CA Report

Miguel Angel Vinas
x22116133

OVERVIEW

This document is submitted as a submission of the CA project of the Distributed Systems module on the Higher Diploma in Computing specializing in Software Development delivered by Yasantha Samarawickrama.

Index

1. Introduction
Introduce the background of the project that you develop.

2. Service Definitions
Define the services selected for this report. Provide a concise description of each service and its role in the overall scenario.
Specify the message formats used for data exchange between services. Include details such as data structure, protocols, and any standards adhered to.

3. Service Implementations
Describe the overall architecture of the chosen services, outlining the components and their interactions.

4. Naming Services
Explain the naming conventions adopted for services.

5. Remote Error Handling & Advanced Features
Describe the mechanisms in place for remote error handling. Detail how errors are detected, reported, and resolved across services.

6. Client - Graphical User Interface (GUI)
Present the design principles and components of the graphical user interface for the client.

7. GitHub Integration
Provide insights into the organization of the GitHub repository, including the arrangement of code, documentation, and other relevant files.

8. Screenshots of the code
Provide screenshots of the code.

9. References
References.

1.

The project I present in this CA is a Smart Zebra Crosswalk system powered by LED lights.

This system aims to enhance pedestrian and traffic safety by implementing an intelligent crosswalk that uses LED lights as the marks on the street to signal the presence of pedestrians when they are around the zebra cross walk and while they are crossing.

The system dynamically activates the LED lights when a pedestrian is within a predefined radius or is actively crossing the road.

For the purpose of this project we are going to define the radius as 4 meters from the zebra walk on each end. These 4 meters can be used in any shape.

2.

This project contains three gRPC services, "CrosswalkMonitoringService", "LEDCrosswalkService" and "PedestrianDetectionService".
The "CrosswalkMonitoringService" handles different requests for the status, logs and history of each crosswalk (based on an ID number).
The "LEDCrosswalkService" activates or deactivates the LED lights of the crosswalks (based on the crosswalk ID).
The "PedestrianDetectionService" is designed to detect the presence of pedestrians within a 4 meters radius.
The message formats for all the services are specified in each protofile.
The server runs on my IP: "127.0.0.1" with the port "7343" and it communicates over gRPC.

3.

A) Crosswalk Monitoring Service.
- Role: Monitors the crosswalks.
    - Components:
    Client side uses HTML and Javascript for the user interface.
    On the gRPC client side we are using Javascript, it communicates with the server to get the crosswalk status.
    On the gRPC server side we are using node.js to handle the requests and provide the status.

B) LED Crosswalk Service.
- Role: Controls the activation or deactivation of the LED lights.
    - Components:
    Client side uses HTML and Javascript for the user interface.
    On the gRPC client side we are using Javascript, it communicates with the server to activate or deactivate the LED lights.

On the gRPC server side we are using node.js to handle the requests.

C) Pedestrian Detection Service.
- Role: Detects the presence of pedestrians within a 4 meters radius from the crosswalk.
- Components:
Client side uses HTML and Javascript for the user interface.
On the gRPC client side we are using Javascript, it communicates with the server to detect pedestrians.
On the gRPC server side we are using node.js to handle the requests.

4.
For the purpose of the project we are following the camelCase convention when writing the name of the services.
We are also aiming to provide clarity with the names and try to be descriptive when choosing them.

5.
Errors: We are not implementing any error handling mechanisms other than telling the user in the GUI that the ID of the crosswalk that the user has entered is not valid.

Advanced Features: We implemented a map in the GUI that pinpoints any particular crosswalk ID in the world via an API call to the Leaflet API.

I looked into making the call to the GoogleMaps API but it was not free and through some research I found two free options.

Option 1 was the Open Street Map API, which was very difficult to implement with my knowledge.

Option 2 was the Leaflet API, which was super simple to implement, it even had a tutorial! [1].

I believe that the implementation of the map offers a powerful tool to users.

In order to create the location of the crosswalks and to be able to implement them with the Leaflet API I created two test crosswalks, one shows the location of the National College of Ireland, the other one shows my house in Dublin.

Security: Our connections and bindings to the server (my computer in this particular case) are Insecure as per standard.
I would have loved to be able to implement a secure connection via SSL between the Server and the Client.

I believe that you can create a secure connection via SSL by loading the SSL certificate and the key in the server file and then loading the SSL certificate in the client file using a path to the SSL certificate.

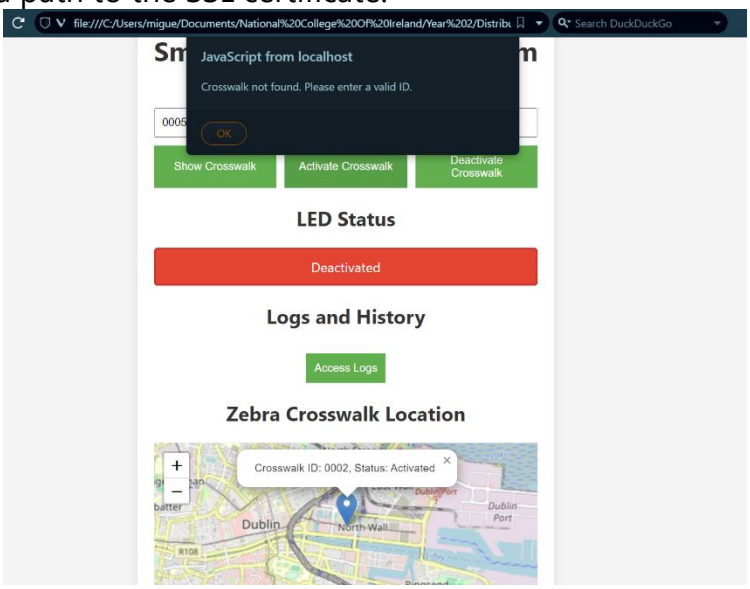

Figure 1 – Error handling when user inputs an invalid Crosswalk ID.

6.

As part of my background in UX design my intention was to create a GUI that was clear, concise and simple to understand and use with a specific use case of a desktop user.

Because I wanted to make the user journey as frictionless as possible I implemented a consistent styling that can be seen across the whole GUI with a defined colour. scheme and layout that is very visual.

The GUI is very clean with Input fields and buttons clearly labeled, big and colourful.

The system offers feedback to the user in four ways:
1) Via a pop up window when the user has entered a Crosswalk ID that is not in the system.
2) Via colour changes in the buttons when hoovering over them so the user knows that they can click on them.
3) On the LED Status button / panel. When the LED is deactivated the button has a red colour, indicating that it is deactivated, when the LED is activated the button has a green colour, indicating that it is activated.
4) Through the Zebra Crosswalk Location map where the user can see exactly where the entered crosswalk is in the world, the ID and the status.
I believe that I implemented a count of pedestrians at some point (I might have pushed the code to Github) and it worked but I found a problem with it, in the event of having an extremely long number it would have broken the page!.

GUI

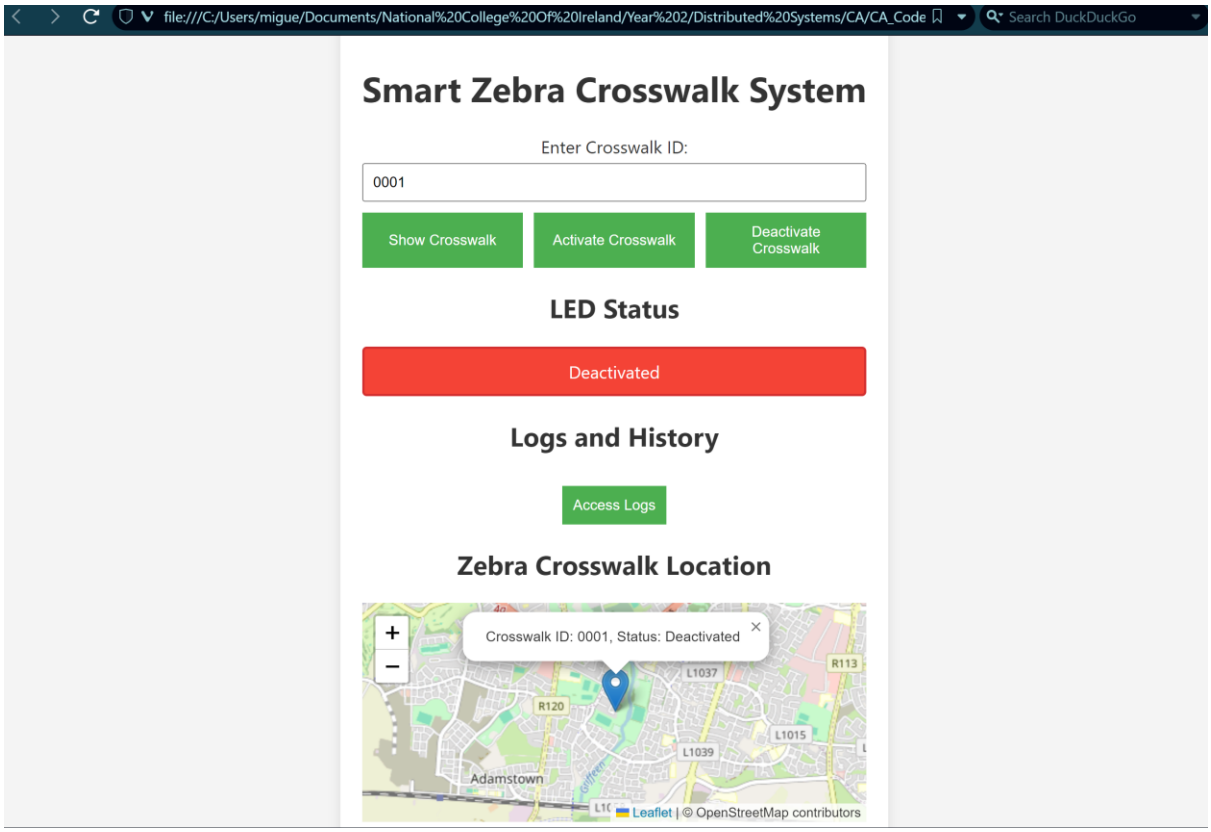

Figure 2: GUI showing the status and location of CrosswalkID with ID "0001"

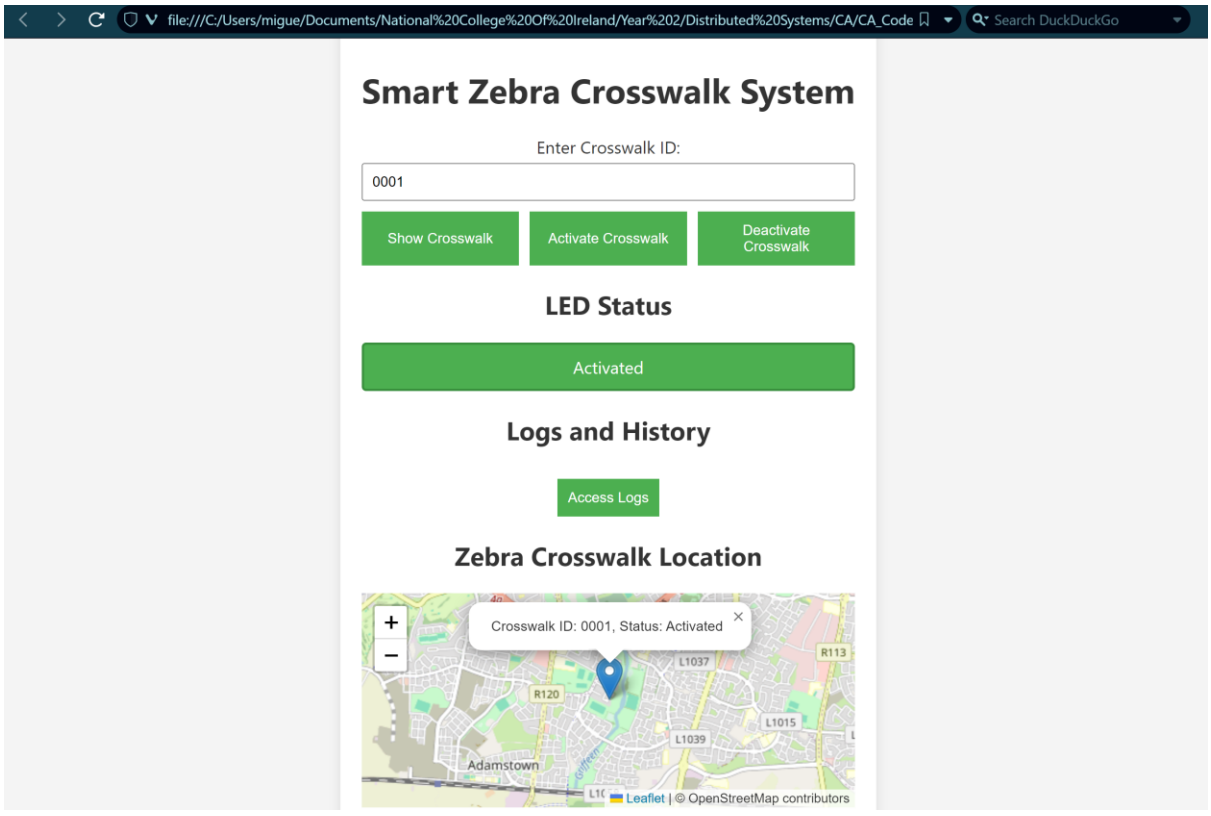

Figure 3: GUI showing the Activated Status (in green) of CrosswalkID with ID "0001" and the real time location and status in the map.

7.

The organization that created the Github repository is my company, Insurgent, which is the company working on this particular Smart Zebra Crosswalk System

I created an account linked to my personal email and then established a way of working through different stages, following the Software Development Lifecycle with a Test branch and a Production branch (called Main).

The test branch was used to write code and send it for a QA test before merging it to the Production branch.

Ideally I would have liked to create a UAT test branch (which I have in the Software Development TABA project from YEAR 1) but it was difficult to implement because I didn't have any user whom to test the GUI.

It is worth mentioning that I was going to create a repository in my Space account from IntelliJ Idea to implement this step but I quickly lost that battle because I couldn't use IntelliJ Idea Community Edition for HTML and Javascript so I used Github instead. I found that it is an easy tool to use.

Link to Insurgent's Github account: https://github.com/insurgent-github

Link to Github's project: https://github.com/insurgent-github/CA_Code

8.

crosswalk_monitoring_client.js

```
/*
Title: crosswalk_monitoring_client.js
Author: Miguel Angel Vinas
Date: 1st January, 2024
Purpose: Distribution's Systems CA for National College Of Ireland
*/

// We are importing the gRPC module
const grpc = require('grpc');

// We are loading the gRPC protocl buffer definition
const crosswalkMonitoringProto = grpc.load('protos/crosswalk_monitoring.proto');

// We are retrieveing the crosswalkMonitoringService definition from the protocol buffer
const crosswalkMonitoringService = crosswalkMonitoringProto.CrosswalkMonitoringService;

// Then we are creating a new gRPC client instance with the address in my computer and the specified credentials.
const client = new crosswalkMonitoringService('localhost:7345', grpc.credentials.createInsecure());

// We are creating the function to GET the status of a crosswalk with a specific ID.
function getCrosswalkStatus(crosswalkId)
{
  const request = new crosswalkMonitoringProto.CrosswalkStatusRequest();
  request.setCrosswalkId(crosswalkId);

  // We are making a gRPC request to get the status of the Crosswalk
  client.getCrosswalkStatus(request, (error, response) =>
  {
    // If there are no errors in the response
    if (!error)
    {
      // We are loging the status, error logs and the activation logs
      console.log(`Crosswalk Status: ${response.getStatus()}`);
      console.log(`Error Logs: ${response.getErrorLogsList()}`);
      console.log(`Activation Logs: ${response.getActivationLogsList()}`);
    }
    // But if there is an issue in the response, we are logging an error.
    else
    {
      console.error(`Error getting crosswalk status: ${error}`);
    }
  });
}

// Call the function to get crosswalk status
getCrosswalkStatus(/* We have to provide the crosswalkId here*/);
```

led_crosswalk_client

```
/*
Title: led_crosswalk_client.js
Author: Miguel Angel Vinas
Date: 1st January, 2024
Purpose: Distribution's Systems CA for National College Of Ireland
*/

// We are importing the gRPC module
const grpc = require('grpc');

// We are loading the gRPC protocol buffer definition
const ledCrosswalkProto = grpc.load('protos/led_crosswalk.proto');

// We are retrieveing the ledCrosswalkProto definition from the protocol buffer
const ledCrosswalkService = ledCrosswalkProto.led_crosswalk.LEDCrosswalkService;

// Then we are creating a new gRPC client instance with the address in my computer and the specified credentials.
const client = new ledCrosswalkService('localhost:7344', grpc.credentials.createInsecure());

// Function to make a request to activate LED lights
function activateLEDLights(crosswalkId, pedestrianPresence)
{
  const request =
  {
    crosswalkId: crosswalkId,
    pedestrianPresence: pedestrianPresence,
  };

  // We are making a gRPC request to activate the LED lights.
  client.ActivateLEDLights(request, (error, response) =>
  {
    // If there are no errors in the response
    if (!error)
    {
      // We will log the successful activation of the LED lights.
      console.log(`LED lights activated for Crosswalk ${crosswalkId}`);
    }
    // But if there is in issue with the request, we will log an error.
    else
    {
      console.error(`There has been an error activating the LED panel in the Crosswalk: ${error}`);
    }
  });
}


function deactivateLEDLights (crosswalkId, pedestrianPresence)
{
  const request =
  {
    crosswalkId: crosswalkId,
    pedestrianPresence: pedestrianPresence,

  };

 // We are making a gRPC request to deactivate the LED lights.
  client.deactivateLEDLights (request, (error, response) =>
  {
    // If there are no errors in the response
    if (!error)
    {
      // We will log the successful activation of the LED lights.
      console.log ('LED lights deactivated for Crosswalk ${crosswalkId}')
    }
    // But if there is in issue with the request, we will log an error.
    else
    {
      console.error ('There has been an error activating the LED panel in the Crosswalk: ${error}')
    }
  });
}

// Call the function to activate LED panel.
activateLEDLights('0001', true);

// Call the function to deactivate the LED panel.
deactivateLEDLights('0001', false);

```

pedestrian_detection_client.js

```
/*
Title: pedestrian_detection_client.js
Author: Miguel Angel Vinas
Date: 1st January, 2024
Purpose: Distribution's Systems CA for National College Of Ireland
*/

// We are importing the gRPC module
const grpc = require('grpc');

// We are loading the gRPC protocol buffer definition
const pedestrianDetectionProto = grpc.load('protos/pedestrian_detection.proto');

// We are retrieveing the pedestrianDetectionProto definition from the protocol buffer
const pedestrianDetectionService = pedestrianDetectionProto.pedestrian_detection.PedestrianDetectionService;

// Then we are creating a new gRPC client instance with the address in my computer and the specified credentials.
const client = new pedestrianDetectionService('localhost:7343', grpc.credentials.createInsecure());

// Function to make a request to detect pedestrians
function detectPedestrian()
{
  client.DetectPedestrian({}, (error, response) =>
  {
    // If there are no errors in the response
    if (!error)
    {
      // We will log the successfull presence of a pedestrian.
      const isPedestrianPresent = response.presence;
      console.log(`Pedestrian is present: ${isPedestrianPresent}`);
    }
    // But if there is in issue with the request, we will log an error.
    else
    {
      console.error(`Error detecting pedestrian: ${error}`);
    }
  });
}

// Call the function to detect pedestrians
detectPedestrian();
```

crosswalk_monitoring_server.js

```
/*
Title: crosswalk_monitoring_server.js
Author: Miguel Angel Vinas
Date: 1st January, 2024
Purpose: Distribution's Systems CA for National College Of Ireland
*/

// We are importing the gRPC module
const grpc = require('grpc');

// We are importing the generated gRPC services and also the message definitions
const { CrosswalkMonitoringService } = require('./crosswalk_monitoring_pb');
const { CrosswalkStatusRequest, CrosswalkStatusResponse } = require('./crosswalk_monitoring_pb');

// We are creating a new gRPC server instance
const server = new grpc.Server();

// And we are defining the implementation for the CrosswalkMonitoringService
class CrosswalkMonitoringServiceImpl
{
  // This is the implementation of te RPC method
  getCrosswalkStatus(call, callback)
  {
    // Implement logic to retrieve crosswalk status, error logs, and activation logs
    const response = new CrosswalkStatusResponse();
    response.setStatus(/* Write the logic to determine the status */);
    response.setErrorLogsList(/* This is an array of error logs */);
    response.setActivationLogsList(/* And this is an array of activation logs */);

    // And this is the callback to send the response back to the client
    callback(null, response);
  }
}

// We are adding the CrosswalkMonitoringService implementation to the server
server.addService(CrosswalkMonitoringService, new CrosswalkMonitoringServiceImpl());

// And we are binding the server to my specific IP address and port with insecure credentials
server.bind('127.0.0.1:7345', grpc.ServerCredentials.createInsecure());

// We are checking and loging that the server is running
console.log('Monitoring server running at http://127.0.0.1:7345');

// And we are starting our gRPC server
server.start();
```

led_crosswalk_server.js

```
/*
Title: led_crosswalk_server.js
Author: Miguel Angel Vinas
Date: 1st January, 2024
Purpose: Distribution's Systems CA for National College Of Ireland
*/

// We are importing the gRPC module
const grpc = require('grpc');

// We are importing the generated gRPC services and also the message definitions
const { LEDCrosswalkService } = require('./led_crosswalk_pb');
const { ActivateLEDLightsResponse } = require('./led_crosswalk_pb');

// We are creating a new gRPC server instance
const server = new grpc.Server();

// And we are defining the implementation for the LEDCrosswalkService
class LEDCrosswalkServiceImpl
{
  // This is the implementation of te RPC method
  activateLEDLights(call, callback)
  {
    const { crosswalkId, pedestrianPresence } = call.request;

    // Placeholder logic - replace with actual logic to activate LED panel
    // We will assume that the activation is successful for demonstration purposes
    const response = new ActivateLEDLightsResponse();
    response.setActivationStatus(true);

    console.log(`LED panel activated for Crosswalk ${crosswalkId}`);

    // And this is the callback to send the response back to the client
    callback(null, response);
  }

    // This is the implementation of te RPC method
   deactivateLEDLights(call, callback)
   {
    const { crosswalkId, pedestrianPresence } = call.request;

    // Placeholder logic - replace with actual logic to deactivate LED panel
    // We will assume that the deactivation is successful for demonstration purposes
    const response = new ActivateLEDLightsResponse();
    response.setActivationStatus(false);

    console.log(`LED panel deactivated for Crosswalk ${crosswalkId}`);

    // And this is the callback to send the response back to the client
    callback(null, response);
  }
}

// We are adding the LEDCrosswalkService implementation to the server
server.addService(LEDCrosswalkService, new LEDCrosswalkServiceImpl());

// And we are binding the server to my specific IP address and port with insecure credentials
server.bind('127.0.0.1:7344', grpc.ServerCredentials.createInsecure());

// We are checking and loging that the server is running
console.log('LED Crosswalk Server running at http://127.0.0.1:7344');

// And we are starting our gRPC server
server.start();

```

Pedestrian_detection_server.js

```
/*
Title: pedestrian_detection_server.js
Author: Miguel Angel Vinas
Date: 1st January, 2024
Purpose: Distribution's Systems CA for National College Of Ireland
*/

// We are importing the gRPC module
const grpc = require('grpc');

// We are importing the generated gRPC services and also the message definitions
const { PedestrianDetectionService } = require('./pedestrian_detection_pb');
const { DetectPedestrianResponse } = require('./pedestrian_detection_pb');

// We are creating a new gRPC server instance
const server = new grpc.Server();

// And we are defining the implementation for the LEDCrosswalkService
class PedestrianDetectionServiceImpl
{
  // This is the implementation of te RPC method
  detectPedestrian(call, callback)
  {
    // Placeholder logic - we will replace it with the actual logic to determine pedestrian presence
    const response = new DetectPedestrianResponse();
    response.setPresence(/* Actual logic to determine the presence of pedestrians */);

    // And this is the callback to send the response back to the client
    callback(null, response);
  }
}

// We are adding the PedestrianDetectionService implementation to the server
server.addService(PedestrianDetectionService, new PedestrianDetectionServiceImpl());

// And we are binding the server to my specific IP address and port with insecure credentials
server.bind('127.0.0.1:7343', grpc.ServerCredentials.createInsecure());

// We are checking and loging that the server is running
console.log('Server running at http://127.0.0.1:7343');

// And we are starting our gRPC server
server.start();

```

Style.css

```
/*
Title: style.css
Author: Miguel Angel Vinas
Date: 30th December, 2023
Purpose: Distribution's Systems CA for National College Of Ireland
*/


/* We are defining the custom font that I chose for my website, Segoe UI, which is included in the files */
@font-face
{
    font-family: 'Segoe UI';
    src: url('.../fonts/segoeui.ttf') format('ttf');
}

/* We are defining the general styling for the entire body of the website */
body
{
    font-family: 'Segoe UI', Tahoma, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
    display: flex;
    align-items: center;
    justify-content: center;
    height: 100vh;
}

/* We are defining the styling for the main container of the website */
.container
{
    max-width: 800px;
    margin: 50px auto;
    padding: 20px;
    background-color: #fff;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    border-radius: 8px;
    text-align: center;
}

/* We are defining the styling for the main heading of the website */
h1
{
    text-align: center;
    color: #333;
}

/* We are defining the styling for the controls section of the website */
.controls
{
    margin-bottom: 20px;
}

```

```css
/* We are defining the styling for the LABELS in the controls section of the website */
.controls label
{
    display: block;
    margin-bottom: 5px;
    color: #333;
}

/* We are defining the styling for the INPUT in the controls section of the website */
.controls input
{
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
    box-sizing: border-box;
}

/* We are defining the styling for the button container in the controls section of the website */
.button-container
{
    display: flex;
    gap: 10px;
    justify-content: center;
}

/* We are defining the styling for the Buttons of the website */
button
{
    background-color: #4caf50;
    color: #fff;
    padding: 10px;
    border: none;
    cursor: pointer;
}

/* We are defining the HOVER effect for the buttons of the website */
button:hover
{
    background-color: #45a049;
}

/* We are defining the styling for the BUTTONS within the button container of the website */
.button-container button
{
    flex: 1;
}

```

```css
/* We are defining the styling for the Activated Status */
.activated
{
    background-color: #4caf50;
    color: #fff;
    padding: 10px;
    border: 2px solid #388e3c;
    border-radius: 4px;
}

/* We are defining the styling for the Deactivated Status */
.deactivated
{
    background-color: #f44336;
    color: #fff;
    padding: 10px;
    border: 2px solid #d32f2f;
    border-radius: 4px;
}

/* We are defining the styling for the Status and the Logs section of the website */
.status, .logs
{
    margin-top: 20px;
}

/* We are defining the styling for the Subheadings of the website */
h2
{
    color: #333;
}

/* We are defining the styling for the buttons within the Log section of the website */
.logs button
{
    margin-top: 10px;
}

/* We are defining the styling for the paragraphs within the Log section of the website */
.logs p
{
    margin-top: 10px;
}

```

Script.js

```
/*
Title: index.html
Author: Miguel Angel Vinas
Date: 30th December, 2023
Purpose: Distribution's Systems CA for National College Of Ireland
*/

//We are going to import the Services that we have implemented.
// The name of the class, like MonitoringServiceClient is the client class that is generated from my protobuf definition for the monitoring service.
// We can choose any name! because they connect to the javascript file (this one) through the https address :)
// The address that we put here and the address and port of the gRPC server have to match of course.

/*
const pedestrianDetectionService = new pedestrianDetectionServiceClient ("https://localhost:7343");
const ledCrossWalkService = new LEDCrosswalkServiceClient ("https://localhost:7344");
const monitoringService = new MonitoringServiceClient ("https://localhost:7345");
*/

class Crosswalk
{
    constructor()
    {
        this.statusElement = document.getElementById('ledStatus');
        this.map = L.map('map').setView([51.505, -0.09], 13);
        L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
        {
            attribution: '© OpenStreetMap contributors'
        }).addTo(this.map);
        this.markers = L.layerGroup().addTo(this.map); // Layer group to manage markers

        //We are going to implement an object to store the logs of each Crosswalk.
        //The object is an array as we don't need anything else.
        this.logs = [];

        // Initialize the map with the default crosswalk state
        this.initializeMap();
    }

    // Function to get our application's crosswalk data
    getCrosswalkData()
    {
        //We will replace this with our data source or logic where to fetch the data from.
        return [
            { id: '0001', latitude: 53.34397088175478, longitude: -6.441806077184068, status: 'Activated' },
            { id: '0002', latitude: 53.34911487357206, longitude: -6.242949873623895, status: 'Activated' }
            // Add more crosswalks as needed
        ];
    }

    initializeMap()
    {
        // Add markers for all crosswalks in the initial state
        this.getCrosswalkData().forEach(crosswalk =>
        {
            this.addMarker([crosswalk.latitude, crosswalk.longitude], crosswalk.id, crosswalk.status);
        });
    }

    activate(crosswalkId)
    {
        this.statusElement.innerText = 'Activated';
        this.statusElement.classList.remove('deactivated');
        this.statusElement.classList.add('activated');

        // Update the map for activation
        this.updateMap(crosswalkId, 'Activated');
    }

    deactivate(crosswalkId)
    {
        this.statusElement.innerText = 'Deactivated';
        this.statusElement.classList.remove('activated');
        this.statusElement.classList.add('deactivated');

        // Update the map for deactivation
        this.updateMap(crosswalkId, 'Deactivated');
    }

    // Function to add a marker to the map
    addMarker(coordinates, crosswalkId, status)
    {
        // Remove previous markers
        this.markers.clearLayers();

        // Add a new marker for the given coordinates, crosswalk ID, and status
        const marker = L.marker(coordinates).addTo(this.markers);
        marker.bindPopup(`Crosswalk ID: ${crosswalkId}, Status: ${status}`).openPopup();
        this.map.setView(coordinates, 13);
    }

```

```
    // Function to update the map based on activation or deactivation
    updateMap(crosswalkId, status)
    {
        const selectedCrosswalk = this.getCrosswalkData().find(crosswalk => crosswalk.id.trim() === crosswalkId.trim());

        if (selectedCrosswalk)
        {
            // Update the map with the selected crosswalk and its new status
            this.map.setView([selectedCrosswalk.latitude, selectedCrosswalk.longitude], 13);
            this.addMarker([selectedCrosswalk.latitude, selectedCrosswalk.longitude], selectedCrosswalk.id, status);
        }
        else
        {
            alert('Crosswalk not found. Please enter a valid ID.');
        }
    }

    // Function to add a log entry
    addLog (crosswalkId, logMessage)
    {
        const logEntry = `Crosswalk ID: ${crosswalkId}, Log: ${logMessage}`;
        this.logs.push (logEntry);
    }

    //Function to display the logs.
    displayLogs()
    {
        const logsElement = document.getElementById ('logsResult');

        //We are going to clear previous logs (not sure if this is something that I might want to implement actually.)
        logsElement.innerText = '';

        //Display each log entry.
        this.logs.forEach (log =>
        {
            const logEntry = document.createElement ('p');
            logEntry.innerText = log;
            logsElement.appendChild (logEntry);
        });
    }

    // Function to remove all markers from the map
    removeMarkers()
    {
        this.map.eachLayer(layer =>
        {
            if (layer instanceof L.Marker)
            {
                this.map.removeLayer(layer);
            }
        });
    }
}

```

Index.html

```
<!--
Title: index.html
Author: Miguel Angel Vinas
Date: 30th December, 2023
Purpose: Distribution's Systems CA for National College Of Ireland
-->


<!DOCTYPE html>

<html lang="en">

    <head>
        <!-- START - This is the START of the Meta tags for character sets and for the viewport -->
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <!-- END - This is the END of the Meta tags for character sets and for the viewport -->

        <!-- START - This is the START to the link of my CSS stylesheet -->
        <link rel="stylesheet" href="css/style.css">
        <!-- END - This is the END to the link of my CSS stylesheet -->

        <!--START - This is the START of  Leaflet API integration for the MAP where the Zebra Crosswalks are -->
        <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"
         integrity="sha256-p4NxAoJBhIIN+hmNHrzRCf9tD/miZyoHS5obTRR9BMY="
         crossorigin=""/>
        <!-- Make sure you put this AFTER Leaflet's CSS (copied from Leaflet's website) -->
        <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"
         integrity="sha256-20nQCchB9co0qIjJZRGuk2/Z9VM+kNiyxNV1lvTlZBo="
         crossorigin=""></script>
        <!-- END - This is the END of  Leaflet API integration for the MAP where the Zebra Crosswalks are -->

        <title>Smart Zebra Crosswalk System</title>
    </head>

    <body>
```

```
        <div class="container">
            <!-- This is the main heading of the page -->
            <h1>Smart Zebra Crosswalk System</h1>

            <!--START - This is the START of the container where the user controls are -->
            <div class="controls">
                <label for="crosswalkId">Enter Crosswalk ID:</label>
                <input type="text" id="crosswalkId" placeholder="Enter Crosswalk Id here please.">

                <div class="button-container">
                    <button id="showCrosswalkButton">Show Crosswalk</button>
                    <button onclick="manualActivateCrosswalk()">Activate Crosswalk</button>
                    <button onclick="manualDeactivateCrosswalk()">Deactivate Crosswalk</button>
                </div>
            </div>
            <!--END - This is the END of the container where the user controls are -->

            <!--START - This is the START of the container where the status display is -->
            <div class="status">
                <h2>LED Status</h2>
                    <p id="ledStatus" class="deactivated">Inactive</p>
            </div>
            <!--END - This is the END of the container where the status display is -->

            <!--START - This is the START of the container where the Logs and History are -->
            <div class="logs">
                <h2>Logs and History</h2>
                    <button onclick="accessLogsAndHistory()">Access Logs</button>
                    <p id="logsResult"></p>
            </div>
            <!--START - This is the START of the container where the Logs and History are -->

        <!--START - This is the START of  Leaflet Map where Zebra Crosswalks are -->
            <div class="map">
                <h2>Zebra Crosswalk Location</h2>
                <div id="map" style = "height: 200px"></div>
            </div>
        <!--END - This is the END of  Leaflet Map where Zebra Crosswalks are -->

        </div>

        <!-- START - This is the START of my gRPC service client files, which is a placeholder for the moment -->
        <script src = ""></script>
        <script src = ""></script>
        <script src = ""></script>
         <!-- END - This is the END of my gRPC service client files, which is a placeholder for the moment -->

    <!-- START - This is the START of my javascript file where the website's logic is -->
    <script src="script/script.js"></script>
    <!-- END - This is the END of my javascript file where the website's logic is -->

    <!-- START - This is the START of the gRPC library -->
    <script src = "https://cdn.jsdelivr.net/npm/grpc-web@1.5.0/index.min.js"></script>

    <!-- <script src="https://cdn.jsdelivr.net/npm/grpc-web@1.2.1/dist/grpc-web.js"></script> this script made the page not work as intended and the console was
    saying that it couldn't find it -->

    <!-- END - This is the END of the gRPC library -->

    <!--START - This is the START of the Leaflet script where Zebra Crosswalks are -->
    <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
    <!--END - This is the END of the Leaflet script where Zebra Crosswalks are -->

</body>
</html>
```

Crosswalk_monitoring.proto

```
/*
Title: crosswalk_monitoring.proto
Author: Miguel Angel Vinas
Date: 1st January, 2024
Purpose: Distribution's Systems CA for National College Of Ireland
*/

// We are defining the syntax (proto3 version) of the protocol buffer file
syntax = "proto3";

// We are declaring the package for the Crosswalk Monitoring service
package crosswalk_monitoring;

//And we are defining the CrosswalkMonitoringService, which contains the RPC method
service CrosswalkMonitoringService
{
  // This is the RPC method to get the status of the crosswalk
  rpc getCrosswalkStatus(CrosswalkStatusRequest) returns (CrosswalkStatusResponse);
}

// We are defining the message for the request to get the crosswalk status
message CrosswalkStatusRequest
{
  string crosswalkId = 1;
}

// And we are defining the message for the response to get the crosswalk status
message CrosswalkStatusResponse
{
  string status = 1;
  repeated string errorLogs = 2;
  repeated string activationLogs = 3;
}

```

Led_crosswalk.proto

```
/*
Title: led_crosswalk.proto
Author: Miguel Angel Vinas
Date: 1st January, 2024
Purpose: Distribution's Systems CA for National College Of Ireland
*/

// We are defining the syntax (proto3 version) of the protocol buffer file
syntax = "proto3";

// We are declaring the package for the led_crosswalk service
package led_crosswalk;

//And we are defining the LEDCrosswalkService, which contains the RPC methods
service LEDCrosswalkService
{
  rpc ActivateLEDLights(ActivateLEDLightsRequest) returns (ActivateLEDLightsResponse);
  rpc DeactivateLEDLights(DeactivateLEDLightsRequest) returns (DeactivateLEDLightsResponse);
}

// We are defining the messages for the different requests
message ActivateLEDLightsRequest
{
  string crosswalkId = 1;
  bool pedestrianPresence = 2;
}

message ActivateLEDLightsResponse
{
  bool activationStatus = 1;
}

message DeactivateLEDLightsRequest
{
  string crosswalkId = 1;
  bool pedestrianPresence = 2;
}

message DeactivateLEDLightsResponse
{
  bool deactivationStatus = 1;
}

```

Pedestrian_detection.proto

```
/*
Title: pedestrian_detection.proto
Author: Miguel Angel Vinas
Date: 1st January, 2024
Purpose: Distribution's Systems CA for National College Of Ireland
*/

// We are defining the syntax (proto3 version) of the protocol buffer file
syntax = "proto3";

// We are declaring the package for the led_crosswalk service
package pedestrian_detection;

//And we are defining the PedestrianDetectionService, which contains the RPC method
service PedestrianDetectionService
{
  rpc DetectPedestrian(DetectPedestrianRequest) returns (DetectPedestrianResponse);
}

// We are defining the messages for the different requests
message DetectPedestrianRequest {}

message DetectPedestrianResponse
{
  bool presence = 1;
}
```

9.

[1] Leaflet – A JavaScript library for interactive maps. Agafonkin, Volodymyr, 2010 – 2023. [Online]. Accessed on January 2nd, 2023. Available: https://leafletjs.com