GNG 1103

Design Project User and Product Manual

TEAM 12

Submitted by:

Group 12

Nicholas Albert Pighin

Shailen Mann

Brayden Baker

Jaron Roy

December 3rd, 2024

University of Ottawa

Table of Contents

List of Figures

List of Tables

List of Acronyms and Glossary

Table 1. Acronyms

Acronym	Definition
AI	Artificial Intelligence
API	Application Programming Interface
GUI	Graphical User Interface
HTML	HyperText Markup Language
IDE	Integrated Development Environment
OS	Operating System
RCP	Robot Containment Program
UE	User Equipment
SMS	Short message service
RCP	Robot Containment Program

Table 2. Glossary

Term	Acronym	Definition
Mini-map	*n/a*	

1.0. Introduction

This User and Product Manual (UPM) provides the information necessary for Shabodi and any client that has purchased this product to effectively use the RCP and for prototype documentation. This document contains the necessary information to properly set up the RCP and use it effectively. It also contains solutions to number of problems that a user might encounter during operations and the best way correct them. Please note that this document contains direct information on the source code of the RCP and access should be restricted only to qualified individuals to reduce the chances of security risks

2.0. Overview

The RCP is designed to track known UEs within a designated zone. With AI and autonomous machines becoming more common in the professional sector, a need to regulate and verify their condition is important. Whether to prevent individuals from harm or to protect proprietary machinery, this device is designed as a warning system for its users. The RCP can track the location of these UEs anywhere with a registered 5G network and will alert administrators of a breach of containment.

The RCP is inspired by the increase in use of autonomous machines and AI in the professional sector. Our program offers an external solution to the tracking of these devices in order to keep your property and individuals safe.

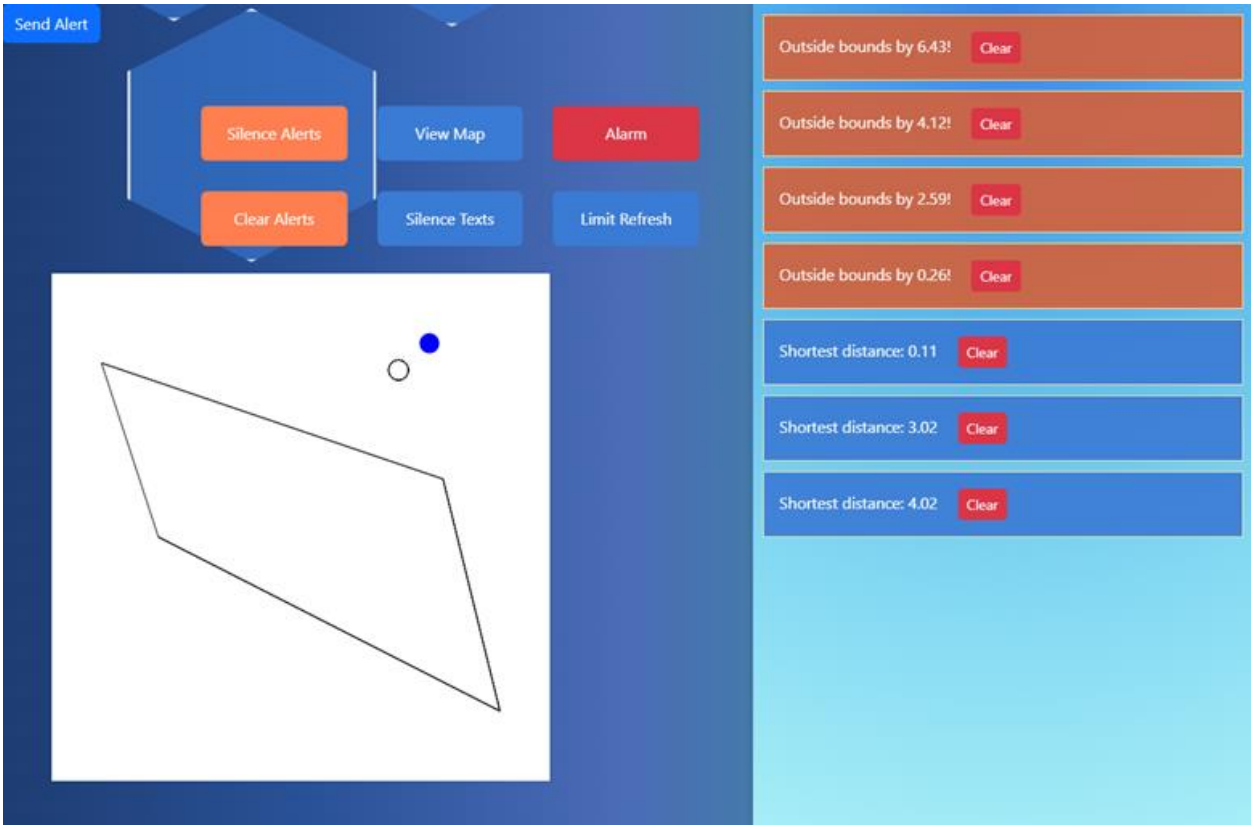


Figure 2.0.1

The program is an executable JavaScript file which when opened displays a GUI to the user and begins to run the function indefinitely. Within the GUI is a number of Buttons with respective functions, a Graphical Interface that shows the position of a UE within the zone, and an Alert Bar that documents any alerts triggered while the program runs.

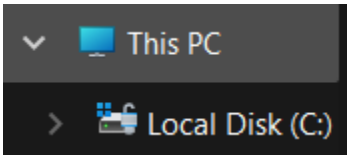
3.0. Getting started

This device functions in accordance with Shabodi's NetAware Program. Before setting up the program, ensure that the UE you wish to track is registered with Shabodi's GetLocation API and Bandwidth API. Once registered the user will update the program with their respective IDs

This prototype was designing using Windows, which is reflected in the documentation. Twilio, node.js, and the other required modules, which are further described in 3.2., likely work on other operating systems, but their implementation will require you to do your own research on how to install these components. Furthermore, they may not even **work** outside of Windows for some unforeseen reason, so, to enable this prototype to enable to work as it should, and in order to benefit from this documentation, use a device running a Windows operating system.

3.1. File Location

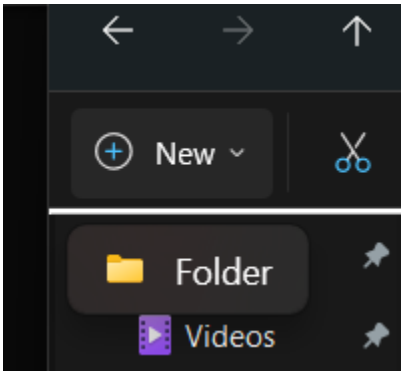
In order for this prototype to work correctly, included files server.js, gng1103website.html, and compass.jpg should be in the same folder, and should not be moved after completing step 3.2.2., or else difficulties with the required modules and interpreters may arise. To make sure that these files will not need to be moved, store them in a folder on a hard drive or SSD on your computer, which can be created by opening File Explorer, navigating to This PC, opening the first



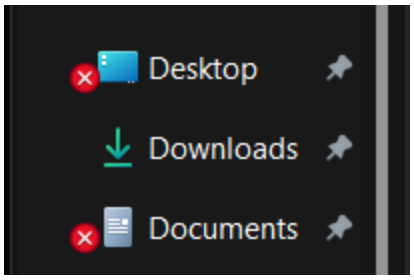
drive titled "Local Disk"

then pressing New->Folder

The title of this folder does not matter, **as long as you remember where to find it**. It also does not matter if this folder is created in a different drive, or resides within other folders, so as its location does not change.



Once this folder has been created, move server.js, gng1103website.html, and compass.jpg to it. Upon download, these files will appear at the top of the Downloads folder in File Explorer



To move them to your newly created folder, open a second instance of File Explorer, locate server.js, gng1103website.html, and compass.jpg in the Downloads folder, then individually drag and drop the files to the folder where they will **stay**.

3.2. Enabling SMS Alerts

3.2.1. Using Twilio

The alerts for this prototype are sent using Twilio, who have an API to send a user-specified message to their phone number. This phone number must first be registered on Twilio's website, found at https://www.twilio.com/en-us. This can be done by pressing "Start for Free"

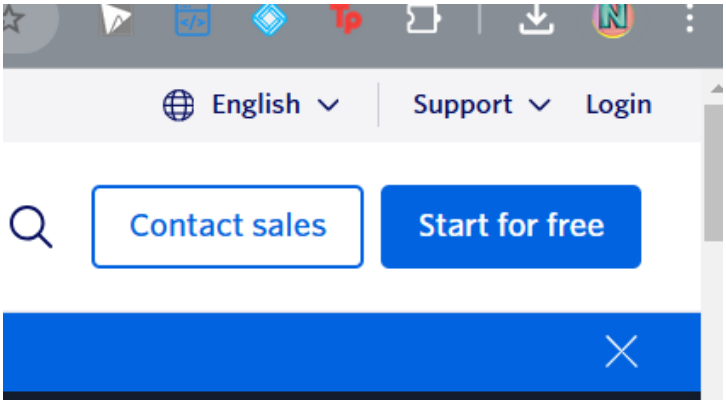


Figure 3.2.1.1

then going through the signup process in order to receive an **Account SID**, **Auth Token**, and **Twilio Phone Number**. They can be located in the Account Info tab in the bottom left of the Account Dashboard page upon signup completion.

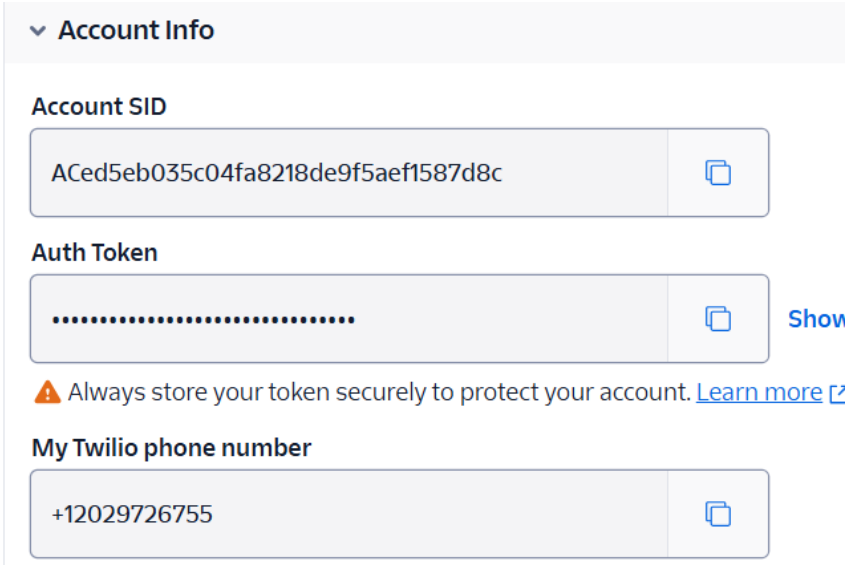
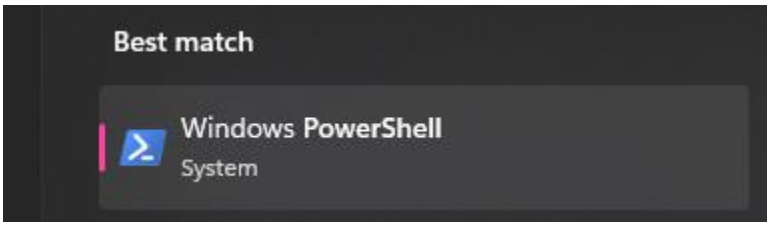


Figure 3.2.1.2

These values, along with the phone number used during signup, will need to be input into the specified places in the included server.js file.

3.2.2. Installing required modules

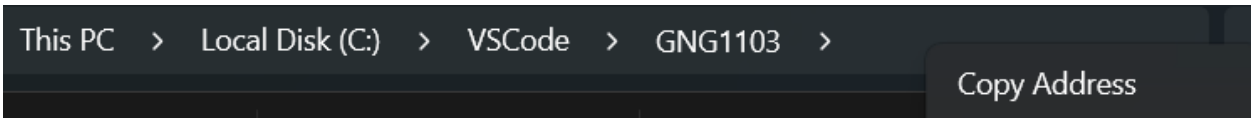
The included server.js file uses node.js, as well as many modules, which may need to be imported/installed. To start, open PowerShell,



then input

cd *FILE_DIRECTORY*

where *FILE_DIRECTORY* is the file directory where your server.js and gng1103website.html files are stored. This can easily be found by opening the folder storing these files, then right clicking the search bar and selecting “Copy Address.”



After this, install node.js by following the instructions on https://nodejs.org/en/download/package-manager. If the following error occurs,

```
PS C:\VSCode> fnm env --use-on-cd
fnm : The term 'fnm' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the
spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ fnm env --use-on-cd
+ ~~~
    + CategoryInfo          : ObjectNotFound: (fnm:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
```

close PowerShell, reopen it, then reexecute

cd *FILE_DIRECTORY*

When running the final line, an error along the lines of "…cannot be loaded because running scripts is disabled on this system" may appear. To resolve this, input

Set-ExecutionPolicy -Scope CurrentUser RemoteSigned

into the same PowerShell terminal, then input

Get-ExecutionPolicy

which should return

RemoteSigned

if execution succeeded. Following the succesful installation of node.js, three modules within server.js will likely need to be installed: express, twilio, cors. To do this, in the same PowerShell terminal, input

npm install express

npm install twilio

npm install cors

3.2.3. Running local server

Once node.js has been installed, server.js has been modified using data acquired from signing up with Twilio, and the needed modules have been installed, it should be possible to run a local server, which is necessary to access Twilio's API. To test this, in the same PowerShell

terminal as before (which can be accessed using the same cd *FILE_DIRECTORY* command as previously described), input

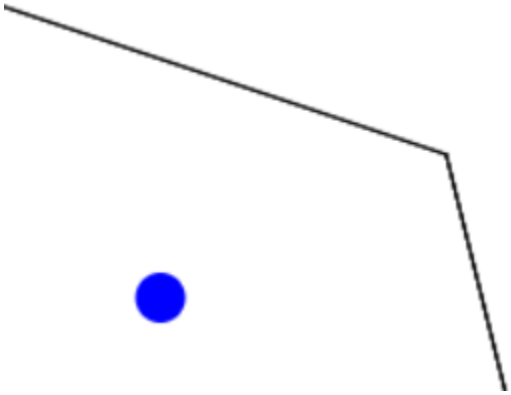
node server.js

If successful, a local server will start, and PowerShell will output

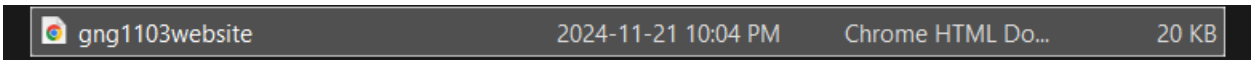
Server running on http://localhost:3000

3.2.4. SMS Alerts on the website

In order to actually receive an SMS, the blue dot representing the UE must cross its. boundary line. At which point the code within gng1103website.html will send a text.



To test this, open



in your preferred browser, then wait until you see a local danger alert appear, which will be in an orange box instead of the ordinary blue for proximity alerts.

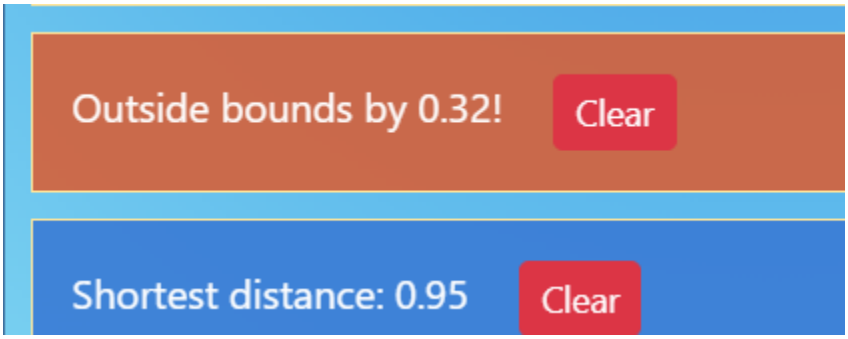
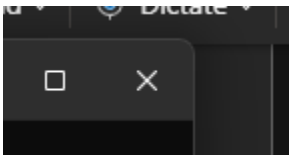


Figure 3.2.4.1

Shortly after seeing a danger alert, a text message will be sent to the phone number specified within server.js. If this does not happen, refer to section 4.1.3., as a local server must be running for an alert to be sent.

3.3. Closing the application

In order to close the local server, which will disable text messages, all that needs to be done is closing the PowerShell terminal by pressing the ‘X’ in the top right corner.

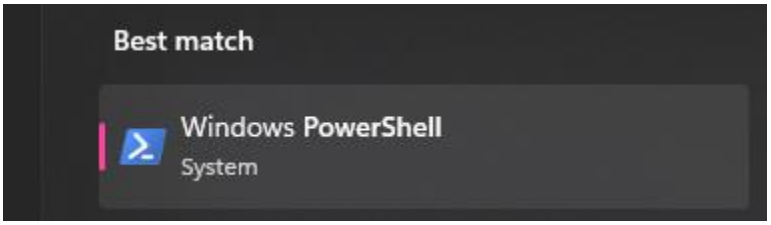


4.0. Using the System

The following sub-sections provide detailed, step-by-step instructions on how to use the various functions or features of the <System Name and/or Acronym>.

4.1. Turning on the Server

Before each and every use, the local server must be started for text messaging to be possible. To do this, open PowerShell



then input

cd *FILE_DIRECTORY*

where *FILE_DIRECTORY* is the file directory where your server.js and gng1103website.html files are stored. Finding this file directory is discussed in 3.2.2. After this, input

fnm env --use-on-cd | Out-String | Invoke-Expression

Without these two steps, running the server is impossible. To actually run the server, only one line must be input in the PowerShell terminal:

node server.js

For any further details, refer to 3.1 and 3.2. While it details the initial setup of the program, the steps for actively using the server are mostly the same.

4.2. Product Configuration

4.2.1. GUI Configuration

4.2.1.1. Mini map is Too Big/Small

A common error that may arise is that the mini-map is not scaled correctly to different monitor display sizes. This is because it is set, by default, to a size of 500 pixels by 500 pixels, which fits well on an ordinary desktop monitor. Because of the way the mini-map is implemented, the code cannot dynamically size it based on monitor size without a significant loss in visual quality. However, the result is that the mini-map may cover some buttons on a small monitor, or may be very small on a very large or high resolution monitor.

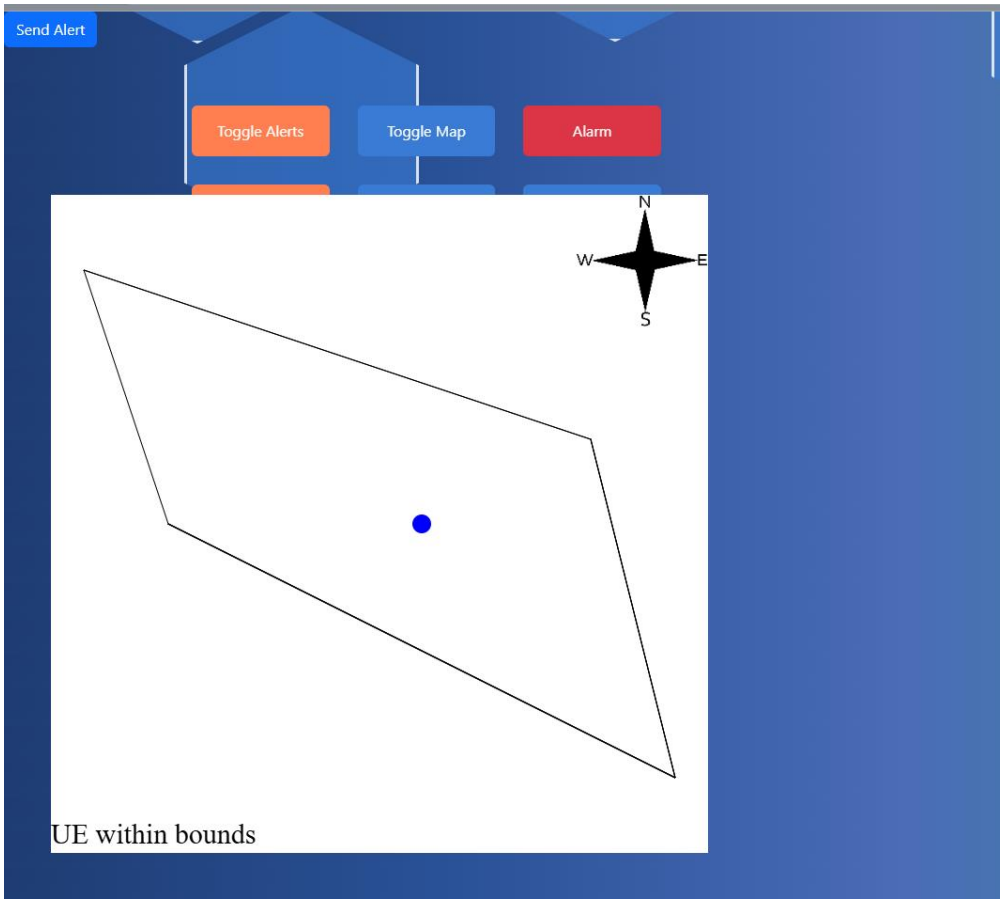
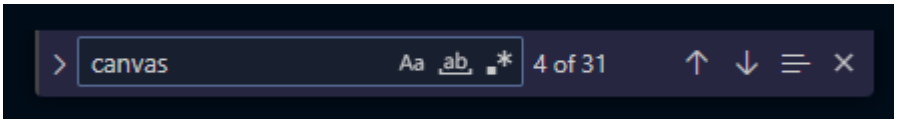


Figure 4.2.1.1.1

To resolve this issue, open the included gng1103website.html file in your preferred IDE or text editor and navigate to this line:

```
198     <canvas id="myCanvas" width="500" height="500" style="position:fixed; left:50px; bottom:50px"></canvas>
```

To make it easier to find this line, most IDEs and text editors have a function to find some input piece of text by pressing ctrl + f. In the search bar that appears, input ‘canvas’



then navigate to the specified line of code. Once there, modify the values included between the quotation marks beside ‘width’ and ‘height.’ These values correspond to the width and height, in pixels, of the white box where the mini-map is drawn. While the code will work as intended for any rectangular canvas, it is designed—and works the best—when the canvas is a square. As such, for best results, it is recommended to input the same value for both its width and height.

4.2.1.2. Text is Too Big/Small

If the text in the bottom-left corner of the canvas is too big or small, it can be easily edited within the gng1103website.html file. Using HTML, text elements must be given a height in pixels, but cannot be dynamically adjusted based on screen size. To edit the text, open the gng1103website.html file in your preferred IDE; this process is described in 5.1.1.2. Locate the following line of code:

```
469         ctx.font = "30px sans serif";
```

This can be done by searching for 'ctx.font,' a process described in 4.2.1.1. To change the size, adjust the number within quotes, but do not change anything else.

4.2.1.3. Compass is Too Big/Small

If the compass in the top-right corner of the canvas is too big or small, it can be easily edited within the gng1103website.html file with a process similar to that described in 4.2.1.2. The compass is set to be 1/5th of the width and height of the total campus, as this was ideal for Team 12's use, but this can be edited based on user preference. To do this, locate these lines:

```
494         img.addEventListener("load", () => {
495             ctx.drawImage(img, (4/5)*myCanvas.width, 0, myCanvas.width/5, myCanvas.height/5);
496         });
```

```
592                 ctx.drawImage(img, (4/5)*myCanvas.width, 0, myCanvas.width/5, myCanvas.height/5);
```

In the brackets beside 'ctx.drawImage,' there are 5 values separated by commas. The 4th and 5th values represent the width and height, respectively, of the compass relative to those of the canvas. In the included example, the width and height are set to 1/5th of their respective values. These fractions can be edited based on user preference. **However**, the 2nd value in the list, which reads '(4/5)*myCanvas.width' and represents the x-axis position of the compass relative to the canvas, must be updated accordingly. It's value, when added to that of the 4th value in the list, much be equal to myCanvas.width. For example, if the 4th value were to be changed to myCanvas.width/10, the 2nd value should be changed to (9/10)*myCanvas.width. This change must be done in both instances of 'ctx.drawImage,' and the dimensions of the compass should remain equal to one another to keep the compass as a square, or else the compass may not appear as intended.

4.3. Functions and Features

4.3.1. Boundary Alert System

- **Feature Description:** The system dynamically calculates the UE's position relative to predefined boundaries. If the UE is close to or outside the boundaries, alerts are displayed.
- **Expected Behavior:**
    - When inside bounds: Displays proximity alerts with the shortest distance to the boundary.

    - When outside bounds: Sends danger alerts specifying the extent of breach and optionally sends text messages.
- **Mastery Required:** Understanding of boundaries and UE movement logic, especially how distances are computed.
- **Caveats and Exceptions:**
    - Alerts may have a delay if the limitRefresh functionality is enabled.
    - UE position updates depend on randomness (randMove()), which might not simulate real-world paths.

```
function limitRefresh() {
    if(limitAlertsBoolean)
        limitAlertsBoolean = false;
    else
        limitAlertsBoolean = true;
}
```

```
function randMove(delay) {

    let coeff;
    if(Math.random() < 0.5)
        coeff = -1;
    else
        coeff = 1;
    return delay*Math.random()*coeff/1000;

}
```

4.3.2. Dynamic Alert Messages

- **Feature Description:** Alerts dynamically populate the alert-container with proximity or danger messages.
    - **Proximity Alerts**: Triggered when UE approaches the boundaries but remains inside.
    - **Danger Alerts**: Triggered when UE exits the boundaries.
- **Controls:**
    - **Clear Alerts:** Removes all existing messages.
    - **Toggle Alerts:** Turns alerts on or off.

- **Expected Behavior:**
  - Alerts should be dismissible.
  - Alerts toggle functionality halts further notifications when disabled.
- **Special Instructions:** Use the clear function periodically to manage high alert volume.

Figure 4.3.2.1

4.3.3. Text Alert Integration

- **Feature Description:** Sends text alerts via Twilio when the UE breaches boundaries. The sendTwilioAlert() function handles this.

- **Expected Behavior:**
  - Sends a notification on the first breach.
  - Logs success or failure of the text alert in the console.
- **Caveats:**
  - Requires a functional Twilio backend and proper API keys.
  - Text alerts can be toggled off if not needed.

```
function sendTwilioAlert(message) {
    fetch('http://localhost:3000/send-alert', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ message }),
    })
        .then((response) => response.json())
        .then((data) => {
            if (data.status === 'success') {
                console.log('Alert sent via Twilio:', data.sid);
            } else {
                console.error('Failed to send Twilio alert:', data.message);
            }
        })
        .catch((error) => {
            console.error('Error sending Twilio alert:', error);
        });
}
```

4.3.4. Dynamic Map Visualization

- **Feature Description:** A canvas element renders the boundaries and dynamically updates UE position and scale to fit the view.
- **Expected Behavior:**
  - Visualize the UE's movement and boundary regions.
  - Canvas updates based on UE proximity, with scaling for optimal visibility.
- **Controls:** Toggle the map on or off.
- **Caveats:**
  - The map visualization only updates when the toggle is active.
  - High delays in boundary redraws may cause inconsistencies in visualization.

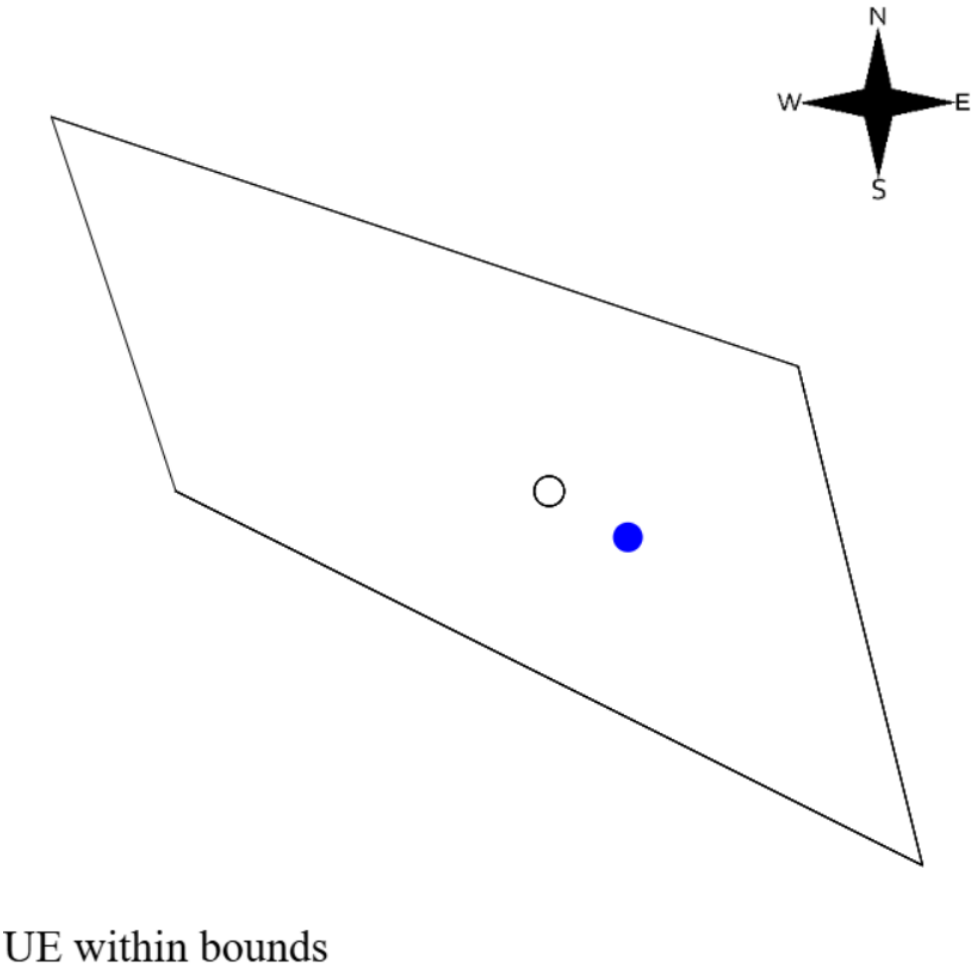


Figure 4.3.4.1

4.3.5. Control Panel Functionality

The control panel includes the following interactive buttons:

- **Toggle Alerts:** Enables or disables the display of alerts on the webpage.
- **Toggle Map:** Toggles the visibility of the dynamic map visualization.
- **Alarm:** Simulates an alarm trigger for emergencies.
- **Clear Alerts:** Clears all alert messages in the alert-container.
- **Toggle Texts:** Enables or disables the sending of text message alerts.
- **Limit Refresh:** Sets alerts and updates to a 10-second delay to reduce frequency.
- **Special Instructions:**
    - Use the buttons in combination based on the situation.
    - Hovering over each button provides a description of its functionality.

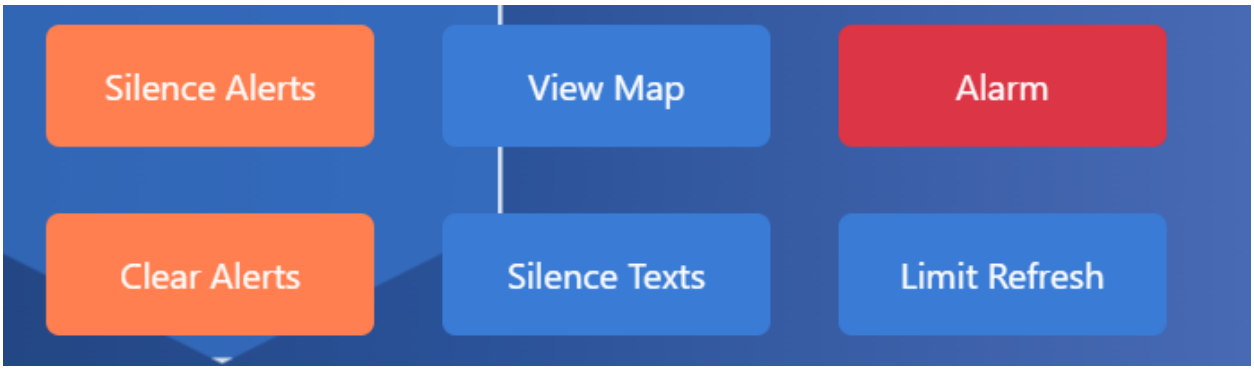


Figure 4.3.5.1

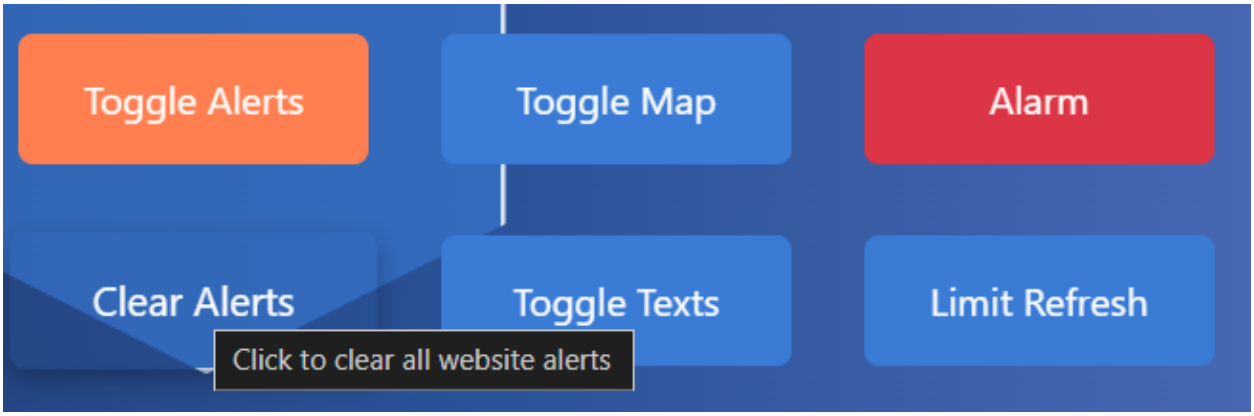


Figure 4.3.5.2

4.3.6. Boundary Logic and UE Movement

- **Feature Description:** Implements complex mathematical calculations for:
  - Checking if UE is within the boundaries (inBoundChecker function).
  - Calculating distances from the UE to boundary segments (getDistances function).
- **Caveats:**
  - The randMove() function simulates random movement, which might not mimic actual scenarios.
  - Errors in defining boundary coordinates could lead to inaccurate results.

```
function inBoundChecker(ueX, ueY, boundaryX, boundaryY) {

    let numVertex = boundaryX.length;
    let rays = 0;

    for(let i = 0; i < numVertex - 1; i++) {

        if(boundaryY[i] > boundaryY[i+1]) {

            if((ueY < boundaryY[i]) && (ueY > boundaryY[i+1])) {

                if(ueX <= ((ueY-boundaryY[i])*(boundaryX[i]-boundaryX[i+1])/(boundaryY[i]-boundaryY[i+1])+boundaryX[i]))
                    rays++;

            }

        }

        else {

            if((ueY > boundaryY[i]) && (ueY < boundaryY[i+1])) {

                if(ueX <= ((ueY-boundaryY[i+1])*(boundaryX[i+1]-boundaryX[i])/(boundaryY[i+1]-boundaryY[i])+boundaryX[i+1]))
                    rays++;

            }

        }

        if((ueY == boundaryY[i]) && (ueX <= boundaryX[i]))
            rays++;

    }

    if(boundaryY[numVertex - 1] > boundaryY[0]) {

        if((ueY <= boundaryY[numVertex - 1]) && (ueY >= boundaryY[0])) {

            if(ueX <= ((ueY-boundaryY[numVertex - 1])*(boundaryX[numVertex - 1]-boundaryX[0])/(boundaryY[numVertex - 1]-boundaryY[0])+boundaryX[numVertex - 1]))
                rays++;

        }

        }

    else {

        if((ueY >= boundaryY[numVertex - 1]) && (ueY <= boundaryY[0])) {

            if(ueX <= ((ueY-boundaryY[0])*(boundaryX[0]-boundaryX[numVertex - 1])/(boundaryY[0]-boundaryY[numVertex - 1])+boundaryX[0]))
                rays++;

        }

    }

    if((ueY == boundaryY[numVertex - 1]) && ueX <= (boundaryX[numVertex - 1]))
        rays++;

    if(rays%2 == 0)
        return false;
    return true;
```

4.4. Sub-Functions and Sub-Features

4.4.1. **Boundary Monitoring Sub-Functions**

These sub-functions determine the UE's status relative to the boundary.

a. Distance Calculation (getDistances**)**

- **Description:** Calculates the shortest distances between the UE and each boundary segment using geometric formulas.

- **Expected Behavior:** Returns an array of distances, enabling identification of the shortest.
- **Caveats:**
    - Assumes the boundary is defined by a closed polygon.
    - Inaccurate boundary coordinates could lead to erroneous calculations.

b. Boundary Inclusion Check (inBoundChecker**)**

- **Description:** Verifies whether the UE is within the defined polygon using a ray-casting algorithm.
- **Expected Behavior:** Returns true if the UE is inside the boundary and false otherwise.
- **Special Instructions:**
    - Ensure boundary arrays (boundaryX, boundaryY) are synchronized.
    - The algorithm assumes a convex or non-complex polygon for accurate results.

4.4.2. Alert Management Sub-Functions

These handle the generation, display, and removal of alerts.

a. Proximity Alert (proximityAlert**)**

- **Description:** Generates alerts when the UE is inside but near the boundary. Adds a yellow-styled alert to the container.
- **Expected Behavior:** Alerts contain the shortest distance and an option to clear individually.
- **Caveats:**
    - Alert styling may need adjustment if additional alert types are introduced.

b. Danger Alert (dangerAlert**)**

- **Description:** Displays alerts when the UE is outside the boundary, with a red-styled warning.
- **Expected Behavior:** Alerts are marked distinctively to signify urgency.
- **Special Instructions:** Used in tandem with the proximityAlert function for comprehensive monitoring.

c. Clear Individual Alert (clearAlert**)**

- **Description:** Provides a close button on each alert to dismiss it individually.
- **Expected Behavior:** Removes only the specific alert clicked.
- **Caveats:** Does not reset the alertCount variable.

d. Clear All Alerts (clearAllAlerts**)**

- **Description:** Removes all alerts from the container simultaneously.

- **Expected Behavior:** Cleans the alert container instantly.
- **Caveats:** Alerts added during the clearing process may persist.

4.4.3. Visualization Sub-Functions

These sub-functions manage graphical elements within the map.

a. Dynamic Boundary Drawing (repeatDraw**)**

- **Description:** Redraws the boundary, UE, and related elements based on their real-time positions.
- **Expected Behavior:**
  - Redraws polygon boundaries and UE position.
  - Adjusts scale dynamically to fit the canvas.
- **Caveats:**
  - Scaling may distort proportions if extreme coordinates are used.
  - Requires the mapBoolean flag to be true.

b. Entity Movement Simulation (randMove**)**

- **Description:** Introduces randomness in UE movement for simulation purposes.
- **Expected Behavior:** Generates small, randomized positional shifts for the UE at each iteration.
- **Caveats:** Intended for simulation, not real-world data.

c. Compass Image Integration

- **Description:** A compass graphic is rendered for orientation purposes.
- **Expected Behavior:** Remains static in the top-right corner of the map.
- **Caveats:** Requires the img source to be available locally or hosted online.

4.4.4. Control Panel Sub-Functions

These sub-functions support user interactivity with the system.

a. Toggle Functionality

- **Description:** Functions to enable or disable specific system behaviors.
  - toggleAlerts: Turns alert notifications on or off.
  - toggleMap: Toggles visibility of the dynamic map.
  - toggleTexts: Activates or deactivates text message alerts.

- **Expected Behavior:** Provides on/off switching for respective features.
- **Special Instructions:** Use cautiously to avoid conflicting system states (e.g., map off while attempting to visualize UE movement).

b. Emergency Alarm Trigger (triggerAlarm**)**

- **Description:** Simulates an emergency alarm for critical situations.
- **Expected Behavior:** Displays an alert popup with the message "Emergency alarm triggered!".
- **Caveats:** No connection to external systems (e.g., sirens or physical alarms).

c. Limit Refresh Rate (limitRefresh**)**

- **Description:** Reduces the frequency of system updates to a 10-second interval.
- **Expected Behavior:** Helps manage system performance during high activity.
- **Caveats:** May delay critical alerts if the system state changes rapidly.

4.4.5. Twilio Integration Sub-Functions

Handles communication with external services to send text alerts.

a. Text Alert Dispatcher (sendTwilioAlert**)**

- **Description:** Sends a POST request to a backend server for Twilio SMS alerts.
- **Expected Behavior:** Text alerts are sent upon boundary breaches or returns.
- **Caveats:**
    - Requires a configured backend running on http://localhost:3000/send-alert.
    - May fail silently if the server is unavailable.

5.0. Troubleshooting & Support

5.1. Known Error Procedures

5.1.1. Text Messaging Errors

5.1.1.1. Server is not Running

If the UE leaves the set boundary but a text message is not sent, the most likely reason is that the server used to send the text is not running. To determine whether this is truly the reason why text messaging is not working, right click anywhere on the website, then click ‘Inspect.’

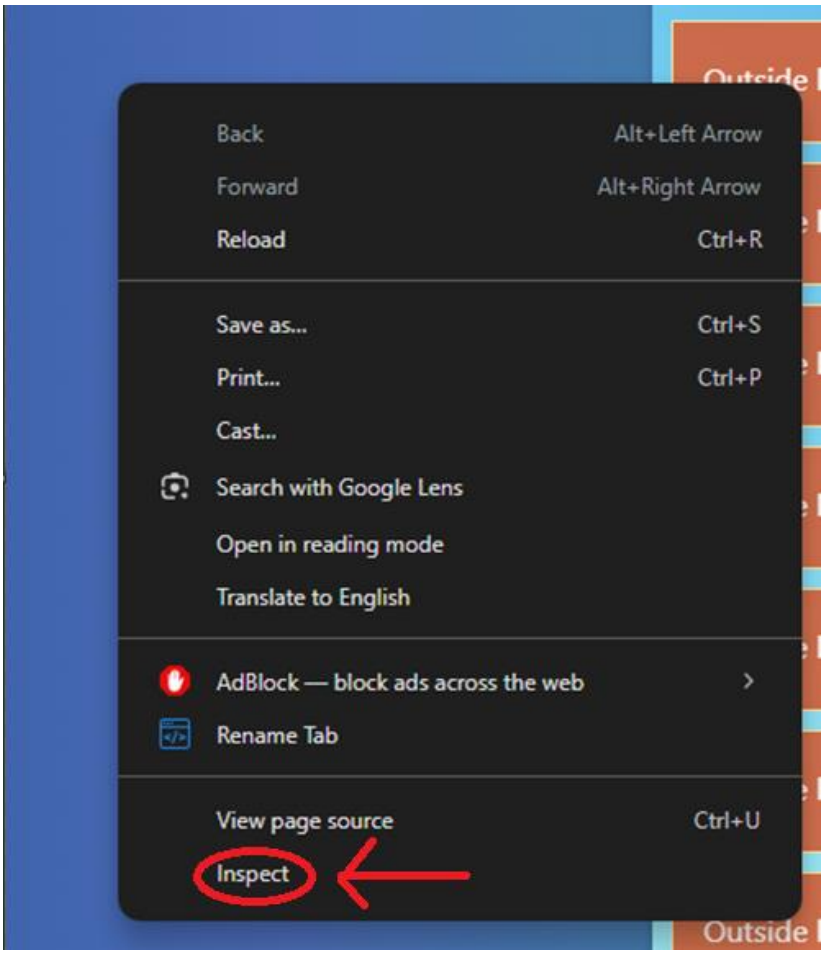
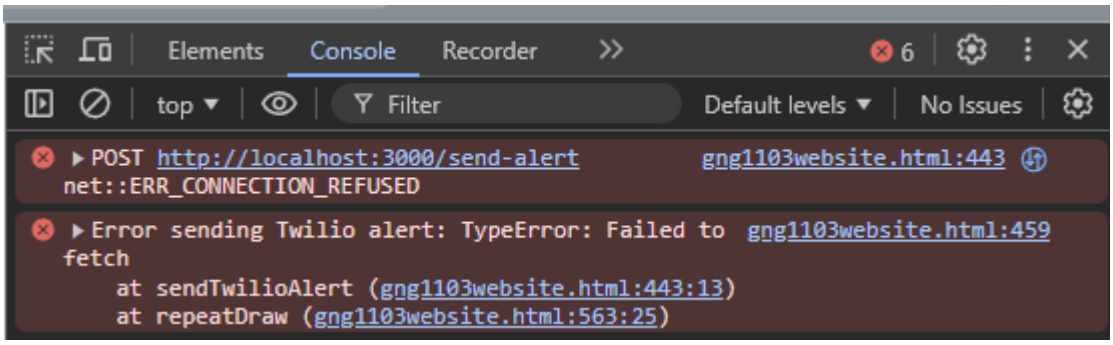


Figure 5.1.1.1.1

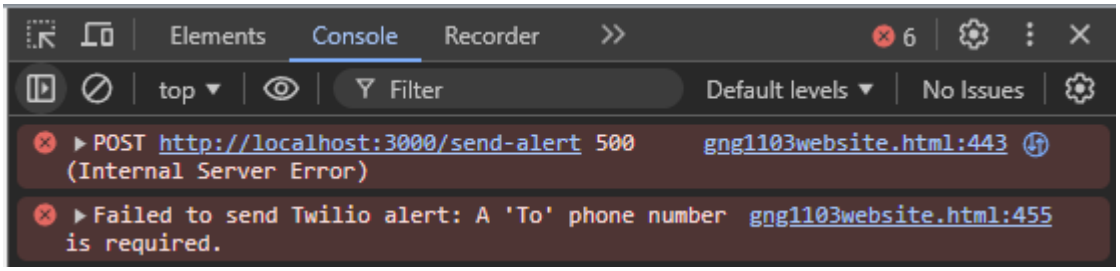
In the menu that that brings up, navigate to the ‘Console’ tab. If the following error is shown



it is because the website has attempted to contact the server, but has not been able to find it, as it is not running. To turn on the server, which should resolve the issue, refer to 4.1.

5.1.1.2. Server File is Configured Wrong

Another potential problem with text messaging is that the included server.js file has not been modified to include valid Twilio information (Account SID, Authentication Token, Twilio phone number, receiving number). If this is the reason why text messaging is not working, there will be 2 signs. The first can be found in the 'Console' tab on the website, which can be located by following the instructions in 5.1.1.1. It will look generally like this



though may specify that something other than 'A 'To' phone number' is missing, depending on what is missing from the server.js file. The second sign that this is the cause of the error can be found in the PowerShell terminal described in 4.1. The error will look like this:

```
RestException [Error]: A 'To' phone number is required.
    at success (C:\VSCode\node_modules\twilio\lib\base\Version.js:79:23)
    at process.processTicksAndRejections (node:internal/process/task_queues:105:5) {
  status: 400,
  code: 21604,
  moreInfo: 'https://www.twilio.com/docs/errors/21604',
  details: undefined
}
```

It contains the same information as that found in the 'Console' tab, but further specifies where to look in Twilio's documentation for potential causes, solutions, etc. To resolve this issue, open the included server.js file in an IDE—like Microsoft Visual Studio Code—then fill in the quotations denoted by the following comments

```
"; //replace value between quotes with Account SID from Twilio website
//replace value between quotes with Auth Token from Twilio website

", //replace value between quotes with 'My Twilio phone number' from Twilio website
//replace value between quotes with personal phone number used during Twilio signup
```

with the specified information.

5.1.2. GUI Errors

5.1.2.1. Mini-map only Updates Once

If the mini-map only updates once, then the program stops visually changing, check to see if the compass is in the top-right corner of the mini-map.

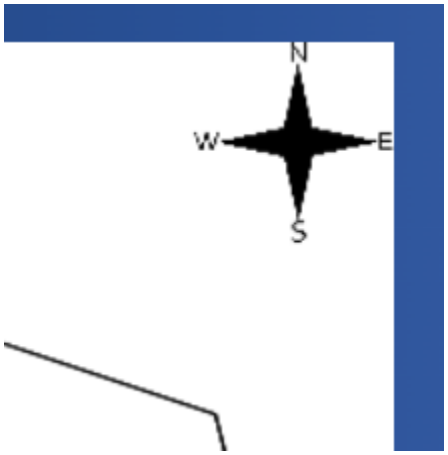


Figure 5.1.2.1.1

If it is not, that is because the code is looking for the compass.jpg file but cannot find it. To resolve this, make sure that the included compass.jpg file has been downloaded, then move it to the same folder as the server.js and gng1103website.html files. Instructions for how to do this are in 3.1.

5.1.3. Distance Calculation Errors

In Team 12's testing, the distance calculation formula was always accurate. However, if this proves not to be the case when compared to real-world measurements, etc., the formula used to calculate the distance between the UE and each boundary wall, developed by Team 12, is included for reference:

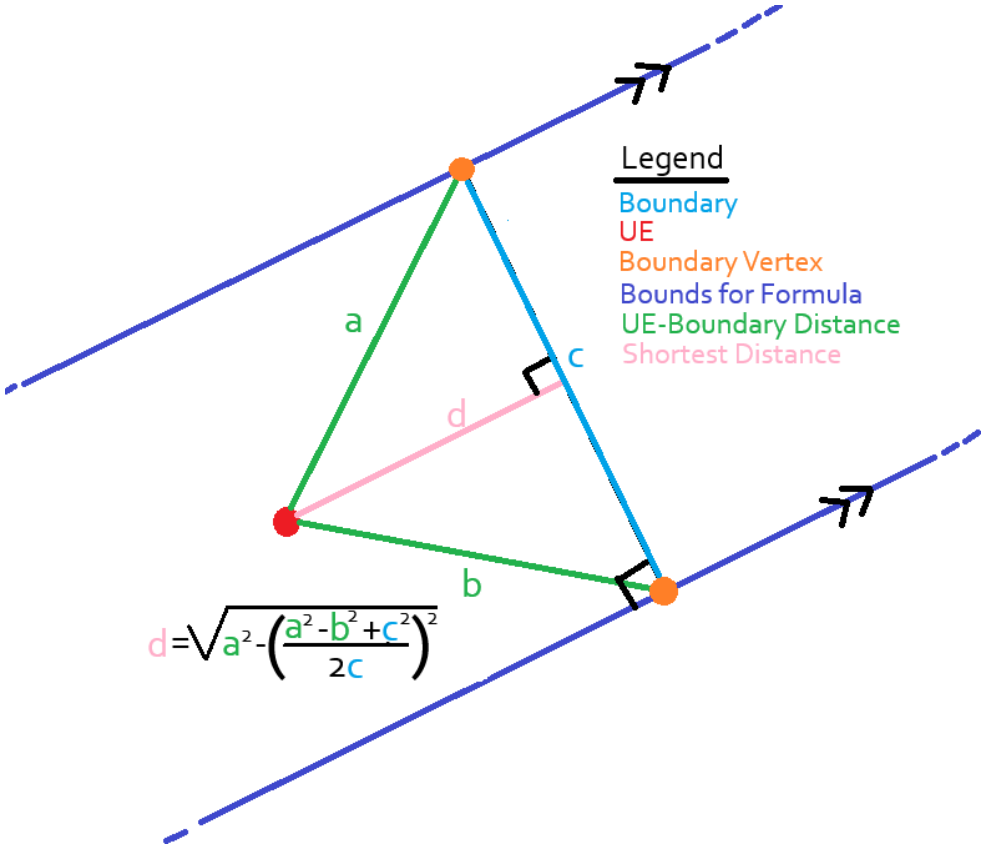


Figure 5.1.3.1

Please note, this formula only applies when the UE, represented by the red dot, is within the lines perpendicular to the boundary, represented by the dark blue. When the UE is not within these bounds, the program simply calculates the distance between the UE and the nearest boundary vertex, represented in orange, using the Pythagorean theorem.

Within the code, this formula is implemented in multiple places within the 'getDistances' function:

```
289        function getDistances(ueX, ueY, boundaryX, boundaryY) {
```

Each line is much longer than any other within the file, and starts as such:

```
distance.push(Math.sqrt((boundaryX[i]-ueX)*(boundaryX[i]-ueX)+(boundaryY[i]-ueY)*(boundaryY[i]-ueY)-Math.pow((((boundaryX[i]-ueX)*(boundaryX[i]-ueX)+(boundaryY[i]-ueY)*
```

The implementation of the formula requires the usage of the Pythagorean formula for each of the required distances, as can be seen in the above line of code, but is otherwise exactly the same as that included in the above diagram. If it is decided that the formula should be changed, all 4 of the longest lines in the 'getDistances' formula should be edited accordingly. Furthermore, the 'distance.push' section must remain. Without a sufficient software background, it is strongly

advised to avoid editing these lines; any of the included Team 12 contacts will be happy to implement a new formula, if so desired, upon contact.

5.1.4. Out of Bounds Checker Errors

Similar to 5.1.3, the out of bounds checker has functioned in all Team 12 testing, but may potentially be inaccurate in usage. In case this situation arises, the algorithm used to determine whether the UE is within its bounds, developed by Team 12, is an implementation of the ray casting algorithm (https://en.wikipedia.org/wiki/Point_in_polygon). It is shown below:

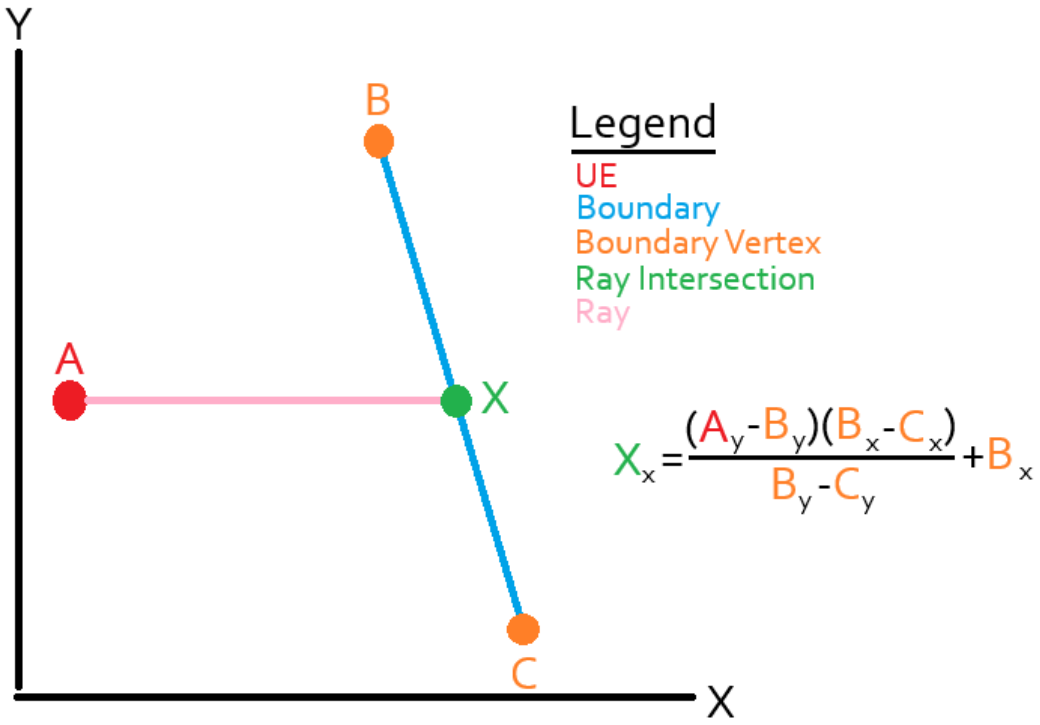


Figure 5.1.4.1

This implementation of the ray casting algorithm casts rays along the positive x-axis, so the algorithm checks if the calculated X_x point is to the right of the UE's x-coordinate, so long as the UE's y-coordinate lies somewhere in the range of that specific boundary, then counts the total number of boundaries crossed by the ray cast. If this number is odd, the UE is within bounds; even means it is outside bounds. The code for this algorithm can be found in the 'inBoundChecker' function:

```
370        function inBoundChecker(ueX, ueY, boundaryX, boundaryY) {
```

For the implementation of the algorithm itself, each of the 4 lines which looks generally like this

```
if(ueX <= ((ueY-boundaryY[i])*(boundaryX[i]-boundaryX[i+1])/(boundaryY[i]-boundaryY[i+1])+boundaryX[i]))
```

does the procedure described above. As in 5.1.3, without a background in software, it is not recommended to edit these lines of code. If an error relating to the within bound checker does arise, it is recommended to contact a member of Team 12.

5.2. Special Considerations

5.2.1. Hardware/Software Specifications

Team 12 developed the RCP exclusively using non-Apple devices running the Windows 11 OS. As such, it is uncertain whether the included files would work on Apple devices, or on devices with a different OS. Team 12 expects that the code should work fine on nearly all systems, but sections 3.2 and 4.1 may not accurately reflect the necessary procedures on devices other than those used in the development process.

5.2.2. Text Messaging Abroad

Team 12 is based in Canada, and chose to use Twilio's API based on this fact. It will work for certain in Canada and the United States, but it is uncertain whether phone numbers registered outside of these regions will be eligible for Twilio's service. In the case that these phone number **do not** work, the program should work as expected and described, **other than** the fact that text messages will not be able to be sent to the specified phone number whenever the UE crosses the user-specified boundary.

5.3. Maintenance

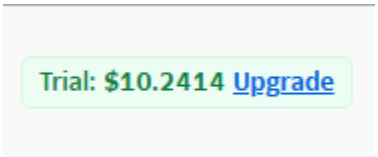
For the most part, there is not much maintenance which needs to be done for the RCP. However, there are still a few elements which should be periodically checked

5.3.1. Text Messaging Maintenance

The RCP uses Twilio's text messaging API to enable messages to be sent whenever a UE crosses the user-specified boundary. This API costs about 0.8₡ per text message:

https://www.twilio.com/en-us/sms/pricing/us

As such, messaging requires that the Twilio account specified in the server.js file has enough credit to send a text. This credit should be periodically checked by logging into Twilio and looking at the green box in the top-right corner:



Team 12 recommends always having enough account credit to maintain the product at expected use volumes for at least one week. As such, this amount should be periodically topped up.

5.3.2. Software Maintenance

All services used for the software should be kept up to date to ensure that no unforeseen errors arise. To update node.js, which is the framework used for the server.js file, follow the instructions in the following website:

https://www.freecodecamp.org/news/how-to-update-node-and-npm-to-the-latest-version/

Team 12 does not expect any of the other services used for their software to update, or, if updates do occur, for them to cause any errors within the code. If the software does break, refer to 5.4.4 for how to contact code functionality support, where a Team 12 member will help determine whether a missing software update is the culprit, as well as how to update that specific piece of software.

5.4. Support

Depending on the nature of the issue, there are different parties who should be contacted.

5.4.1. Device Tracking Support

For support related to the tracking of a real-world device, all questions and concerns should be directed to Shabodi by email, as their Location API is leveraged to get UE longitude and latitude. Their support email is as follows:

developer.support@shabodi.com

In Team 12 experience, Shabodi can be unreliable in responding to emails. As such, if Shabodi does not respond, please reach out to Team 12 directly instead. The process for doing so is described in 5.4.5.

5.4.2. Text Messaging Support

For support related to text message functionality, all questions and concerns should be directed to Twilio, as their text messaging API is leveraged to send customizable text messages. For AI support, the following website can be used to answer prompts:

https://help.twilio.com/

If this does not suffice, human support can be reached by purchasing a Support Plan:

https://www.twilio.com/en-us/support-plans

If neither of these options work, contacting Team 12, which is described in 5.4.5, is always possible. However, Team 12 does not know much about the functionality of Twilio's product, and may not be of much help as a result.

5.4.3. GUI Support

For support related to the GUI (excluding the mini-map; refer to 5.4.4), contact Shailen Mann, a member of Team 12, by email at:

smann016@uottawa.ca

Please provide him with a text description of the problem, along with as many pictures as possible, as this is a visual problem.

5.4.4. Code Functionality Support

For support related to the functionality of the code itself, whether it be server.js or gng1103website.html, contact Nicholas Pighin, a member of Team 12, by email at:

npigh082@uottawa.ca

Please provide him with as much text description of the problem as possible, along with pictures, if possible.

5.4.5. General Support

If support is required but does not fit within any of the previously described categories, any member of Team 12 may be contacted. Their emails are:

bbake087@uottawa.ca

smann016@uottawa.ca

npigh082@uottawa.ca

jroy069@uottawa.ca

As problems that do not fit into other categories may be inherently more difficult to solve, the contacted member of Team 12 may take a considerable amount of time to resolve the issue. To reduce this team, please provide as many text details as possible about the problem and system specifications, as well as any applicable pictures.

6.0. Product Documentation

As this product is mainly software based the only category that will be brought into consideration will be software. The final prototype was created by compiling the subsystems defined in the previous subsystems into one comprehensive and functional prototype. The subsystems designated in the previous prototypes were an alert sending system, a GUI and a location tracking system. Several past tests were already performed on each of these past subsystems, so the sole focus of this final prototype was to properly integrate these systems into each other. In these prototypes the code was divided into two separate coding languages with that being JavaScript and Python. It was decided that it would be simpler to have all of the code in the same language when trying to compile it into one prototype. The group decided to move all the code into JavaScript as this coding language is more tailored to the building of a website which is an important factor of this project.

In the final prototype the alert system used SMS messaging to alert the user as opposed to an email which the group had considered using at the beginning of the project. SMS was chosen as they are proven to be a more secure form of messaging which is important for the important information being sent. Another reason why SMS was chosen was that it is faster than an email and speed is a very important factor when considering the implications of what could occur when an autonomous UE leaves its designated zone.

In the final prototype it was also decided to use a formula created by one of our own members for the location tracking instead of a pre-existing library. This was decided as the formula created is tailored closely to the use case of our product while the general library which exists is more

general. In a situation where the code faces any issues it would be easier to fix a code which was created by one of our own members than one taken from a online library.

6.1. Alert Sending Subsystem

6.1.1. BOM (Bill of Materials)

Component	Quantity	Unit	Unit Cost	Total Cost	Link
Visual Studio Code	4	EA	0.00	0.00	https://code.visualstudio.com/
Twilio API	2	EA	0.00	0.00	Twilio
ECMAScript 2024 (JavaScript)	4	EA	0.00	0.00	https://nodejs.org/en/download/package-manager
node.js	1	EA	0.00	0.00	https://nodejs.org/en/download/package-manager
				Total	
			Added Total	0.00	
			Final	0.00	

6.1.2. Equipment list

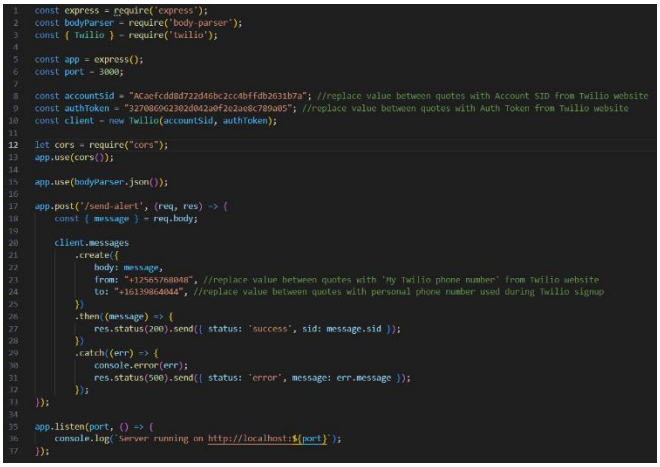
-Computer/Laptop (Device which code can be accessed from)

-Visual Studio Code

-Internet Connection

6.1.3. Instructions

1. Open a JavaScript file in vscode and download the necessary modules.
2. Insert the correct code to send a text message.



```
const express = require('express');
const bodyParser = require('body-parser');
const { Twilio } = require('twilio');

const app = express();
const port = 3000;

const accountSid = "ACaefcdd8d722d4bbc2cc4bffdb2631b7a"; //replace value between quotes with Account SID from Twilio website
const authToken = "32708696230?d042a0f2e2ae8c789a05"; //replace value between quotes with Auth Token from Twilio website
const client = new Twilio(accountSid, authToken);

let cors = require("cors");
app.use(cors());

app.use(bodyParser.json());

app.post('/send-alert', (req, res) => {
    const { message } = req.body;

    client.messages
        .create({
            body: message,
            from: "+12565768048", //replace value between quotes with 'My Twilio phone number' from Twilio website
            to: "+16139864044", //replace value between quotes with personal phone number used during Twilio signup
        })
        .then((message) => {
            res.status(200).send({ status: 'success', sid: message.sid });
        })
        .catch((err) => {
            console.error(err);
            res.status(500).send({ status: 'error', message: err.message });
        });
});

app.listen(port, () => {
    console.log(`Server running on http://localhost:${port}`);
});
```

3. Change the data correlating to the given Twilio account.

6.2. GUI and Tracking Subsystem

6.2.1. BOM (Bill of Materials)

Component	Quantity	Unit	Unit Cost	Total Cost	Link
Visual Studio Code	4	EA	0.00	0.00	https://code.visualstudio.com/
HTML 5	4	EA	0.00	0.00	https://html.spec.whatwg.org/multipage/
CSS3	4	EA	0.00	0.00	https://www.w3.org/TR/CSS/#css
ECMAScript 2024 (JavaScript)	4	EA	0.00	0.00	https://nodejs.org/en/download/package-manager
				Total	
			Added Total	0.00	

6.2.2. Equipment list

Equipment list:

-Computer/Laptop (Device which code can be accessed from)

-Visual Studio Code

-Internet Connection

6.2.3. Instructions

1. Open a HTML file in vscode.

2. Copy in the desired code from the file.

3. Change the desired size of the mini map to fit the display properly.

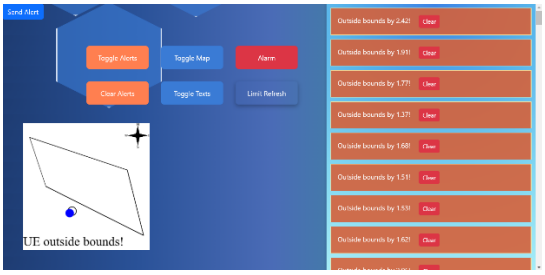
```
205     <canvas id="myCanvas" width="300" height="300" style="position:fixed; left:50px; bottom:50px"></canvas>
```

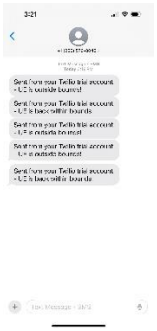
6.3. Testing & Validation

6.3.1. Alerts Tests

Test Descriptions	Verifies whether the prototype correctly detects when a user crosses a boundary triggers an appropriate alert in real-time
Reason for Prototype	Ensure Safety and Operational Efficiency --- Identifying breaches and addressing them promptly
Evaluation Criteria	Alerts are triggered upon boundary crossing and text is sent --- Alert contains accurate information
Level of Prototype	High fidelity --- Focused
Kind of Prototype	Visual
Metrics	Proportion (%) --- Delay (s)
Analysis Method	Log occurrences and times of triggered alerts for review
Stopping Criterion	When the UE completes 10 crosses without any unexpected system behaviour
Results	The system successfully detected all 10 crossings and displayed accurate alerts in real time and SMS are sent. Alerts were correctly dismissed when

	the clear button was clicked. No false positives or missed detections were observed.
Interpretation	Pass





This test was performed on the prototypes to determine the functionality of the alert system. This is a necessary test as it is needed to confirm that the alert system works correctly as the product is obsolete without it.

If a separate phone number is ever added or another Twilio account is used this test should be run again to ensure functionality. This test should also be done if the area of the containment zone is altered.

6.3.2 System Compatibility Test

Test Descriptions	Evaluates how different components of the zone restriction system work together to provide a seamless experience.
Reason for Prototype	Testing integration of multiple functions --- Individual components communicate effectively, preventing system failures
Evaluation Criteria	Alerts are generated when boundaries are crossed --- All components respond within acceptable time limits
Level of Prototype	High fidelity ---

	Focused
Kind of Prototype	Visual
Metrics	Proportion (%) --- Delay (s)
Analysis Method	Log system timestamps for each process (location update, alert generation, zone modification) and calculate response times.
Stopping Criterion	The test stops after successful verification of component interactions under two different scenarios: 1. No boundary crossings (system remains silent). 2. Boundary crossing detected and alert generated.
Results	The system demonstrated effective integration across all components. Location tracking maintained an accuracy of ±1 meter, alerts were generated in under 1 second. No conflicts or lags were observed, confirming seamless communication between components.
Interpretation	Pass

This test tests the functionality of the systems together in the final product. This test is necessary to determine if the systems can work together in a real-world scenario and to see if they work effectively and efficiently together.

If any new subsystems were introduced to the product this test would have to be redone to determine if the new systems can be properly and effectively integrated.

6.3.3 Button functionality test

Test Descriptions	Evaluates if the buttons can output their desired functions and properly modify the file.
Reason for Prototype	Testing functionality of buttons --- Buttons can work in conjunction with each other

Evaluation Criteria	Buttons can output their desired function --- The buttons can output their desired function within an acceptable time frame
Level of Prototype	High fidelity --- Focused
Kind of Prototype	Visual
Metrics	Delay (s)
Analysis Method	Test system to see if file is changing to output buttons desired functionality
Stopping Criterion	Buttons desired function is performed and can be reversed.
Results	Each button was able to perform its desired function.
Interpretation	Pass

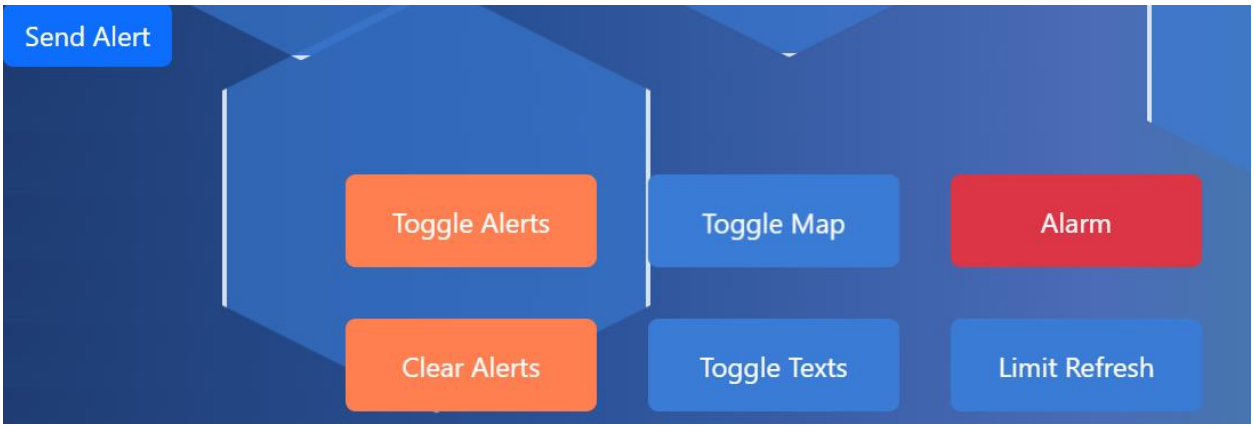


Figure 6.3.3.1

This test would need to be performed again if a new functionality for a button is created by the user.

7.0. Conclusions and Recommendations for Future Work

In conclusion, through the creation of the of the product we were able to create an effective and efficient GUI to access the zone restriction program completed by a SMS alert system. In the process of creating the final prototype it was determined that it is much easier to compile everything together in the same coding language if it is needed to be integrated together. Another productive avenue which the group decided to take was to create all the different prototypes in separate files and to have them work separately before having them work together. By doing this it is much easier to discover the flaws in each subsystem and identify why they would not work well together.

If more time was allocated to work on this project, one thing which the team would abandon would most likely be to create a more comprehensive GUI system which isn't mainly focused on the alerts. Due to the time restraints on this project, it was decided that the GUI would be as simplistic as possible as that is much more feasible though if more time was given a more comprehensive GUI would be a positive addition to the program.

8.0. Bibliography

Youtube.com:

Net Ninja. (2021, June 14). *Boostrap 5 Crash Course Tutorial #1 – Intro & Setup* [Video]. YouTube. https://www.youtube.com/watch?v=O_9u1P5YjVc&ab_channel=NetNinja

Net Ninja. (2021, June 15). *Boostrap 5 Crash Course Tutorial #3 – Colours & Typography* [Video]. YouTube. https://www.youtube.com/watch?v=iUCyU_U0J2E&t=204s&ab_channel=NetNinja

Net Ninja. (2021, June 16). *Boostrap 5 Crash Course Tutorial #4 – Buttons & Button Graphs* [Video]. YouTube. https://www.youtube.com/watch?v=ZZXGmoQ4PdI&ab_channel=NetNinja

ChatGPT:

https://chatgpt.com/share/674fbb17-215c-8004-befd-0f870dab3d06

https://chatgpt.com/share/674fbd0f-432c-8010-b0c3-27463167ad1b

Amazon Reviews:

AiPITE GPS Pet Tracker, Activity Monitor for Dog, Real-Time Location & Escape Alerts & Virtual Fences, No Monthly Fee & Support Multi User Co-Monitoring, Waterproof & Drop-Proof on Amazon

Tractive XL GPS Tracker for Large Dogs - Waterproof, GPS Location & Smart Pet Activity Tracker, Unlimited Range, Works with Any Collar (Green) on Amazon

GPS Wireless Dog Fence System, Electric Dog Fence by GPS Signal, Range 65-3280 ft, Adjustable Warning Strength, Rechargeable, Upgraded Pet Containment System, Harmless and Suitable for All Dogs on Amazon

W3Schools:

https://www.w3schools.com/

APPENDICES

APPENDIX I: Design Files

The user manual integrates all project stages, translating the team's work into a guide ensuring users can operate the system effectively. Ultimately, the manual bridges the technical design with practical implementation, showcasing the project's success through a user-focused perspective.

Table 3. Referenced Documents

Document Name	Document Location and/or URL	Issuance Date
Deliverable A	https://makerepo.com/NicholasPighin/2106.gng1103-group-12-zone-restriction	22/09/24
Deliverable B	https://makerepo.com/NicholasPighin/2106.gng1103-group-12-zone-restriction	29/09/24
Deliverable C	https://makerepo.com/NicholasPighin/2106.gng1103-group-12-zone-restriction	06/10/24
Deliverable D	https://makerepo.com/NicholasPighin/2106.gng1103-group-12-zone-restriction	13/10/24
Deliverable E	https://makerepo.com/NicholasPighin/2106.gng1103-group-12-zone-restriction	27/10/24
Deliverable F	https://makerepo.com/NicholasPighin/2106.gng1103-group-12-zone-restriction	03/11/24
Deliverable G	https://makerepo.com/NicholasPighin/2106.gng1103-group-12-zone-restriction	10/11/24
Deliverable H	https://makerepo.com/NicholasPighin/2106.gng1103-group-12-zone-restriction	24/11/24
Deliverable I	https://makerepo.com/NicholasPighin/2106.gng1103-group-12-zone-restriction	27/11/24
Deliverable J	https://makerepo.com/NicholasPighin/2106.gng1103-group-12-zone-restriction	11/11/24