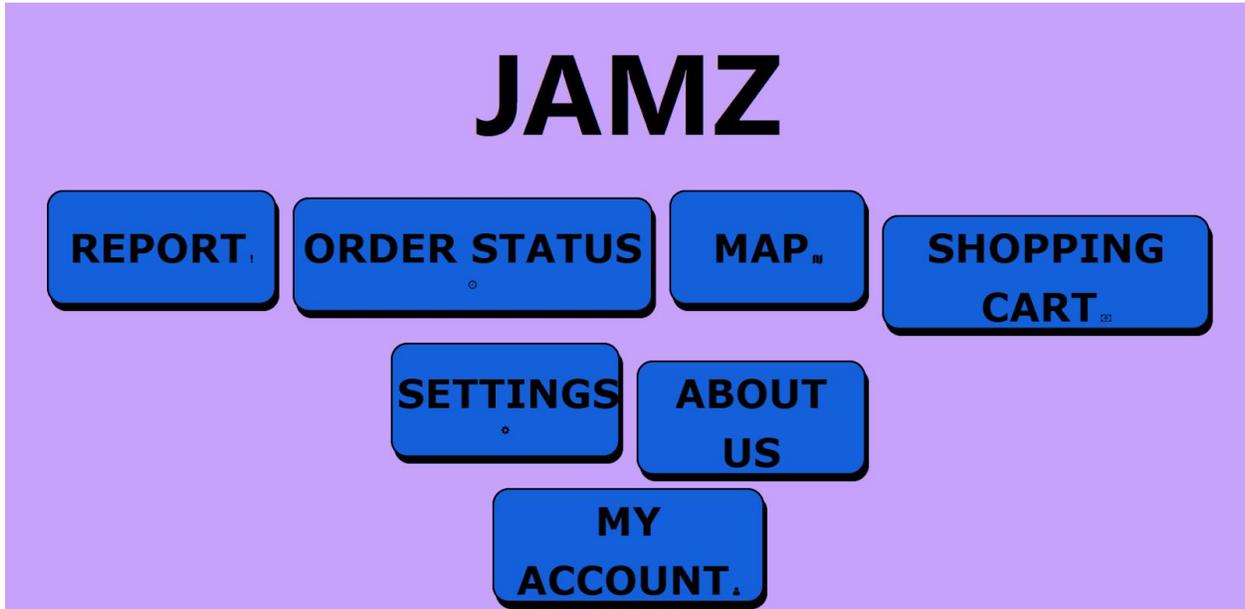


Figure 9: Home page

Terms and Conditions Page:

Users can get to this page from the sign up page should they wish to read terms and conditions before confirming they understand. The terms and conditions themselves are mostly specified by JAMZ.

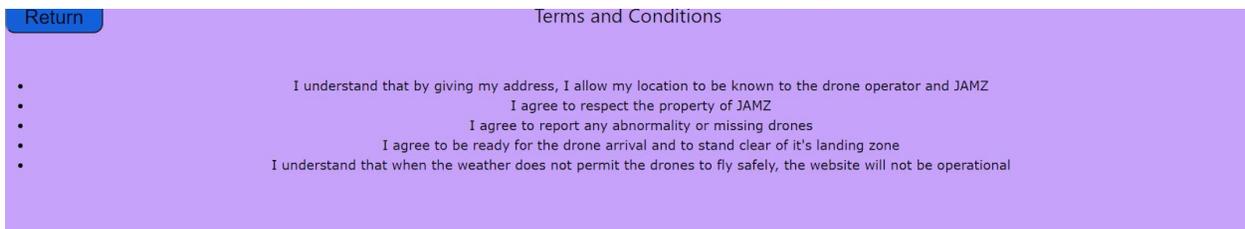


Figure 10: Terms and Conditions

About Us Page:

The information displayed on this page is very similar to what can be found on the current dummy website for JAMZ themselves. There is little to no functionality here however the main purpose of this page is to give the user more information about their provider, their goals and to create that connection.

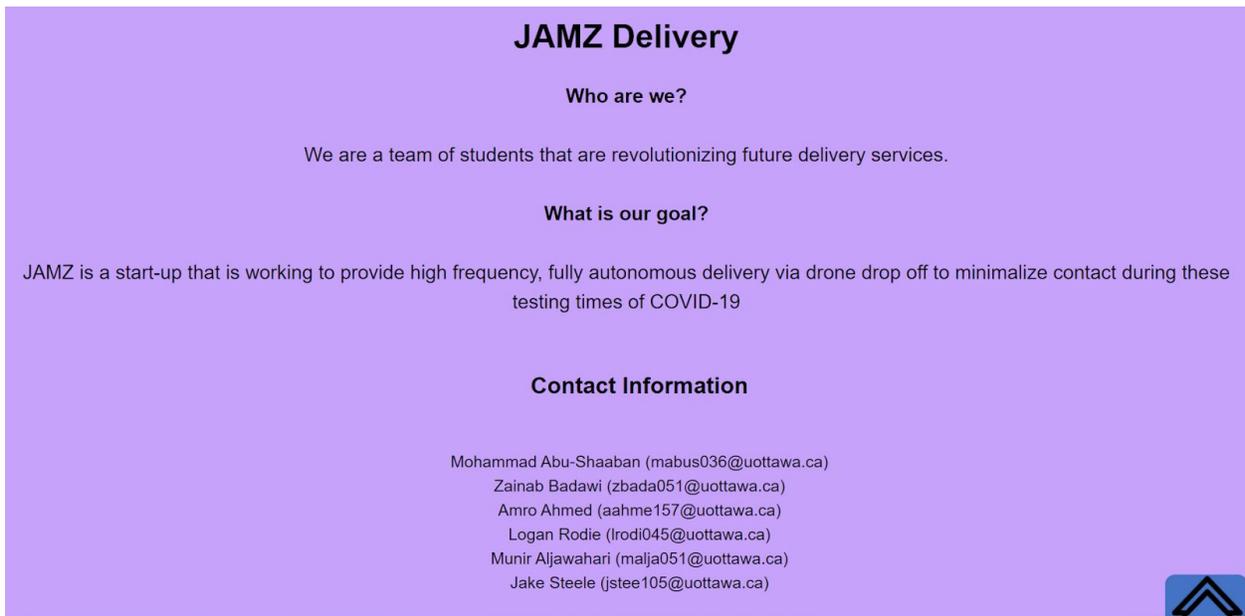


Figure 11: About us page

My Account:

This page displays the information of the user who is currently signed in to the website. If they wish to sign out, they can do so and this will return them to the login page. They can also update their information.

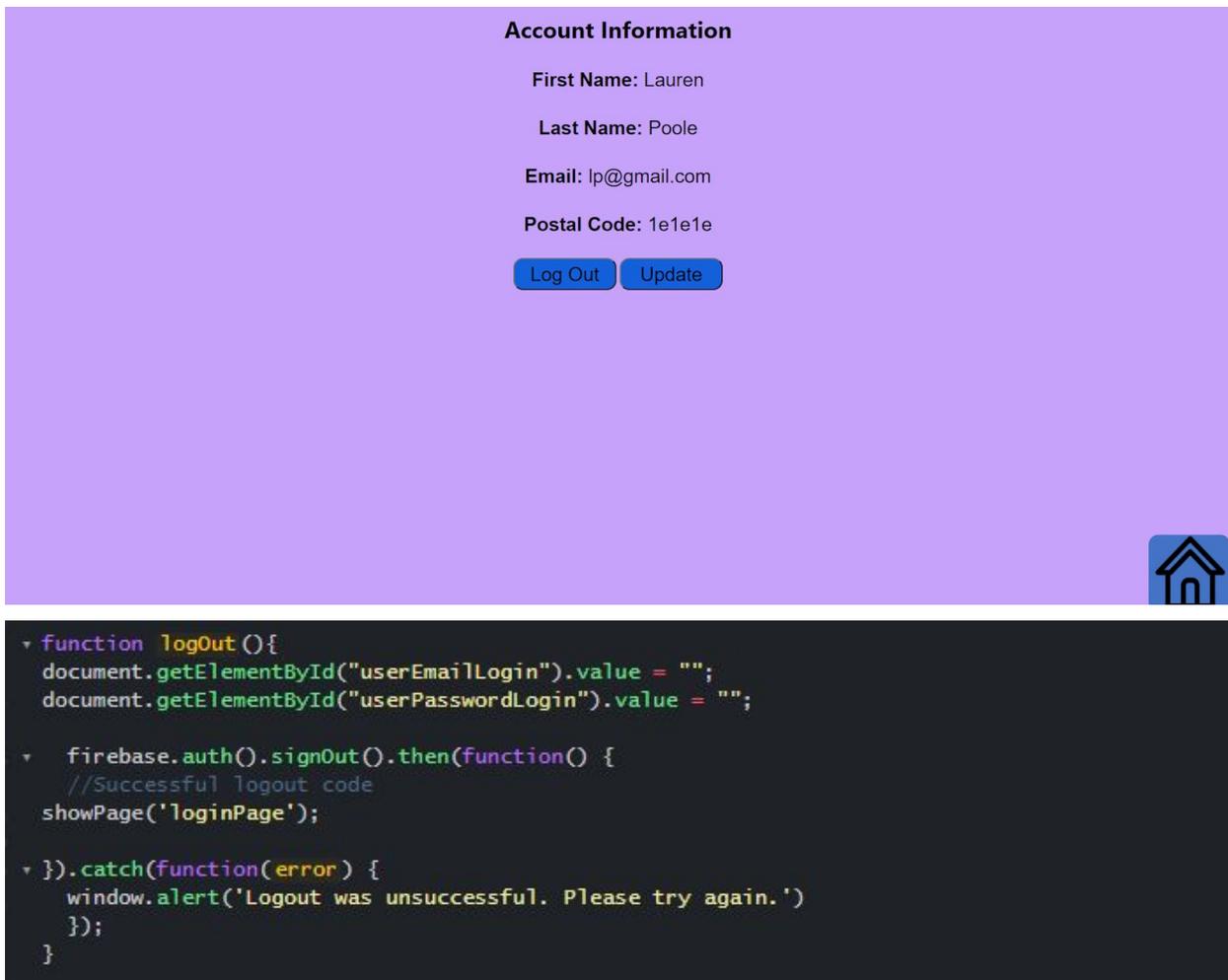


Figure 12: Account information page and code

Settings page:

This page is where the user has the freedom to control many preferences such as activating dark mode, switching languages between French and English, and personalizing the drone name. Users can also view order history and submit feedback on the food quality and drone service. Finally, users can choose to go back to the home page through the home button on the bottom right. Basic HTML and CSS styling was used to create this page, both of which will be provided in the makerrepo project folder.

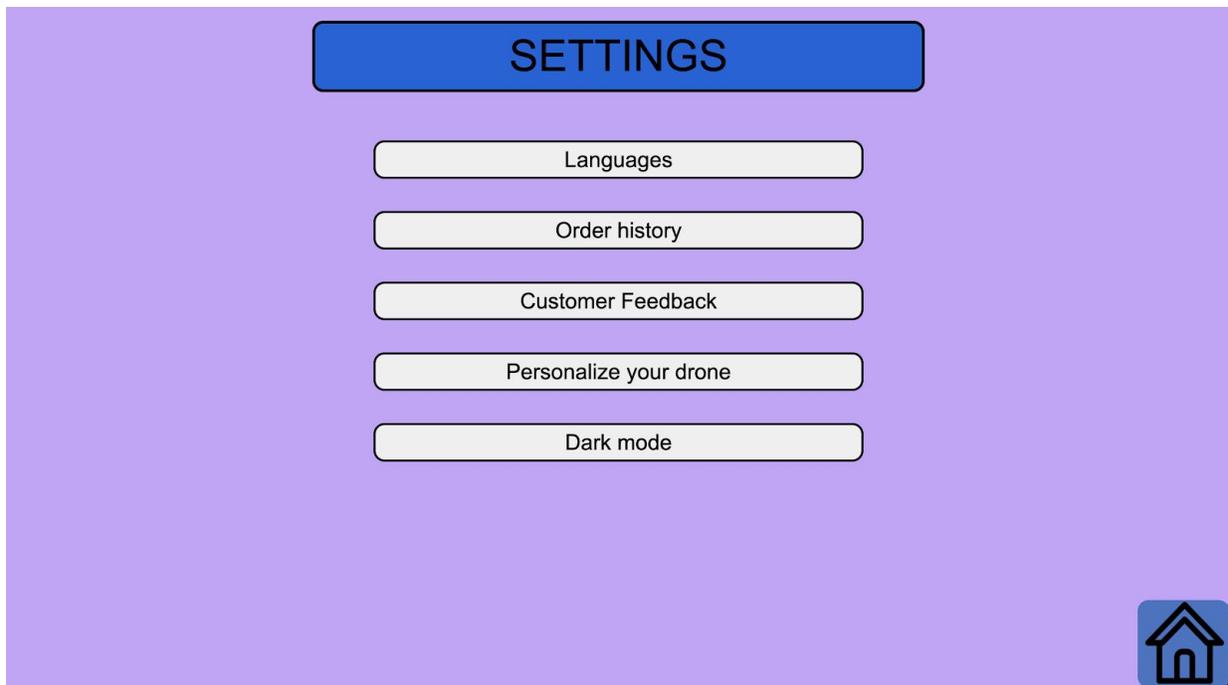
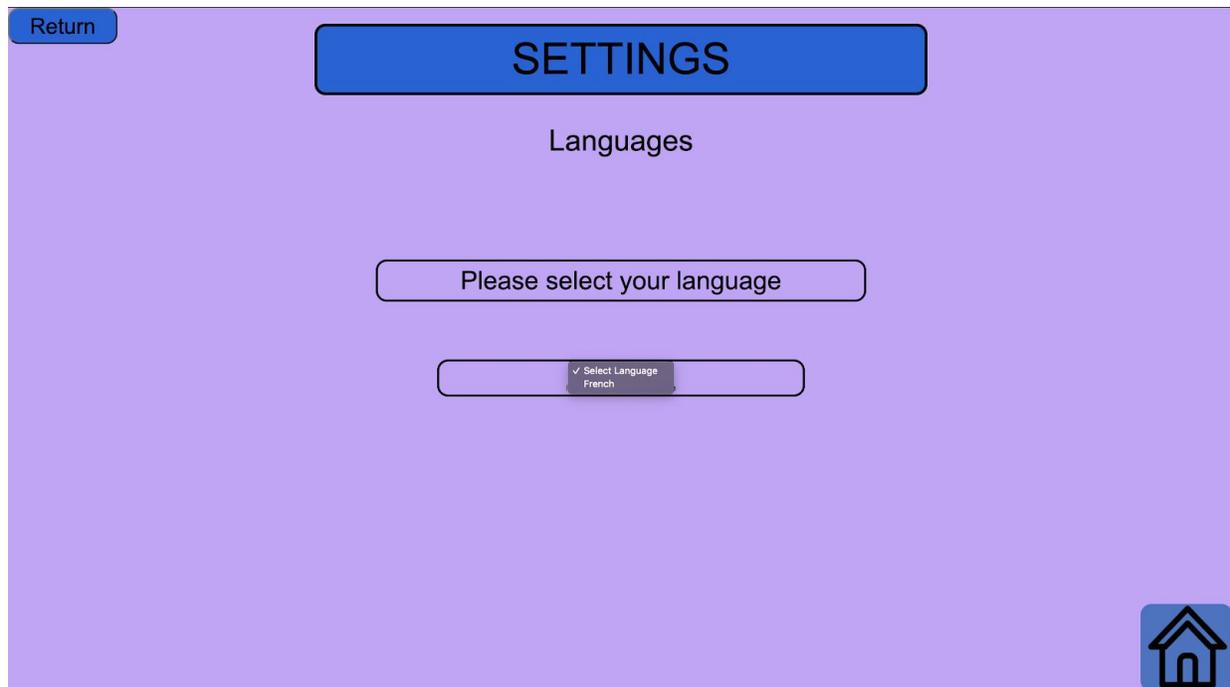


Figure 13: Settings page

Languages page:

This page is where the user can freely switch between English and French. Google translate was implemented through the use of an API key which is provided in the [resources list](#). Basic HTML and CSS styling was used to create this page, both of which will be provided in the makerrepo project folder.



```
5 <script type="text/javascript">
6   function googleTranslateElementInit(){
7     new google.translate.TranslateElement({pageLanguage: 'en',
8       includedLanguages: 'en,fr'}, 'google_translate_element');
9   }
10  </script>
```

Figure 14: Languages page and code

Order History page:

This page is where the user can view past orders along with details such as price and date. Users can also replace these orders at their own will. Basic HTML and CSS styling was used to create this page, both of which will be provided in the appendix.



Figure 15: Order history page

Customer Feedback page:

This is the page where the user can input feedback about the quality of drone delivery and food quality. When the user submits their feedback, a message will replace the text box and star rating to thank them for the feedback. The star rating was styled using css from an external [source](#). The text box and submit button were created using javascript functionality. The whole page was designed through basich HTML and CSS styling, both of which are included in the project folder in makerrepo.

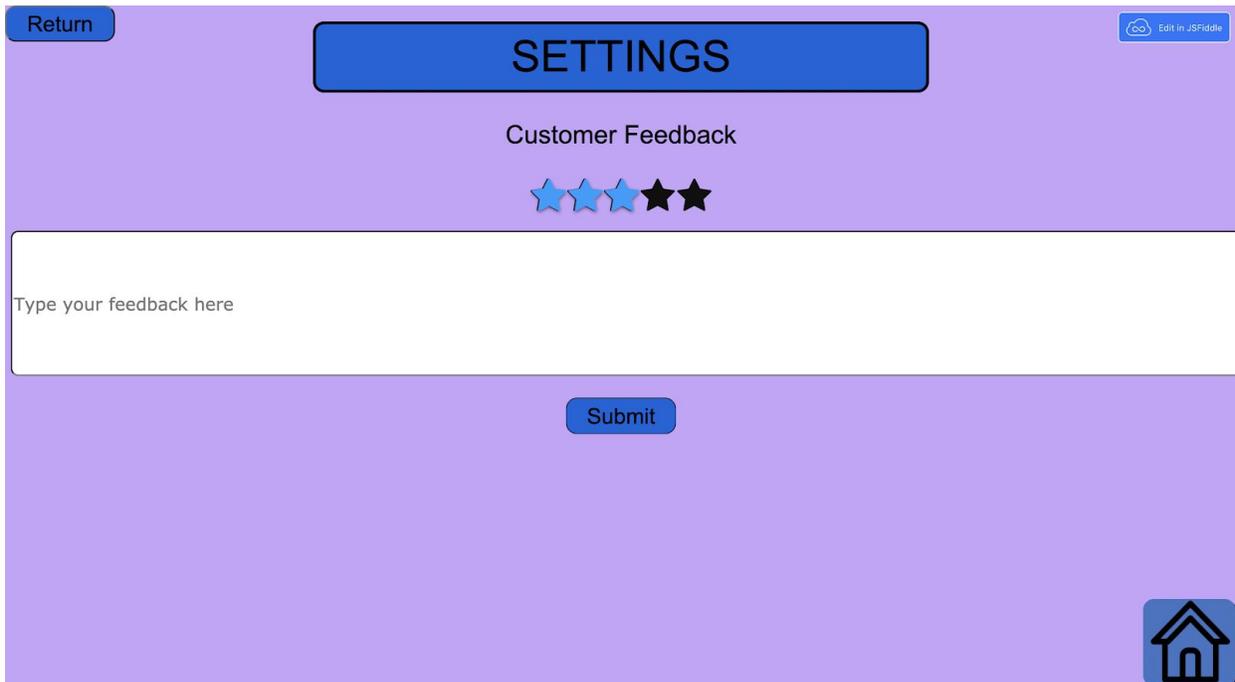
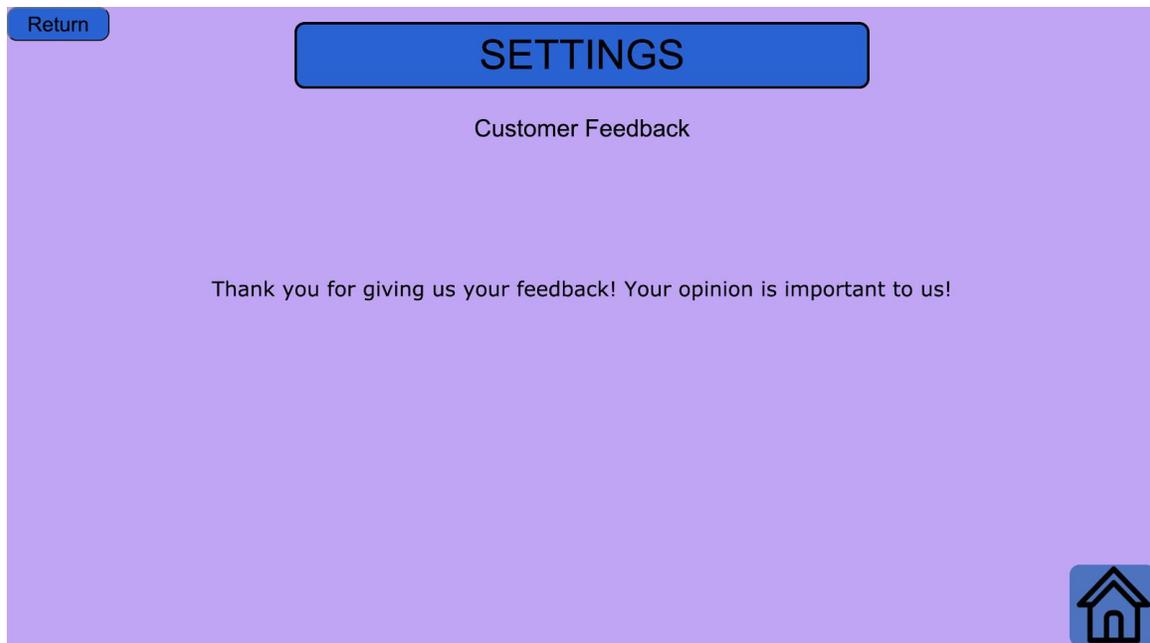


Figure 16: Customer feedback page



```
function feedbackFunction() {
  document.getElementById("rating").style.display="none";
  document.getElementById("feedback").style.display="none";
  document.getElementById("submitFeedback").style.display="none";
  document.getElementById("commentsBox").style.display="none";
  document.getElementById('feedbackSubmissionMessage').innerHTML = "Thank you for giving us your feedback! Your opinion is important to us!";
}
</script>
```

Figure 17: Customer feedback page and code

Personalize your drone page:

In this page, the user has full control over naming their delivery drone. When the user submits a name, a pop-up message will replace the text box to thank the user for submitting a drone name. The submit button uses javascript which clears the page and replaces it with a message of desired text. Meanwhile, it also stores the drone name that was submitted into firebase with the rest of the account information. Basic HTML and CSS styling was used for this page, both of which are in the makerrepo folder.

Return

SETTINGS

Personalize your drone!

Name your drone:

Type the drone name here

Submit



Figure 18: Personalize your drone page



```

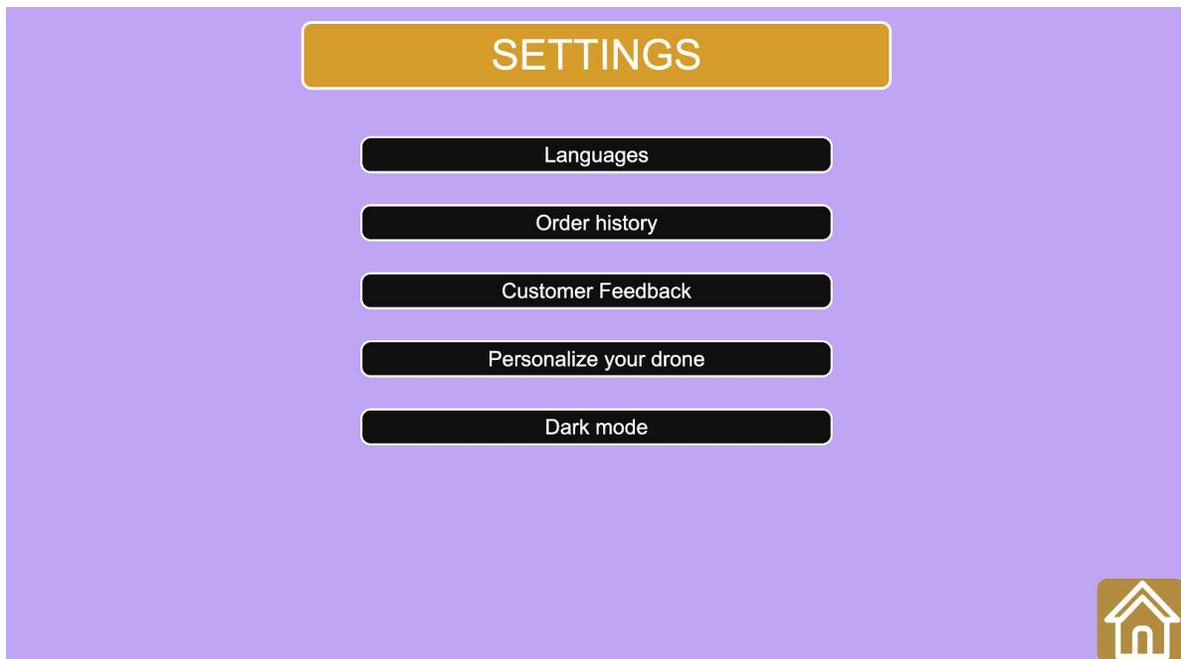
5 function personalizeDrone(){
6
7 var email = document.getElementById("userSignUpEmail").value;
8 var password = document.getElementById("userSignUpPassword").value;
9 var firstName = document.getElementById("userFirstName").value;
10 var lastName = document.getElementById("userLastName").value;
11 var postalCode = document.getElementById("userPostalCode").value;
12 var droneName = document.getElementById("droneName").value;
13
14 var user = firebase.auth().currentUser;
15
16 user.updateProfile({
17
18 }).then(function() {
19
20 let UID = user.uid;
21 //let orderNumber = user.orderNumber;
22
23 firebase.database().ref('users/' + UID).update({
24   droneName: droneName
25 });
26 document.getElementById("droneName").style.display="none";
27 document.getElementById("personalizeDroneSubmit").style.display="n
28 one";
29 document.getElementById("personalizeDroneHeading").style.display
30 ="none";
31 document.getElementById("onSubmitMessage").innerHTML = "<br>Thank
32 you and enjoy your food!";
33 }
34 });

```

Figure 19: Personalize your drone page and code

Dark mode feature:

The dark mode feature is where the user can willingly choose whether to browse the pages in dark or light mode. The button was initially fully functional. However, for unidentifiable reasons the functionality is not as exhaustive as it once was and the purple wallpaper refuses to invert. The switching between light and dark mode consists of a simple CSS line of code which inverts the colors of all elements on all the pages. The HTML code is provided in the files in makerrepo.



```
1  .dark-mode {  
2  filter: invert(100%);  
3  }  
4
```

Figure 20: Dark mode feature and code

Report page:

This page is where the user can request the phone number of the restaurant and drone operator if any misfortune occurs during the delivery. Basic HTML and CSS styling was used to create this page, both of which will be provided in the appendix.



Figure 21: Report page

Phone numbers page:

This page is where the user can view the requested phone numbers to make calls. Basic HTML and CSS styling was used to create this page, both of which will be provided in the appendix.

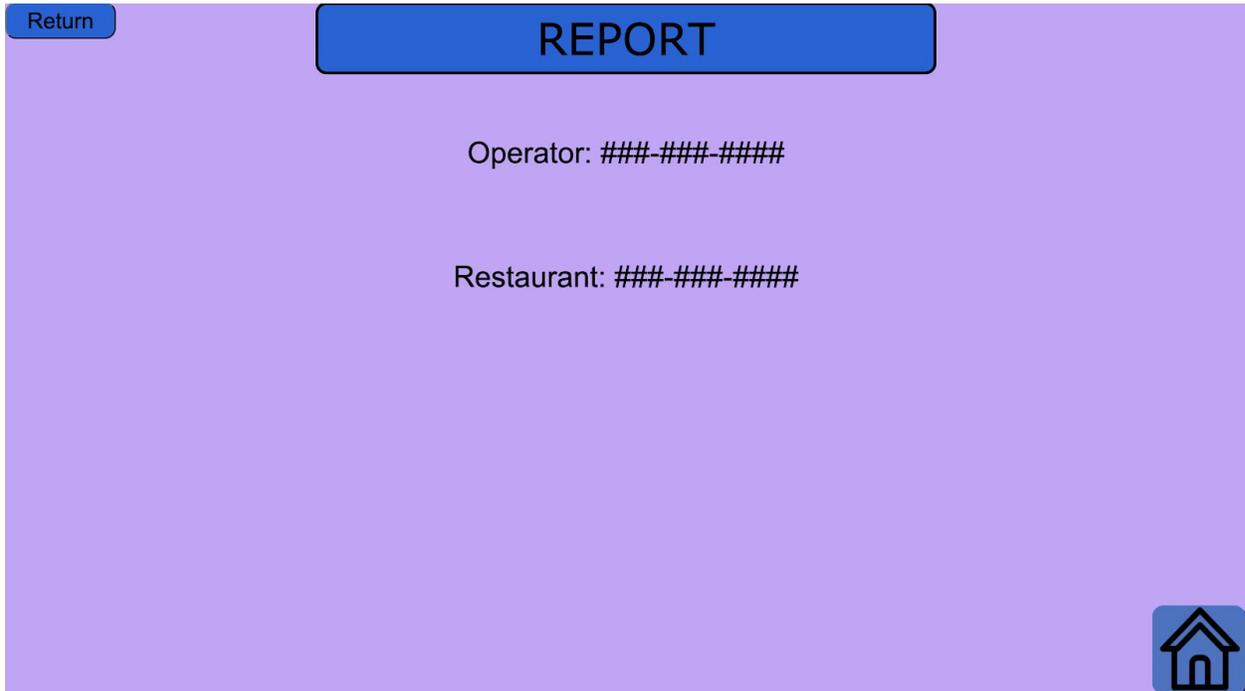


Figure 22: Phone numbers page

Order Status:

The order status subsystem is used to display the current status, a short order verification, and a QR code that will be used later on when the drone is nearby. This subsystem is accessed through either the home page or at the end of checkout when the user has finished paying for their order.

Order Status



Task	Status
Order is processing	Complete
Food is being prepared	Complete
Order is being picked up	Complete
Order is in transit	Now
Order is in the process of delivery	Pending
Order has been delivered	Pending

pelota is picking up your food! Get the QR code ready to display!

Verification

```

function displayDroneName(){
  var user = firebase.auth().currentUser;
  let UID = user.uid;

  firebase.database().ref("users/" + UID).once('value', function(snapshot) {
    snapshot.forEach(function(childSnapshot) { //Callback function
      var childKey = childSnapshot.key;
      var childData = childSnapshot.val();
      console.log(childData);
      console.log(childKey + " : " + childData);
      if (childKey == "droneName") {
        document.getElementById("dronePickUpOrder").innerHTML = childData +
        " is picking up your food! Get the QR code ready to display!";
        document.getElementById("droneOnMap").innerHTML = childData +
        " is getting your order! View current location below and get your QR code ready.";
      }
    })
  })
}

```

Figure 23: Order status page and code

This page is used to inform the user updates regarding their recent order and what is the state of the delivery. It includes a table with a description of the status on the left. On the right it will either indicate if the task has been completed, currently being completed, or still pending. An animation of a drone will show beside whichever task is currently being completed. Currently this table is not functional, the status which is shown on the right side of the table are only words that have been typed directly into the table. In order to be able to have full functionality of the table, more resources will be needed. Firstly, a software that the food establishments will use to collect orders needs to be created. This way, it is possible for the order status to change based on what the food establishment sets the status at. Also, access to the backend software of the drone will be needed. The order status table also requires information regarding the status of the delivery after the drone has left the restaurant. Updates regarding the proper delivery of the food are needed to determine if certain tasks in the table are completed or not. Furthermore, if the user has personalized their drone by naming it, this is displayed here as well .

The text below the table reminds the user to get the QR code ready. The name of the drone, which can be personalized in the settings, will show here. The default name is "myDroneName". More information and instructions come on the next page, which is the order verification page. To access, simply press the blue button below, with "Verification".

Return

Order Verification

Time

Estimated Time	15 minutes
----------------	------------

Details

Postal Code	1e1e1e
Total	\$40

Instructions

In order for the drone to land, it needs to scan the appropriate code. Follow the instructions below:

- 1) When the drone is near, open the QR code below.
- 2) Hold the device so that the code is facing upwards or place the device on the ground facing up.
- 3) Ensure you remain a safe 5 meters from the drone as it lands.

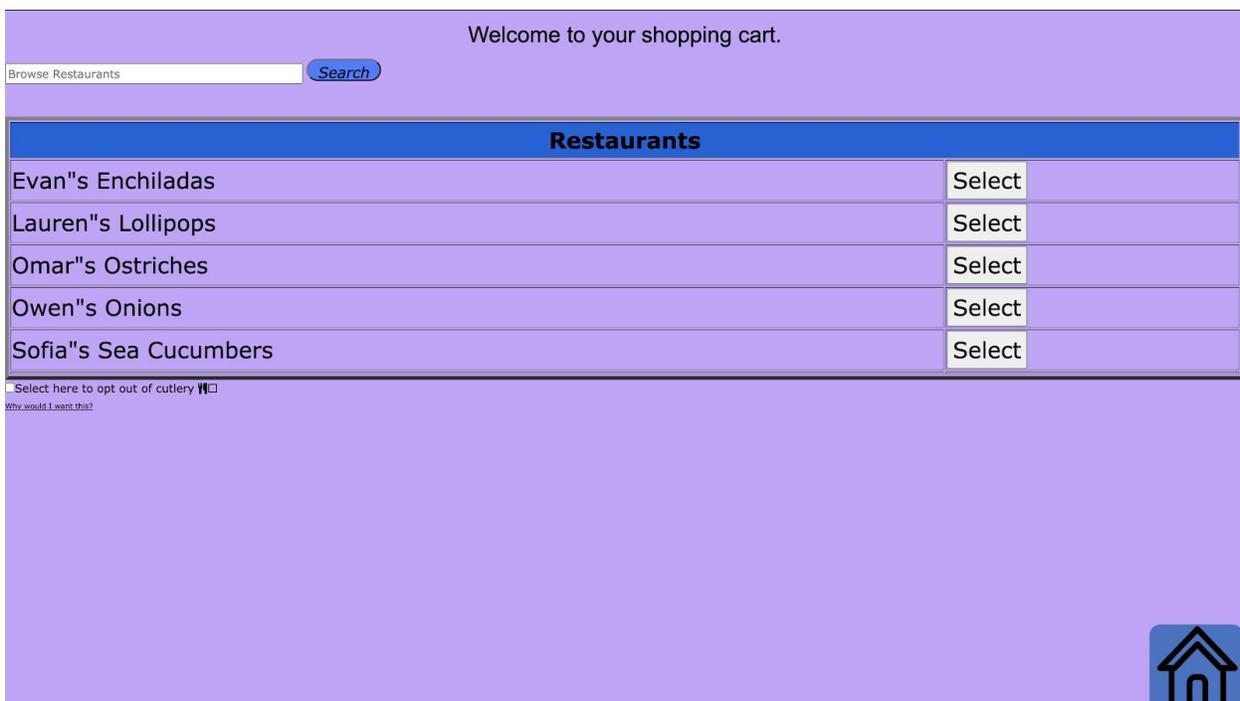


Figure 24: Order verification page

This page states additional information regarding the recent order. The estimated time until delivery is stated at the top. This is also non functional as more resources are needed to display a correct time. As mentioned above, access to the backend drone software and restaurant UI are needed to correctly predict a delivery time. Additionally, it is impossible to estimate a delivery time when there is not an actual delivery happening. Below the estimated time, the postal code is stated an additional time so the user is able to confirm that the drone is headed to the right location. Also the order total is stated. Based on the feedback received from testing, instructions are included on what to do with the QR code. The QR code is used to ensure that the food is delivered to the right customer. The customer will be asked to take the QR code to the landing location of the drone. Once the customer is 5m away from the landing location of the drone, it will proceed to scan the code. If the code matches the one in the database of the specific order, the food will be approved to deliver. The QR code currently is not unique for every order. Again, there needs to be access to the backend software to correctly match QR codes to specific deliveries. Lastly, an option to email the QR code exists. For users who are on stationary PCs, it is not convenient to bring the computer to the landing site. Therefore, this option will benefit those who are not on a laptop or smartphone.

Shopping Cart:

This is where users can place and pay for their orders. At this point, the shopping cart page is fairly functional, with all the buttons working as they should and leading to the right pages. There are some placeholder restaurant and food item names that are able to be selected and have the prices added up and tax calculated. The script for the restaurant information declaration is included below. It incorporates the data that the user entered upon signing up on the order summary page. There is an option to opt out of cutlery as well, something which the customer liked that the team had. The next steps to improve this page would be to add a backend to this website that links to actual restaurants and allows the user to order and pay for their orders. Some code can also be refined, with some bugs to be fixed, such as being able to cycle back through the site without the restaurant and food names duplicating. To make work easier in the future, it might be a good idea to switch work over to a website builder like Wix to have some pre-built templates for the shopping cart. If possible, existing code can be incorporated into a template in order to keep the website true to the original ideas that we had and that the customer required.



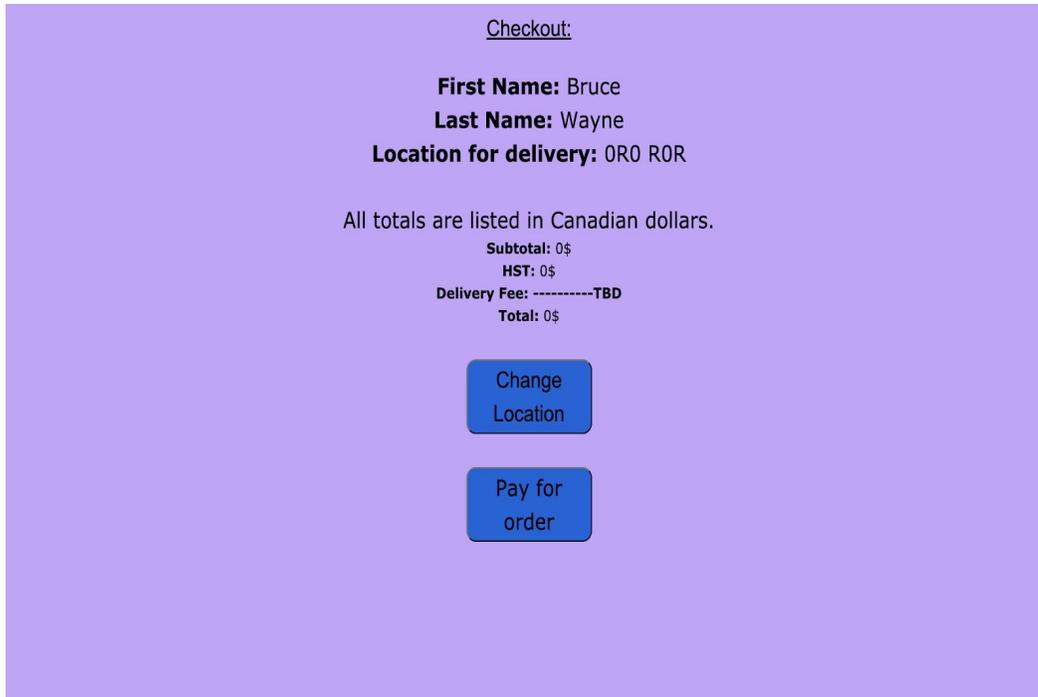
```
function initializeRestaurantData() { //This function is only for initializing restaurant data to be used
to demonstrate the shopping cart
  firebase.database().ref('restaurants/Lauren"s Lollipops').set({
    name: "Lauren's Lollipops",

  });
  firebase.database().ref('restaurants/Lauren"s Lollipops/Item_1').set({
    name: "blue lollipop",
    price: "5.00",
  });
  firebase.database().ref('restaurants/Lauren"s Lollipops/Item_2').set({
    name: "red lollipop",
    price: "10.00",
  });
  firebase.database().ref('restaurants/Lauren"s Lollipops/Item_3').set({
    name: "yellow lollipop",
    price: "15.00",
  });
}
```

Figure 25: Restaurant browsing page and code

Options			
Name	Price (\$CAD)	Quantity	
cheese enchilada	5.00	<input type="text"/>	
beef enchilada	10.00	<input type="text"/>	
bean enchilada	15.00	<input type="text"/>	

Figure 26: Menu browsing page



```

var menuTable = document.getElementById('menuTable');
var menuRows = menuTable.rows;
var i;
var subtotal = 0;
var taxes = 0;
var total = 0;
for (i=2; i<5; i++)
{
    var quantity = document.getElementById('item' + i).value;
    var row = menuRows[i];
    var price = row.cells[1].innerHTML;
    subtotal = subtotal + (price*quantity);
    console.log(price + " : " + quantity + " : " + subtotal);
}
document.getElementById('subtotal').innerHTML = subtotal + "&#36;";
taxes = subtotal*0.15;
document.getElementById('taxes').innerHTML = taxes + "&#36;";
total = subtotal + taxes;
document.getElementById('total').innerHTML = total + "&#36;";
showPage('checkoutPage');
}
    
```

Figure 27: Order summary/checkout page and code

Payment Method:

Once you finish paying for your order, you can view your the status of your order or view the current location of your drone.

Delivery Fee:

Taxes:

Total:

Your Details

Billing

Cardholder Name

Billing Address

Same as Delivery

Card Number

CVS

Expiry Date

[Place Order](#)

Figure 28: Payment information input page

Thank you for choosing JAMZ! You can view your the status of your order or view the current location of your drone.

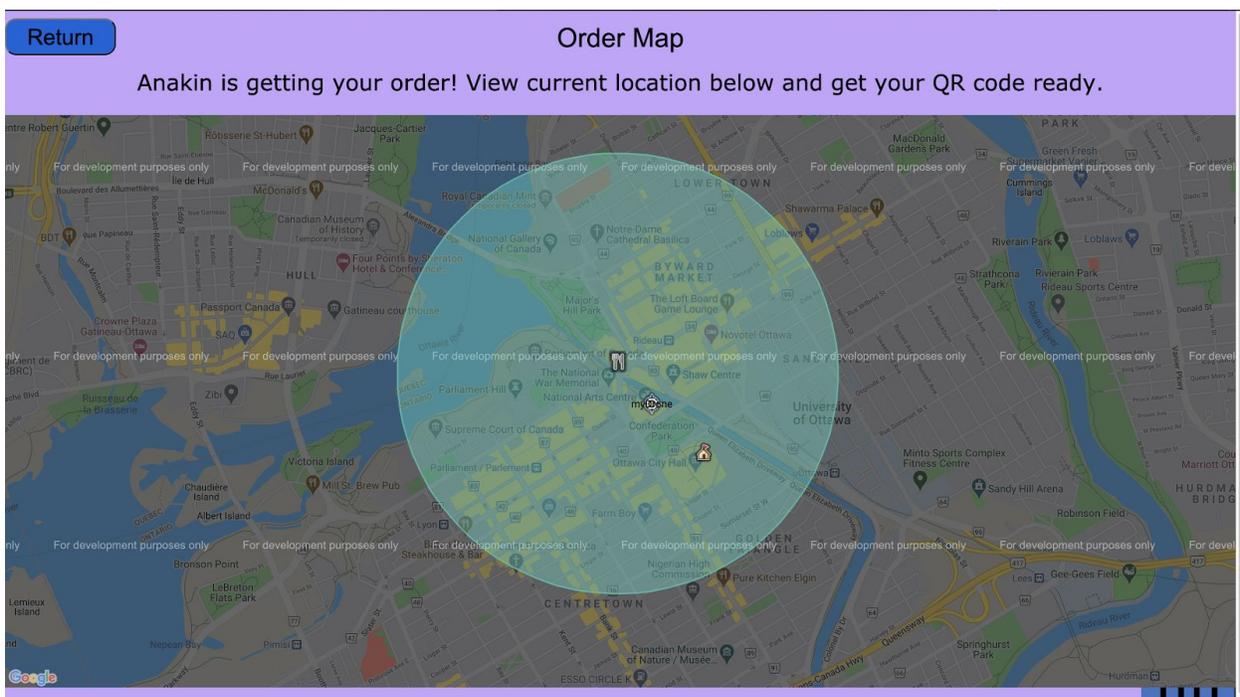
Don't forget to give your drone a personalized name in the settings!

[View Status of Order Here](#) [View Location Of Drone](#)

Figure 29: Order confirmation page

Order Map:

The entirety of this paragraph uses **Figure 30**. This is the page where users can view the map to both find restaurants in the users area, and watch their drone in real time traveling with their order. It is fully functional with the exception of actually connecting to the backend of JAMZ servers to interact with real drones. This map uses Google's API interface to display an interactive and fully functional Google Maps. It does so by using an API key which is given after you have explained what you want to use it for and is limited unless paid for. This is to be noted as the words "for development purposes only" will appear after the data used in Google's API has not been touched in a while or is not owned by the user hosting the website. This display contains a ring around the centerpoint to show the exact distance to which your drone will order food from and this distance is set to 10km, anything past this will not be an option to order food from as the drone needs to have enough battery to operate. There are also icons displaying different restaurants and places that the drone can go to order from. In the top left corner and the bottom right there are buttons that will take the user back to the home page.



```

function initMap() {
  const Ottawa = { lat: 45.424721, lng: -75.695000 };
  const drone = { lat: 45.423, lng: -75.693 };
  const userLocation = { lat: 45.421, lng: -75.690 };
  const map = new google.maps.Map(document.getElementById("displayMap"), {
    zoom: 14.7,
    center: Ottawa,
  });

  const restaurantImage = "https://maps.google.com/mapfiles/kml/pal2/icon63.png"
  const droneImage = "https://maps.google.com/mapfiles/kml/pal3/icon40.png"
  const userLocationImage = "https://maps.google.com/mapfiles/kml/pal3/icon31.png"

  const cityPin = new google.maps.Marker({
    position: Ottawa,
    map: map,
    icon: restaurantImage,
  });
  const dronePin = new google.maps.Marker({
    position: drone,
    map: map,
    icon: droneImage,
    label: "myDrone",
  });
  const userLocationPin = new google.maps.Marker({
    position: userLocation,
    map: map,
    icon: userLocationImage,
  });
  var Circle = {
    strokeColor: "#47C4B0",
    strokeOpacity: 0.8,
    strokeWeight: 2,
    fillColor: "#54ECD3",
    fillOpacity: 0.35,
    map: map,
    center: Ottawa,
    radius: 1000 // in meters
  };
  cityCircle = new google.maps.Circle(Circle)
  cityCircle.bindTo('center', cityPin, 'position');
}

```

Figure 30: Order map and code

Materials:

Images (image addresses are taken from google images):

- https://miro.medium.com/max/1424/1*sHmqYIYMV_C3TUhucHrT4w.png
- <https://static01.nyt.com/images/2017/03/14/science/14DRONE2/14DRONE2-facebookumbo.jpg?year=2017&h=550&w=1050&sig=0x8ee9a6dfff4e1c8b84dc76c8deb20b8d>
- <https://media3.giphy.com/media/SXC4WO8YO3eWQ/giphy.gif>
- <https://i.postimg.cc/pX2CDv9h/homebutton.jpg>
- "Noun Projects: Icons and Photos for Everything." <https://thenounproject.com/>

Total cost: \$1.37

- Domain to launch website from namecheap (purchased the domain 'droneontheloose.club' for one year beginning on November 28th, 2020

Table 4: Bill Of Materials and Respective Prices

<i>Item number</i>	<i>Part Name</i>	<i>Description (Identify prototype #)</i>	<i>Quantity</i>	<i>Unit Costs (\$CAD)</i>	<i>Extended Cost (Qty x \$)</i>
1	Mouse	Computer Input	1	FREE	FREE
2	Keyboard	Computer Input	1	FREE	FREE
3	jsfiddle	Coding development software	1	FREE	FREE
4	Computer	Operating System	1	FREE	FREE
5	Icons	Graphics	20+	3-10\$ per	60-100\$
6	Beta Testers	People	5+	FREE	FREE

Lessons learned:

- When carrying out an exhaustive task such as creating a website, it was found that using website builders would have greatly saved time and effort, rather than coding it all from scratch. Developing a website using website builders such as [wix.com](https://www.wix.com) or [godaddy.com](https://www.godaddy.com) would solve a lot of aesthetic and functionality requirements in a much shorter time. As well, many of our desired features would be easily implemented as the templates already exists with such tools so the extensive research which took a lot of time would be eliminated
- Effective and ongoing feedback is necessary for a smooth and successful progression of events. It was perceived that the more that frequent and detailed communication occurred, the team progressed faster. Using social media platforms such as discord.com to cooperate verbally and zoom when video was needed greatly helped with this.

- It is essential to keep track of time and ensure that tasks are completed on time. This was done by monitoring and updating the team's progress using trello.com's gantt chart feature.
- The separate files were hard to combine. The different coding styles limited the project to be combined on JSfiddle. Consistent, more scheduled/routine communication is something that needs to be done in order to effectively understand each other's parts and how everyone's work will affect each other when generating different subsystems in a team setting. This ties into the need for an overall goal when performing such a project. It can be very general but will ensure all team members have the vision and can work accordingly throughout the term.
- Using JS Fiddle was a difficult task when wanting to add pictures, features and integrating separate subsystems together. Using a different code management system would be ideal, one that allows for live updates with multiple people applying changes at once. A great example is GitHub, which in hindsight, should have been used from the beginning as it allows for all of these key features and more.
- Having prior knowledge of coding is important when trying to make several functions and subsystems collaborate. Learning using a course beforehand or whilst coding helps a lot for newer users to understand the language.
- Instead of having each team member working on a separate subsystem and integrating all the subsystems in the end, it would have been more efficient to work on the whole project together. This would save time, avoid integration problems and leave more time for testing bugs, working out kinks and refining the project to something truer to the original intent.
- Choosing to generate an app or a website is a decision that needs to be made firsthand and not changed later as the work's original intentions were to make an app. Aiming focus on this and knowing the requirements to create one from scratch would need to be fully outlined at the beginning.

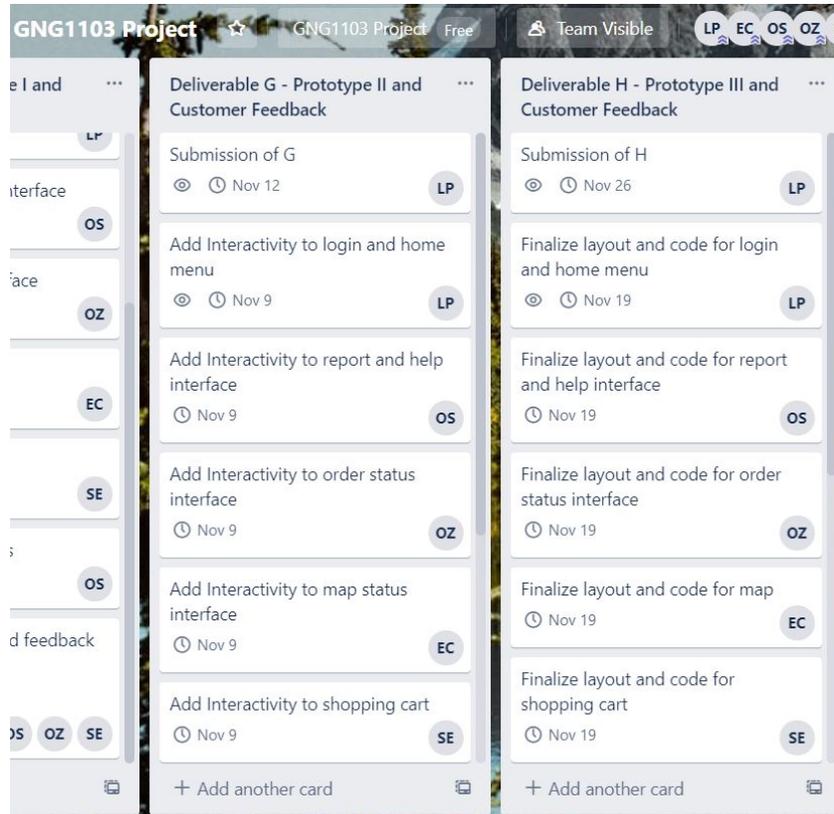


Figure 31: Trello board schedule

Resources/References:

Benchmarking resources:

"Order Food and Grocery Online from Delivery Restaurants and Groceries in UAE | Talabat." <https://www.talabat.com/uae>.

"Uber Eats | Food Delivery and Takeout - Order Online from Restaurants Near You." <https://www.ubereats.com/>.

"Order Restaurant Food Delivery Online & Take Out | SkipTheDishes." <https://www.skipthedishes.com/>.

RegEx Keys for Comparison:

Password format: `/(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,10}/`

Email format: `/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$/ig`

Firestore:

- <https://firebase.google.com/docs/auth>
- <https://firebase.google.com/docs/database/security>

Google maps API key:

- AlzaSyCifive-kREVNsu7E7yUdShzrsatGBET3A

Google translate API key:

- https://translate.google.com/translate_a/element.js?cb=googleTranslateElementInit

Customer feedback star rating CSS styling:

- <https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css>
- Video tutorial: <https://www.youtube.com/watch?v=Ep78KjstQuw>

Conclusions and Future Trials:

To sum up the project, we feel we have accomplished what we set out to do based off of the design concept. We have successfully implemented the design process to solve a complex and open ended problem. We have learned many, many lessons in terms of time management and planning as well as developing team skills and project processing strategies. The importance of communication and updating team members has become increasingly evident as well. This augmented particularly when nearing the end of the term when the deliverables became increasingly larger and more time consuming (too large to be completed by one person in one night).

There really is no upper bound to the final functionality of an interface like this. However as defined earlier, the stopping criteria provide limits as to the potential of the website, mainly in terms of time constraints. The basic features have been included and these have been judged based on the prioritized design criteria. Thus, the order map, user login and sign up as well as the shopping cart have taken the bulk of our functionality design. However,

the major improvements of aesthetics are primarily implemented by applying a combination of our user's feedback. Some of the features below are in the progress of being designed however some are out of the scope of our time frame and are documented to show future progression should we have had more time with the project.

- 1) Upon signing up, the user's location is stored in Firebase along with other information. Since this value can now be known across all subsystems, a future idea is to convert the postal code to latitude and longitude coordinates via geocoder. With these coordinates, when the order map is displayed, it could start by showing the user where they are and opening the map would find it centered at this location. A step further could include using the postal code from the restaurant the user ordered from and using its longitude and latitude coordinates to also display on the map.
- 2) As of right now, the shopping cart is only partially functional. With more time and more research, it could be completed however since these are both limited as per the team's abilities as well, the current functionality is set. There is a section under settings called order history where desired storage of past user information would be stored. When the user checks out from the shopping cart after placing an order, ideally, the price, date, items ordered and billing address would be stored (again via Firebase) and sent to be displayed under order history. As of right now, each user has an order details attribute stored and inside this, order number. A picture is shown below. The idea would be that each consecutive order per user (identified via UID) would be assigned an order number. As can be seen, each order number has price, date, location and restaurant attributes and this is the information that would need to be stored. With this, it would also be possible, as already stated to display the order information across multiple pages. For example, this would clean up the shopping cart progress because the order details would be carried over (less recall for the user).

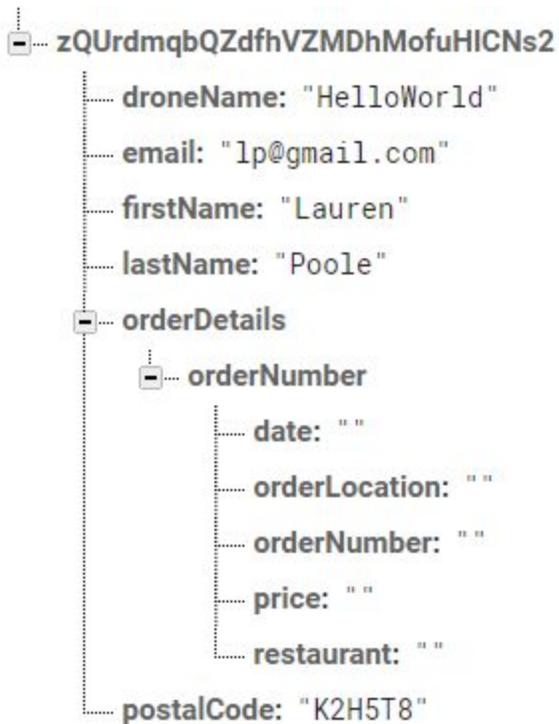


Figure 30: Details of User Pathway in Firebase

- 3) Finally, a last future plan for this project is to exempt the need for the dummy data stored in our restaurants and restaurant item selections. This can be done two ways: Firstly by searching up individual restaurant information and coding it into the required placeholders (this would be purely html however and would not be a very good use of time especially if functionality takes precedence). Second by researching/asking restaurants for their code to document how they want their options to appear on the website (this case really is for if this design would be chosen to be used for more and more real applications)
- 4) Settings pages: Since the reviews overall indicated a good feedback, major changes to the idea of these pages will not occur. However, all functionality except dark mode works. Therefore more concern will be put into implementing a different, but working dark mode color scheme. Order history would include the feature of receiving and sending orders to and from the shopping cart. Also, there is a need for a feature to increase the size of the text for users that are visually impaired. Ideally, we plan to

create varying text sizes to accommodate all users. This would be done through the learning of more complex coding.

- 5) Report pages: Since the reviews overall indicated a good feedback, major changes to the idea of these pages will not occur. However, the next step would be to include a "chat with an operator" for drone issues, and a "chat with a restaurant employee" feature would be implemented to further enhance the user experience of fixing errors.