Group C15

Deliverable G – Prototype 2 and Customer Feedback

Engineering Design – GNG 1103 – Section C

Group Members:

- **Eleftheria Sarsaroudi: 300189060**
- **Boyu Zhao: 300069815**
- **Adrian Perras: 8231683**
- **Illia Negovora: 300070880**
- **Rui Pang: 300118019**

Table of Contents

Table of Figures

Abstract

This report lays out the details of development and testing of the second prototype. Further information will be given on how testing is done and of the new components added or refined for this second stage. Customer feedback on the prototype was obtained and will be included, showing that the client had questions on the scalability and quality of the dataset that would sit behind the product, and the concerns were addressed directly to the client, as written below.

1. Introduction

In the last report, the first prototype was developed, tested and presented. After cursory testing, the first project showed that it can meet the customer's basic requirements and can run normally. It was concluded that it can meet the client's needs within budget.

In this report, further development of the project to improve the feasibility, functionality and aesthetics is demonstrated. This report will continue to analyze the design concept and operation data of the project and show any marked improvement. In addition, Customer feedback will be mentioned with the teams' responses summarized. In future work, the project prototype will be improved based on this customer feedback and continue to transform and upgrade the project prototype on the premise of reducing various risks.

2. Prototype 2

2.1. Find location

In this prototype, the range of the latitude is set from –90 to 90 and of longitude from –180 to 180. The following image shows an example of what happens when invalid values of latitude and longitude are inserted (view of terminal). The inputted value of latitude is 100, which is invalid because it exceeds the range of latitude in of the prototype, and the inputted value of longitude is –200, which is also invalid because it exceeds the range of longitude as well. Therefore, the image shows the view of the terminal and the error message at the end.

Figure 1 Invalid Input Terminal Example

The following image shows what happens when valid values of latitude and longitude are inserted (view of browser).
The inputted value of latitude is 90, which is valid because it is in the range of latitude in the prototype, and the inputted value of longitude is 180, which is also valid because it is in the range of longitude in the prototype. Therefore, the image shows the view of the browser and there is empty data because there are no registered bins in this location at the moment.

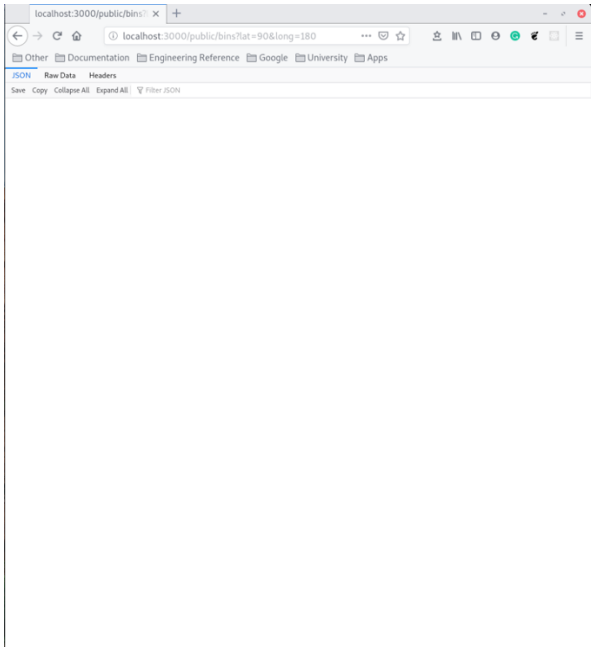

Figure 2 Empty Data Example

In order to add a bin on the server, first the appropriate information must be inserted in the system: the location (longitude and latitude), the waste type, and the colour of bin.
In the following image, the orange location icon represents the searched location, and the black location icon represents a recycling bin that matches the inserted information.

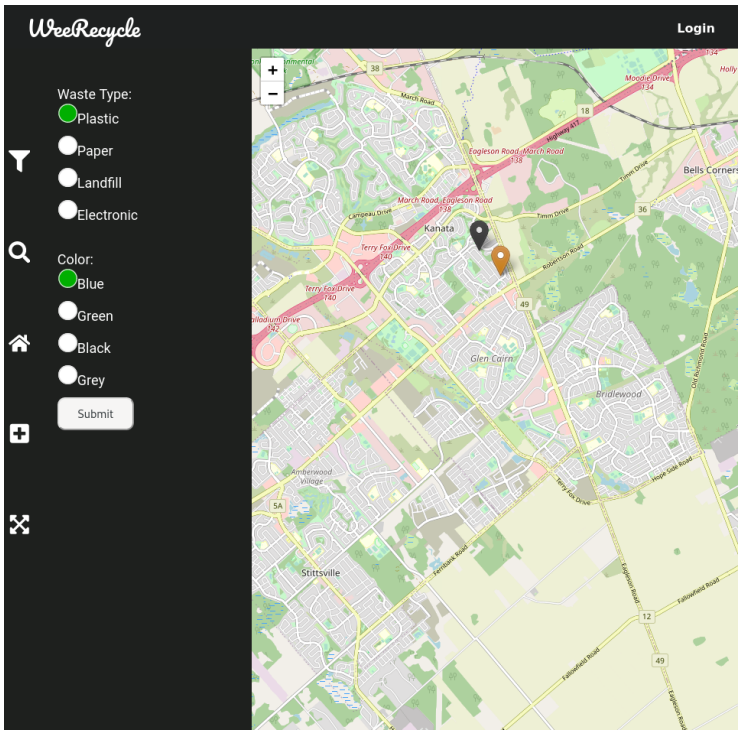

Figure 3 Searched Location Example

The system collects the data inputted (the following image is an example of the data):

```
{
  latitude: 45.30524722258941,
  longitude: -75.88034453938353,
  type: 'plastic',
  color: 'blue'
}
```

Figure 4 Data Collection Example

Then, in the current searched location, a bin may be added by clicking on the desired area in the map and is therefore added in the map/system.
In the following image, the green location icon represents the added bin, and the black location icon represents a recycling bin that matches the inserted information as well.

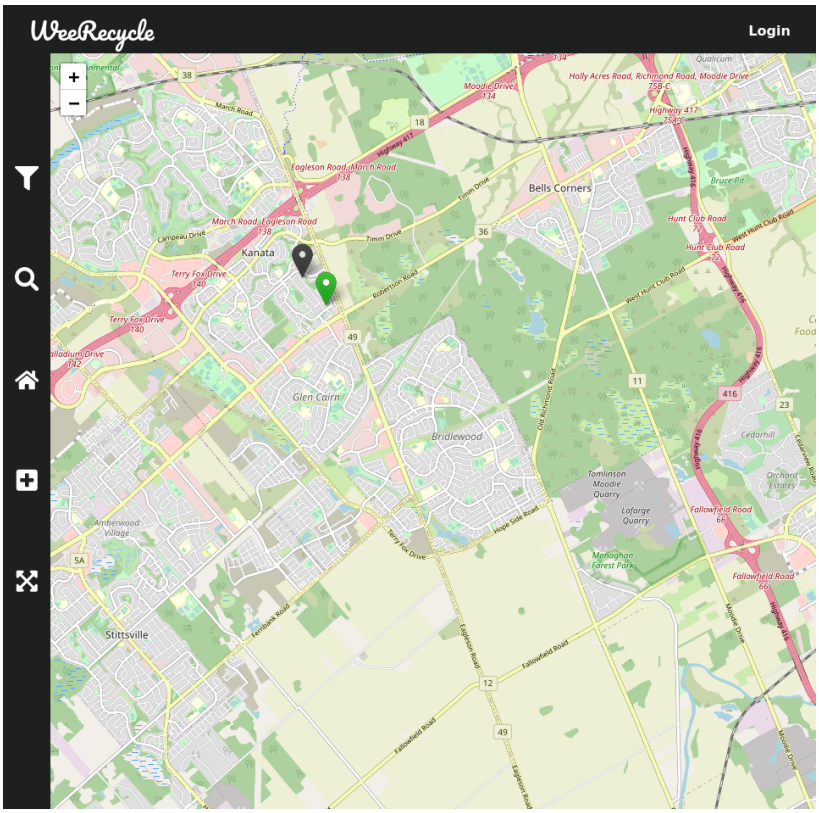

Figure 5 Add Bin Example

2.2. Full screen:

When the user wants the map to go into full screen, they just need to click on the “full screen” icon that is located on the side bar. After doing that, the top bar disappears, and the map is in full screen. The following images show how the full screen looks like:

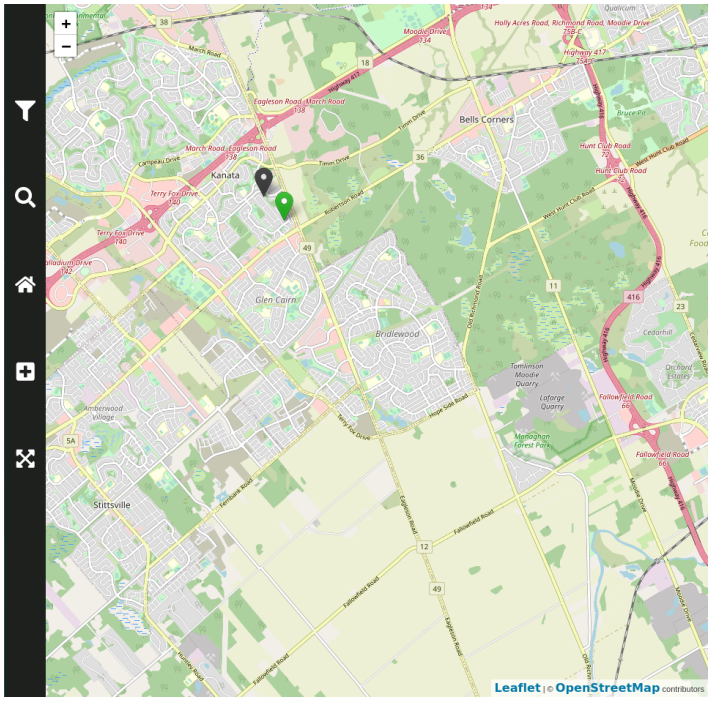

Figure 6 Full-Screen Example

2.3. Return to home:

When the user wants to return to the home screen, they just need to click on the “home” icon that is located on the side bar.
The following images represent the “return to home” function. The following image shows the user’s current map view after searching a recycling location, and the next image shows the home view after the user has clicked on the “home” icon.

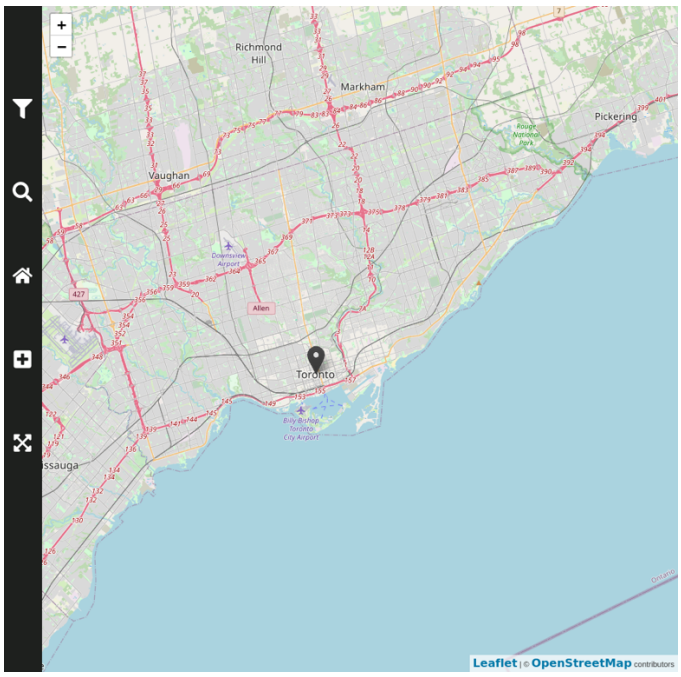

Figure 7 Return to Home Example 1

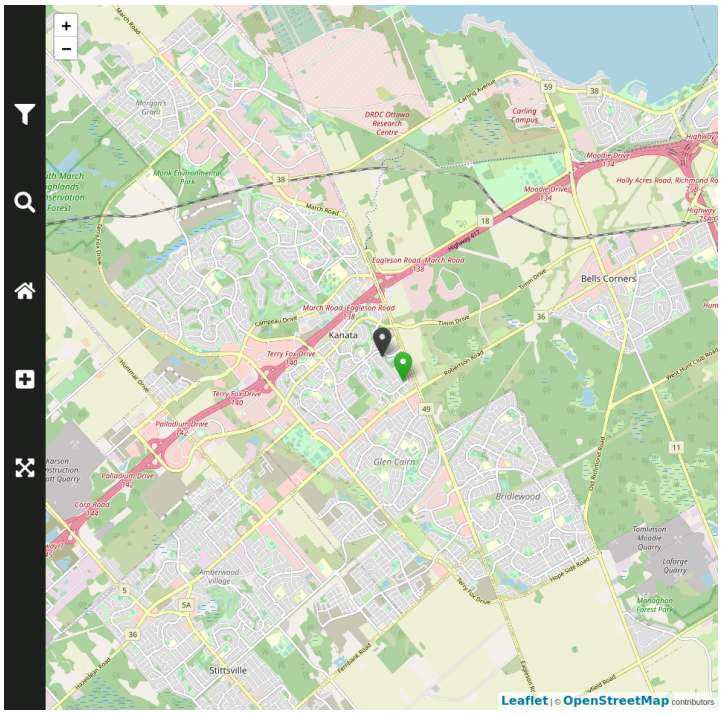

Figure 8 Return to Home Example 2

3. Test Plan

3.1. Why are we doing these tests?

3.1.1. Test objectives

The purpose is to be able to automatically test the integrity of the system, after any change made to the code base. In addition, manual testing will be used to test the overall integrity of the system. In general, the types of tests that will be performed for the prototype are unit testing (using Jest Framework) and manual testing. Unit testing is used to assert expected output under defined input of isolated functions. Manual testing is used to test the overall system (from request to respond).

```
const sum = require('./sum');

test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```

Figure 9 Unit test performance example

In Prototype 1, we tested if the radius function works properly (find location). In Prototype 2 we want to test out the functions of adding a recycling bin, putting the map into full screen, and returning to the home screen. We will also test the location function just in case an error occurs because of the added functions.

3.1.2. Possible types of results

3.1.2.1. Find location:

As mentioned in Prototype 1, the user needs to insert valid coordinates to find a location. Coordinates have boundaries. So, if the function is developed correctly, then no errors will occur when acceptable latitude and longitude is passed to the function. When invalid latitude and longitude is passed, we expect the server to respond with an error message.

3.1.2.2. Adding recycling bin:

When this function works, a bin can be added to the desired location on the system's map with no glitches. In case if bin cannot be added to the database by the server, browser console will print the error. Currently there is no error message on the UI.

3.1.2.3. Map into full screen:

When the function works, the user will click on the "full screen" icon and the top bar will vanish and therefore, the map will enter full screen. The feature does not rely on database, and therefore no error can occur during hiding and displaying process. This feature is yet to be tested on mobile devices.

3.1.2.4. Return to home screen:

When the function works, no errors will occur. The user will be able to return to the home location (current geographical location) by simply clicking on the "home" icon. If this function fails (geolocation of person cannot be obtained), location will not be changed.

3.1.3. How results will be used to make decisions or select concepts

Positive results will certainly increase the functionality of the prototype, but even if the results are errors, we will know what the system can handle and can therefore take appropriate actions.

3.1.4. Criteria for test success or failure

Whenever unit test or manual test fails, this indicates that the system has a serious bug which has to be fixed. Also, the feedback from the client is critical. If the client does not like thefinal prototype, then we are not meeting the expectations. In the recent client meet, the client liked what we presented and did not suggest anything to add to the prototype.

3.2. What is the prototype and what is the test?

3.2.1. Type of prototype and reason of selection

This prototype is *comprehensive* because 4 functions were tested out and for the first time all together: finding location, adding bin, entering full screen, and returning to home page. The

map is considered the “heart” of the prototype since the goal is to help users identify their recycling waste and find recycling bins near them. All the current functions are focused on the map.
Because we had more time to prepare and complete this deliverable, a *comprehensive* prototype was chosen.

3.2.2. Description of testing process

For the testing of the radius (find location), manual system testing was done. Http request with invalid parameters was sent to the server and expected error was return. However, when http request with valid permeameters was sent, the expected result was received (bins within the radius were fetched). For the rest of the functions (adding bin, full screen, and return to home screen) manual system testing was done as well. By applying manual testing, it is possible to determine immediately if the functions successfully work.

3.2.3. Required materials

Node.js, Express Framework, ReactJS, and other open-source libraries.

3.2.4. Approximation of estimated cost of the prototype

As mentioned in Deliverable E and Prototype 1, any material used for the prototypes is free because this is a student project and all the remaining technologies involved are free and open-source software.

3.2.5 Work to be done (Testing, construction, modeling work, research)

All the required technologies are researched, and the software architecture is modelled. API subsystem is capable of fetching bins within the radius and adding new bins to the system. UI subsystem can display the map, overall user interface and add fetched bins on the map.

3.3. How is the prototype used?

3.3.1. Recorded information

The behavior of the system, and failure of isolated function, or complete features.

3.3.2. How the results will be recorded

The result will be logged in console and it will be clearly seen when the system fails.

3.3.3. Is this important data for the project?

The radius is not functional part of the project, but it indicates the overall quality of the software and functionality of certain features.
Fetch and add bins are functional components of the system, and therefore testing these features will represent the overall quality.
While UI is an important part of the system, the entire UI is only a gateway to API. Therefore, failures of the UI do not indicate overall failures of the system.

3.4. When is the testing happening and how long will it take?

3.4.1. Duration of test and dependencies

The unit tests must be developed before they can run. In this case, test cases will be developed using jest framework before after or during the functional elements of the system are developed. The manual testing will be developed after every new feature is added.

3.4.2. When are the results required

The results from the testing are required before the due date of this deliverable, which has been extended to March 21st 11:59 pm. First, the results need to be gathered and then are added in this deliverable.

3.4.3. Test planning Gantt chart

For the protype stages, the team is still following the same Wrike Gantt chart that was created in Deliverable E and used for Prototype 1.

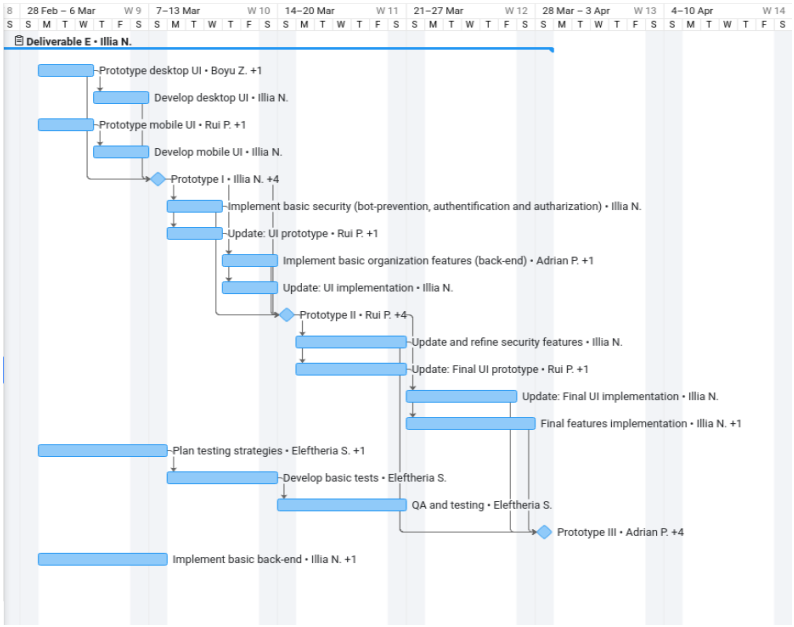

Figure 10 Gantt Chart Sample for Prototyping Stages

A different Wrike chart though has been created to finish writing this specific deliverable (submitted in separate PDF document).

4. Client Feedback

The prototype was demonstrated to the client in the recent client meet, giving the chance for questions and feedback on the current design and implementation. The client had no negative comments and did not suggest any improvements, which shows confidence in the current approach / direction for the product. Some Questions were asked, to see how well this product could handle certain future scenarios, helping guide the later development and hint at extra requirements for the product.

First question was: “What happens if a lot of users try to access the site simultaneously? “, to which the robustness of the solution was further explained. The current implementation could handle roughly 150 simultaneous requests, with the ability to scale to 1000s or more simultaneous requests with a server cluster model.

Second Question was: “How can you prevent users from adding fake bin locations on the service’s map? “, To which the next stages of security implementation in later prototypes were explained. Implementation of session-based or JWT-based authentication. If any visiting user is to be authenticated, restrictions can be imposed on them (for example add bin only within certain radius or add only n number bins per hour). In addition, users can be banned from adding more bins.

5. Conclusion

Not much change to the overall structure or feature-set of the project, instead with a focus on cleaning up code and working on understanding how to approach thorough testing of the code in a time-effective way. Further work was done for the full screen, add bin, and return to home features, through making code more readable and getting it ready to unit test. Though these don’t make for directly noticeable changes, these are very valuable considering the product is the API itself (code readability and ease of modification is extremely important). Further work on understanding how to do testing, and some simple implementations of unit-testing were highlighted. Using the customer feedback, it can be assured that future prototypes demonstrate the true capabilities of the product that the client was curious about.

6. Appendix

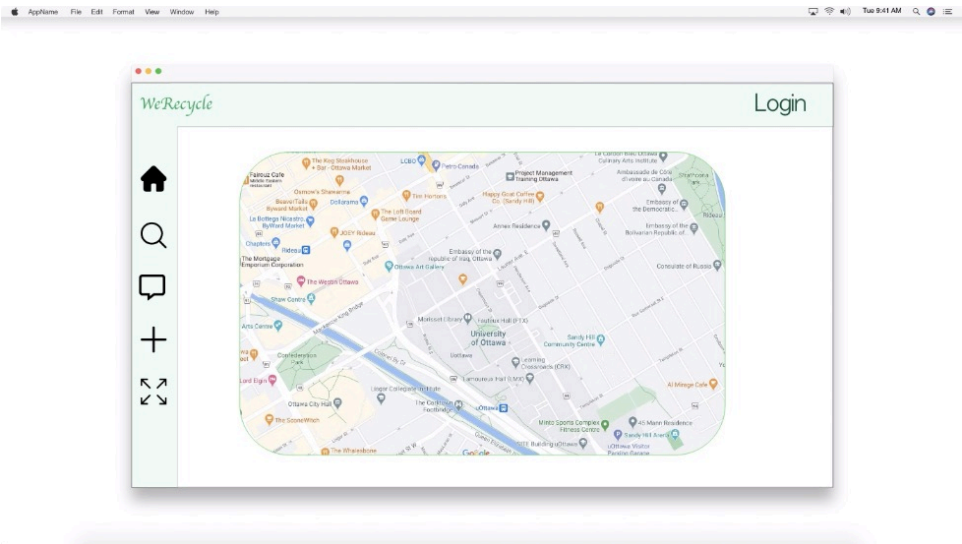

Figure 11 macOS-Main-1

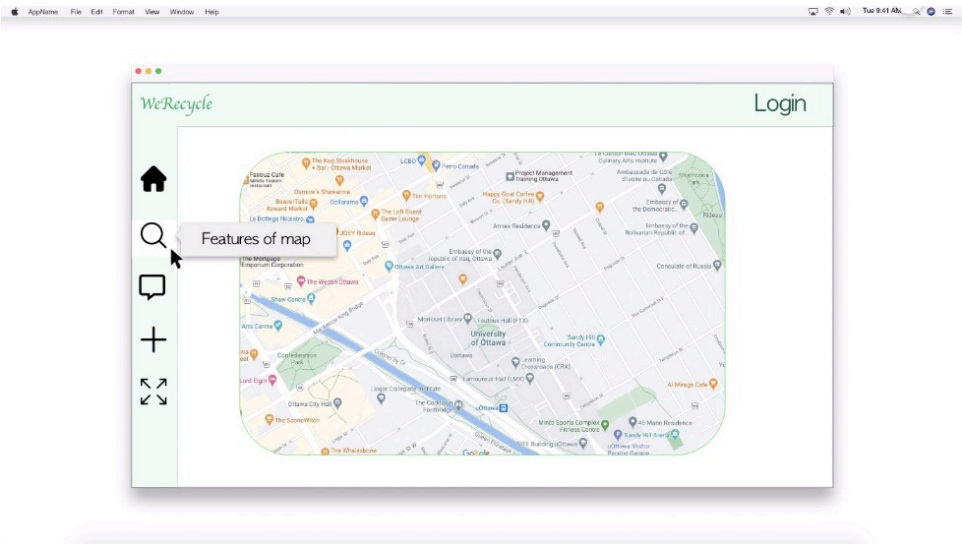

Figure 12 macOS-Main-2

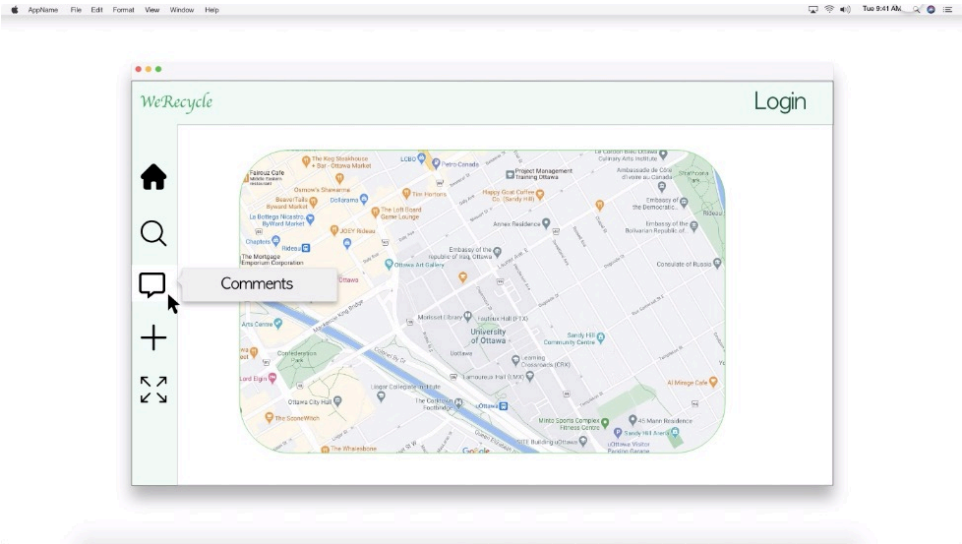

Figure 13 macOS-Main-3

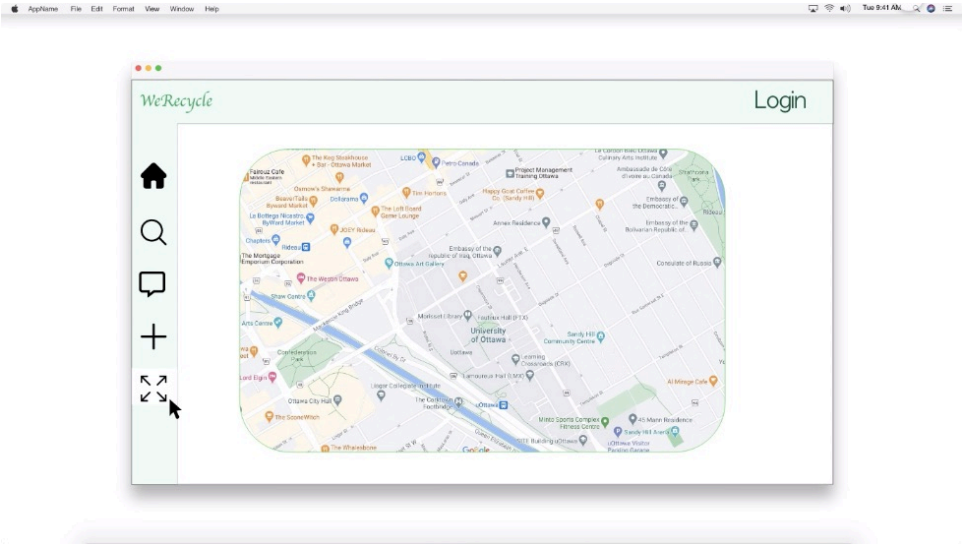

Figure 14 macOS-Main-4

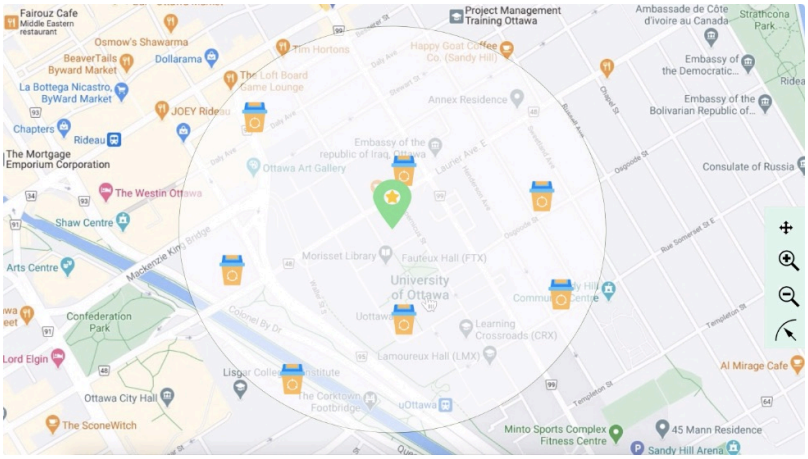

Figure 15 macOS-Main-5

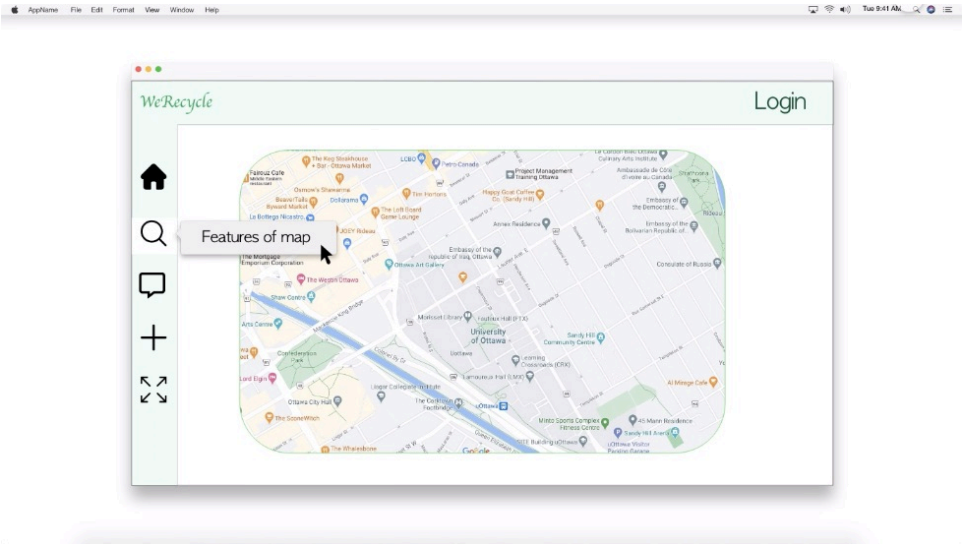

Figure 16 macOS-Find Bin-1

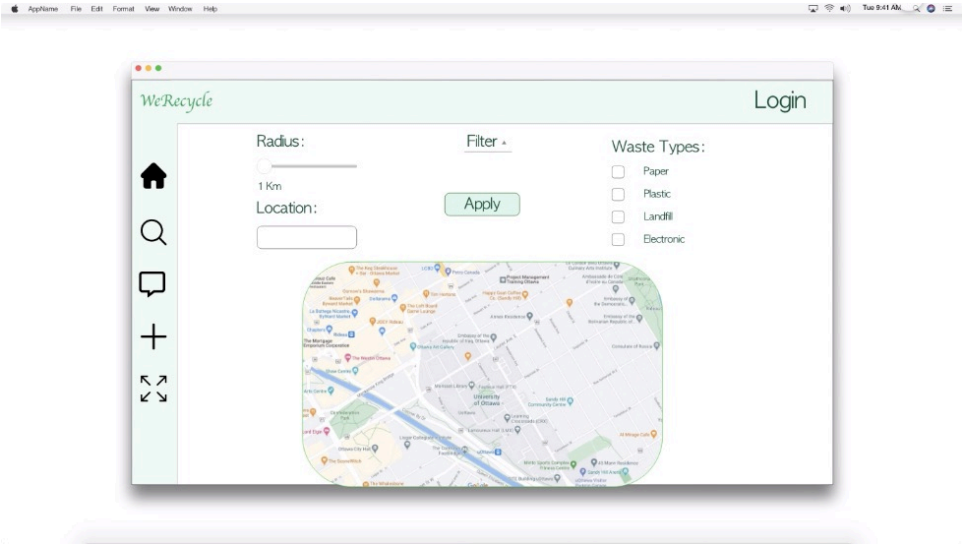

Figure 17 macOS-Find Bin-2

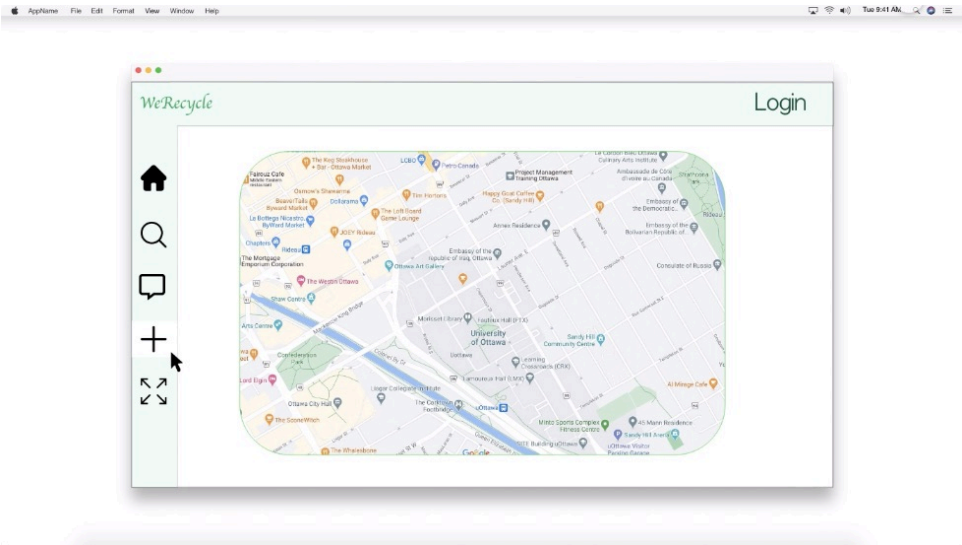

Figure 18 macOS-Add Bin-1

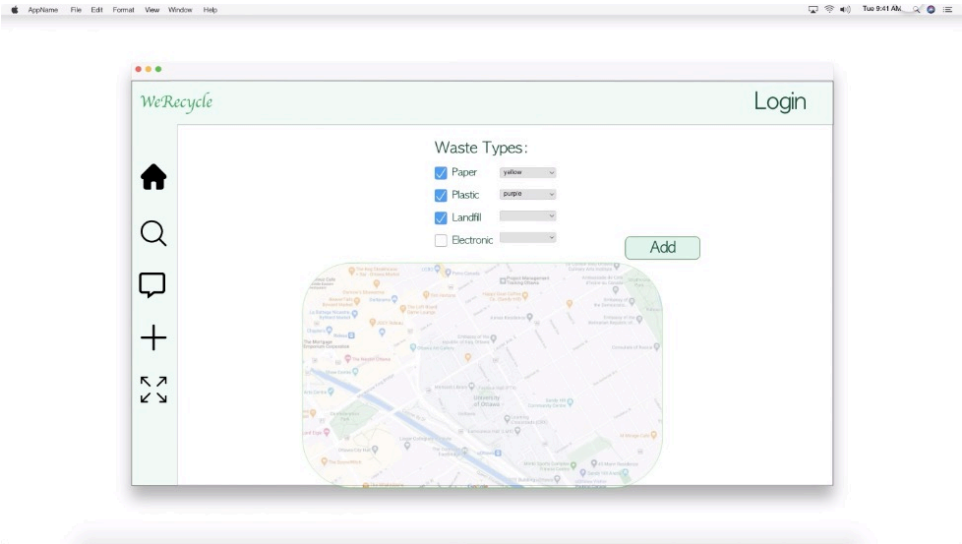

Figure 19 macOS-Add Bin-2

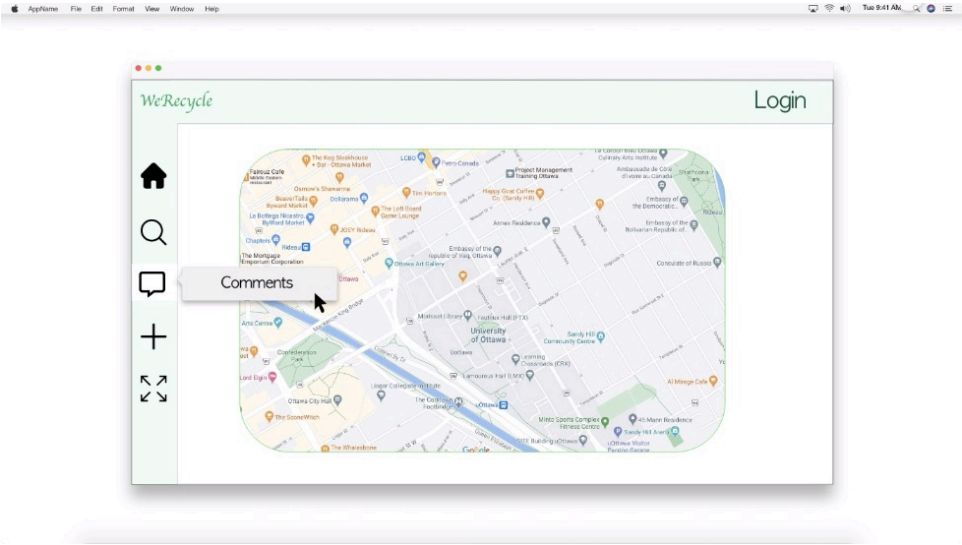

Figure 20 macOS-Info-1

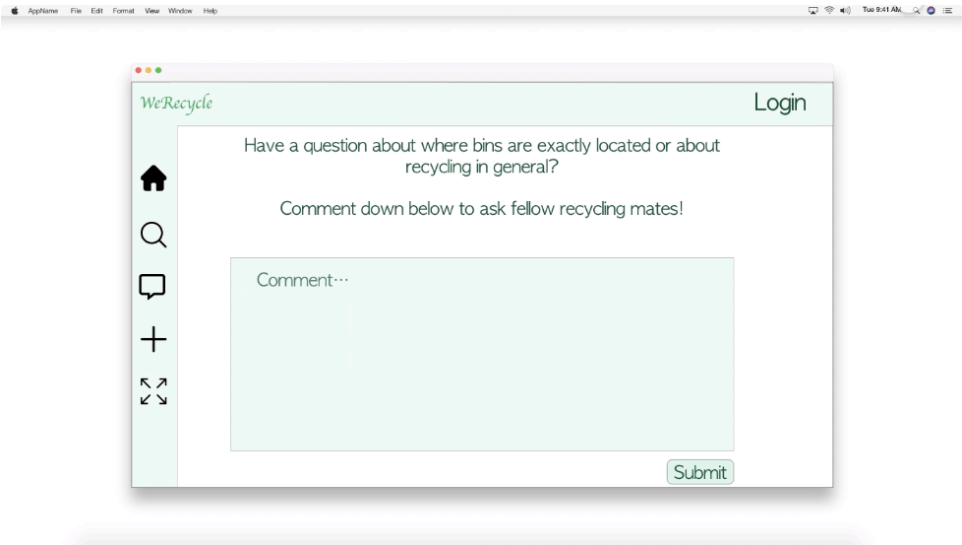

Figure 21 macOS-Info-2

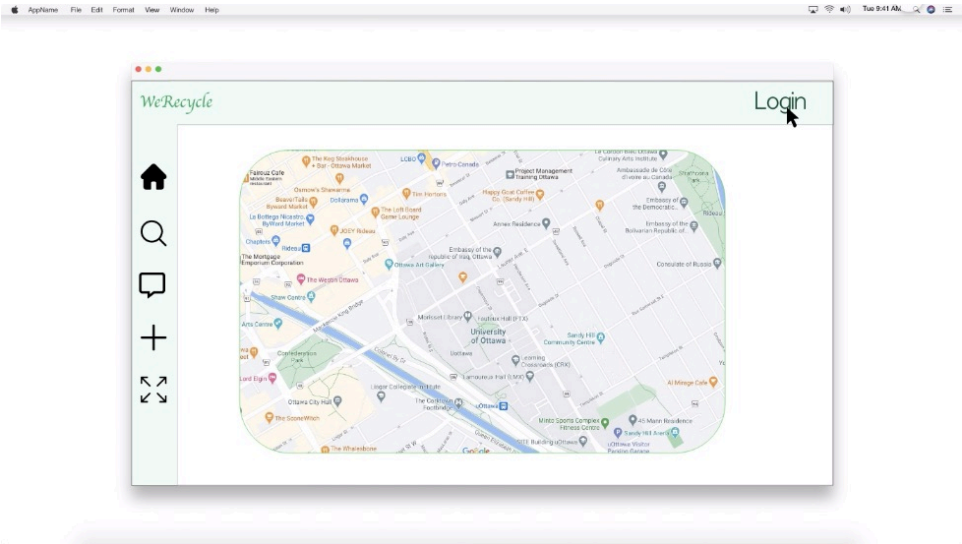

Figure 22 macOS-Login-1

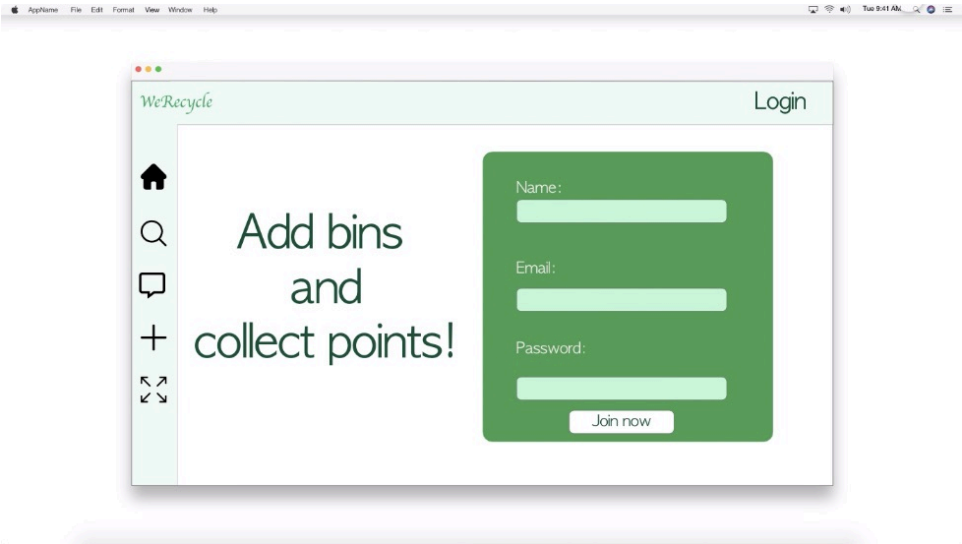

Figure 23 macOS-Login-2

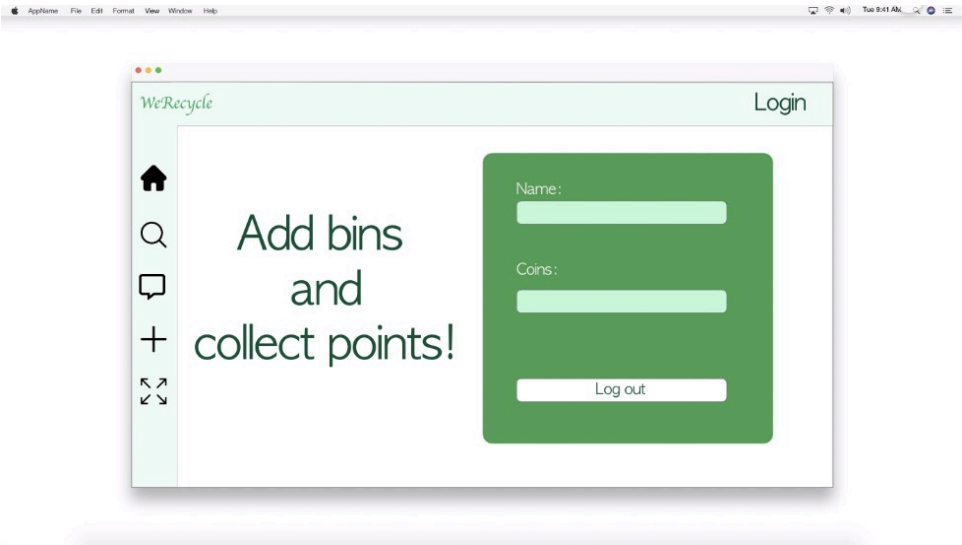

Figure 24 macOS-Login-3

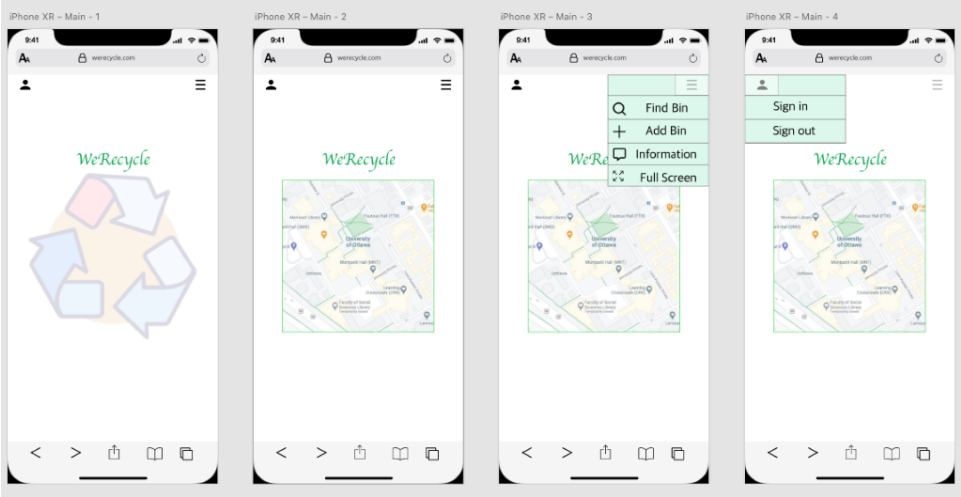

Figure 25 iOS-Main

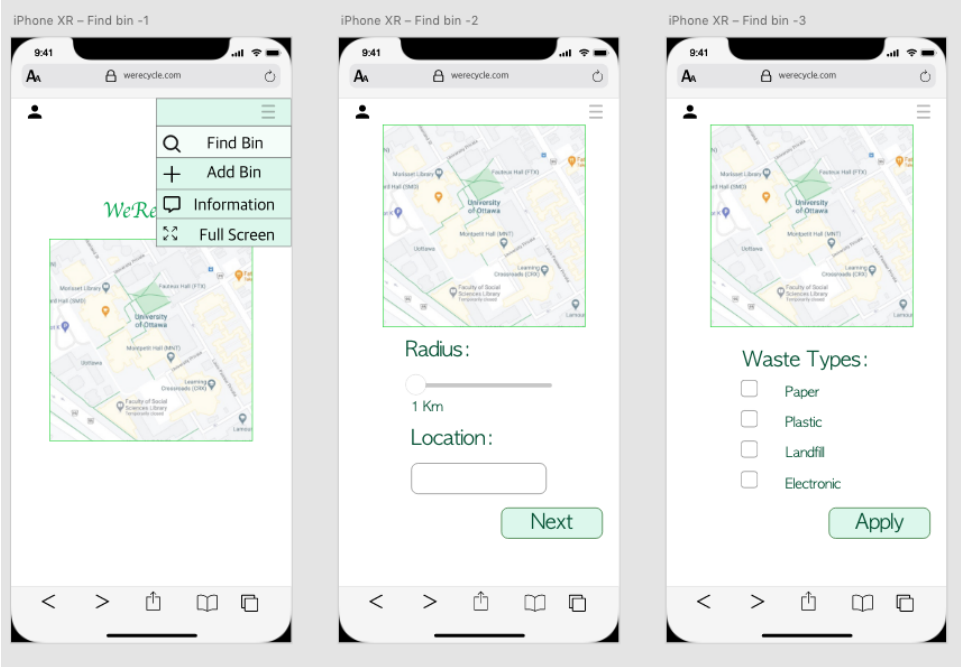

Figure 26 iOS-Find Bin

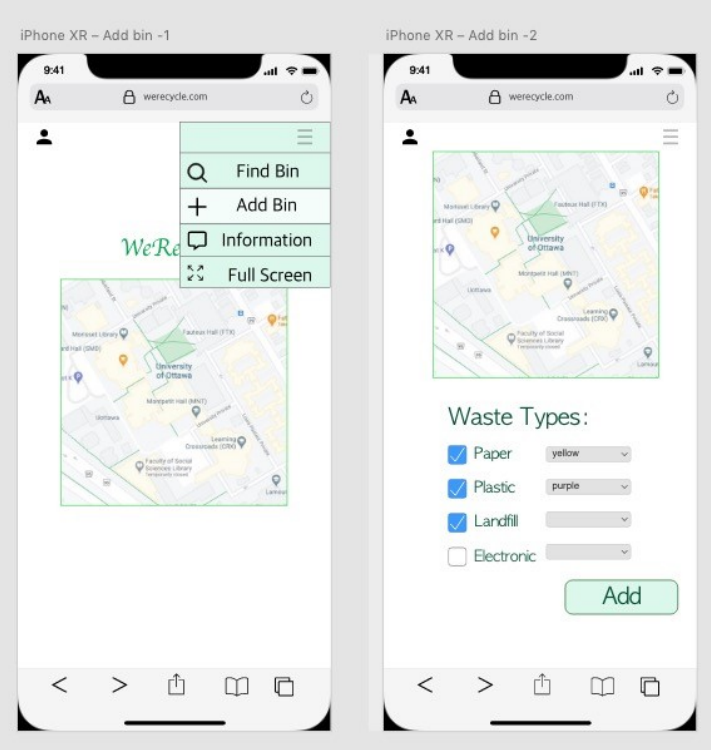

Figure 27 iOS-Add Bin

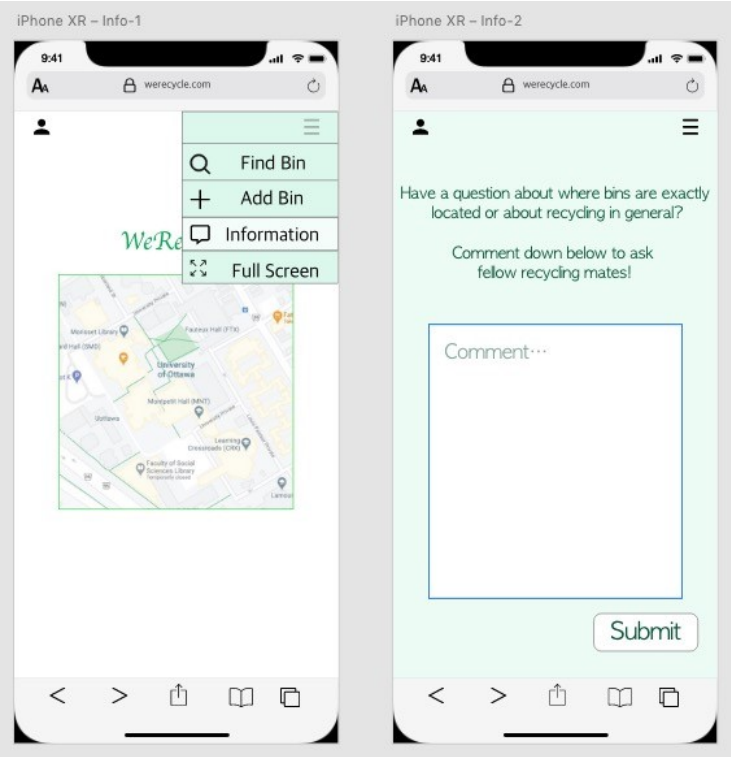

Figure 28 iOS-Info

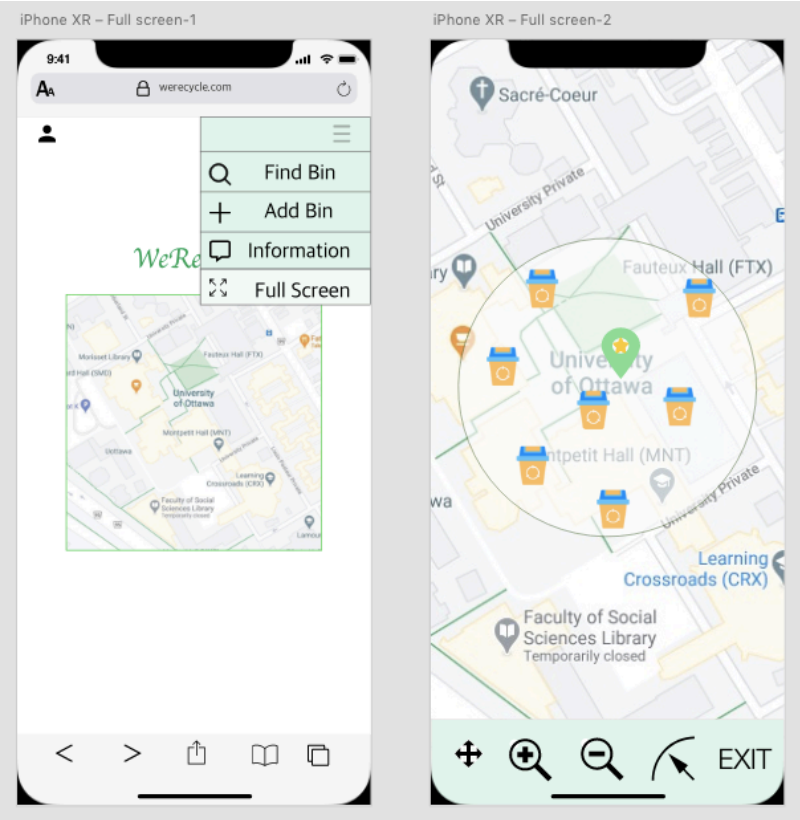

Figure 29 iOS-Full Screen

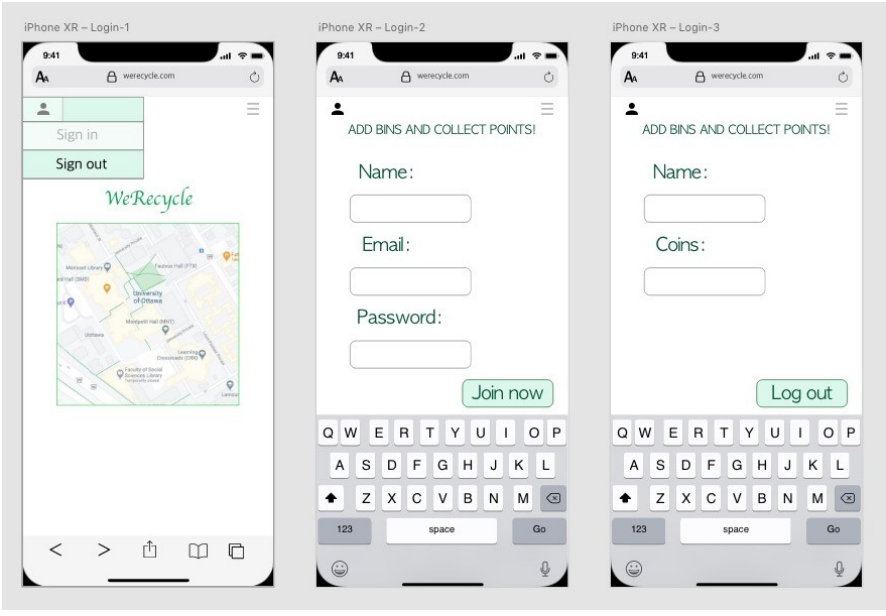

Figure 30 iOS-Login