Group C15

Deliverable F – Prototype 1 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 will lay out the work completed for prototype 1, focusing primarily on UI elements. The chosen methodologies for testing the platform, and a brief overview as to how software-based testing works (Mainly Unit Testing) will follow. Unfortunately due to timing of initial presentations, a prototype was not in development such that customer feedback could have been given. Presentation of initial concept was successful, and further feedback from in-lab client feedback sessions will be highlighted in upcoming deliverables.

1. Introduction

In the last report, our team proposed the design concept of the project, planned the project outline and schedule, and evaluated the project budget and risks.

In this presentation, our team will develop a basic interactive interface based on the conceptual design and customer requirements. We will devote ourselves to studying the usability and possibility of the web page and testing the functionality of the web page. In this deliverable, we will focus on whether the basic functionality of the web page can meet the basic needs of customers and whether the design of the web page is simple.

In the following reports, we will focus on solving problems in the project, optimizing the web application, and optimizing the web design, aiming to follow the concepts of aesthetics and utility.

2. Prototype 1

2.1. Purpose of the Prototype

- We've collected feedback from the client
- Feedback was overall positive
- The major purpose of this prototype is to implement basic features
- In addition we would like to check if this features are matching client's expectation
- Expecting to use this prototype to collect additional client's feedback to improve prototype in next integration

2.2. Features Implemented

In the current prototype stage, some basic features have been implemented. Since our prototype consists of 2 subsystems (API and UI), let's look at each subsystem separately.

2.2.1 API Subsystem:

The overall architecture of API was laid. The subsystem is implemented as a classic REST API with multiple modules. Some of the modules include the public module (/public/) which allows users, who are not part of the organization, to fetch and add bins. Currently, there is no distinction between authorized and non-authorized users, but in future adding a bin will require authentication and authorization.
Also, a comprehensive error system was developed to notify users of incorrect requests. Such a system will play important role in future since it will inform the user when they are authorized to act, and when they are not.

2.2.2. UI Subsystem:

Basic UI was designed, and basic desktop UI was implemented. It is important to differentiate between design and implementation. The team uses Adobe XD to do both

mockups and the later implementations. Design in this context means that UI was mocked-up but does not imply it was implemented in code. Currently, only desktop UI designs are implemented, with mobile UI to come in later versions. Below are samples of the MacOS UI elements. For the remaining UI elements that were designed for prototype 1, please refer to the Appendix.

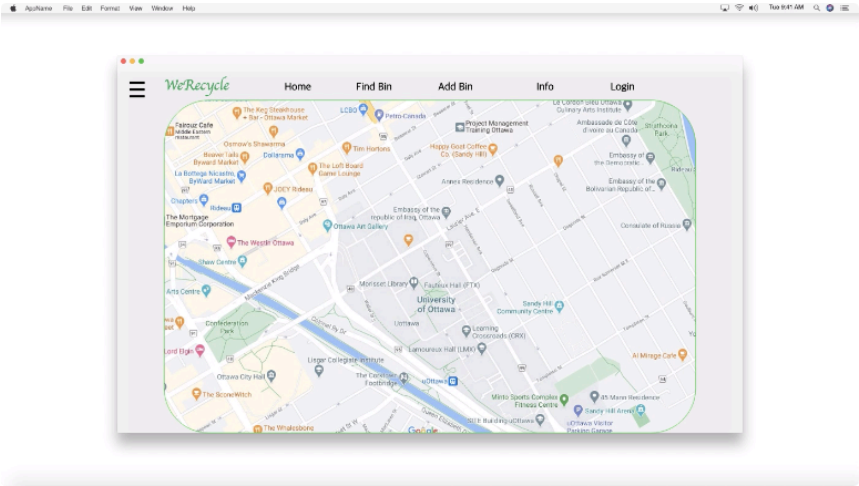


Figure 1 macOS-Main-1

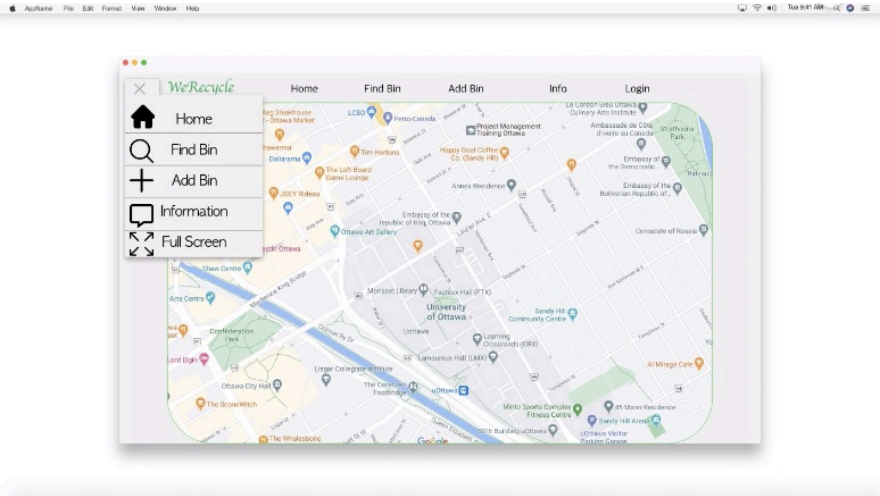


Figure 2 macOS-Main-3

2.3. Prototype Technologies

Our current prototype uses 7 major technologies across both UI and API subsystems. Let's examine how each subsystem uses this technology.

2.3.1. API Subsystem

The major role of the API subsystem is to handle HTTP requests from users and clients, perform certain computations and send HTTP to respond to the user/client. Node.js was chosen to perform this task on the server-side. Node.js is a runtime library that allows us to run JavaScript code (which is conventionally run in the browser) locally on the server

machine. In this way, our team needs to familiarize ourselves with only 1 programming language (JavaScript) for both API and UI development.

While Node.js on its own is enough to handle HTTP requests, it does it in a semantically unfavorable way (Promise chaining), and the large application would be hardly maintainable if written in pure JavaScript and Node.js. To solve this problem Express framework was picked to be used on top of native Node.js. This framework allows us to utilize what's called "middleware" to separate our requests in multiple modules (for example /public/, /organization/, etc.), and perform some computations on requests simply bypassing request or response objects to the aforementioned "middleware" function.

Also, the Jest framework is used to perform unit tests. It allows us to assert function return bypassing pre-defined input. In such a way, many unit tests can be used to check if any change to the code base changes the behavior of any isolated function, simplifying the process of debugging and manual testing.

2.3.2. UI Subsystem

Major role of UI subsystem is a visual interface through which users and organizations can interact with our API without using scripts or code. The system is highly dynamic and have a lot of data that must be retrieved from the server and displayed to the user. Therefore, beside the conventional HTML/CSS and JavaScript, reactive framework, namely ReactJS, was chosen to manage the state of each UI component. React allows programmer to bind dynamic variables to elements of component, which will greatly simplify state management.

3. Test Plan

3.1. Why are we doing these tests?

3.1.1Test 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 our overall system (from request to respond).

3.1.2. Unit test performance example:

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

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

From this prototype 1 we want to determine whether the basic function of finding the radius works properly.
There was very limited time to work on prototype 1, so that is the reason why only this functionality was tested out.

3.1.3. Possible types of results

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.

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

The results will help develop our prototype whether they are good or bad. If something does not go as expected, the appropriate steps will be taken in order to correct unwanted errors.

3.1.5. Criteria for test success or failure

Whenever unit test or manual test fails, this indicates that our system has a serious bug which has to be fixed. Also, the feedback from the client is critical. If the client does not like our final prototype, then we are not meeting the expectations.

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

3.2.1. Type of prototype and reason of selection

This prototype is *focused* at this moment because only one functionality was tested out: the radius functionality. The map is the most important part of the project and the radius is one

of the components that will show the user where the nearest recycling bins are located in the radius that they inputted in the system.
As mentioned previously, because of the limited time to complete this deliverable, this deliverable includes a focused prototype.

3.2.2. Description of testing process

For the testing of the radius, 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)

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 the previous deliverable (Deliverable E), 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 that needs to be done

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?

No, it is not functional part of the project, but it indicates the overall quality of the software and functionality of certain features.

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 our 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 is March 7th 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, our team is following the Wrike Gantt chart that was created in the previous deliverable (Deliverable E). To complete this deliverable, a new Wrike Gantt chart has been created (submitted in separate PDF document).

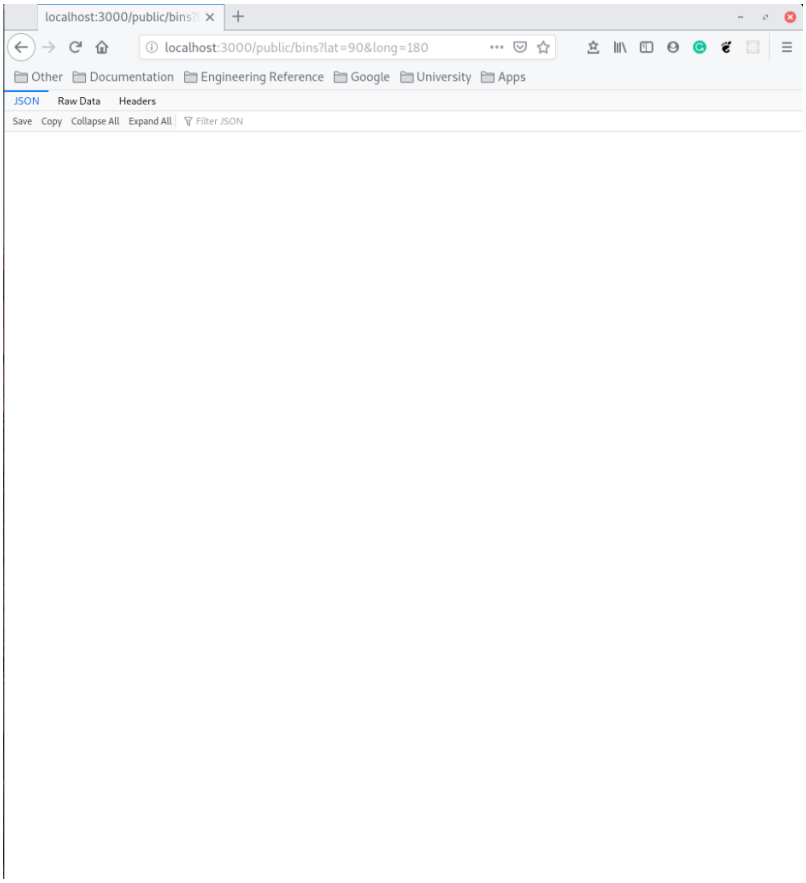
3.4. Prototype 1

In our prototype, we have set the range of the latitude 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 our prototype, and the inputted value of longitude is –200, which is also invalid because it exceeds the range of longitude in our prototype. Therefore, the image shows the view of the terminal and the error message at the end.

```
illian@illiaLaptop:~$ curl -i -X GET http://localhost:3000/public/bins?lat=100&long=-200
[1] 167520
illian@illiaLaptop:~$ HTTP/1.1 400 Bad Request
X-Powered-By: Express
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 80
ETag: W/"50-ZjSncG1yOdmRwB1vXvB40q1hT6M"
Date: Sun, 07 Mar 2021 23:58:11 GMT
Connection: keep-alive
Keep-Alive: timeout=5

"STATUS (400): insufficient query parameters ||| RESOURCE: /public/bins?lat=100"
```

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 our prototype, and the inputted value of longitude is 180, which is also valid because it is in the range of longitude in our 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.



4. Conclusion

Based on customer feedback and the user experience information we have collected; our customers and users are generally satisfied with our project prototype. Our project prototype basically meets the customer's needs of garbage sorting and finding the specific location of garbage bins. In addition, our team made our project applicable to more platforms through web services, and made our project be accepted by more people through easy-to-understand user interface. In the subsequent development, we will mainly improve the aesthetics of the user interface and improve the functionality of the project, so that our project will be more in line with the expectations of customers.

5. Appendix

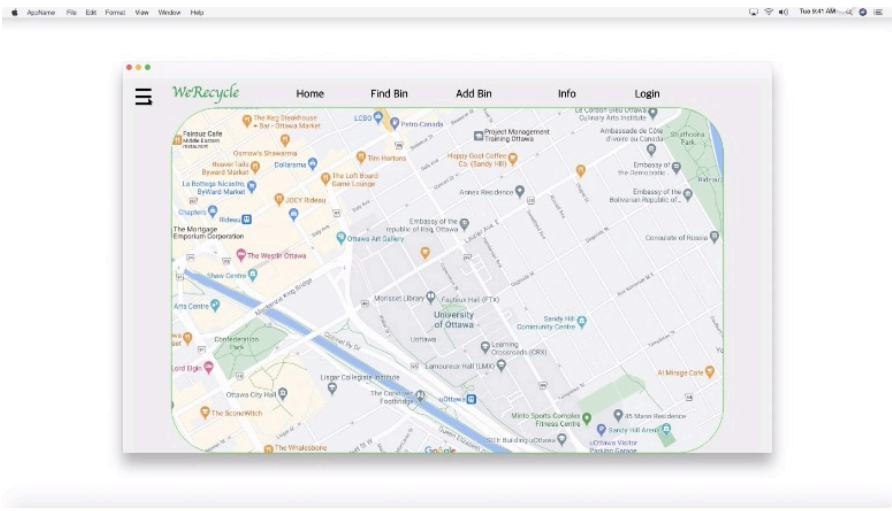


Figure 3 macOS-Main-2

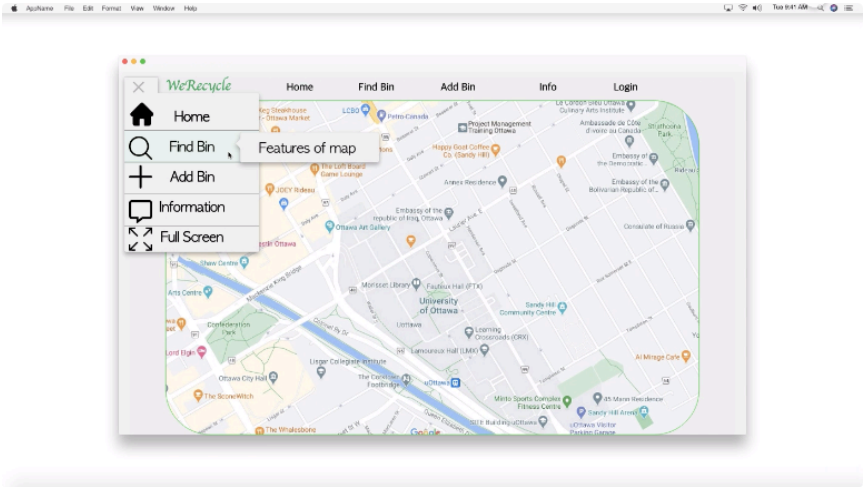


Figure 5 macOS-Main-4

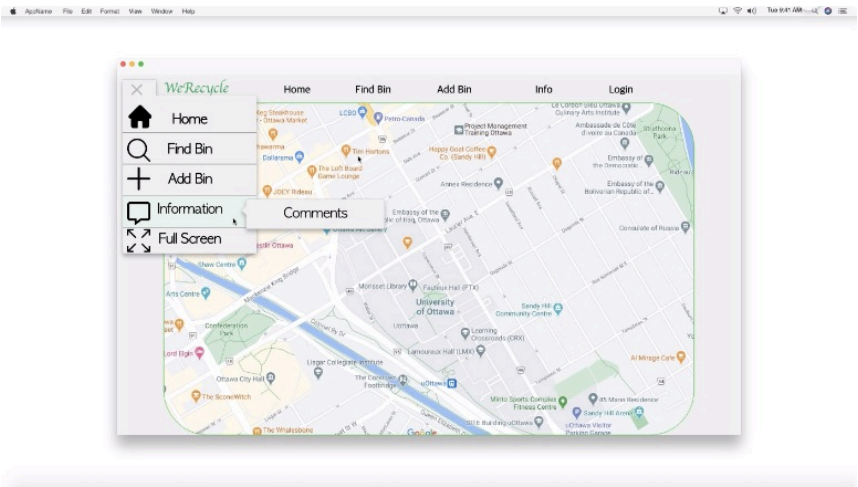


Figure 6 macOS-Main-5

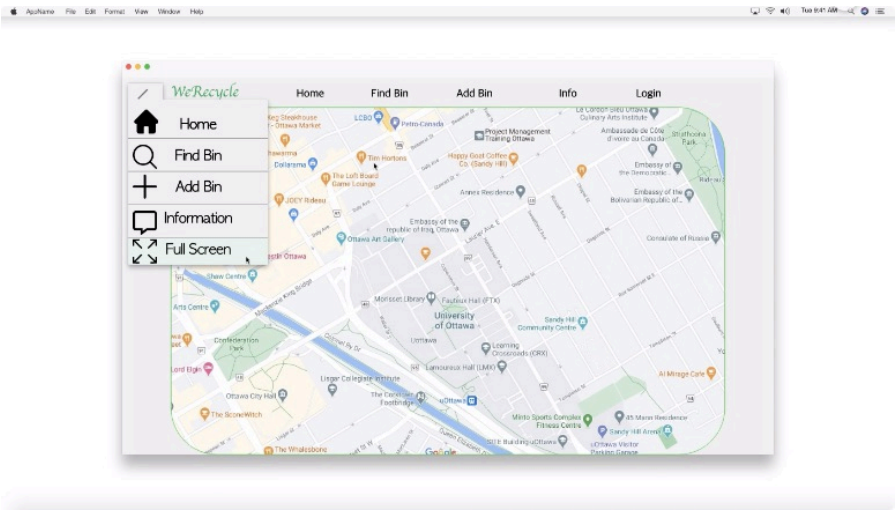


Figure 7 macOS-Main-6

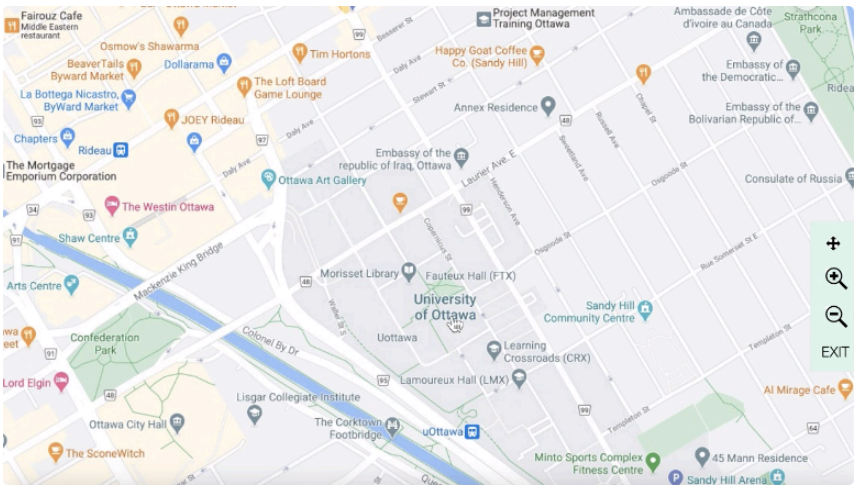


Figure 8 macOS-Main-7

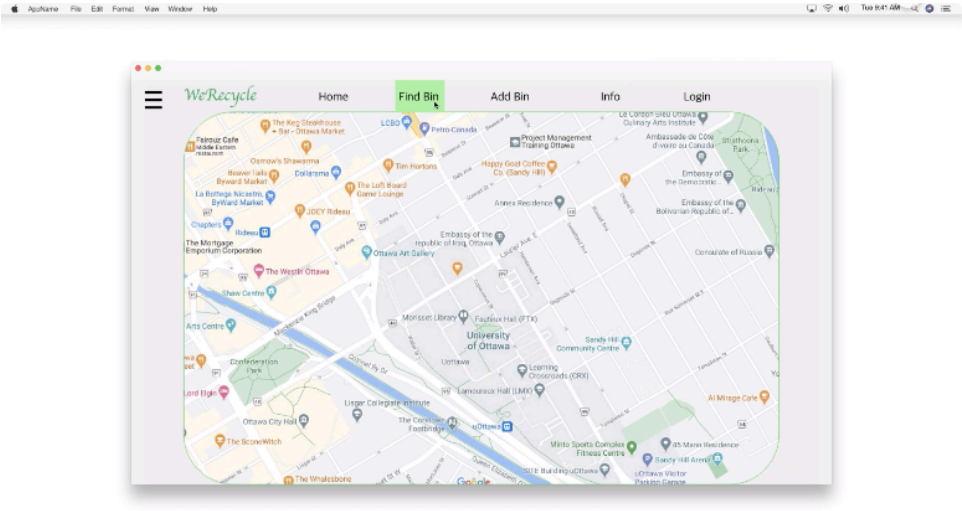


Figure 9 macOS-Find Bin-1

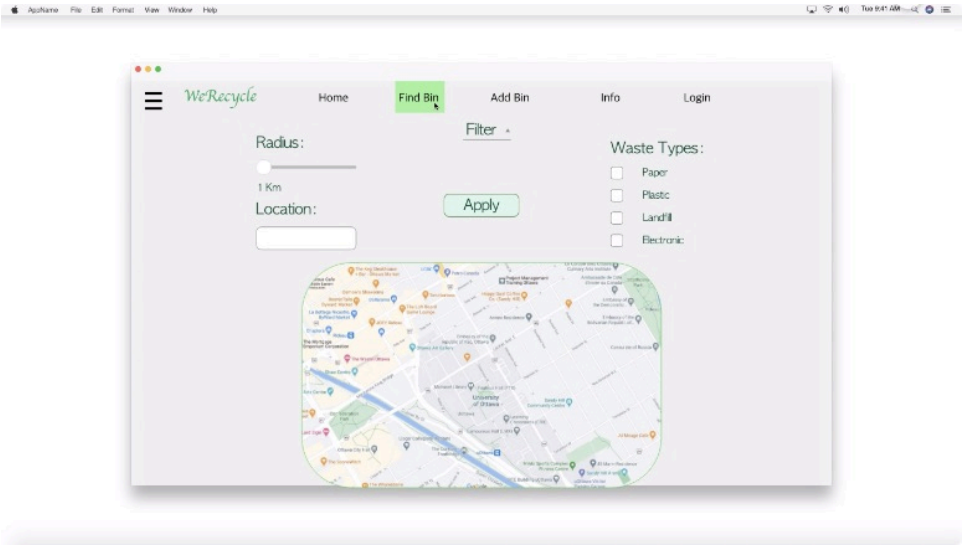


Figure 10 macOS-Find Bin-2

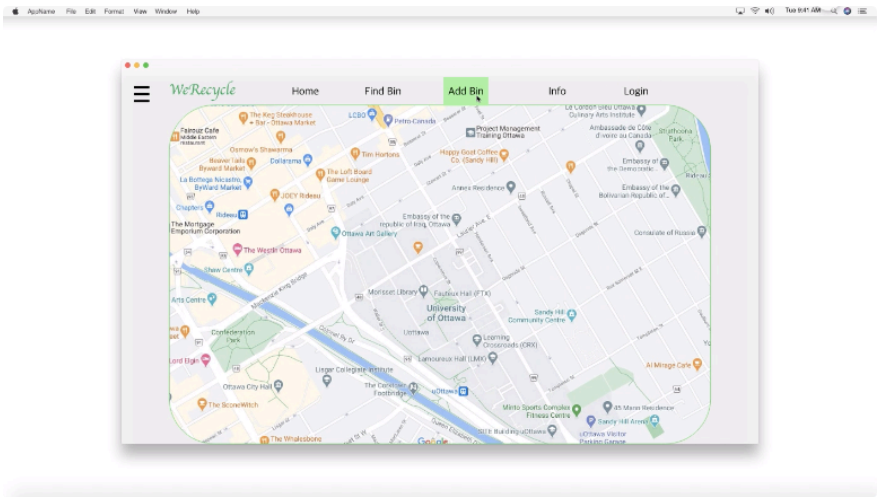


Figure 11 macOS-Add Bin-1

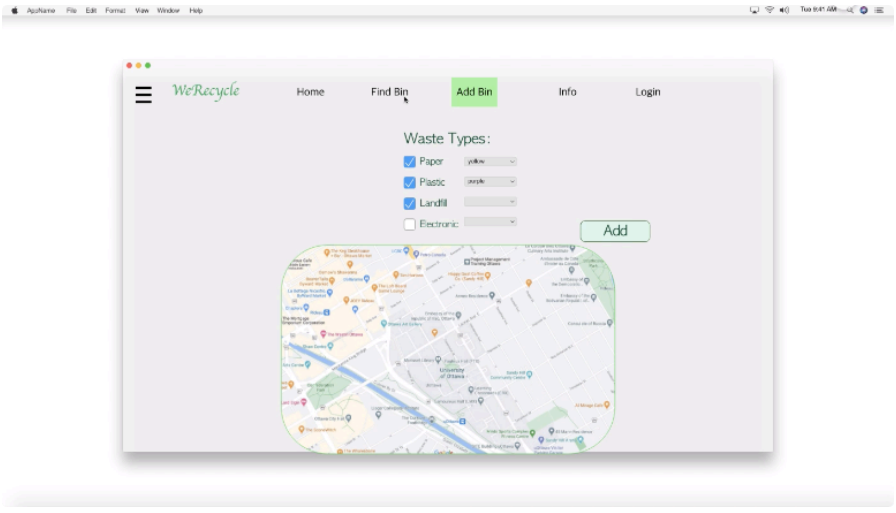


Figure 12 macOS-Add Bin-2

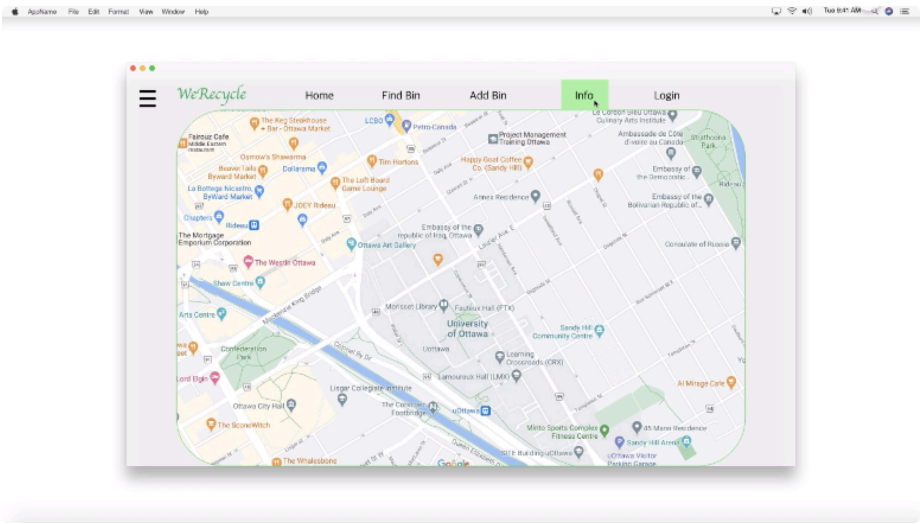


Figure 13 macOS-Info-1

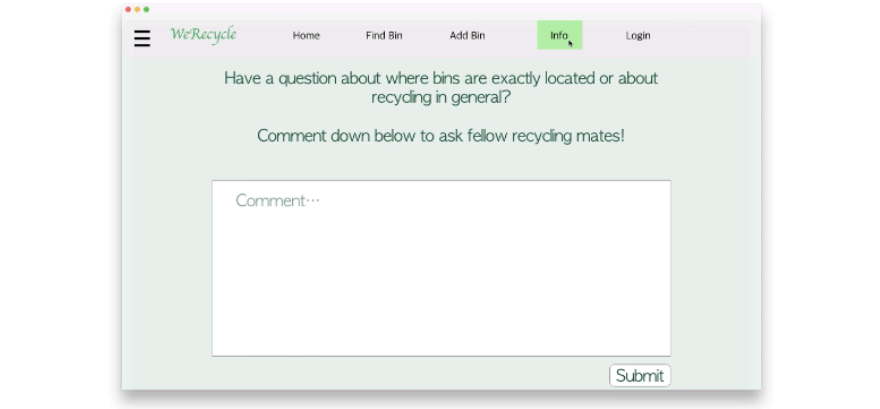


Figure 14 macOS-Info-2

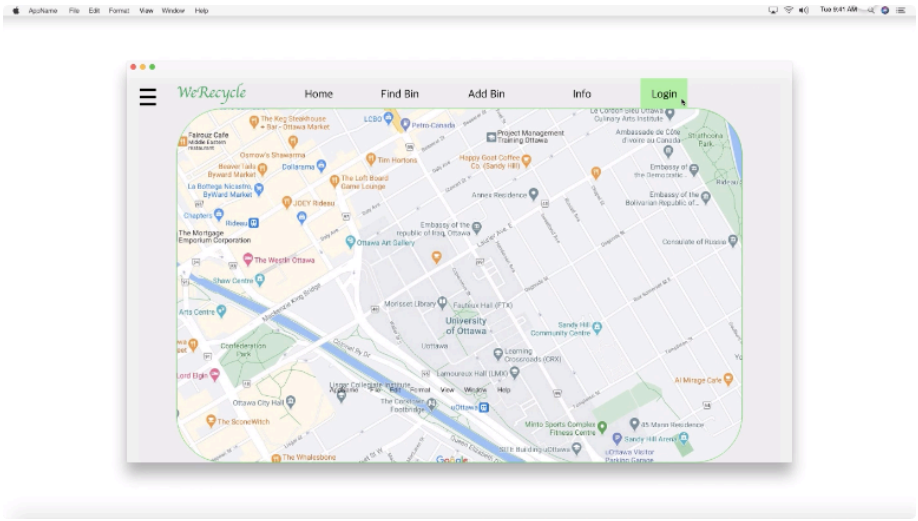


Figure 15 macOS-Login-1

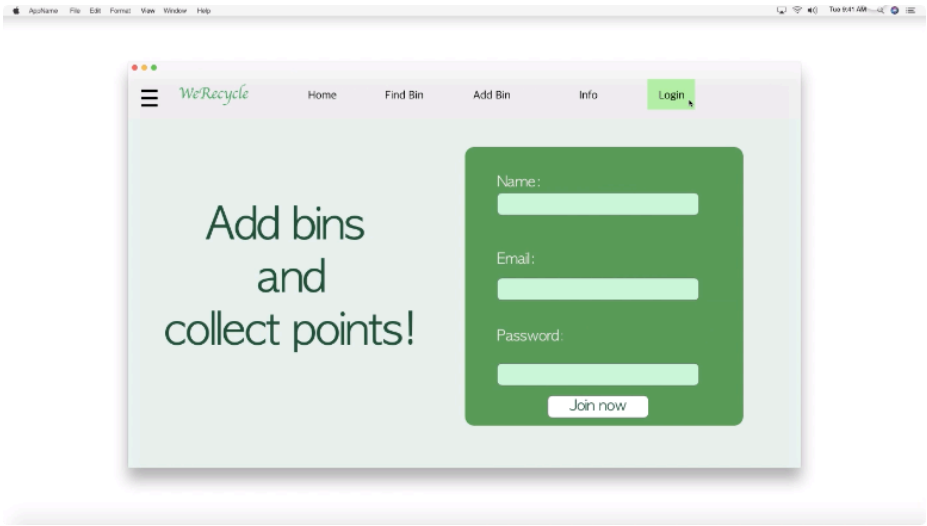


Figure 16 macOS-Login-2

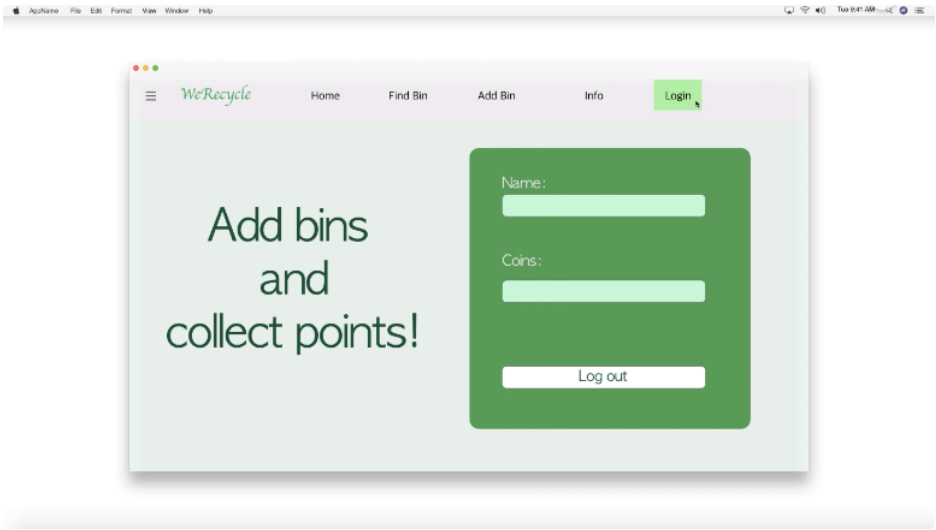


Figure 17 macOS-Login-3

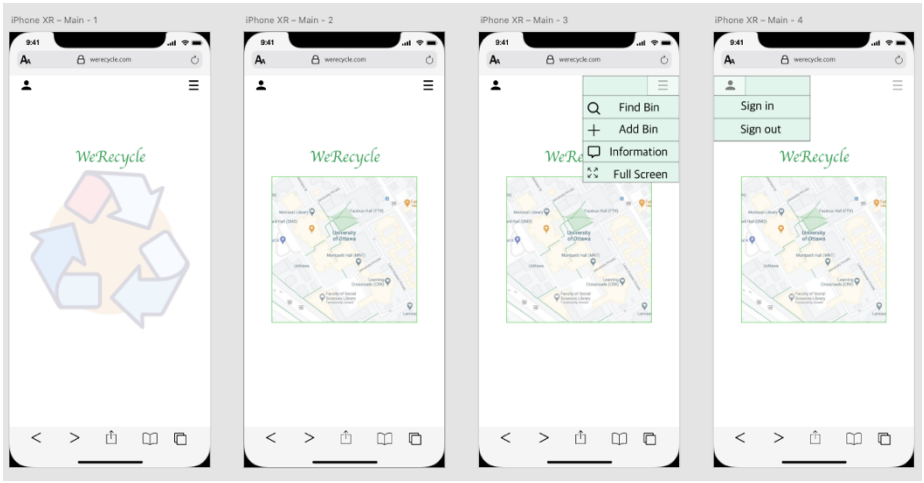


Figure 18 iOS-Main

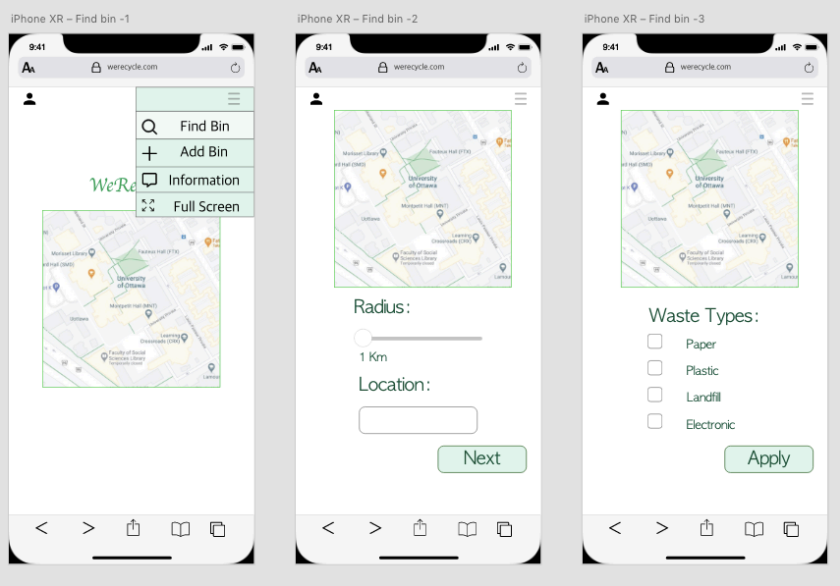


Figure 19 iOS-Find Bin

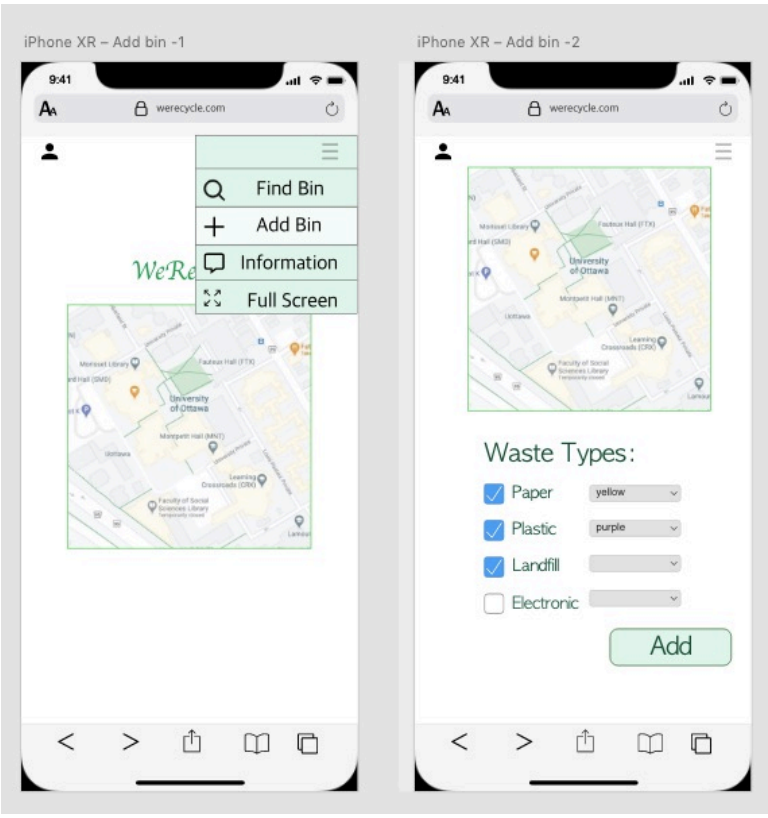


Figure 20 iOS-Add Bin

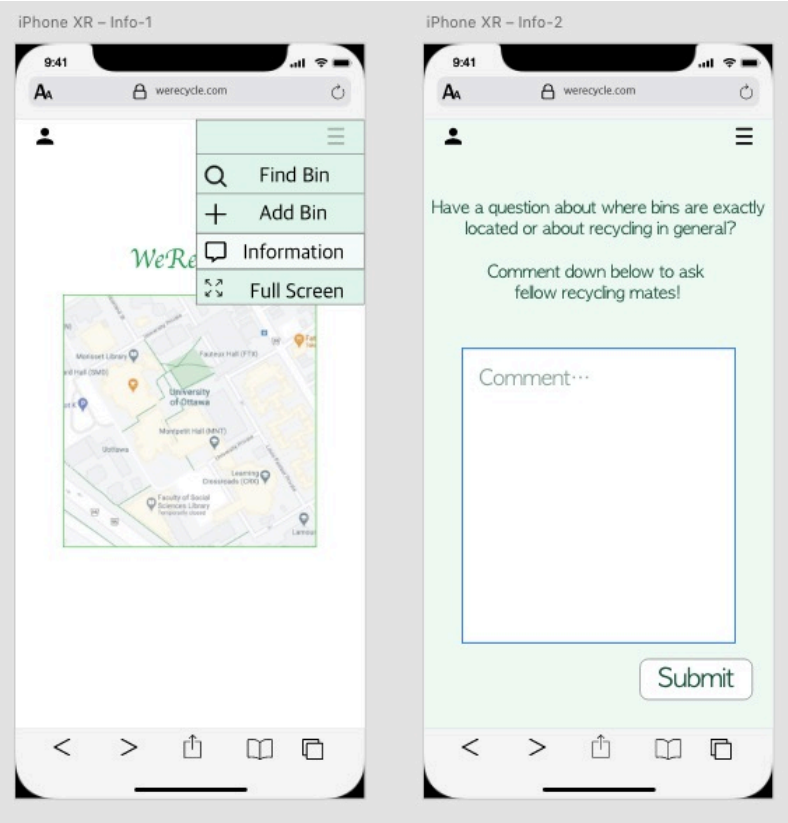


Figure 21 iOS-Info

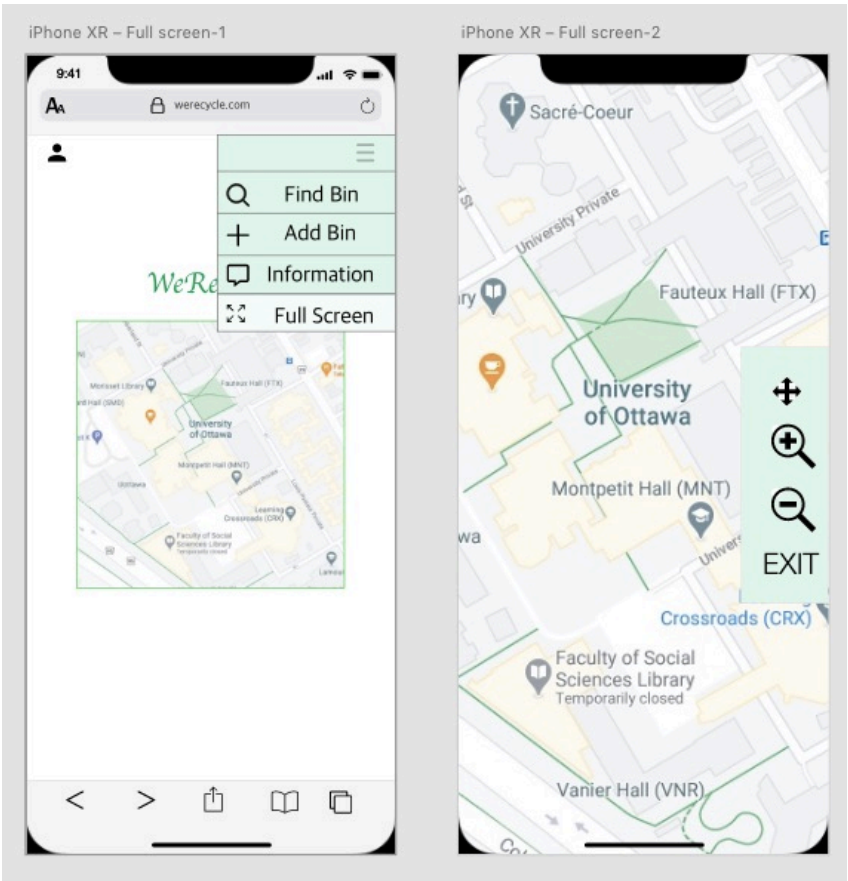


Figure 22 iOS-Full Screen

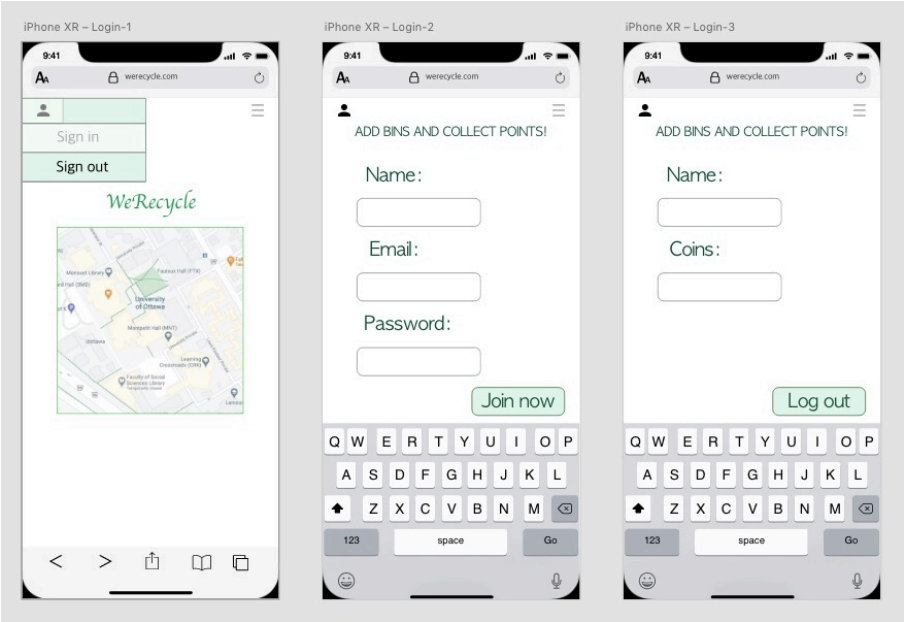


Figure 23 iOS-Login