Circle Finder (Flask app)

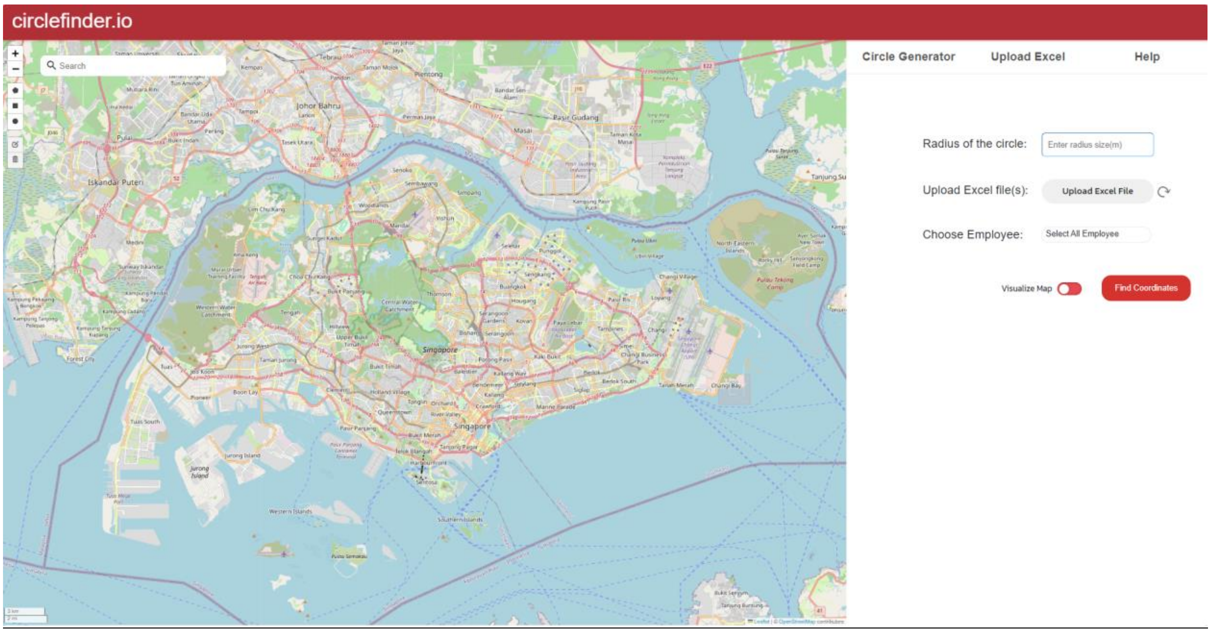
EXE application

Version 2.0 (LATEST)

Add on Purpose:

For the higher ups or the Head of QSD department (Quality and Service Department) to monitor mobile clock-ins of QSD employees at project sites.

Main Page:



Contents

Updates from V1.0

New Features:

- Circle Generator
  - Before Processing
    - Added auto generation of circles (Example)
  - After Processing
    - Able to edit the visuals (aka circles and markers) on the map, which will automatically update the corresponding details in the output container. (Example)

            - To ensure that the user can identify the new circle's coordinates or the deleted circle's coordinates is deleted from the output.
- Upload Excel
    - Before Processing
        - Clock Records
            - Able to auto read the latest excel sheet that contains the Clock Records of QSD employees from OneDrive (Code) (Encountered Problem)
            - It has Project Sites database (Code)
            - Filter employee (for faster processing) (Example)
            - Option to have map visualization (for faster processing)
            - Able to configure the radius of the project site (Default: 100m)
        - Other Excel Spreadsheets (Example)
            - Supports uploading of excel spreadsheets that mainly have Longitude and Latitude (and Radius if have)
            - Able to indicate the radius of the coordinates
            - Good for visualizing the previous generated circles for reference again
    - After Processing
        - Clock Records
            - For not filtering before processing
                - Able to filter the output changing the visuals on the map and the output container
            - Both
                - Able to zoom in to project sites and zoom out (Example)
                - Able to filter the dates (Example)
            - Able to download a zip file of Excel spreadsheet of the employee's details, status in a csv format, and a txt file format of the filtered employee, date output. (Example)
        - Other Excel Spreadsheets
            - No filtering
            - Shows the visuals
                - If there is radius in the excel, follows the radius in the excel spreadsheet
                - If there is radius in the input box, it will replace the initial radius value of each circle
            - Able download the text document of the output container
- Both features (Example)
    - The visuals can interact with the output container
        - For example, when you click the marker or the circle, it will show the name of it and will scroll to the details of it in the output box

Improvements from previous version (Circle Generator):

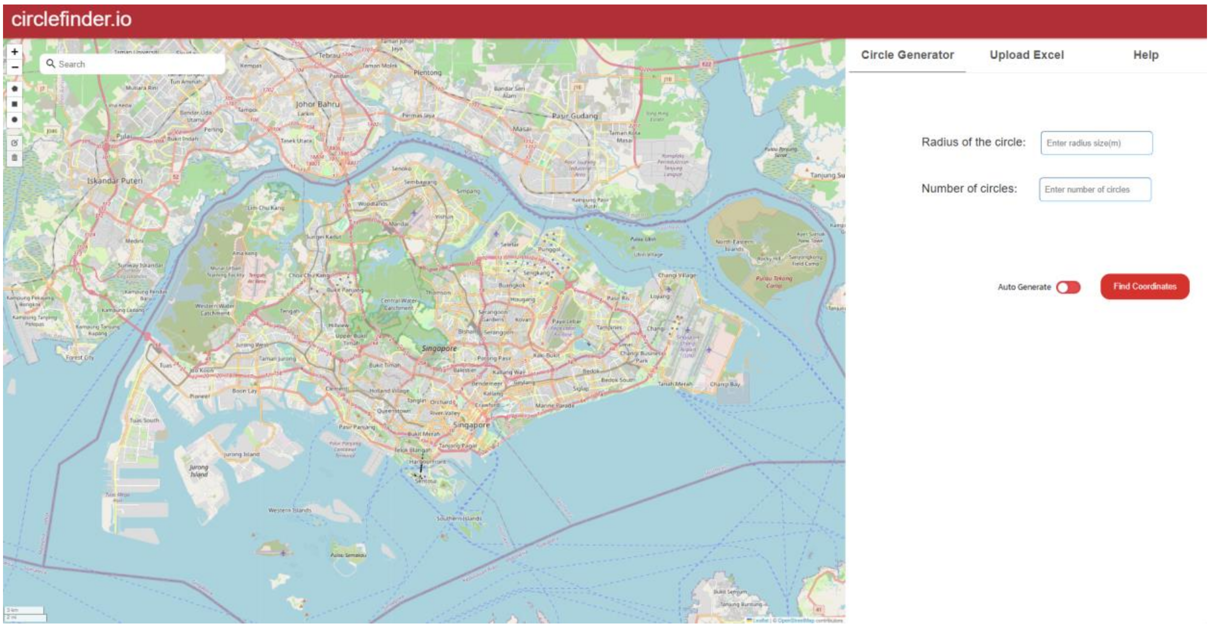
- Better algorithm to generate the circles to cover the whole land (Code)
- Previously, you could only download the txt file of the output container. But now you can download a zip file containing the txt file and the excel spreadsheet of the circle's coordinates, radius. (Example)

How to use

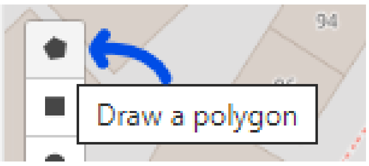
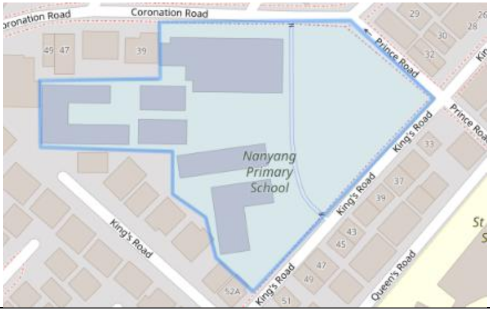
2 Functions:

- Circle Generator (Always Randomized)
- Upload Excel

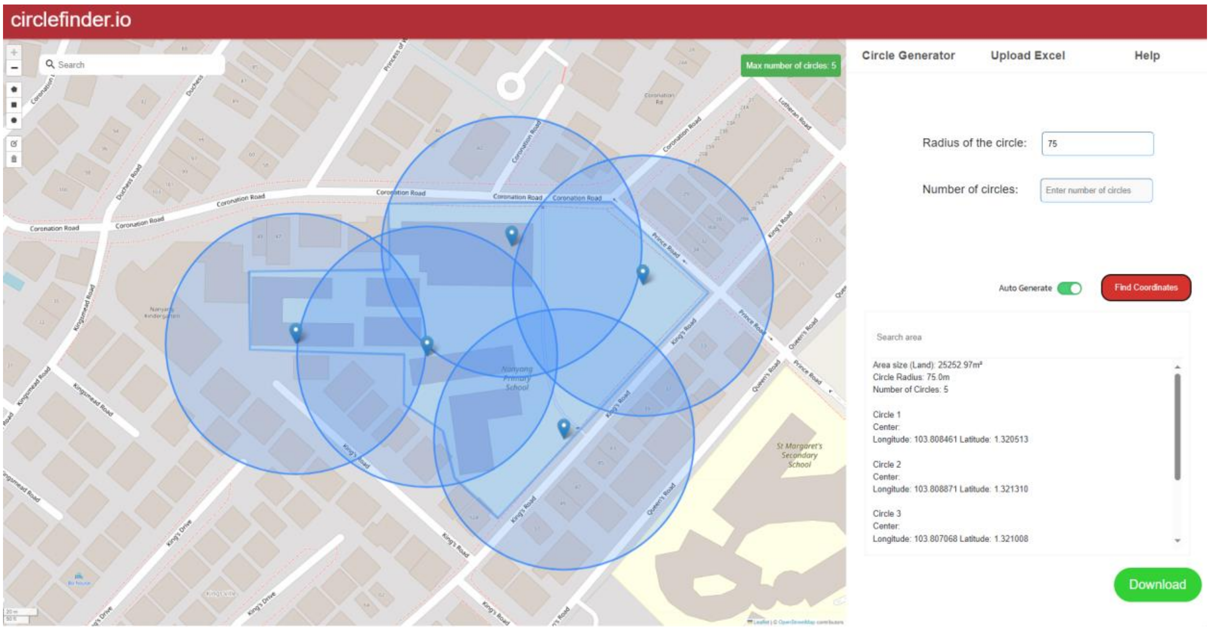
Circle Generator



Features before process:

	Draw a polygon on the map (Required)  Coronation Road Nanyang Primary School King's Road Prince Road Queen's Road
Radius of the circle: Enter radius size(m)	Enter the radius of the circle and define it in meters (Required) Example: 50, 24.5,100
Number of circles: Enter number of circles	Enter the number of circles that you would like (Optional – if you switched on auto generate)
Auto Generate	If you do not know how many circles you want or want to cover the whole land, switch this on. When this is switched on, it will ignore the input of the number of circles and disable the input box. (Optional)
Find Coordinates	Press the button to start the process

- With Auto Generate (Code)
  Example: 75m



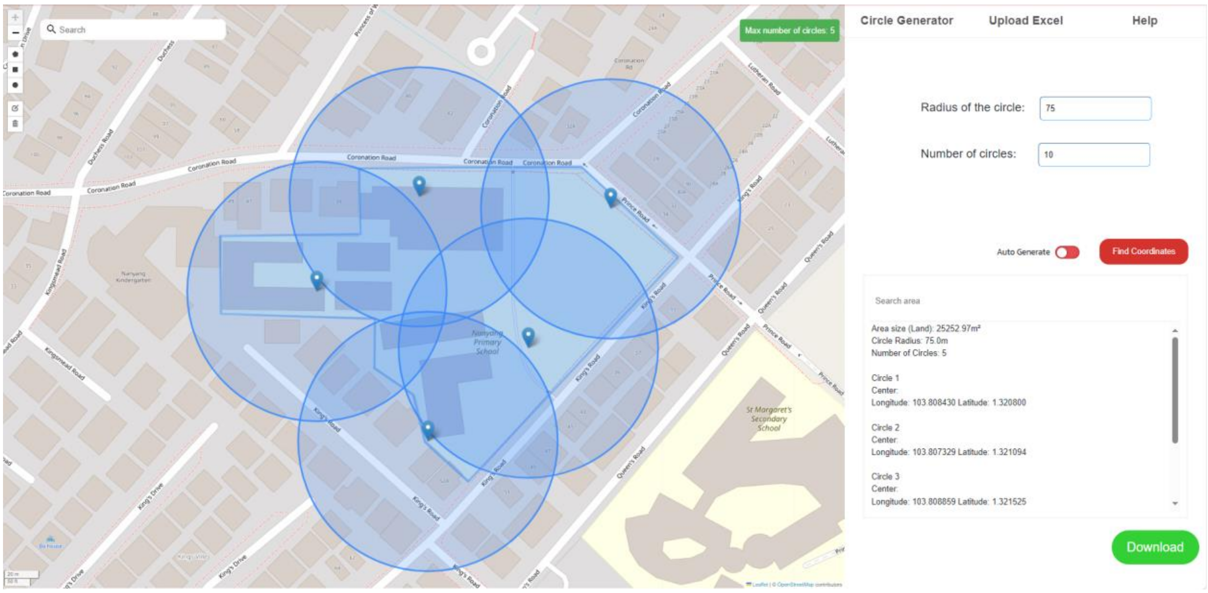
Radius of the circle: 75

Number of circles: Enter number of circles

Auto Generate ● (on)

```
Area size (Land): 25252.97m²
Circle Radius: 75.0m
Number of Circles: 5

Circle 1
Center:
Longitude: 103.808461 Latitude: 1.320513

Circle 2
Center:
Longitude: 103.808871 Latitude: 1.321310

Circle 3
Center:
Longitude: 103.807068 Latitude: 1.321008
```

- Without Auto Generate
  If the user specifies more circles than necessary to cover the land, adjustments will be made based on the actual area of the land.



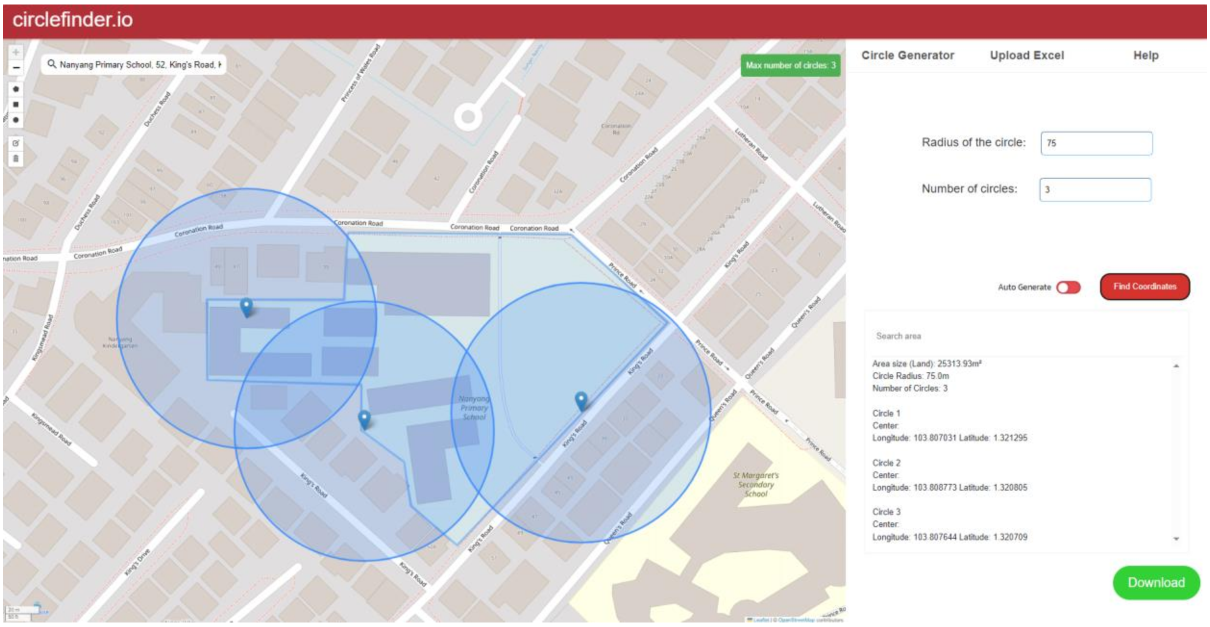
Radius of the circle: 75

Number of circles: 10

Auto Generate ○ (off)

```
Area size (Land): 25252.97m²
Circle Radius: 75.0m
Number of Circles: 5

Circle 1
Center:
Longitude: 103.808430 Latitude: 1.320800

Circle 2
Center:
Longitude: 103.807329 Latitude: 1.321094

Circle 3
Center:
Longitude: 103.808859 Latitude: 1.321525
```

Normal Output



Radius of the circle: 75

Number of circles: 3

Search area

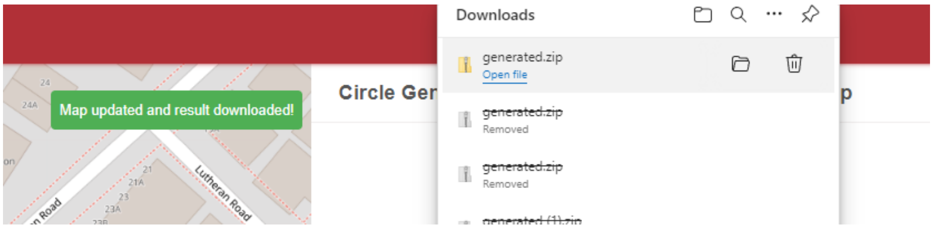
Area size (Land): 25313.93m²
Circle Radius: 75.0m
Number of Circles: 3

Circle 1
Center:
Longitude: 103.807031 Latitude: 1.321295

Circle 2
Center:
Longitude: 103.808773 Latitude: 1.320805

Circle 3
Center:
Longitude: 103.807644 Latitude: 1.320709

- Download



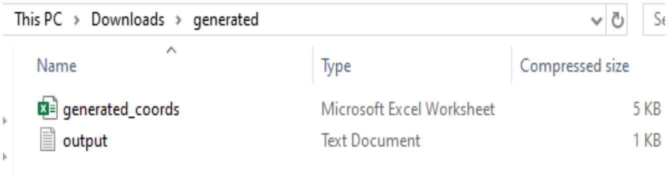
Map updated and result downloaded!

generated.zip

~~generated.zip~~ Removed

~~generated.zip~~ Removed

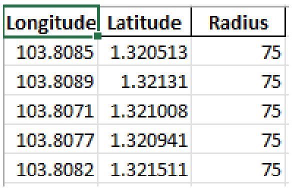
~~generated (1).zip~~



This PC › Downloads › generated

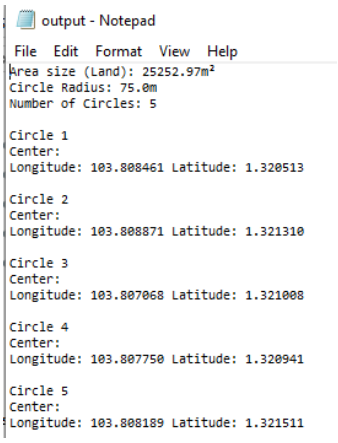
Name	Type	Compressed size
generated_coords	Microsoft Excel Worksheet	5 KB
output	Text Document	1 KB

  - generated_coords.xlsx (from auto generate)



Longitude	Latitude	Radius
103.8085	1.320513	75
103.8089	1.32131	75
103.8071	1.321008	75
103.8077	1.320941	75
103.8082	1.321511	75

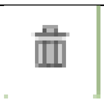
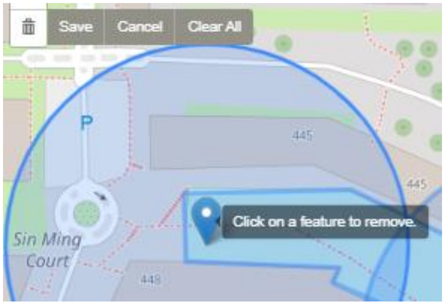
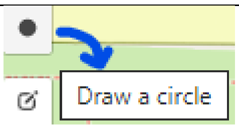
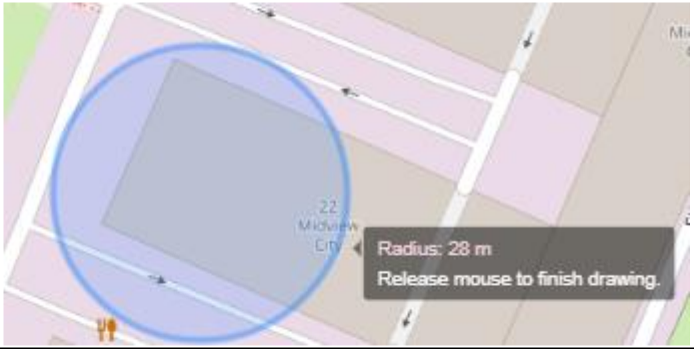
  - output.txt



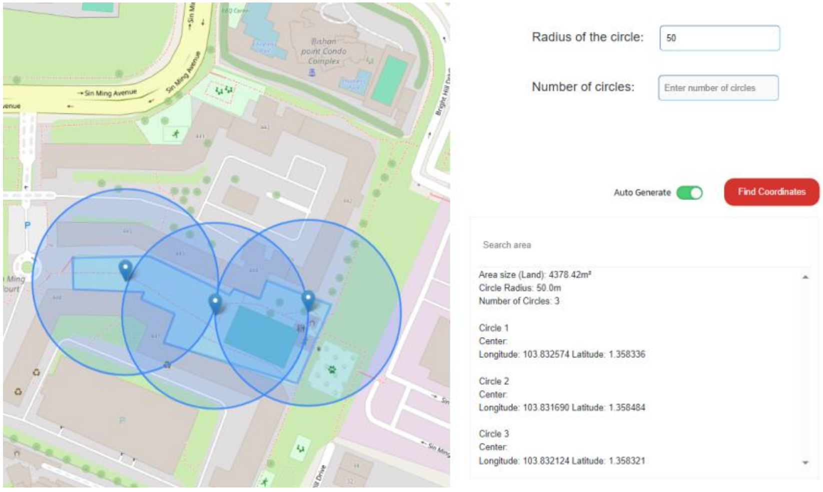
```
Area size (Land): 25252.97m²
Circle Radius: 75.0m
Number of Circles: 5

Circle 1
Center:
Longitude: 103.808461 Latitude: 1.320513

Circle 2
Center:
Longitude: 103.808871 Latitude: 1.321310

Circle 3
Center:
Longitude: 103.807068 Latitude: 1.321008

Circle 4
Center:
Longitude: 103.807750 Latitude: 1.320941

Circle 5
Center:
Longitude: 103.808189 Latitude: 1.321511
```

- Editing generated circle output

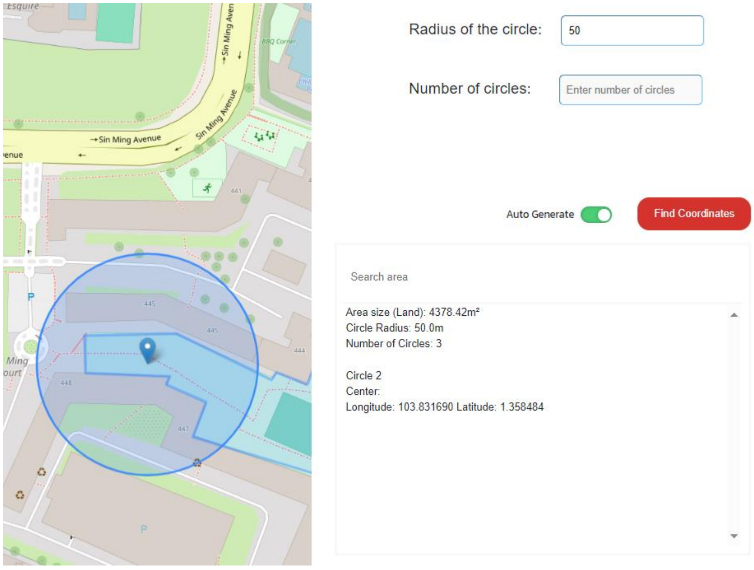
Features after process:

	To delete a circle, you need to click on the trash bin icon, it will prompt this out.  Save Cancel Clear All Click on a feature to remove. Delete the circles that you do not like and delete the marker in the center of the circle to have your output affected. Then, press save to secure the visuals and the output.
 Draw a circle	To draw a circle, click on the circle icon, and click on the spot that you want and drag.  Radius: 28 m Release mouse to finish drawing.

Example:

Generated output	Map and the output text:  Radius of the circle: 50 Number of circles: Enter number of circles Auto Generate Find Coordinates Search area Area size (Land): 4378.42m² Circle Radius: 50.0m Number of Circles: 3 Circle 1 Center: Longitude: 103.832574 Latitude: 1.358336 Circle 2 Center: Longitude: 103.831690 Latitude: 1.358484 Circle 3 Center: Longitude: 103.832124 Latitude: 1.358321

<table>
<tr>
<td></td>
<td>
Dataframe:
<table>
<tr><th>Longitude</th><th>Latitude</th><th>Radius</th></tr>
<tr><td>103.8326</td><td>1.358336</td><td>50</td></tr>
<tr><td>103.8317</td><td>1.358484</td><td>50</td></tr>
<tr><td>103.8321</td><td>1.358321</td><td>50</td></tr>
</table>
</td>
</tr>
<tr>
<td>Delete circle
(Code)</td>
<td>
Map and the output text:



Radius of the circle: 50

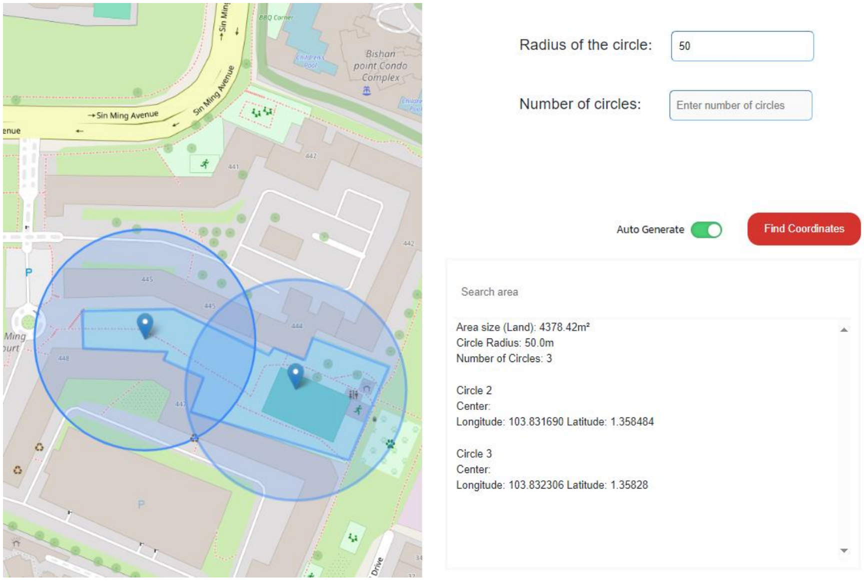
Number of circles: Enter number of circles

Search area

Area size (Land): 4378.42m²
Circle Radius: 50.0m
Number of Circles: 3

Circle 2
Center:
Longitude: 103.831690 Latitude: 1.358484

Dataframe:
<table>
<tr><th>Longitude</th><th>Latitude</th><th>Radius</th></tr>
<tr><td>103.8317</td><td>1.358484</td><td>50</td></tr>
</table>
</td>
</tr>
<tr>
<td>Create Circle
(Code)</td>
<td>
Map and the output text:



Radius of the circle: 50

Number of circles: Enter number of circles

Search area

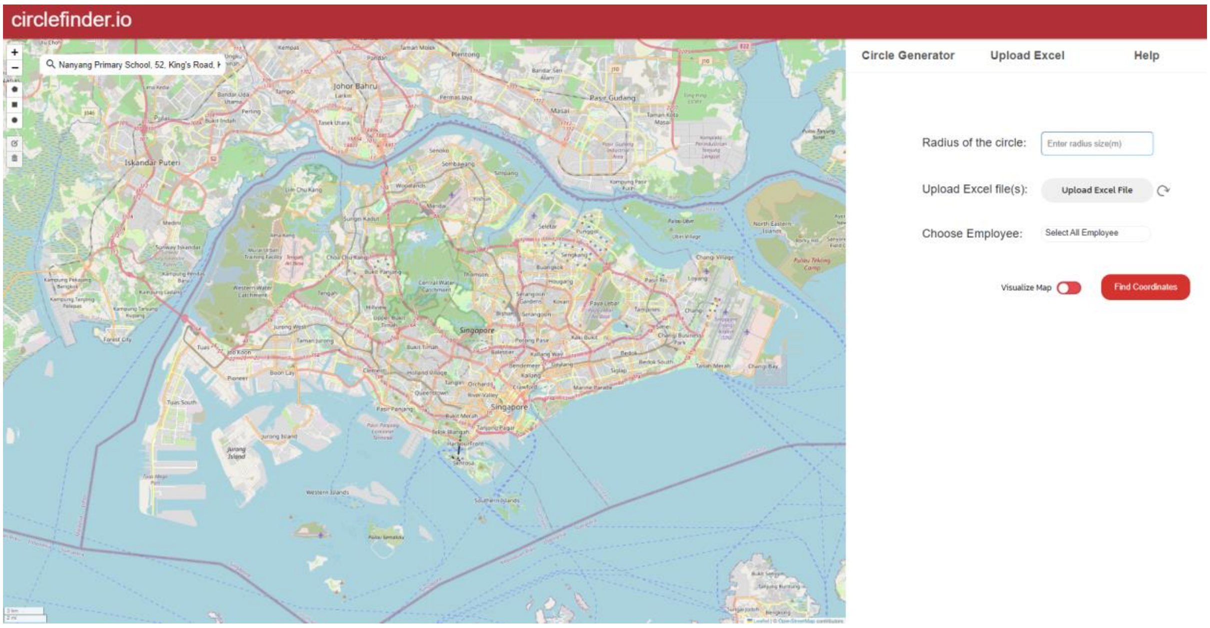
Area size (Land): 4378.42m²
Circle Radius: 50.0m
Number of Circles: 3

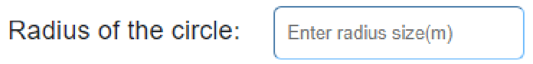
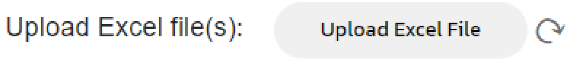
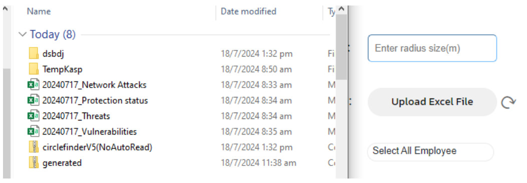
Circle 2
Center:
Longitude: 103.831690 Latitude: 1.358484

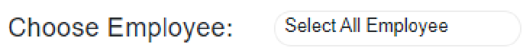
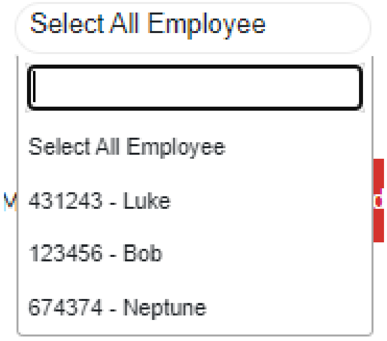
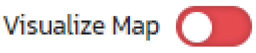
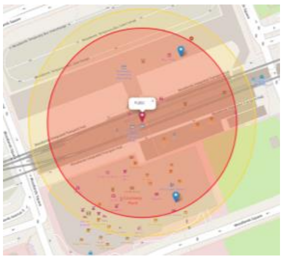
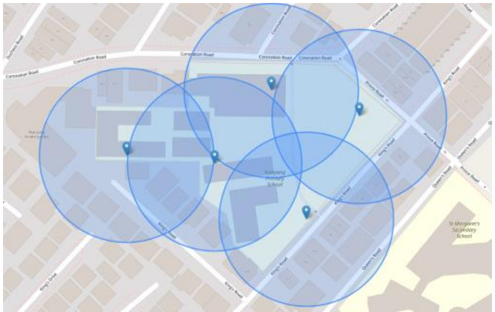
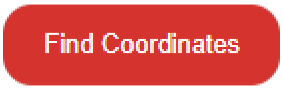
Circle 3
Center:
Longitude: 103.832306 Latitude: 1.35828

Dataframe:
<table>
<tr><th>Longitude</th><th>Latitude</th><th>Radius</th></tr>
<tr><td>103.8317</td><td>1.358484</td><td>50</td></tr>
<tr><td>103.8323</td><td>1.35828</td><td>50</td></tr>
</table>
</td>
</tr>
</table>

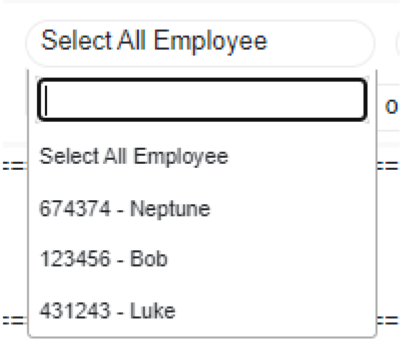
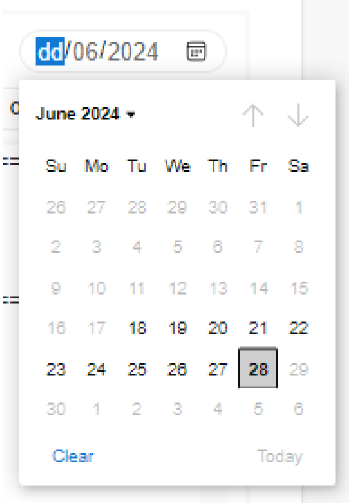
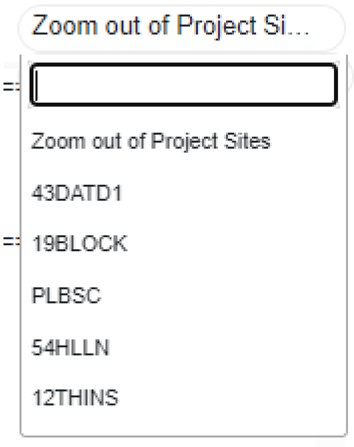
Upload Excel



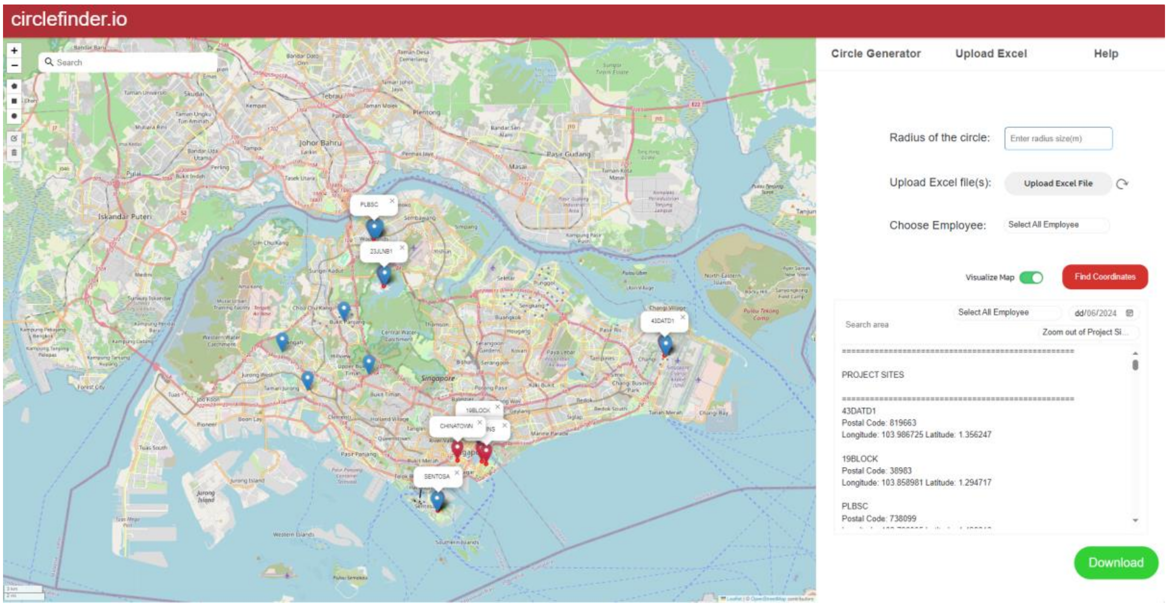
 Radius of the circle: Enter radius size(m)	Enter the radius of the circle and define it in meters (Optional) (The radius can be adjusted for the project site or for the other excel) Example: 50, 24.5,100
 Upload Excel file(s): Upload Excel File ⟳ (Manual Upload for Clock Records Code) (Auto read for Clock Records Code)	If the latest Clock Records cannot be auto read or you want to upload a spread sheet that has Longitude, Latitude and may have Radius. Press the button and it will lead you here, to choose the file. (Required) Since the auto read is not working for now, please upload it manually for Clock Records.  Name / Date modified Today (8) dsbdj 18/7/2024 1:32 pm TempKasp 18/7/2024 8:50 am 20240717_Network Attacks 18/7/2024 8:33 am 20240717_Protection status 18/7/2024 8:34 am 20240717_Threats 18/7/2024 8:34 am 20240717_Vulnerabilities 18/7/2024 8:35 am circlefinderV5(NoAutoRead) 18/7/2024 1:32 pm generated 18/7/2024 11:38 am Enter radius size(m) / Upload Excel File / Select All Employee (If Auto Read is working) Other case if you are done visualizing the uploaded file and would like to refer to the previous data that was auto read click here ⟳

 	If you have uploaded the file, or the auto read was successful, you will be able to choose the employee that you would like to see only before processing. (Optional) (This is to process faster than selecting all the employee)  Select All Employee Select All Employee 431243 - Luke 123456 - Bob 674374 - Neptune
 	This is to visualize the data that you have sent through when you switch it on (Optional) (This is for faster processing as it takes time to place down the layers) Clock Records:  Others (Previous data from Auto Generate): 
 	Press the button once done choosing

- Clock Records
    - Filters that you will encounter

 Select All Employee Select All Employee 674374 - Neptune 123456 - Bob 431243 - Luke (Code)	If you choose all employee before processing, you will get to filter the output container and the map visuals Search Bar: To search the name or the code Example use
 dd/06/2024 June 2024 Su Mo Tu We Th Fr Sa 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 Clear Today (Code)	The range is automatically determined based on the earliest and latest Clock Record dates, providing a clear timeframe for the user. This eliminates the need for users to manually select each day to understand the span of the records, streamlining the process. Example use
 Zoom out of Project Si… Zoom out of Project Sites 43DATD1 19BLOCK PLBSC 54HLLN 12THINS (Code)	To easily zoom into the project site area or zoom out. Search Bar: To search for the project site or code for faster search Example use

- If the filter is set to Select All Employee (Takes longer)



Radius of the circle: Enter radius size(m)

Upload Excel file(s): Upload Excel File

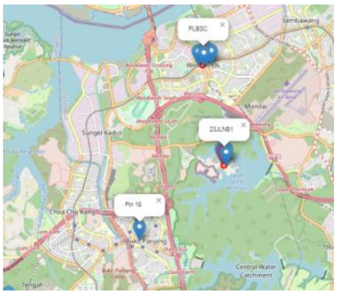
Choose Employee: Select All Employee

Search area | Select All Employee | dd/06/2024 | Zoom out of Project Si...

```
=====================================================
PROJECT SITES
=====================================================
43DATD1
Postal Code: 819663
Longitude: 103.986725 Latitude: 1.356247

19BLOCK
Postal Code: 38983
Longitude: 103.858981 Latitude: 1.294717

PLBSC
Postal Code: 738099
```

How closest project site is determined: (Code)

It takes 23JLNB1 because it's close to the employee.



Pin 19

Employee Name: 674374 - Neptune

Badge ID: 144-593-845

Date: 22/06/2024 08:43:00

Project Site: 23JLNB1

Location Name: Yishun Avenue 1, Singapore 769130

Center:

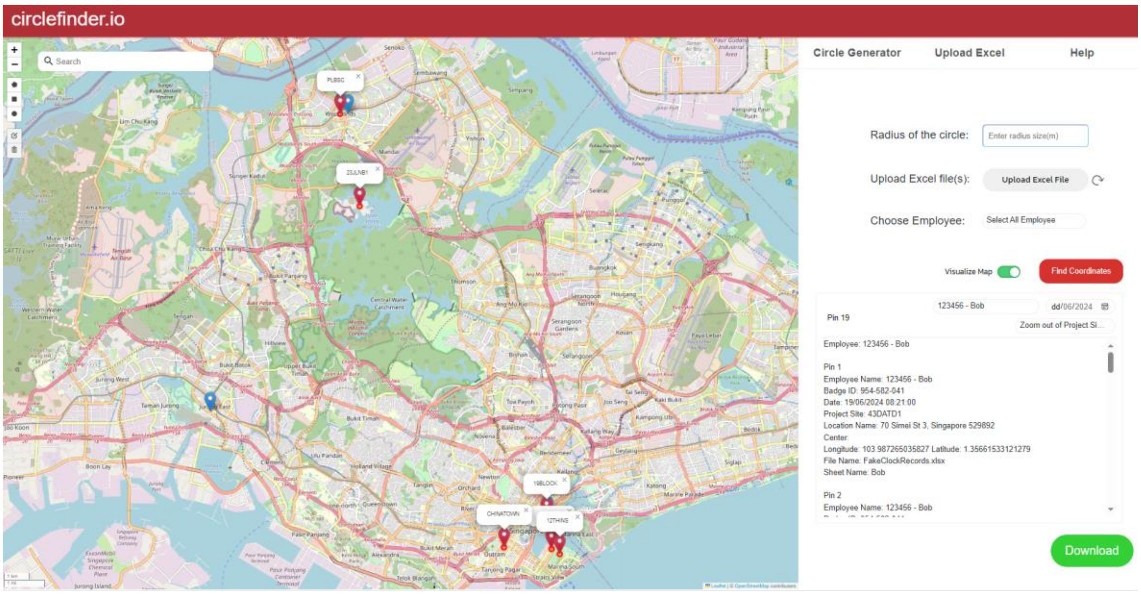
Longitude: 103.7656 Latitude: 1.3797

File Name: FakeClockRecords.xlsx

Sheet Name: Neptune

It will still provide an employee filtering for the output container and the visuals on the map

Example of employee filtering:



Radius of the circle: Enter radius size(m)

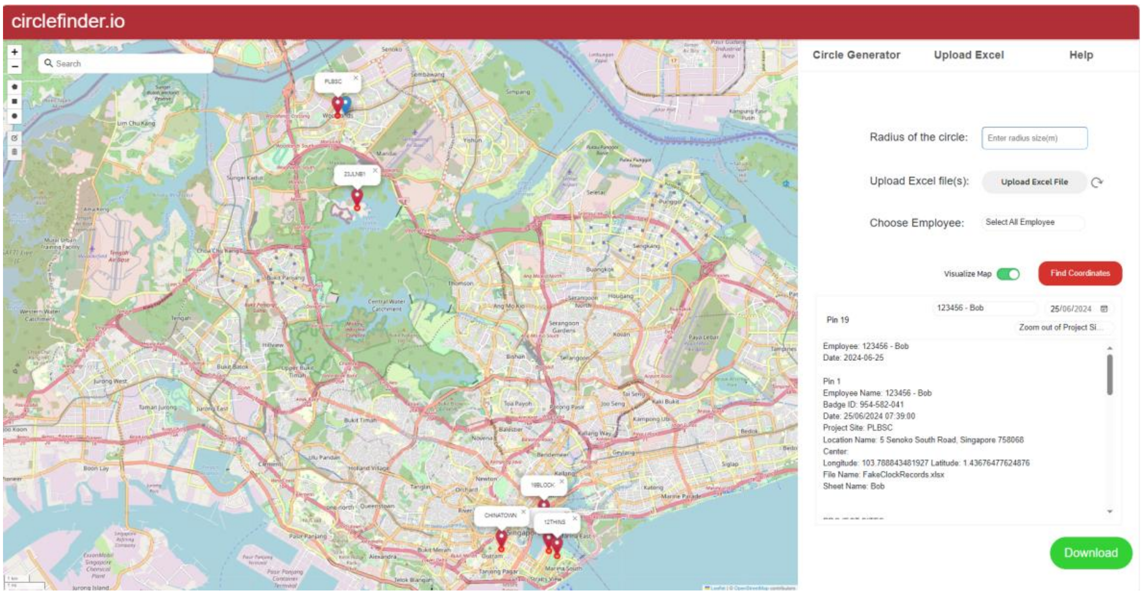
Upload Excel file(s): Upload Excel File

Choose Employee: Select All Employee

Pin 19 | 123456 - Bob | dd/06/2024 | Zoom out of Project Si...

```
Employee: 123456 - Bob

Pin 1
Employee Name: 123456 - Bob
Badge ID: 954-582-041
Date: 19/06/2024 08:21:00
Project Site: 43DATD1
Location Name: 70 Simei St 3, Singapore 529892
Center:
Longitude: 103.987265035827 Latitude: 1.35661533121279
File Name: FakeClockRecords.xlsx
Sheet Name: Bob

Pin 2
Employee Name: 123456 - Bob
```

Example of employee filtering and date filtering:



Radius of the circle: Enter radius size(m)

Upload Excel file(s): Upload Excel File

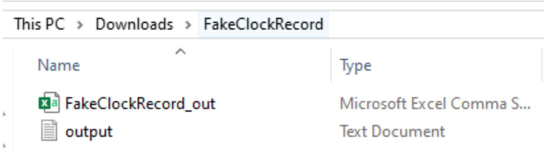
Choose Employee: Select All Employee

Pin 19 123456 - Bob 25/06/2024 Zoom out of Project Si...

Employee: 123456 - Bob
Date: 2024-06-25

Pin 1
Employee Name: 123456 - Bob
Badge ID: 954-582-041
Date: 25/06/2024 07:39:00
Project Site: PLBSC
Location Name: 5 Senoko South Road, Singapore 758068
Center:
Longitude: 103.788843481927 Latitude: 1.43676477624876
File Name: FakeClockRecords.xlsx
Sheet Name: Bob

Downloading the filtered output:



This PC › Downloads › FakeClockRecord

Name	Type
FakeClockRecord_out	Microsoft Excel Comma S...
output	Text Document

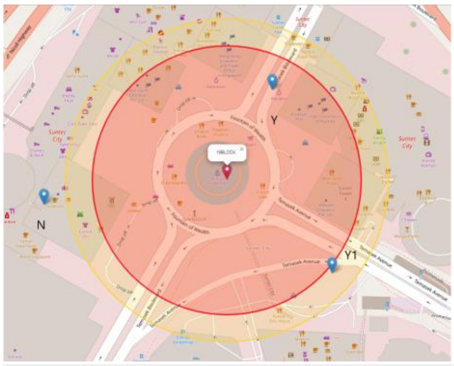
CSV file (Will not be filtered) (Code)

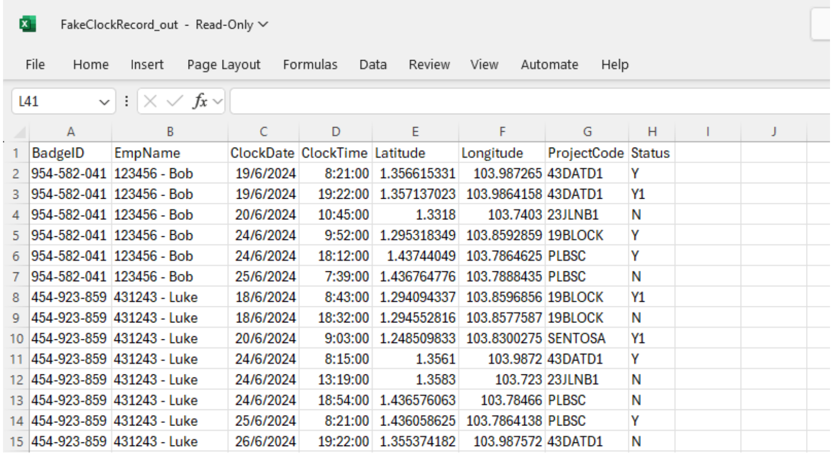
Status:

Y – within the specified radius or 100m (default)

Y1 – within radius + 20m

N - outside the 2 range



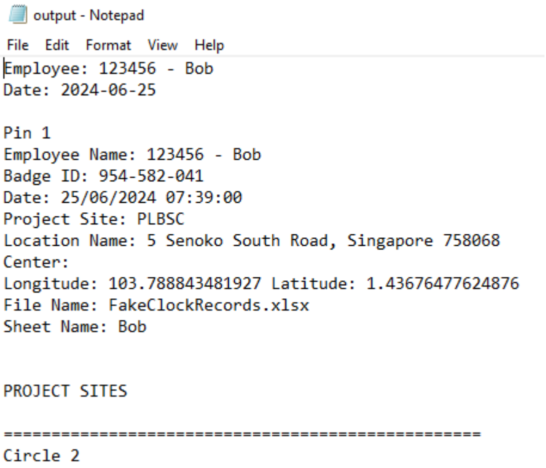


FakeClockRecord_out - Read-Only

L41

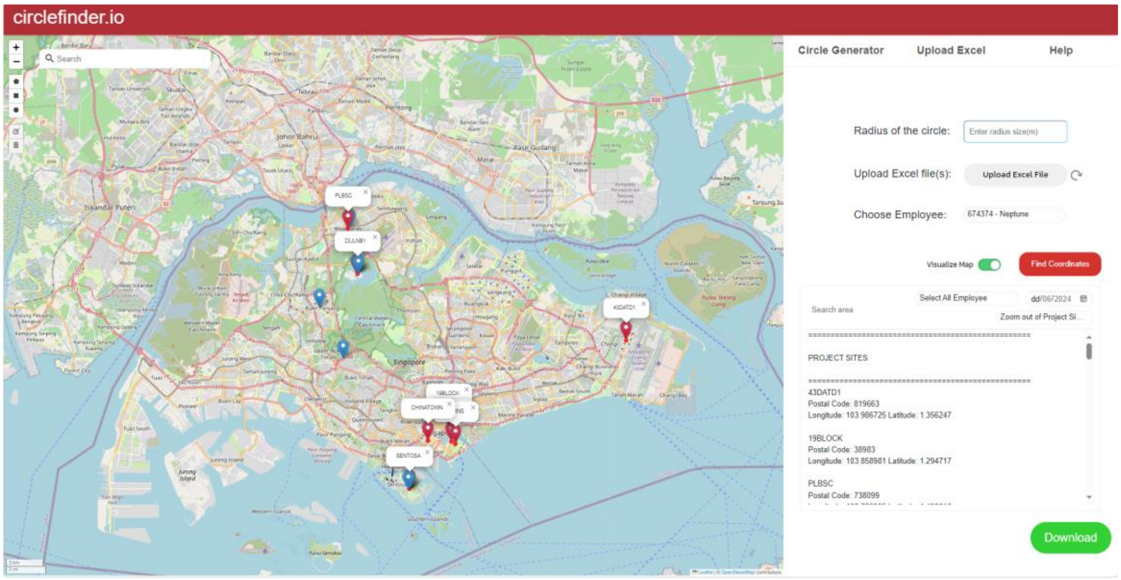
	A	B	C	D	E	F	G	H	I	J
1	BadgeID	EmpName	ClockDate	ClockTime	Latitude	Longitude	ProjectCode	Status		
2	954-582-041	123456 - Bob	19/6/2024	8:21:00	1.356615331	103.987265	43DATD1	Y		
3	954-582-041	123456 - Bob	19/6/2024	19:22:00	1.357137023	103.9864158	43DATD1	Y1		
4	954-582-041	123456 - Bob	20/6/2024	10:45:00	1.3318	103.7403	23JLNB1	N		
5	954-582-041	123456 - Bob	24/6/2024	9:52:00	1.295318349	103.8592859	19BLOCK	Y		
6	954-582-041	123456 - Bob	24/6/2024	18:12:00	1.43744049	103.7864625	PLBSC	Y		
7	954-582-041	123456 - Bob	25/6/2024	7:39:00	1.436764776	103.7888435	PLBSC	N		
8	454-923-859	431243 - Luke	18/6/2024	8:43:00	1.294094337	103.8596856	19BLOCK	Y1		
9	454-923-859	431243 - Luke	18/6/2024	18:32:00	1.294552816	103.8577587	19BLOCK	N		
10	454-923-859	431243 - Luke	20/6/2024	9:03:00	1.248509833	103.8300275	SENTOSA	Y1		
11	454-923-859	431243 - Luke	24/6/2024	8:15:00	1.3561	103.9872	43DATD1	Y		
12	454-923-859	431243 - Luke	24/6/2024	13:19:00	1.3583	103.723	23JLNB1	N		
13	454-923-859	431243 - Luke	24/6/2024	18:54:00	1.436576063	103.78466	PLBSC	N		
14	454-923-859	431243 - Luke	25/6/2024	8:21:00	1.436058625	103.7864138	PLBSC	Y		
15	454-923-859	431243 - Luke	26/6/2024	19:22:00	1.355374182	103.987572	43DATD1	N		

txt file (Will be filtered)



```
Employee: 123456 - Bob
Date: 2024-06-25

Pin 1
Employee Name: 123456 - Bob
Badge ID: 954-582-041
Date: 25/06/2024 07:39:00
Project Site: PLBSC
Location Name: 5 Senoko South Road, Singapore 758068
Center:
Longitude: 103.788843481927 Latitude: 1.43676477624876
File Name: FakeClockRecords.xlsx
Sheet Name: Bob


PROJECT SITES

==================================================
Circle 2
```

- If the filter is set to the specific employee (Faster than all employee)



Radius of the circle: Enter radius size(m)

Upload Excel file(s):

Choose Employee: 674374 - Neptune

Visualize Map

Search area

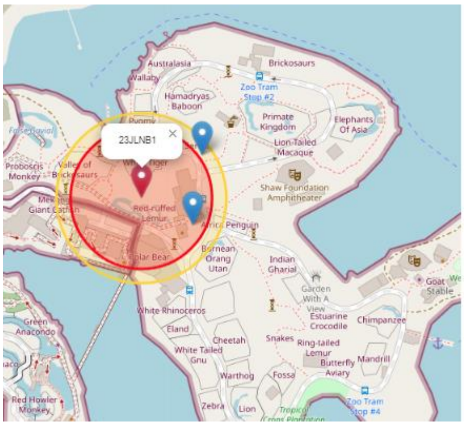
Select All Employee

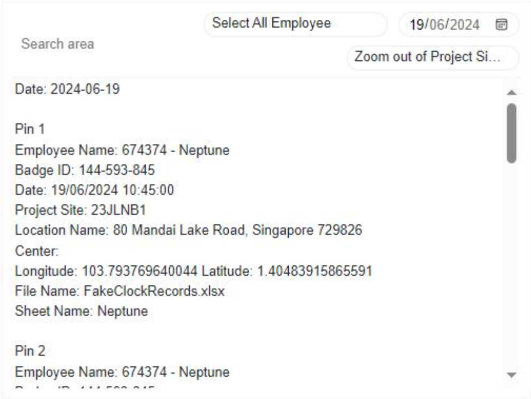
dd/06/2024

Zoom out of Project Si...

```
==================================================

PROJECT SITES

==================================================
43DATD1
Postal Code: 819663
Longitude: 103.986725 Latitude: 1.356247

19BLOCK
Postal Code: 38983
Longitude: 103.858981 Latitude: 1.294717

PLBSC
Postal Code: 738099
```

Can only be filtered by the date





Search area

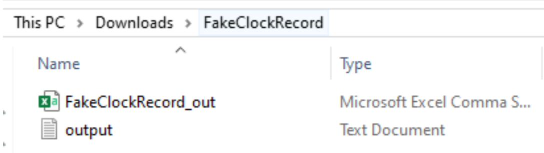
Select All Employee

19/06/2024

Zoom out of Project Si...

```
Date: 2024-06-19

Pin 1
Employee Name: 674374 - Neptune
Badge ID: 144-593-845
Date: 19/06/2024 10:45:00
Project Site: 23JLNB1
Location Name: 80 Mandai Lake Road, Singapore 729826
Center:
Longitude: 103.793769640044 Latitude: 1.40483915865591
File Name: FakeClockRecords.xlsx
Sheet Name: Neptune

Pin 2
Employee Name: 674374 - Neptune
```

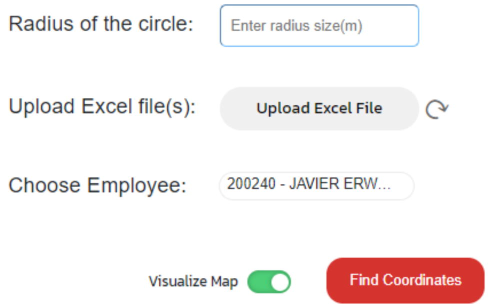
Downloading the filtered output:



This PC › Downloads › FakeClockRecord

Name	Type
FakeClockRecord_out	Microsoft Excel Comma S...
output	Text Document

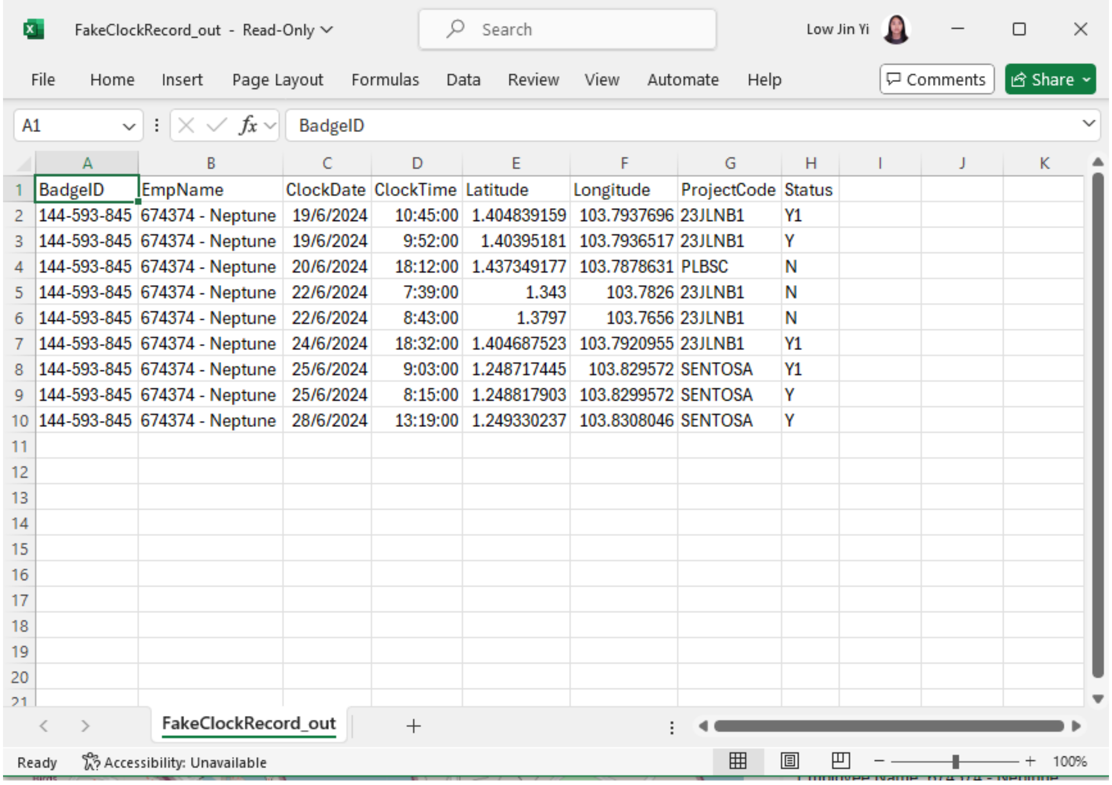
CSV file (filtered based on before processing):



Radius of the circle: Enter radius size(m)

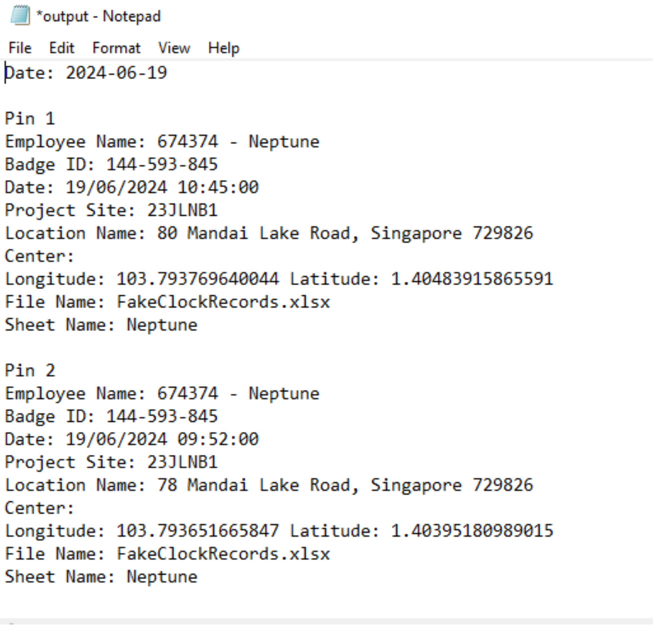
Upload Excel file(s): Upload Excel File

Choose Employee: 200240 - JAVIER ERW...



BadgeID	EmpName	ClockDate	ClockTime	Latitude	Longitude	ProjectCode	Status
144-593-845	674374 - Neptune	19/6/2024	10:45:00	1.404839159	103.7937696	23JLNB1	Y1
144-593-845	674374 - Neptune	19/6/2024	9:52:00	1.40395181	103.7936517	23JLNB1	Y
144-593-845	674374 - Neptune	20/6/2024	18:12:00	1.437349177	103.7878631	PLBSC	N
144-593-845	674374 - Neptune	22/6/2024	7:39:00	1.343	103.7826	23JLNB1	N
144-593-845	674374 - Neptune	22/6/2024	8:43:00	1.3797	103.7656	23JLNB1	N
144-593-845	674374 - Neptune	24/6/2024	18:32:00	1.404687523	103.7920955	23JLNB1	Y1
144-593-845	674374 - Neptune	25/6/2024	9:03:00	1.248717445	103.829572	SENTOSA	Y1
144-593-845	674374 - Neptune	25/6/2024	8:15:00	1.248817903	103.8299572	SENTOSA	Y
144-593-845	674374 - Neptune	28/6/2024	13:19:00	1.249330237	103.8308046	SENTOSA	Y

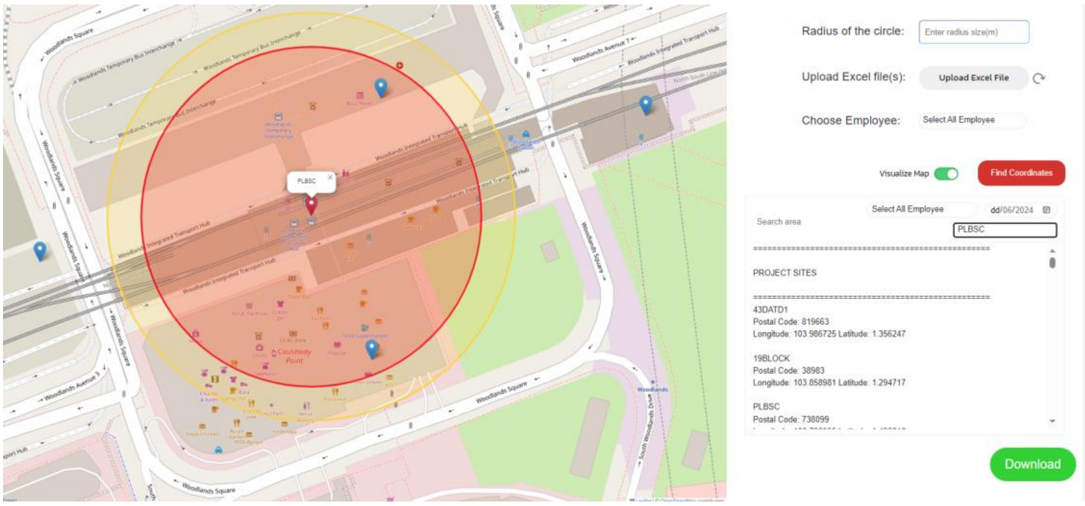
txt file (filtered based on the data picker)



```
Date: 2024-06-19

Pin 1
Employee Name: 674374 - Neptune
Badge ID: 144-593-845
Date: 19/06/2024 10:45:00
Project Site: 23JLNB1
Location Name: 80 Mandai Lake Road, Singapore 729826
Center:
Longitude: 103.793769640044 Latitude: 1.40483915865591
File Name: FakeClockRecords.xlsx
Sheet Name: Neptune

Pin 2
Employee Name: 674374 - Neptune
Badge ID: 144-593-845
Date: 19/06/2024 09:52:00
Project Site: 23JLNB1
Location Name: 78 Mandai Lake Road, Singapore 729826
Center:
Longitude: 103.793651665847 Latitude: 1.40395180989015
File Name: FakeClockRecords.xlsx
Sheet Name: Neptune
```

- Zoom in and zoom out feature
    - Using the drop-down menu (Code)

      Zooming into a project site



Radius of the circle: Enter radius size(m)

Upload Excel file(s): Upload Excel File

Choose Employee: Select All Employee

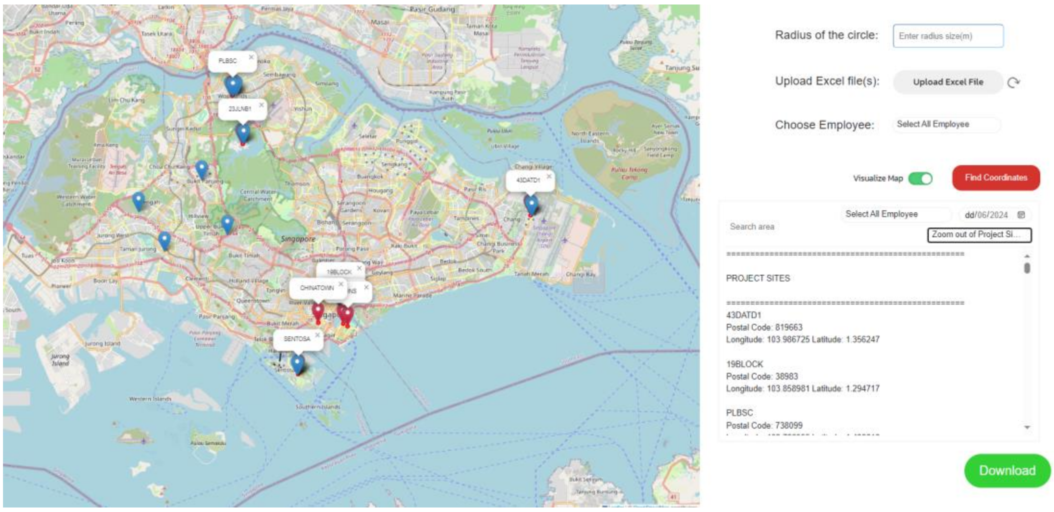
Visualize Map Find Coordinates

Select All Employee dd/06/2024

Search area PLBSC

```
=====================================================
PROJECT SITES
=====================================================
43DATD1
Postal Code: 819663
Longitude: 103.986725 Latitude: 1.356247

19BLOCK
Postal Code: 38983
Longitude: 103.858981 Latitude: 1.294717

PLBSC
Postal Code: 738099
```

      Zooming out of a project site



Radius of the circle: Enter radius size(m)

Upload Excel file(s): Upload Excel File

Choose Employee: Select All Employee

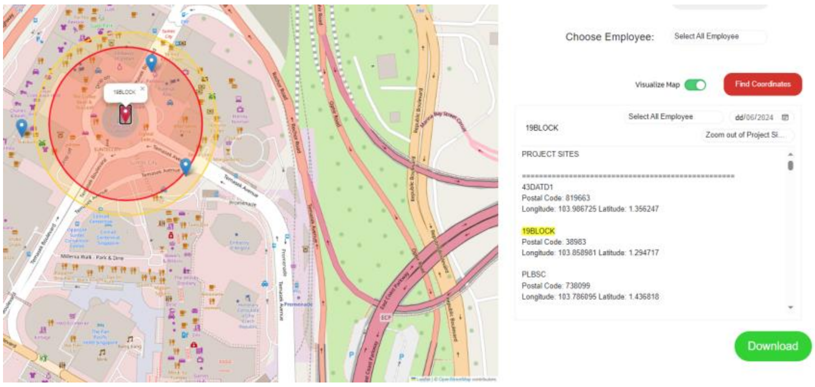
Visualize Map Find Coordinates

Select All Employee dd/06/2024

Search area Zoom out of Project Si...

```
=====================================================
PROJECT SITES
=====================================================
43DATD1
Postal Code: 819663
Longitude: 103.986725 Latitude: 1.356247

19BLOCK
Postal Code: 38983
Longitude: 103.858981 Latitude: 1.294717

PLBSC
Postal Code: 738099
```

    - Just by clicking the project site pin can zoom into the project site and at the same time highlight the project site details (coordinates and address) (Code)



Choose Employee: Select All Employee

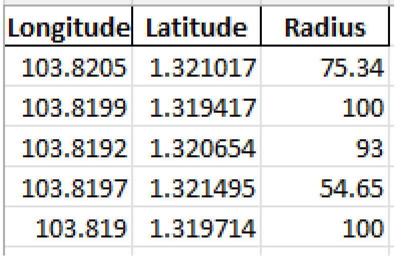
Visualize Map Find Coordinates

19BLOCK Select All Employee dd/06/2024

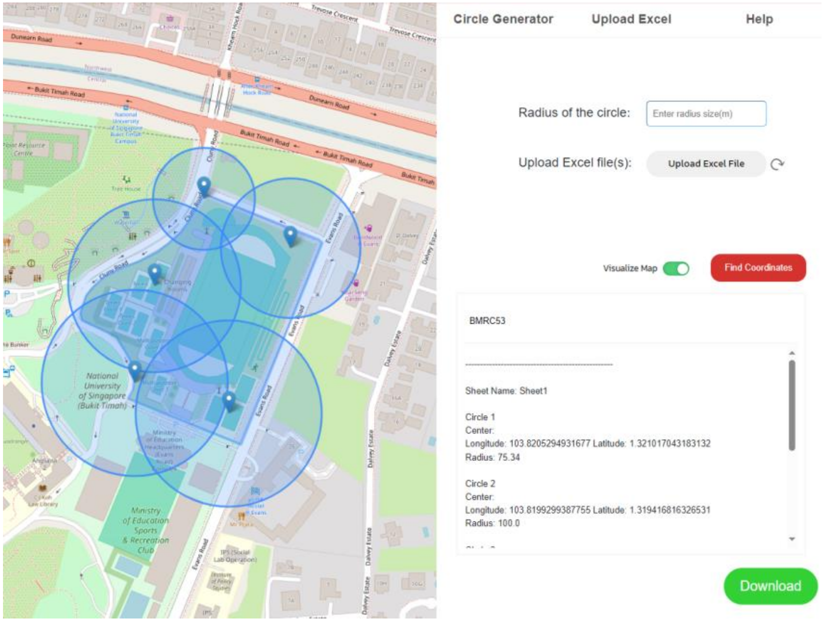
Zoom out of Project Si...

```
PROJECT SITES
=====================================================
43DATD1
Postal Code: 819663
Longitude: 103.986725 Latitude: 1.356247

19BLOCK
Postal Code: 38983
Longitude: 103.858981 Latitude: 1.294717

PLBSC
Postal Code: 738099
Longitude: 103.786095 Latitude: 1.436818
```

- Other Excel spreadsheets (Code)
  Example:
  With Radius Column and no Radius input



Longitude	Latitude	Radius
103.8205	1.321017	75.34
103.8199	1.319417	100
103.8192	1.320654	93
103.8197	1.321495	54.65
103.819	1.319714	100



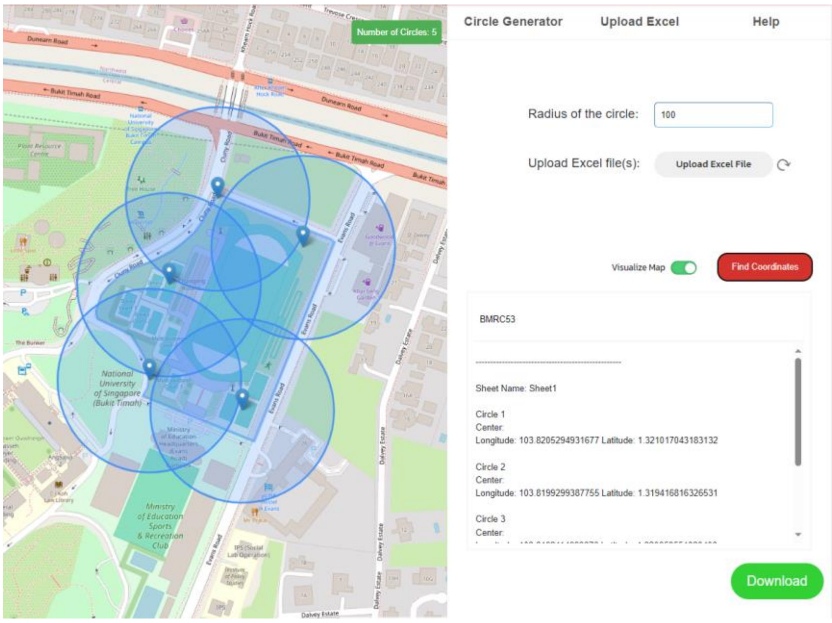
Radius of the circle:

Upload Excel file(s):

Visualize Map

```
BMRC53

----------------------------------------

Sheet Name: Sheet1

Circle 1
Center:
Longitude: 103.8205294931677 Latitude: 1.321017043183132
Radius: 75.34

Circle 2
Center:
Longitude: 103.8199299387755 Latitude: 1.319416816326531
Radius: 100.0
```

(With Radius Column and Radius Input) or (Without Radius Column and Radius Input)



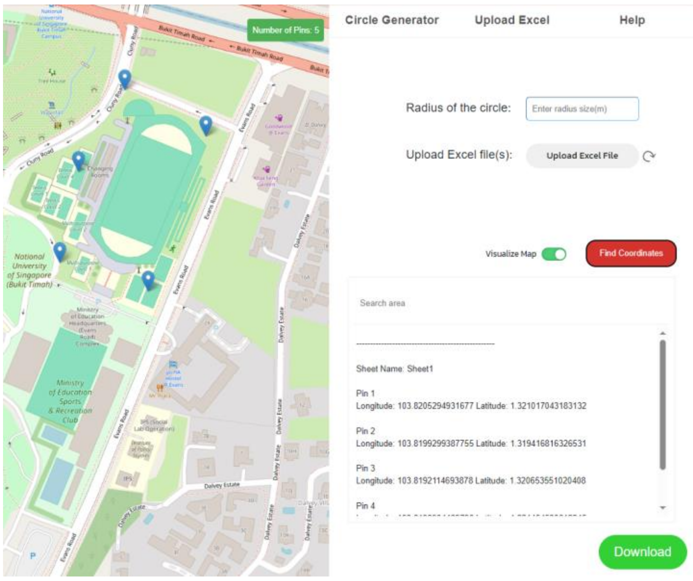
Radius of the circle: 100

Upload Excel file(s):

Visualize Map

```
BMRC53

----------------------------------------

Sheet Name: Sheet1

Circle 1
Center:
Longitude: 103.8205294931677 Latitude: 1.321017043183132

Circle 2
Center:
Longitude: 103.8199299387755 Latitude: 1.319416816326531

Circle 3
Center:
```

Without Radius Column and no Radius Input (No circles)



Radius of the circle: Enter radius size(m)

Upload Excel file(s):

Search area

Sheet Name: Sheet1

Pin 1
Longitude: 103.8205294931677 Latitude: 1.321017043183132

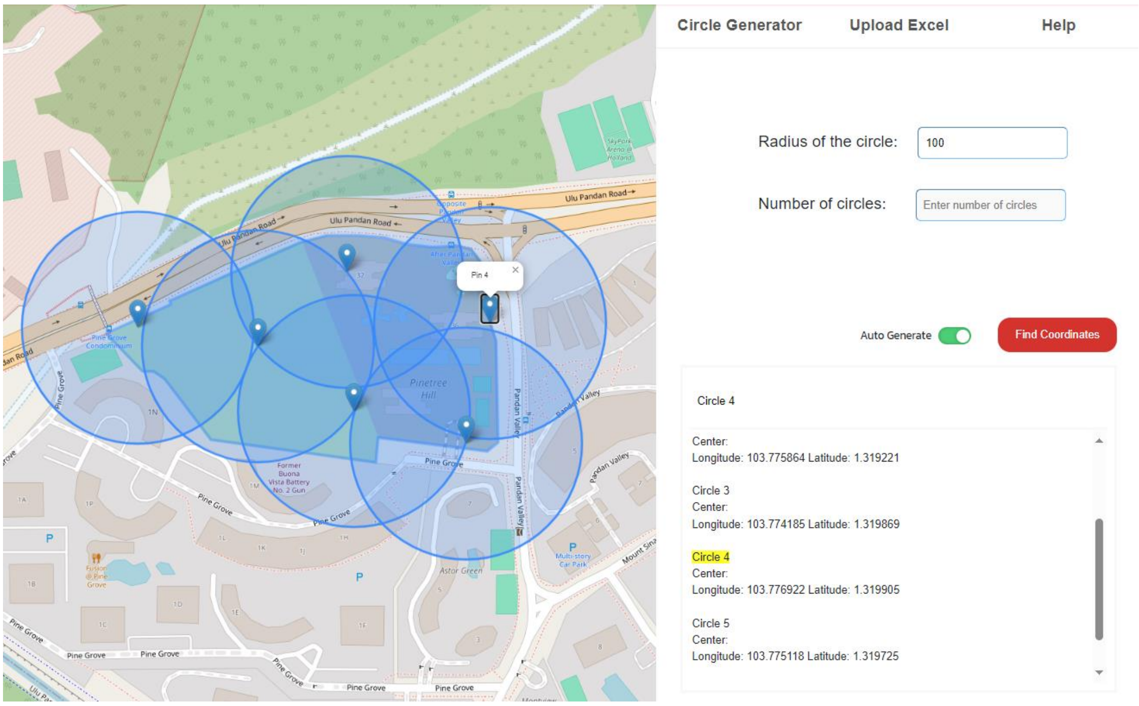
Pin 2
Longitude: 103.8199299387755 Latitude: 1.319416816326531

Pin 3
Longitude: 103.8192114693878 Latitude: 1.320653551020408

Pin 4

Similar features for the 2 Functions:

- Able to scroll to the marker or circle's detail and highlight it just by clicking on it
    - Generate Circle and other Excel Spreadsheet



Radius of the circle: 100

Number of circles: Enter number of circles

Circle 4

Center:
Longitude: 103.775864 Latitude: 1.319221

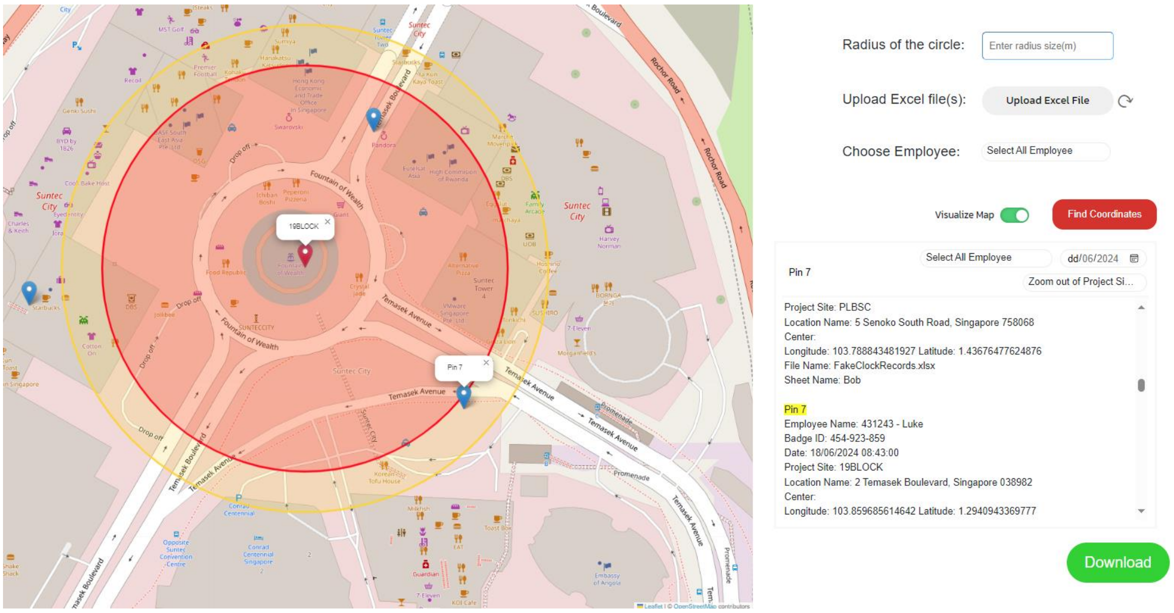
Circle 3
Center:
Longitude: 103.774185 Latitude: 1.319869

Circle 4
Center:
Longitude: 103.776922 Latitude: 1.319905

Circle 5
Center:
Longitude: 103.775118 Latitude: 1.319725

- Clock Records

Clicking on employee markers:



Radius of the circle: Enter radius size(m)

Upload Excel file(s): Upload Excel File

Choose Employee: Select All Employee

Visualize Map | Find Coordinates

Select All Employee | dd/06/2024

Pin 7

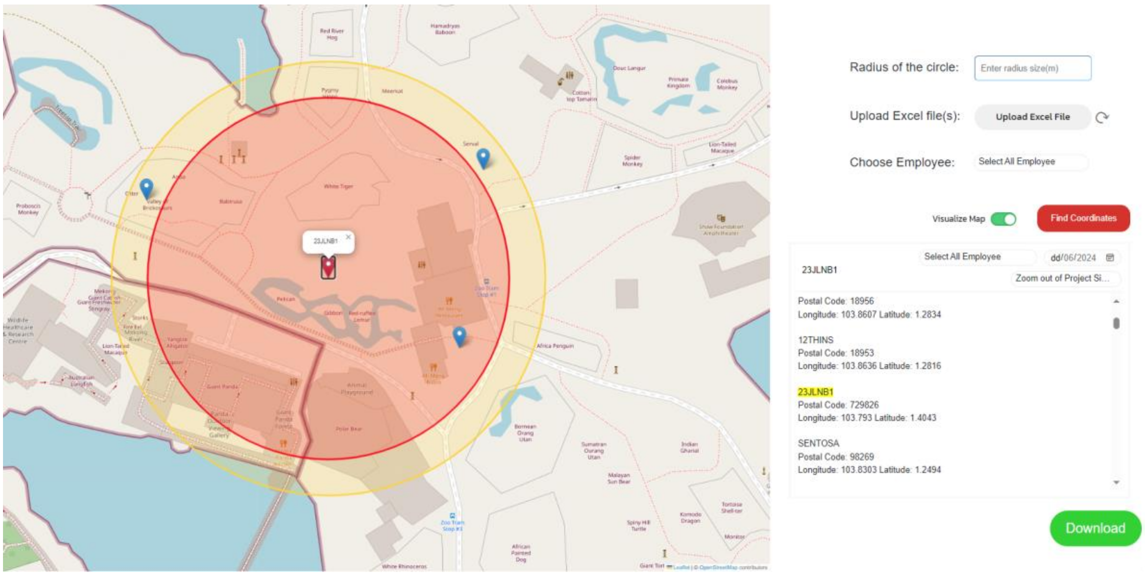
Zoom out of Project Si...

Project Site: PLBSC
Location Name: 5 Senoko South Road, Singapore 758068
Center:
Longitude: 103.788843481927 Latitude: 1.43676477624876
File Name: FakeClockRecords.xlsx
Sheet Name: Bob

Pin 7
Employee Name: 431243 - Luke
Badge ID: 454-923-859
Date: 18/06/2024 08:43:00
Project Site: 19BLOCK
Location Name: 2 Temasek Boulevard, Singapore 038982
Center:
Longitude: 103.859685614642 Latitude: 1.2940943369777

Download

Clicking on Project Site markers:



Radius of the circle: Enter radius size(m)

Upload Excel file(s): Upload Excel File

Choose Employee: Select All Employee

Visualize Map | Find Coordinates

Select All Employee | dd/06/2024

23JLNB1

Zoom out of Project Si...

Postal Code: 18956
Longitude: 103.8607 Latitude: 1.2834

12THINS
Postal Code: 18953
Longitude: 103.8636 Latitude: 1.2816

23JLNB1
Postal Code: 729826
Longitude: 103.793 Latitude: 1.4043

SENTOSA
Postal Code: 98269
Longitude: 103.8303 Latitude: 1.2494

Download

Explanation of Codes

Content:

- Circle Generator
- Upload Excel
- Download Output

Circle Generator

Cover only auto generate since Version 1.0 covers on the manual input.

Version 1.0 also covered how to get the polygon coordinates from map.html, get radius.

- Auto Generate (Click here to see the output)
  1) When the Auto Generate button is switched on, it will trigger the script to store True in a variable for the processing and at the same time disable the Number of Circle input.

Creating the Auto Generate switch

```
<div class="item">
                <label class="autogenerate">Auto Generate</label>
                <div class="toggle-pill-color">
                    <input type="checkbox" id="pill3" name="check">

                    <label for="pill3"></label>
                </div>
            </div>
```

Script

index.html (line 485)

```
var isChecked = false;
        document.getElementById("pill3").addEventListener("change", function
() {
            isChecked = this.checked;
            console.log(isChecked);
            if (this.checked) {
                document.getElementById("circle-number").disabled = true;
            } else {
                document.getElementById("circle-number").disabled = false;
            }
        });
```

2) After the radius is entered and the auto generate switch is on, the user will press the Find Coordinates button. index.html will send the data to app.py

Triggering the function

```
<button id="shapearea" class="calculate-button">Find Coordinates</button>
document.getElementById('shapearea').addEventListener('click',
getPolygonCoordinates)
```

Script

It takes in radius from the input box which is from circle-radius id. It helps to get the coordinates from the map.html (covered in Version 1.0) and then sends to app.py /calculate_area.

index.html (line 566) (getPolygonCoordinates)

```
 function getPolygonCoordinates(callback) {
                const generatingnotify =
document.getElementById('generatingnotify');
                const searchForm = document.getElementById('searchForm');
                var downloadbut = document.getElementById('downloadbutton');
                var coordsOutput = document.getElementById('coordsOutput');
                var textbox = document.getElementById('text');
                var coordinatesdata = localStorage['Coordinates'];
                var radius = document.getElementById('circle-radius').value;
                var circlenumber = document.getElementById('circle-
number').value;
                const suggestionsContainer =
document.getElementById('suggestions');

                generatingnotify.style.display = 'block';
                generatingnotify.textContent = 'Generating...';
                if (coordinatesdata) {
                    if (radius && (circlenumber || (circlenumber == false &&
isChecked))) {
                        if (circlenumber == false) {
                            circlenumber = 1;
                        }
                        var clearing =
coordinatesdata.replace('{"type":"Feature","properties":{},"geometry":{"type":
"Polygon","coordinates":[[[', '');
                        var coordinates = clearing.replace(']]]}}', '');
                        coordsOutput.textContent = coordinates;

                        fetch('/calculate_area', {
                            method: 'POST',
                            headers: {
                                'Content-Type': 'application/json'
                            },
              body: JSON.stringify({ 'coordinate': coordinates,
'radius': radius, 'circlenumber': circlenumber, 'isChecked': isChecked }),
//Send to app.py to let it calculate

            })
```

3) In app.py, it retrieves the necessary data and process it using an algorithm (If you want to know more about polygon coordinates, circle_to_geojson, circles_overlap, check with version 1.0)

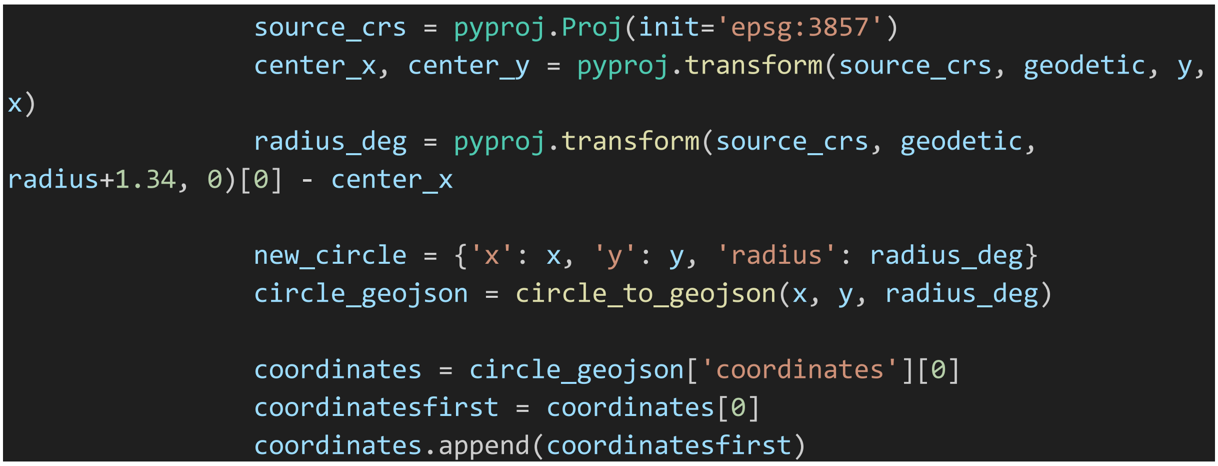
   The algorithm will randomly place an initial circle (here) and then generate a mesh grid using NumPy with 50 points distributed across the Polygon to identify uncovered areas (here). Then, it will go through that list of uncovered points and randomly select a point and place the circle there until the whole place is covered (here). If it exceeded a certain time limit which is 30 seconds, it will show the output that the application can achieve.

Initial circle

It goes through one time and gets a random coordinate based on the range from min x, y to max x, y of the polygon. Then, it will check if the center coordinate of a circle is in the polygon.

app.py (line 296)

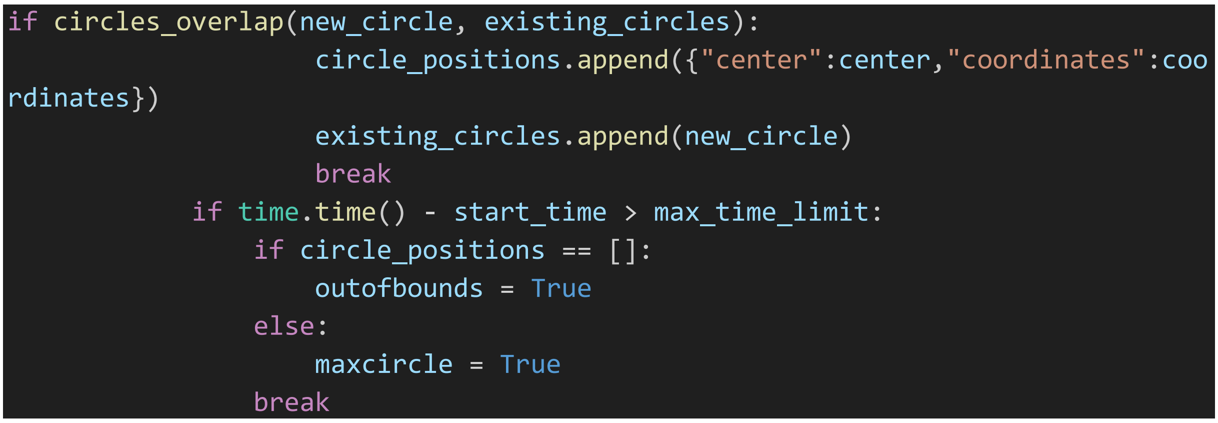
```
if data['isChecked'] == True:
    circlenumber = 1

  for _ in range(circlenumber):
    if time.time() - start_time > max_time_limit:
      if circle_positions == []:
        outofbounds = True
      else:
        maxcircle = True
      break
    while True:
      if time.time() - start_time > max_time_limit:
        if circle_positions == []:
          outofbounds = True
        else:
          maxcircle = True
        break

      min_x, max_y, max_x, min_y= polygon.bounds
      x = random.uniform(min_x, max_x)
      y = random.uniform(min_y, max_y)

      center = [x, y]

      if polygon.contains(Point(x, y)):
```



```
                source_crs = pyproj.Proj(init='epsg:3857')
                center_x, center_y = pyproj.transform(source_crs, geodetic, y,
x)
                radius_deg = pyproj.transform(source_crs, geodetic,
radius+1.34, 0)[0] - center_x

                new_circle = {'x': x, 'y': y, 'radius': radius_deg}
                circle_geojson = circle_to_geojson(x, y, radius_deg)

                coordinates = circle_geojson['coordinates'][0]
                coordinatesfirst = coordinates[0]
                coordinates.append(coordinatesfirst)
```

Then it will check if the circle overlap each other and if they overlap more than the radius length, it will be rejected. This is by the work of a function called circles_overlap.

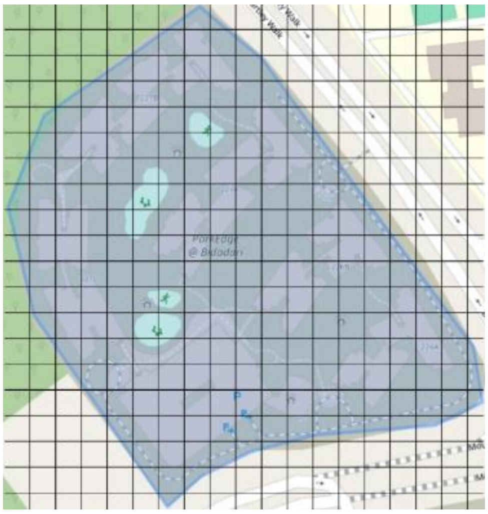
app.py (line 340)



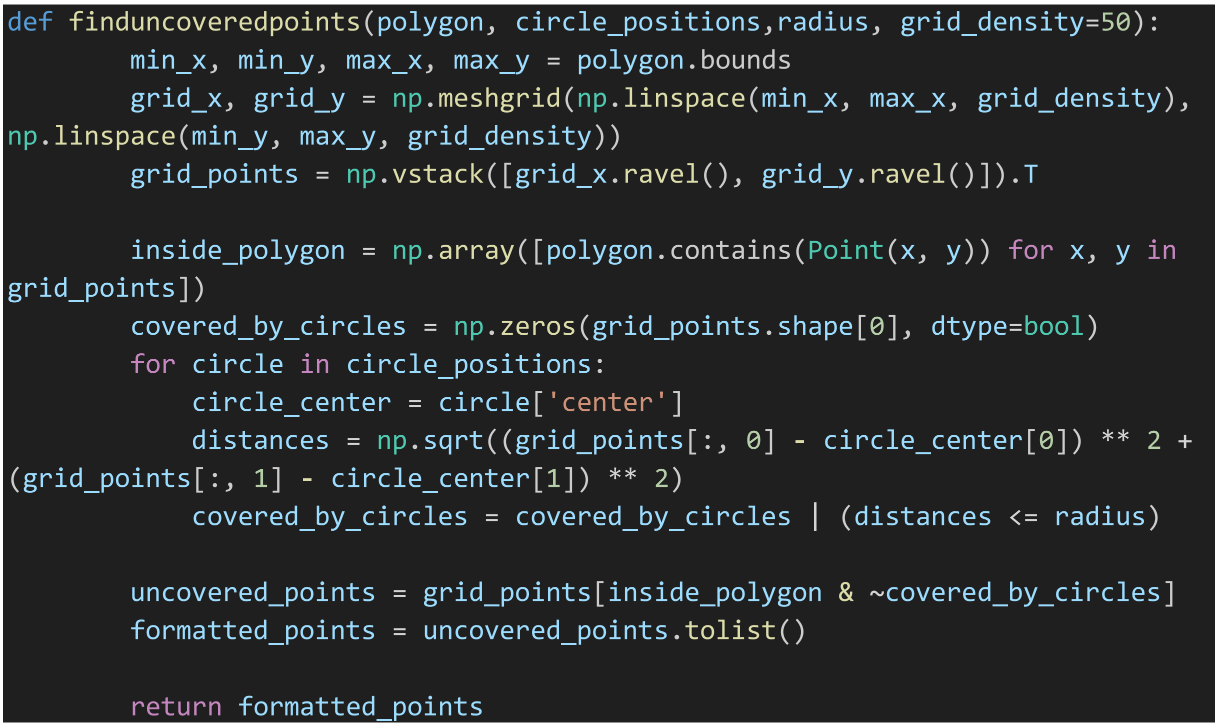
```
if circles_overlap(new_circle, existing_circles):
                    circle_positions.append({"center":center,"coordinates":coo
rdinates})
                    existing_circles.append(new_circle)
                    break
            if time.time() - start_time > max_time_limit:
                if circle_positions == []:
                    outofbounds = True
                else:
                    maxcircle = True
                break
```

Generating a mesh grid

It takes the min x, y and max x, y of the polygon and create a mesh grid of equally spaced points within the bounding box using np.meshgrid function from NumPy. Then, it stores the coordinates of the 50 grid points.



app.py (line 274) (finduncoveredpoints)

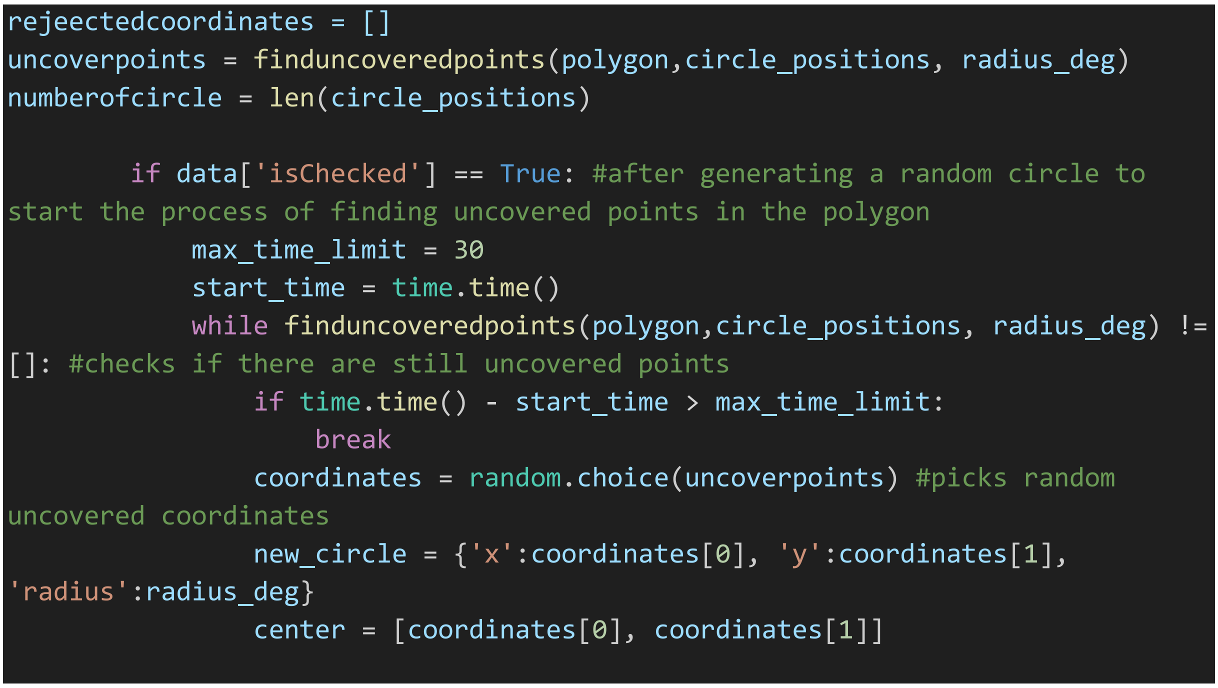


```
def finduncoveredpoints(polygon, circle_positions,radius, grid_density=50):
        min_x, min_y, max_x, max_y = polygon.bounds
        grid_x, grid_y = np.meshgrid(np.linspace(min_x, max_x, grid_density),
np.linspace(min_y, max_y, grid_density))
        grid_points = np.vstack([grid_x.ravel(), grid_y.ravel()]).T

        inside_polygon = np.array([polygon.contains(Point(x, y)) for x, y in
grid_points])
        covered_by_circles = np.zeros(grid_points.shape[0], dtype=bool)
        for circle in circle_positions:
            circle_center = circle['center']
            distances = np.sqrt((grid_points[:, 0] - circle_center[0]) ** 2 +
(grid_points[:, 1] - circle_center[1]) ** 2)
            covered_by_circles = covered_by_circles | (distances <= radius)

        uncovered_points = grid_points[inside_polygon & ~covered_by_circles]
        formatted_points = uncovered_points.tolist()

        return formatted_points
```

Fully cover the land

The list that was produced out will be checked whether all the uncovered points have been covered by checking whether the list is empty. If the list is not empty, it will randomly pick the uncovered points in the list for faster coverage. There will be a time limit for this, just in case it takes too long.

app.py (line 389)



```
rejeectedcoordinates = []
uncoverpoints = finduncoveredpoints(polygon,circle_positions, radius_deg)
numberofcircle = len(circle_positions)

        if data['isChecked'] == True: #after generating a random circle to
start the process of finding uncovered points in the polygon
            max_time_limit = 30
            start_time = time.time()
            while finduncoveredpoints(polygon,circle_positions, radius_deg) !=
[]: #checks if there are still uncovered points
                if time.time() - start_time > max_time_limit:
                    break
                coordinates = random.choice(uncoverpoints) #picks random
uncovered coordinates
                new_circle = {'x':coordinates[0], 'y':coordinates[1],
'radius':radius_deg}
                center = [coordinates[0], coordinates[1]]
                #getting the coordinates of the circumference and make it as a
"polygon variable"
                circle_geojson = circle_to_geojson(coordinates[0],
coordinates[1], radius_deg)
                #making it a complete polygon by mentioning the first
coordinates at the end of the list
                coordinates = circle_geojson['coordinates'][0]
                coordinatesfirst = coordinates[0]
                coordinates.append(coordinatesfirst)
```

This checks if the circle is inside the polygon

app.py (line 241) (circle_inside_polygon)

```
def circle_inside_polygon(coordinates): #check whether the center of the
circle is inside the polygon
        number = 0
        for coordinate in coordinates:
            x = coordinate[0]
            y = coordinate[1]

            if polygon.contains(Point(x, y)):
                #number += 1
                return True
```

If the coordinates are accepted for all the conditions, it will be stored in a dictionary. Then, gets all the data that is needed to send it to index.html

app.py (line 411)

```
if circle_inside_polygon(coordinates) and coordinates not in
rejeectedcoordinates:
                    if circles_overlap(new_circle, existing_circles):
                        if finduncoveredpoints(polygon,circle_positions,
radius_deg) == []:
                            break
                        circle_positions.append({"center":center,"coordinates"
:coordinates}) #store coordinates
                        existing_circles.append(new_circle)
                        numberofcircle += 1
                    else:
                        rejeectedcoordinates.append(coordinates)
                if time.time() - start_time > max_time_limit:
                    break
        if numberofcircle == len(circle_positions):
            notify = f'Recommended number of circle: {numberofcircle}'
        content += f'Number of Circles: {numberofcircle}\n'
for positions in circle_positions: #for many circles
            number += 1
            longitude = positions['center'][0]
            longitude = '{:.6f}'.format(longitude)
            latitude = positions['center'][1]
            latitude = '{:.6f}'.format(latitude)

            content += f'\nCircle {number}\nCenter:\nLongitude: {longitude}
Latitude: {latitude}\n'
            circlecoordinateslist.append(positions['center'])

        if maxcircle == False and notify == '': #if the circles can fit in the
polygon
            notify = ''
        else: #if it reaches the maximum number of circles to fit in the
polygon
            notify = f'Max number of circles: {number}'

    htmlcontent = content.replace('\n', '<br>')

    circlegenerateddf = pd.DataFrame(circlecoordinateslist,
columns=['Longitude', 'Latitude'])
    circlegenerateddf['Radius'] = radius


    return json.dumps({'area': printarea, 'content':htmlcontent,
'projected_coords':coordinateslist, 'circle_coords': circlecoordinateslist
                       , 'notify': notify})
```

4) Now back to index.html function getPolygonCoordinates after processing all the suitable coordinates of the circles and the details of them. The function will act as a messenger to send to map.html and store all the necessary values inside a localStorage.

index.html (line 605) (getPolygonCoordinates)

```
.then(response => {
                                return response.json();
                            })
.then(data => { //get the data after app.py calculates
                                localStorage.removeItem('dropdownprojectsitech
anged')
                                coordsOutput.textContent = coordinates;
                                console.log(data.content);
                                textbox.innerHTML = data.content; //fill up
the box with the content that lets the user know the center points of the
circle

                                var projected_coords = data.projected_coords;
                                centroid =
calculateCentroid(projected_coords);
```

```
                                console.log('Message sending to map.html:',
data);

                                //now implement the zoom into this coordinate
then draw it out on the map2.html
                                localStorage.setItem('polygoncoords',
JSON.stringify(projected_coords));
                                localStorage.setItem('longitudesecondmap',
centroid[0]);
                                localStorage.setItem('latitudesecondmap',
centroid[1]);
                                localStorage.setItem('circlecoords',
JSON.stringify(data.circle_coords)); //store as JSON string

                                localStorage.setItem('radius', radius);
                                var mapFrame =
document.getElementById('mapFrame');
                                mapFrame.contentWindow.postMessage('postlocati
on', '*');
```

5) At the mapFrame.contentWindow.postMessage is to alert map.html about the message so there will be a window.addeventlistener for a message specifically saying postlocation which is what we did for the search bar in Version 1.0. So, with the data from index.html, it will take the center coordinates of the circles and the input of the radius and create circles on the map. Adding to the layer, there will be markers for the center of the circles. For user ease when they want to check on the details of the circle.

map.html (line 401)

```
if (event.data === 'postlocation') {
            console.log('Message received in map.html:', event.data);
            var circlecoords = localStorage['circlecoords'];
            var radius = localStorage.getItem('radius');

            localStorage.setItem('shapetype', 'circle');

           //code here for deleting existing layers…

                radius = parseFloat(radius);
                let number = 1
                circlecoords.forEach(coords => {
                    var circle = L.circle([coords[1], coords[0]], {
                        radius: radius,
                    }).addTo(map_9a5fe2b4c0a85f46e0a2eaef8761d87c);
                    circle.bindPopup("Circle " + number).on('click', function
(e) {
```

```
                        localStorage.setItem('clickedpin', circle.getPopup().getContent());
                        localStorage.setItem('status', event.data);
                    });
                    circle_array.push(circle);
                    drawnItems_draw_control_a6dd22a64b6a6130aecf47f240d50ce6.addLayer(circle);
                    number++;
                });
                number = 1;
                circlecoords.forEach(coords => {
                    var pin = L.marker([coords[1], coords[0]], {
                        radius: radius,
                    }).addTo(map_9a5fe2b4c0a85f46e0a2eaef8761d87c);
                    pin.bindPopup('Pin ' + number).on('click', function (e) {
                        localStorage.setItem('clickedpin', pin.getPopup().getContent());
                        localStorage.setItem('status', event.data);
                    });
                    coordmarkers.push(pin);
                    drawnItems_draw_control_a6dd22a64b6a6130aecf47f240d50ce6.addLayer(pin)
                    number++;

                });
            } else {
                console.error("Circle coordinates is missing or not formatted correctly");
            }
        }
```

- Edit the output of the generated circles

  When the generated circles are not satisfactory to the users, they are able to edit the output which will update the output container and the download data so that they would not need to change the output that the system generated.

  - Deleting circles (Example)
    - In map.html, they can find out what layer was deleted and the layers that are deleted such as circles and marker, it will only get the marker's content. Thus, the user must delete the circle and the marker at the same time.

When the user deletes a layer or many layers at once on the map, it will be recorded down. So, the deleted items will be checked if it is a Marker and if the status is postlocation so it will know to only check the marker that was created in Circle Generator and not in Upload Excel. Then, it sends the list of deleted layer's name to index.html

map.html (line 175)

```
map_9a5fe2b4c0a85f46e0a2eaef8761d87c.on('draw:deleted', function (e) {
        const deleteditem = [];
        var layers = e.layers;
        layers.eachLayer(function (layer) {
            // Access the deleted layer's properties
            if (layer instanceof L.Marker) {
                if (localStorage['status'] === 'postlocation') {
                    deleteditem.push(layer.getPopup().getContent())
                }
            }
        });
        localStorage.setItem('deleteditem', JSON.stringify(deleteditem));
    });
```

Then, index.html will be notified if the localStorage is being modified. It checks whether it is the deleteditem being modified and fetches the /editoutput. Then, sends the data that it gathered from map.html to app.py and notify app.py that the layer is being deleted. So how it notify is telling app.py that 'add' is false.

index.html (line 1227)

```
window.addEventListener('storage', () => { //keeps an eye on the localStorage
if (localStorage['deleteditem']) {
                    console.log('here del')
                    fetch('/editoutput', {
                        method: 'POST',
                        headers: {
                            'Content-Type': 'application/json'
                        },
                        body: JSON.stringify({ 'deleteditem':
localStorage['deleteditem'], 'htmlcontent':
document.getElementById('text').innerHTML, 'add': false }), //Send to app.py
to let it calculate
                    })
                        .then(response => response.json())
                        .then(data => {
                            console.log(data.content)
                            document.getElementById('text').innerHTML =
data.content;
                            removeHighlights(document.getElementById('text'));
                        })
                    localStorage.removeItem('createdlayer')
                    localStorage.removeItem('deleteditem');
                }
            }
```

In app.py, it will check if data['add'] is false. Then, it will find the pattern that matches and removes the output text and the dataframe that stores the coordinates and the radius.

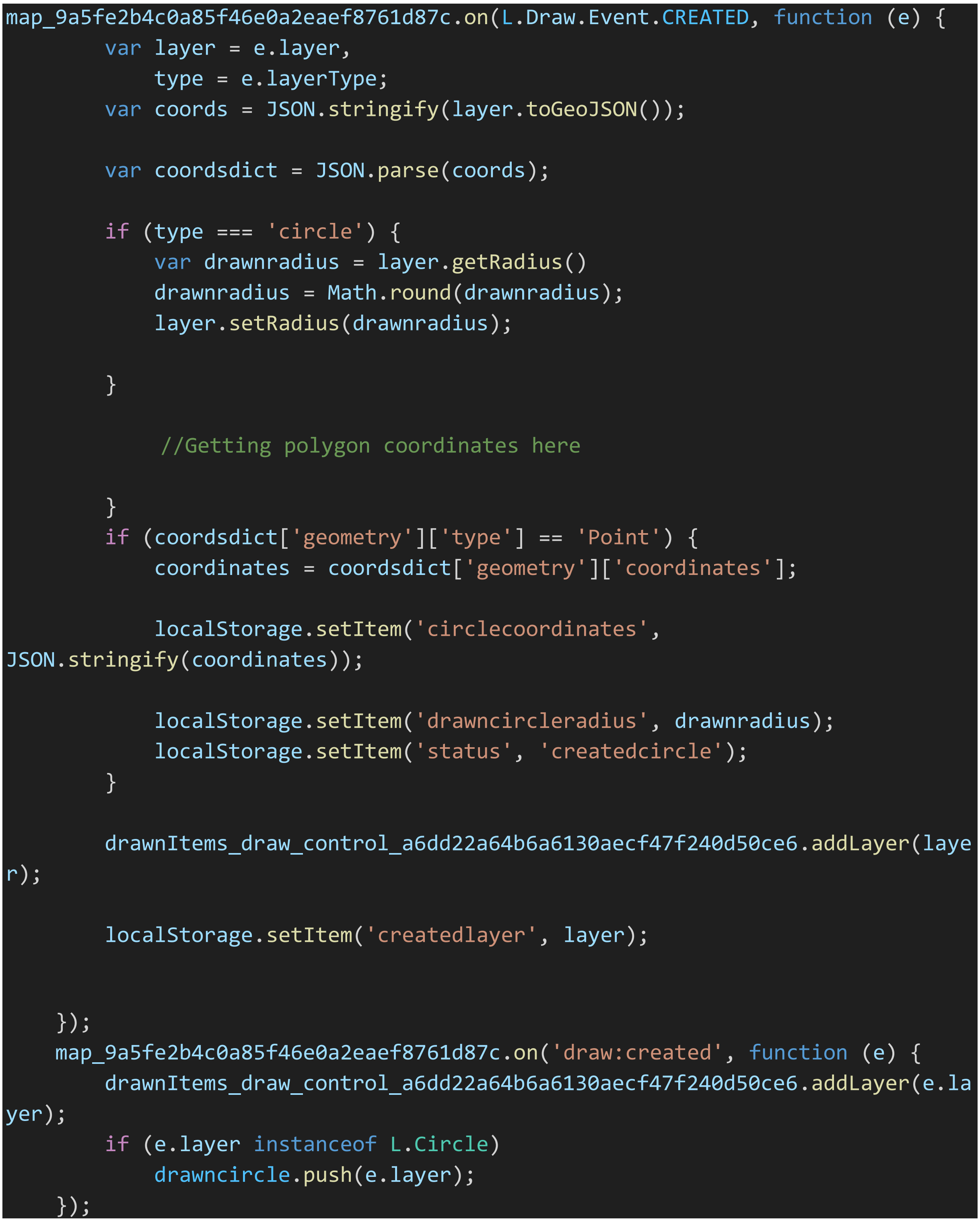
app.py (line 1093)

```
somestring = replacecontent.replace('<br>', '\n')

        somestring = somestring.replace('<span class="highlight">', '')
        somestring = somestring.replace('</span>', '')

        deleteditem = json.loads(data['deleteditem'])


        for item in deleteditem:
            item = int(item.replace('Pin ', ''))
            pattern = re.compile(rf"\nCircle {item}\nCenter:\nLongitude: [0-
9.]+ Latitude: [0-9.]+\n")

            circlegenerateddf.drop(index=item - 1, inplace=True)


            somestring = re.sub(pattern, "", somestring)
        replacecontent = somestring
        htmlcontent = somestring.replace('\n', '<br>')


        return json.dumps({'content': htmlcontent})
```

- Creating circles (Example)
    - In map.html, it will know whether the circle is being created in the map and it will update the output adding in the circle's details such as the coordinates. For dataframe, it will be updated and at the same time, it will have the new created circle radius in whole number.

When the user creates a layer that is a circle, it will get the radius and round it up to a whole number. Hence, the user does not need to be worried about making the radius accurate or not. With the whole number, it will also reflect on the map correcting it accurately to the whole number. With the details of the circle, it will be sent to index.html. It also stores the circle layer in a list to delete it for the next process.

map.html(line 138)



```
map_9a5fe2b4c0a85f46e0a2eaef8761d87c.on(L.Draw.Event.CREATED, function (e) {
        var layer = e.layer,
            type = e.layerType;
        var coords = JSON.stringify(layer.toGeoJSON());

        var coordsdict = JSON.parse(coords);

        if (type === 'circle') {
            var drawnradius = layer.getRadius()
            drawnradius = Math.round(drawnradius);
            layer.setRadius(drawnradius);

        }

            //Getting polygon coordinates here

        }
        if (coordsdict['geometry']['type'] == 'Point') {
            coordinates = coordsdict['geometry']['coordinates'];

            localStorage.setItem('circlecoordinates',
JSON.stringify(coordinates));

            localStorage.setItem('drawncircleradius', drawnradius);
            localStorage.setItem('status', 'createdcircle');
        }

        drawnItems_draw_control_a6dd22a64b6a6130aecf47f240d50ce6.addLayer(laye
r);

        localStorage.setItem('createdlayer', layer);


    });
    map_9a5fe2b4c0a85f46e0a2eaef8761d87c.on('draw:created', function (e) {
        drawnItems_draw_control_a6dd22a64b6a6130aecf47f240d50ce6.addLayer(e.la
yer);
        if (e.layer instanceof L.Circle)
            drawncircle.push(e.layer);
    });
```

In index.html, it will check if the layer is created and sends the data from map.html to app.py, providing the center of the circle coordinates and setting add to true.

index.html (line 1205)

```
window.addEventListener('storage', () => {
if (localStorage['createdlayer']) {
                    fetch('/editoutput', {
                        method: 'POST',
                        headers: {
                            'Content-Type': 'application/json'
                        },
                        body: JSON.stringify({ 'circlecoordinates': localStorage['circlecoordinates'], 'thisradius': localStorage['drawncircleradius'], 'htmlcontent': document.getElementById('text').innerHTML, 'add': true }), //Send to app.py to let it calculate
                    })
```

app.py will receive the coordinates and the radius of the new circle. It will find the maximum index in the text output so that it can add on to the list. In the dataframe, it will be put at the end of the dataframe.

app.py (line 1070)

```
if data['add']:
        replacecontent = replacecontent.replace('<br>', '\n')
        somestring = re.findall('Circle \d+', replacecontent)
        if somestring != []:
            somestring = [int(s.replace('Circle ', '')) for s in somestring]
            maxnumber = max(somestring)
        else:
            maxnumber = 0
        circlecoordinates = json.loads(data['circlecoordinates'])
        replacecontent += f'\nCircle {maxnumber + 1}\nCenter:\nLongitude: {circlecoordinates[0]} Latitude: {circlecoordinates[1]}\n'
        htmlcontent = replacecontent.replace('\n', '<br>')

        newcirclelist = [circlecoordinates[0], circlecoordinates[1], round(float(data['thisradius']),2)]

        if circlegenerateddf.empty is not True:
            circlegenerateddf.loc[circlegenerateddf.index.max()+ 1] = newcirclelist
        if circlegenerateddf.empty:
            circlegenerateddf.loc[0] = newcirclelist
        return json.dumps({'content': htmlcontent, 'number': maxnumber})
```

Upload Excel

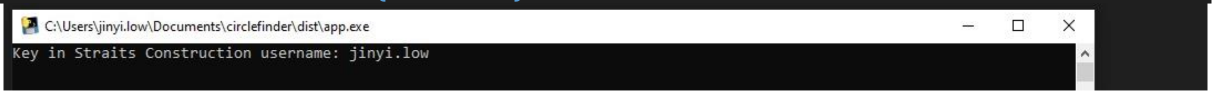
- Before the application starts
  - Reading Project Data
    - Gets the Project data from an excel sheet that is in the same directory as the python script. It will store the data as a dataframe and create a text output of the project site's detail which will be added on to the future employee text output. Hence, storing the project site details temporarily

app.py (line 88)

```
global projectplace
    global locationlist
    global projectplacecoords
    global projectplacecontent
    global chooseemployee

    projectplacecontent = ''

    excel_file = Geo-location Of All SCS Construction Sites.xlsx'

    projectplacecoords = []
    locationlist = []
    projectplace = []

    inputfiledf = pd.read_excel(excel_file)
    projectplacecontent += '=' * 50
    projectplacecontent += '\n\nPROJECT SITES\n\n'
    projectplacecontent += '=' * 50


    for latitude, longitude, location, address in zip(inputfiledf['Latitude'],
inputfiledf['Longitude'], inputfiledf['Location'], inputfiledf['Address']):
        if (longitude is not None and latitude is not None) and (longitude !=
0 and latitude != 0):
            center = [longitude, latitude]
            projectplacecoords.append(center)
            locationlist.append(location)
            projectplace.append({'Location':location,'Address': address,
'Center':[longitude, latitude], 'shapetype': 'circle'})

            projectplacecontent += f'\n{location}\nAddress:
{address}\nLongitude: {longitude} Latitude: {latitude}\n'

   #Auto Read Code here…


return render_template('index.html', chooseemployee=chooseemployee)
```

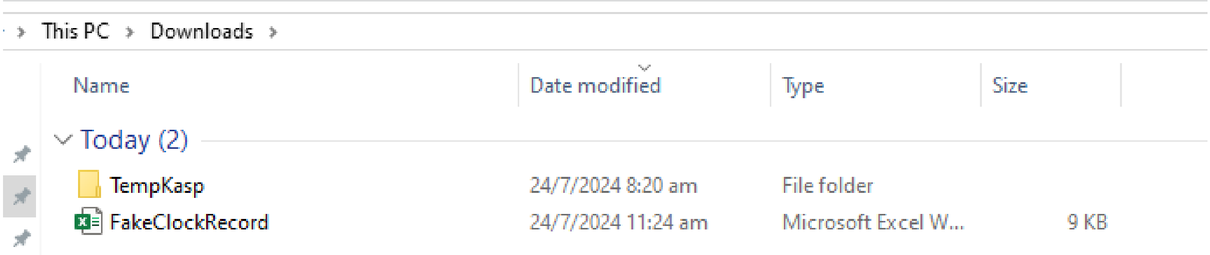
- Clock Record Auto Read
    - It asks the user, their company username and completes the file path directory. Then, it will read the data in the file and allow the user to pick the filter

```
username = input('Key in Straits Construction username: ')
clockfile = rf'C:\Users\{username}\Downloads\FakeClockRecord.xlsx'
```


C:\Users\jinyi.low\Documents\circlefinder\dist\app.exe

```
Key in Straits Construction username: jinyi.low
```

Pretend that it is a OneDrive file path



Name	Date modified	Type	Size
Today (2)			
TempKasp	24/7/2024 8:20 am	File folder	
FakeClockRecord	24/7/2024 11:24 am	Microsoft Excel W...	9 KB

app.py (line 131)

```
chooseemployee = []

    #to get the employeename so user can filter to just getting the data for
that employee
    if os.path.exists(clockfile):
        wb = openpyxl.load_workbook(clockfile)
        sheetnames = wb.sheetnames
        for sheet in sheetnames:
            inputfiledf = pd.read_excel(clockfile, sheet_name=sheet)
            for latitude, longitude, employeenamecode in
zip(inputfiledf['Latitude'], inputfiledf['Longitude'],
inputfiledf['EmployeeCodeName']):
                #checks if the coordinates are empty and get the
employeenamecode
                if (longitude != None and latitude != None) and (longitude !=
0 and latitude != 0):
                    chooseemployee.append(employeenamecode)

    chooseemployee = list(set(chooseemployee))
```

Problem Facing:

Every user may have a different file path to OneDrive. However, if I tried threading or os.walk in the C: drive, it would take a long time to search all the files until it finds the matching relative path. Hence, my approach would be to make sure the users have a consistent file path to the OneDrive

- Manual Upload for Clock Records (Example)
  - The Process
    1) When the user uploads Clock Records excel spreadsheet, it will let the user choose the excel files. It will send the files over to app.py to store it in the UPLOAD_FOLDER.

Creating Upload Button

```
<label class="uploadbuttontext">Upload Excel file(s):</label>
                <button id="uploadbutton" class="uploadbutton"
                    onclick="document.getElementById('fileInput').click()">Upload
                    Excel File</button>
                <input type="file" id="fileInput" class="hidden" multiple>
```

Script

index.html (line 389)

```
const fileInput = document.getElementById('fileInput');
            fileInput.addEventListener('change', (e) => {
                handleFiles(e.target.files);
            });

            function handleFiles(files) { //sends the files to app.py to put it in the UPLOAD_FOLDER
                const formData = new FormData();
                for (const file of files) {
                    formData.append('files', file);
                }
                fetch('/upload', {
                    method: 'POST',
                    body: formData,
                })
```

app.py (line 538)

```
@app.route('/upload', methods=['POST'])
def upload_files():
    deletefiles()
    if 'files' not in request.files:
        return jsonify({'error': 'No files part'}), 400

    files = request.files.getlist('files')
    for file in files:
        if file:
            filename = file.filename
            file.save(os.path.join(UPLOAD_FOLDER.name, filename))
```

2) After putting the files in the UPLOAD FOLDER, it will go through the files in there. It checks whether the excel has Clock Records. If the dataframe column contains ‘EmployeeCodeName’, it will store all the employee’s name into a list. The list will be cleaned to not have any repeated names. This list will be used to let the user choose which employee; they would like to filter to before processing.

app.py (line 550)

```
chooseemployee = []

    for inputfiles in os.listdir(UPLOAD_FOLDER.name): #this is to check if
that file is a clock record file (incase user wants to access the past files)
        filepath = f'{UPLOAD_FOLDER.name}/{inputfiles}'
        wb = openpyxl.load_workbook(filepath)
        sheetnames = wb.sheetnames
        for sheet in sheetnames:
            inputfiledf = pd.read_excel(filepath, sheet_name=sheet)
            if 'EmployeeCodeName' in inputfiledf.columns:
                for latitude, longitude, employeenamecode in
zip(inputfiledf['Latitude'], inputfiledf['Longitude'],
inputfiledf['EmployeeCodeName']):
                    if (longitude != None and latitude != None) and (longitude
!= 0 and latitude != 0):
                        chooseemployee.append(employeenamecode)

    chooseemployee = list(set(chooseemployee)) #it will be stored in the drop
down that allows the user to choose the employee before processing

    return jsonify({
        'message': 'Files uploaded successfully',
        'chooseemployee': chooseemployee
    }), 200
```

3) In index.html, the list will be turned into a drop-down box allowing the user to choose the filter for faster processing compared to choosing all the employees. (Example of how the drop-down box look like)

index.html (line 405)

```
.then(response => response.json())
                    .then(data => {
                        const chooseemployeedropdown =
document.getElementById('chooseemployeedropdown');
                        chooseemployeedropdown.innerHTML = '';
```

```
            alert('Files uploaded successfully');
            localStorage.clear();
            if (data.chooseemployee) {
              console.log(data.chooseemployee)
              const option = document.createElement('option');
              option.value = 'allemployee';
              option.textContent = 'Select All Employee';
              chooseemployeedropdown.appendChild(option);

              data.chooseemployee.forEach(name => {
                const option =
document.createElement('option');
                option.value = name;
                option.textContent = name;
                chooseemployeedropdown.appendChild(option);
              })
              localStorage.setItem('chosenemployeevalue',
'allemployee');
            }

            $(document).ready(function () {
              $('#chooseemployeedropdown').select2({
                theme: 'theme3'
              });
              localStorage.setItem('chosenemployeevalue',
'allemployee');
              $('#chooseemployeedropdown').on('change', function
() {

                localStorage.setItem('chosenemployeevalue',
this.value);
              })
            })

          })
          .catch(error => {
            console.error(error);
            alert('Error uploading files');
          });
```

index.html (line 689)

```
fetch('/process', { //to fetch the process in app.py
          method: 'POST',
          headers: {
            'Content-Type': 'application/json'
          },
                    body: JSON.stringify({ 'coordradius': coordradius,
'overlapchecked': overlapchecked, 'chosenemployeevalue':
localStorage['chosenemployeevalue'] }), //Send to app.py to let it calculate
                })
```

4) The user will get to select the filter and press the button to have it processed.
    a. Select All Employee
       There will be no change in the drop-down box. It will send an empty variable which is chosenemployee. Then, it will process the files producing the output.

Firstly, it will check whether there is a radius input in the application. If there is no radius, it will set it as 100m by default.

app.py (line 738)

```
elif checklocation and data['coordradius'] == '':
        radius = 100
        content += 'Project Site Area\n'
        content += f'Circle Radius: 100m\n'
    elif checklocation and data['coordradius'] != '':
        radius = float(data['coordradius'])
        content += 'Project Site Area\n'
        content += f'Circle Radius: {radius}m\n'
```

Since the user did not change anything in the drop-down box, it will set it as allemployee

app.py (line 755)

```
chosen_employee_value = data.get('chosenemployeevalue', None)
    if not chosen_employee_value:
        data['chosenemployeevalue'] = 'allemployee'
```

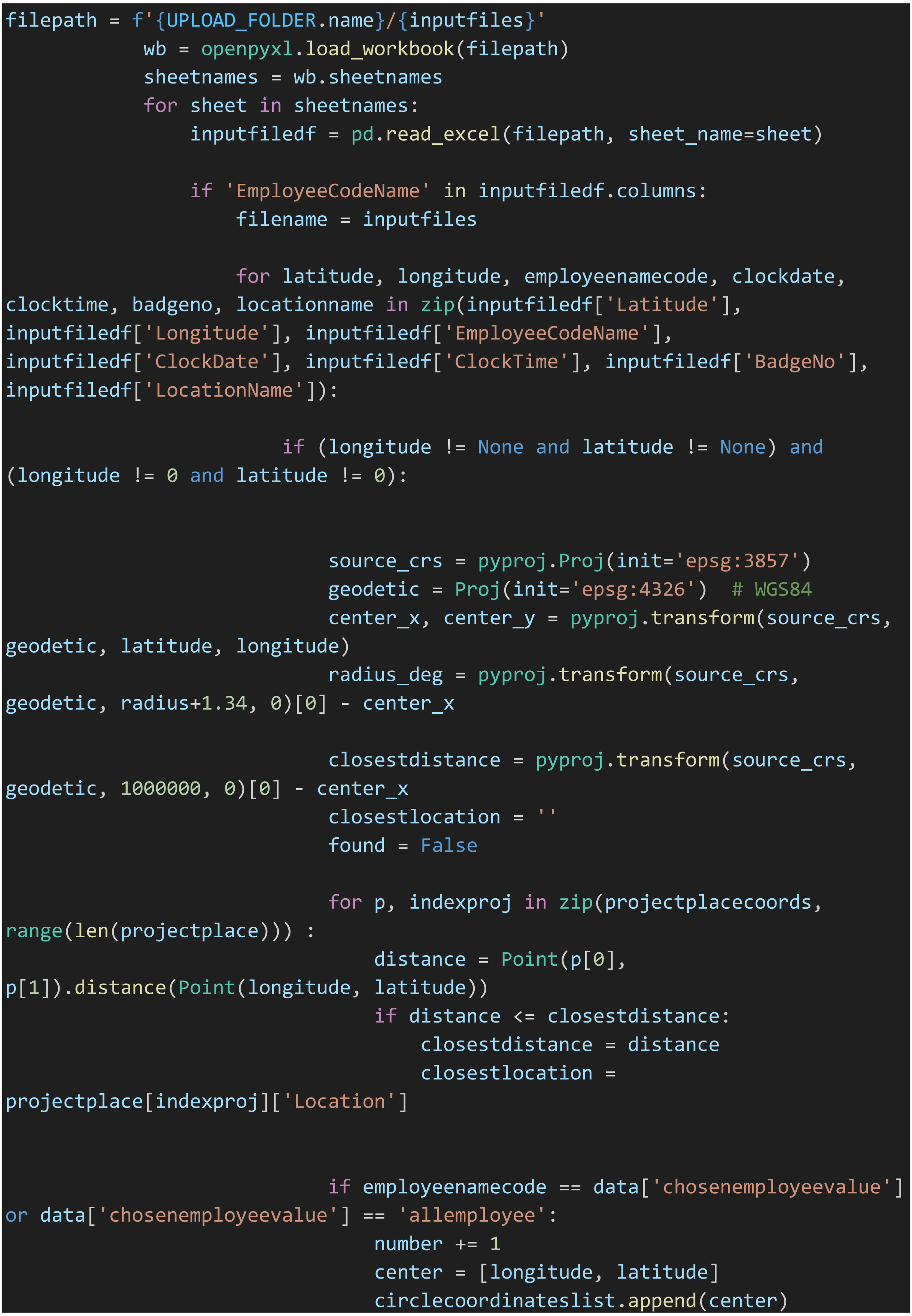
    b. Select One Employee
       When there is a change occur, the chosenemployee value will have the selected employee name and sent to app.py.

5) Now chosenemployee has a value, the output will be conditioned based on the value.

It will iterate the clock record dataframe and find the closest project site to the employee by measuring the distance between the center of the project site and the employee. Then, create the output text and store the values in the dictionary. (Example)

app.py (line 800)

```
for inputfiles in os.listdir(UPLOAD_FOLDER.name):
```



```
filepath = f'{UPLOAD_FOLDER.name}/{inputfiles}'
            wb = openpyxl.load_workbook(filepath)
            sheetnames = wb.sheetnames
            for sheet in sheetnames:
                inputfiledf = pd.read_excel(filepath, sheet_name=sheet)

                if 'EmployeeCodeName' in inputfiledf.columns:
                    filename = inputfiles

                    for latitude, longitude, employeenamecode, clockdate,
clocktime, badgeno, locationname in zip(inputfiledf['Latitude'],
inputfiledf['Longitude'], inputfiledf['EmployeeCodeName'],
inputfiledf['ClockDate'], inputfiledf['ClockTime'], inputfiledf['BadgeNo'],
inputfiledf['LocationName']):

                        if (longitude != None and latitude != None) and
(longitude != 0 and latitude != 0):


                            source_crs = pyproj.Proj(init='epsg:3857')
                            geodetic = Proj(init='epsg:4326')  # WGS84
                            center_x, center_y = pyproj.transform(source_crs,
geodetic, latitude, longitude)
                            radius_deg = pyproj.transform(source_crs,
geodetic, radius+1.34, 0)[0] - center_x

                            closestdistance = pyproj.transform(source_crs,
geodetic, 1000000, 0)[0] - center_x
                            closestlocation = ''
                            found = False

                            for p, indexproj in zip(projectplacecoords,
range(len(projectplace))) :
                                distance = Point(p[0],
p[1]).distance(Point(longitude, latitude))
                                if distance <= closestdistance:
                                    closestdistance = distance
                                    closestlocation =
projectplace[indexproj]['Location']


                            if employeenamecode == data['chosenemployeevalue']
or data['chosenemployeevalue'] == 'allemployee':
                                number += 1
                                center = [longitude, latitude]
                                circlecoordinateslist.append(center)
                                filteremployee.append({'Number': number,
'EmpName': employeenamecode, 'BadgeID':
badgeno,'Longitude':longitude,'Latitude':latitude,
                                                                'Center':cente
r,'ClockDate': clockdate.date(),'ClockTime' : clocktime,'sheetname': sheet,
'inputfile': inputfiles, 'shapetype': 'pin','closestlocation' :
closestlocation, 'locationname': locationname})
                                filteremployeename.append(employeenamecode)


                                content += f'\nPin {number}\nEmployee Name:
{employeenamecode}\nBadge ID: {badgeno}\nDate:
{clockdate.date().strftime("%d/%m/%Y")} {clocktime}\nProject Site:
{closestlocation}\nLocation Name: {locationname}\nCenter:\nLongitude:
{longitude} Latitude: {latitude}\nFile Name: {inputfiles}\nSheet Name:
{sheet}\n'
```

6) With the dictionary, I would need to still find the status of the employee which is whether they are within the range of the specified radius or 100m default. (Example)

app.py (line 929)

```
if filteremployee != [] and projectplacecoords != []:

        for i in range(len(filteremployee)):
            longitude = filteremployee[i]['Longitude']
            latitude = filteremployee[i]['Latitude']
            source_crs = pyproj.Proj(init='epsg:3857')
            geodetic = Proj(init='epsg:4326')  # WGS84
            center_x, center_y = pyproj.transform(source_crs, geodetic,
latitude, longitude)
            radius_deg = pyproj.transform(source_crs, geodetic, radius+1.34,
0)[0] - center_x #Y
            outerradius_deg = pyproj.transform(source_crs, geodetic,
radius+21.34, 0)[0] - center_x #Y1

            closestdistance = pyproj.transform(source_crs, geodetic, 1000000,
0)[0] - center_x
            closestlocation = ''
            found = False

            #to check if the user is in range and produce the status Y, Y1. N
            #Y - within 100m
            #Y1 - within 120m
            #N - outside the range
            for p, indexproj in zip(projectplacecoords,
range(len(projectplace))) :
                distance = Point(p[0], p[1]).distance(Point(longitude,
latitude))
                if distance <= radius_deg:
                    filteremployee[i]['ProjectCode'] =
projectplace[indexproj]['Location']
                    filteremployee[i]['Status'] = 'Y'
                    found = True
                    break
                elif distance <= outerradius_deg:
                    filteremployee[i]['ProjectCode'] =
projectplace[indexproj]['Location']
                    filteremployee[i]['Status'] = 'Y1'
                    found = True
                    break
                else:
                    if distance <= closestdistance:
                        closestdistance = distance
                        closestlocation = projectplace[indexproj]['Location']
                    filteremployee[i]['Status'] = 'N'

            if found == False:
                filteremployee[i]['ProjectCode'] = closestlocation
        employeedf = pd.DataFrame.from_dict(filteremployee)

        employeedf = employeedf[['BadgeID', 'EmpName', 'ClockDate',
'ClockTime', 'Latitude', 'Longitude', 'ProjectCode', 'Status']]

        employeedict = employeedf.to_dict('records')
```

7) It will store the data on one side to be displayed and to make it ready for download. After that, it will get the list of dates that the clock record occurred and get the earliest to the latest date. Then the date picker can give that range. (Making the date picker)

app.py (line 981)

```
for i in range(len(filteremployee)):
            getdate.append(filteremployee[i]['ClockDate'])

        getdate = list(set(getdate))
        getdate.sort()

        startdate = str(getdate[0])
        enddate = str(getdate[-1])
```

8) It will also have a list to put inside the employeename filter and send it back to index.html

app.py (line 997)

```
filteremployeename = list(set(filteremployeename))
return json.dumps({'circle_coords': circlecoordinateslist,
'content':htmlcontent, 'notify':notify, 'employeenames':filteremployeename,
'startdate': startdate, 'enddate': enddate,
'checkmobileclock':checkmobileclock, 'locationlist': locationlist})
```

9) In index.html, the zoom in filter will be filled by the locationlist allowing the user to zoom into a project site or zoom out. (Example)

index.html (line 719)

```
                            const option = document.createElement('option');
                            option.value = 'allprojectsite';
                            option.textContent = 'Zoom out of Project Sites';
                            dropdownplace.appendChild(option);

                            data.locationlist.forEach(site => {
                                const option =
document.createElement('option');
                                option.value = site;
                                option.textContent = site;
                                dropdownplace.appendChild(option);
                            })
                            //to zoom in the project sites
                            $(document).ready(function () {
                                $('#dropdownprojectsite').select2({
                                    theme: 'theme1'
                                });
                                $('#dropdownprojectsite').on('change',
function () {
                                    localStorage.setItem('dropdownprojectsitec
hanged', this.value);
                                });

                            });
```

10) Then same thing for the employeename list which is the same as the drop-down list before the process. (You can refer here)

11) The earliest date and the latest date will be used for setting the date range in the date picker. (Example) (Getting the data range)

index.html (line 803)

```
const startDate = data.startdate;
const endDate = data.enddate;

const datePicker = document.getElementById('datepicker');
datePicker.style.display = 'block';
datePicker.min = startDate;
datePicker.max = endDate;
```

12) Setting the datepicker to activate when the selection changes sending the selected filter back to app.py to get the output of the selected filter. Then app.py sends the response back to index.html

index.html (line 511)

```
const datepicker = document.getElementById('datepicker');
        datepicker.addEventListener('change', function () {
            var employee = '';
            if (localStorage['dropdownchanged']) {
                var employee = document.getElementById('dropdown').value;
            }
            var coordradius = document.getElementById('circle-radius-
coords').value;
            console.log('Selected value:', datepicker.value);
            console.log(employee);
            fetch('/searchoutput', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ 'finddate': this.value, 'radius':
coordradius, 'employee': employee })
            })
```

13) It will send the coordinates of the employee, location and radius to map.html to set up the layers for visualizing.

index.html (line 817)

```
localStorage.setItem('shapetype', 'circle')
                                localStorage.setItem('coordradius',
coordradius);
                                localStorage.setItem('circlecoords',
JSON.stringify(data.circle_coords));
                                localStorage.setItem('checkmobileclock',
data.checkmobileclock);
```

```
                                localStorage.setItem('locationlist',
JSON.stringify(data.locationlist))
                                localStorage.setItem('projectplacecoords',
JSON.stringify(data.projectplacecoords));
                                downloadbut.style.display = 'block';
                                generatingnotify.style.display = 'none';
                                const coordtextbox =
document.getElementById('coordtextbox');
                                coordtextbox.style.display = 'block';
                                showNotification(data.notify, true)
                                console.log('Message sending to map.html:',
data);
                                var mapFrame =
document.getElementById('mapFrame');
                                mapFrame.contentWindow.postMessage('coordlocat
ion', '*');
```

14) In map.html, it will check the message event.data whether it is coordlocation and trigger the visualizeemployee function. The function creates the layers, which is the project site area and the markers

map.html (line 470)

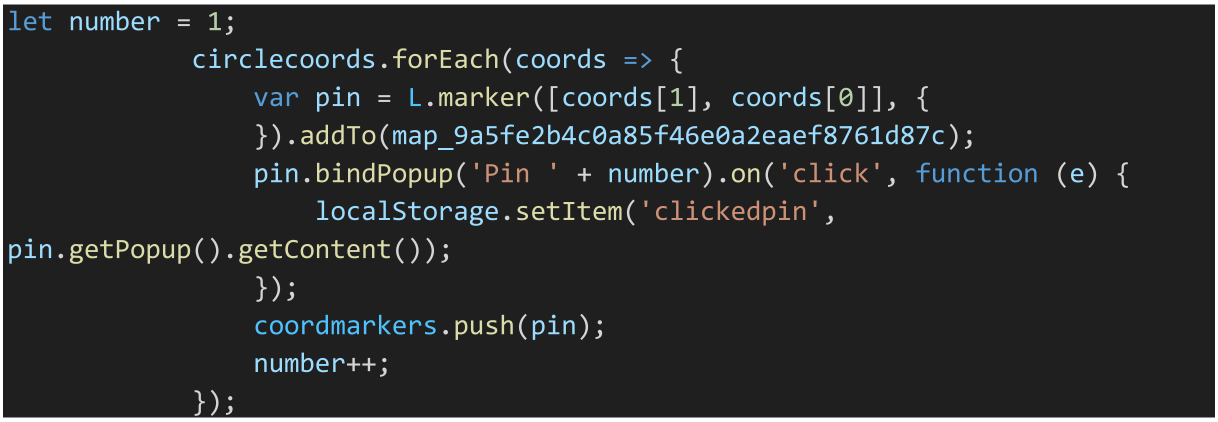
```
window.addEventListener("message", function (event) {
      if (event.data === 'coordlocation' && localStorage['visualize'] ===
'true') {
            visualizeemployee(
```

map.html (line 211) (visualizeemployee)

```
function visualizeemployee() {
        var radius = localStorage.getItem('coordradius');
        var circlecoords = localStorage['circlecoords'];
        var projectplacecoords = localStorage['projectplacecoords'];
        var radiuslist = localStorage['radiuslist'];
        var locationlist = localStorage['locationlist']
        circlecoords = JSON.parse(circlecoords);
        if (radiuslist) {
            radiuslist = JSON.parse(radiuslist);
        }
        if (projectplacecoords) {
            projectplacecoords = JSON.parse(projectplacecoords);
        }
        if (locationlist) {
            locationlist = JSON.parse(locationlist);
        }

        //remove existing layer
```

Creating the employee markers

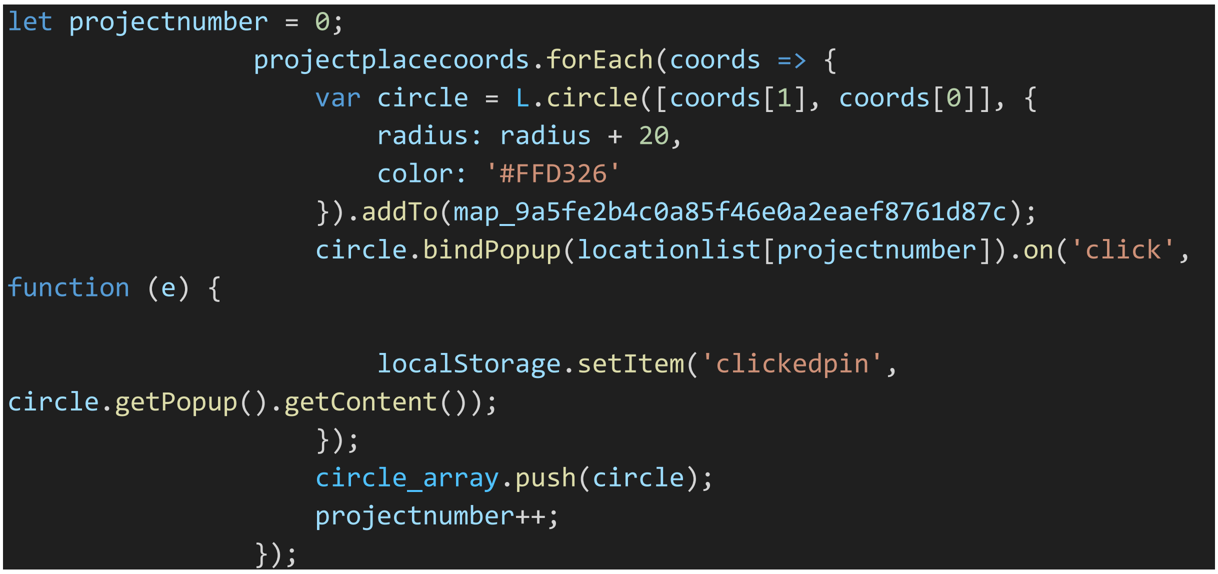
map.html (line 248) (visualizeemployee)



```
let number = 1;
      circlecoords.forEach(coords => {
        var pin = L.marker([coords[1], coords[0]], {
        }).addTo(map_9a5fe2b4c0a85f46e0a2eaef8761d87c);
        pin.bindPopup('Pin ' + number).on('click', function (e) {
          localStorage.setItem('clickedpin',
pin.getPopup().getContent());
        });
        coordmarkers.push(pin);
        number++;
      });
```

Creating the project site radius + 20m (Yellow Circle)

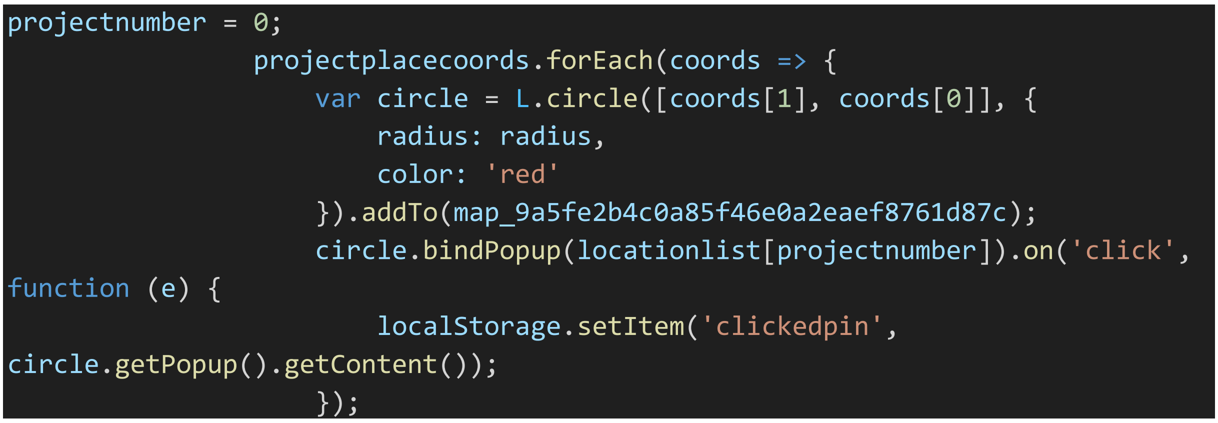
map.html (line 266) (visualizeemployee)



```
let projectnumber = 0;
        projectplacecoords.forEach(coords => {
          var circle = L.circle([coords[1], coords[0]], {
            radius: radius + 20,
            color: '#FFD326'
          }).addTo(map_9a5fe2b4c0a85f46e0a2eaef8761d87c);
          circle.bindPopup(locationlist[projectnumber]).on('click',
function (e) {

            localStorage.setItem('clickedpin',
circle.getPopup().getContent());
          });
          circle_array.push(circle);
          projectnumber++;
        });
```

Creating the project site radius (Red Circle)

map.html (line 283) (visualizeemployee)



```
projectnumber = 0;
        projectplacecoords.forEach(coords => {
          var circle = L.circle([coords[1], coords[0]], {
            radius: radius,
            color: 'red'
          }).addTo(map_9a5fe2b4c0a85f46e0a2eaef8761d87c);
          circle.bindPopup(locationlist[projectnumber]).on('click',
function (e) {
            localStorage.setItem('clickedpin',
circle.getPopup().getContent());
          });
```

```
                    circle_array.push(circle);
                    projectnumber++;
                });
```

Creating the center marker in the project site (Red Pin). It is forced to have its content pop up, so user don’t need to click on every pin to figure out what project site they are. Additionally, when the user clicks on the pin, the map will zoom in to that project site. (Example)
map.html (line 301) (visualizeemployee)

```
projectnumber = 0;
                projectplacecoords.forEach(coords => {
                    var pin = L.marker([coords[1], coords[0]], {
                        icon: redIcon
                    }).addTo(map_9a5fe2b4c0a85f46e0a2eaef8761d87c);
                    var popup = L.popup({ closeOnClick: false, autoClose:
false })
                        .setContent(locationlist[projectnumber])
                        .setLatLng([coords[1], coords[0]]);
                    pin.bindPopup(popup).on('click', function (e) {
                        localStorage.setItem('clickedpin',
pin.getPopup().getContent());
                        if (!pin.getPopup().isOpen()) {
                            pin.openPopup();
                        }
                        zoominprojectsite(coords[1], coords[0], 18);
                    });
                    popup.addTo(map_9a5fe2b4c0a85f46e0a2eaef8761d87c);
                    coordmarkers.push(pin);
                    if (!pin.getPopup().isOpen()) {
                        pin.openPopup();
                    }
                    projectnumber++;
```

The redIcon is from a URL and can be adjusted. For more information: Documentation - Leaflet - a JavaScript library for interactive maps (leafletjs.com)

You can put any image to represent as the marker.

map.html (line 240) (visualizeemployee)

```
var redIcon = new L.Icon({
                iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-
color-markers/master/img/marker-icon-red.png',
                iconSize: [25, 41],
                iconAnchor: [12, 41],
                popupAnchor: [1, -34],
            });
```

- Employee Filter feature & Date Picker feature (after processing)
    1) The making of the filter is the same as before process, but the trigger is different. When it changes, it will fetch the search output in app.py. That will give the filtered output in the container and at the same time changing the visuals on the map.

index.html (line 775)

```
fetch('/searchoutput', {
                                        method: 'POST',
                                        headers: {
                                            'Content-Type': 'application/json'
                                        },
                                        body: JSON.stringify({ 'employee':
this.value, 'radius': coordradius, 'finddate': finddate })
                                    })
```

2) This will call out the function in app.py and check the value of the filtered option.
It will match the filtered name to the option changing the output of the text except for the csv file. It also takes the coordinates of the filtered employee or date or both.

app.py (line 597)

```
for i in range(len(filteremployee)):
        shapetype = filteremployee[i]['shapetype']

        if filteremployee[i]['EmpName'] == findemployee and
filteremployee[i]['ClockDate'] == finddate: #if they are the same then ot will
produce this
            number += 1
            if filteremployee[i]['shapetype'] == 'circle':
                searchcontent += f"\nCircle {number}\nEmployee Name:
{filteremployee[i]['EmpName']}\nBadge ID:
{filteremployee[i]['BadgeID']}\nDate:
{filteremployee[i]['ClockDate'].strftime('%d/%m/%Y')}
{filteremployee[i]['ClockTime']}\nProject Site:
{filteremployee[i]['closestlocation']}\nLocation Name:
{filteremployee[i]['locationname']}\nCenter:\nLongitude:
{filteremployee[i]['Center'][0]} Latitude:
{filteremployee[i]['Center'][1]}\nFile Name:
{filteremployee[i]['inputfile']}\nSheet Name:
{filteremployee[i]['sheetname']}\n"
            else:
                 #same searchontent layout
            coordlist.append(filteremployee[i]['Center'])
            shapetype = filteremployee[i]['shapetype']

            canbefound = True

        #to prevent producing the one that cannot be found
        elif (filteremployee[i]['EmpName'] == findemployee and finddate == '')
or ((findemployee == '' or findemployee == 'allemployee') and
filteremployee[i]['ClockDate'] == finddate):
            number += 1
            if filteremployee[i]['shapetype'] == 'circle':
                #same searchontent layout
            else:
                #same searchontent layout
            canbefound = True

    if canbefound == False:
        searchcontent = '<b>Not Found</b>'

#More codes…

return json.dumps({'findcoords': coordlist, 'content':htmlcontent,
'shapetype': shapetype,'projectplacecoords': projectplacecoords})
```

3) Once app.py sent the content and the coordinates list that was filtered out, index.html will sent it over to map.html

```
.then(response => response.json())
                                        .then(data => {
                                            var textbox =
document.getElementById('coordtext');
                                            textbox.innerHTML = data.content;
                                            localStorage.setItem('shapetype',
data.shapetype);
                                            localStorage.setItem('circlecoords
', JSON.stringify(data.findcoords));
                                            if (coordradius) {
                                                localStorage.setItem('coordrad
ius', coordradius);
                                            }
                                            var mapFrame =
document.getElementById('mapFrame');
                                            mapFrame.contentWindow.postMessage
('coordlocation', '*');
                                        })
```

4) It will go through the same function in map.html. Removing the existing layers and replacing them with the filtered ones. (Code Reference: visualizeemployee)

- Zooming in and out of the project site using the dropdown box (Example)
  1) At index.html (line 719), I mentioned when the dropdown box selection is changed, it will store the value of the selected option which will notify map.html to zoom into that project site or zoom out.

map.html (line 525)

```
window.addEventListener('storage', () => {
if (localStorage['dropdownprojectsitechanged']) {
```

map.html (line 206) (zoominprojectsite)

```
function zoominprojectsite(lat, lng, zoom) {
        map_9a5fe2b4c0a85f46e0a2eaef8761d87c.setView(new L.LatLng(lat, lng),
zoom);
    }
```

If the selected filter is all project site, it will zoom out to see the overview of Singapore

map.html (line 529)

```
if (localStorage['dropdownprojectsitechanged'] === 'allprojectsite') {
                zoominprojectsite(1.3521, 103.8198, 12)
            }
```

If it is a specific project site, it will search the markers' popup content and if the name matches, it will zoom in to that marker.

map.html (line 531)

```
else {
                coordmarkers.forEach(marker => {

                    if (marker.getPopup().getContent() ===
localStorage['dropdownprojectsitechanged']) {
                        console.log(`Popup content:
${marker.getPopup().getContent()}`);
                        zoominprojectsite(marker.getLatLng().lat,
marker.getLatLng().lng, 19);
                    }
                });
```

- Uploading of other excel files (Example)
  - The process
    1) It will go through the files in the UPLOAD_FOLDER and checks if there is radius in the column

app.py (line 855)

```
elif 'Radius' in inputfiledf.columns:
                    content = ''
                    print('here')
                    otherfilename = inputfiles
                    content += '\n' + '-' * 50 + '\n\n'
                    content += f'Sheet Name: {sheet}\n'
                    for latitude, longitude, fileradius in zip(inputfiledf['Latitude'], inputfiledf['Longitude'], inputfiledf['Radius']):
                        if (longitude != None and latitude != None) and (longitude != 0 and latitude != 0):
                            if data['coordradius'] == '':
                                circlecoordinateslist.append([longitude, latitude])
                                radiuslist.append(fileradius)
                                number += 1
                                content += f'\nCircle {number}\nCenter:\nLongitude: {longitude} Latitude: {latitude}\nRadius: {fileradius}\n'
                                print(radiuslist)
                            else:
                                center = [longitude, latitude]
                                circlecoordinateslist.append(center)
                                number += 1
                                content += f'\nCircle {number}\nCenter:\nLongitude: {longitude} Latitude: {latitude}\n'

                    withradius =True
```

2) Deciding what layer to create

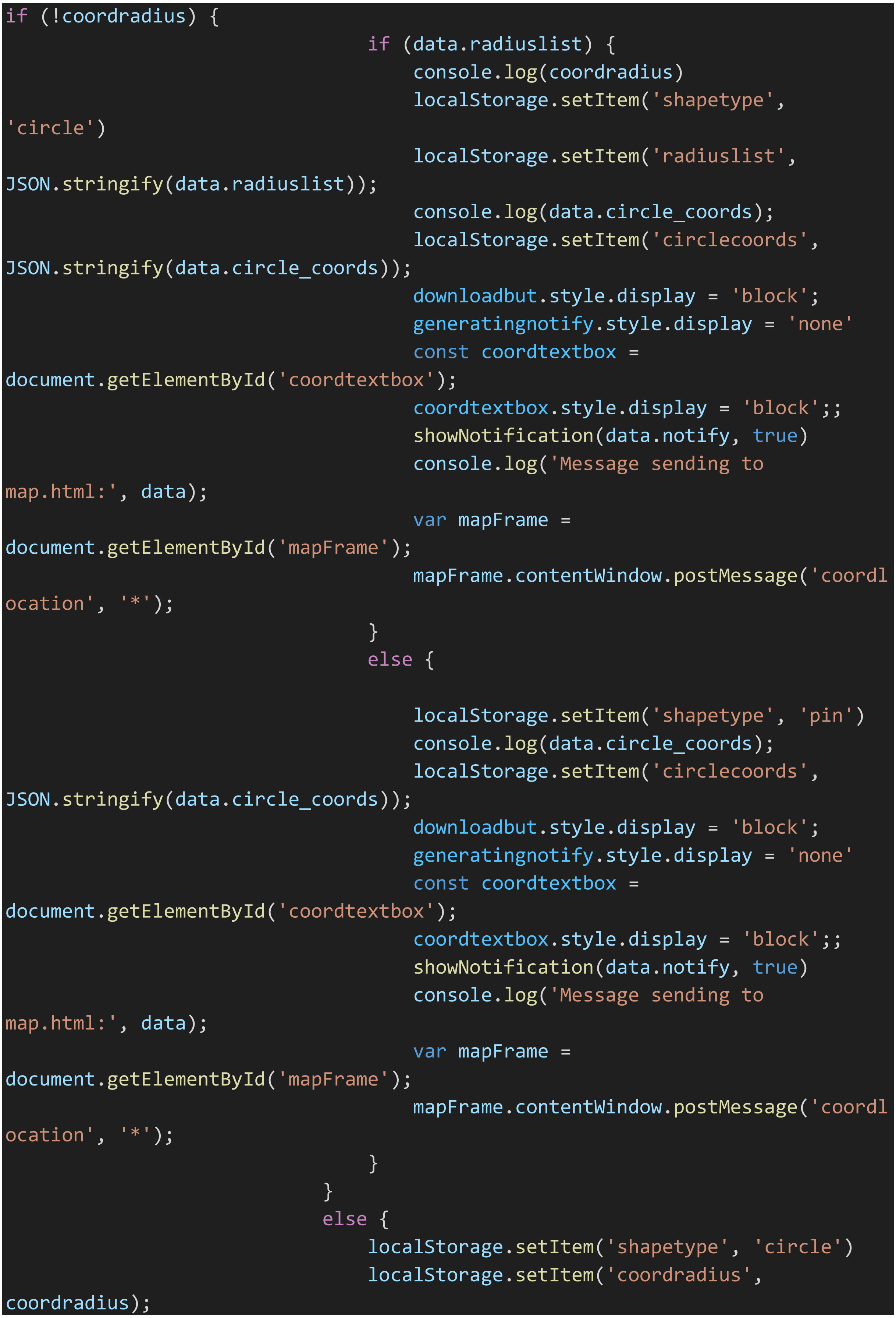
Create circle

- No specified radius, have radius column
- Have specified radius, have radius column (Will stick to specified radius)
- Have specified radius, no radius column

Create Marker

- Do not have both

app.py (line 850)



```
if (!coordradius) {
                                if (data.radiuslist) {
                                    console.log(coordradius)
                                    localStorage.setItem('shapetype', 
'circle')
                                    localStorage.setItem('radiuslist', 
JSON.stringify(data.radiuslist));
                                    console.log(data.circle_coords);
                                    localStorage.setItem('circlecoords', 
JSON.stringify(data.circle_coords));
                                    downloadbut.style.display = 'block';
                                    generatingnotify.style.display = 'none'
                                    const coordtextbox = 
document.getElementById('coordtextbox');
                                    coordtextbox.style.display = 'block';;
                                    showNotification(data.notify, true)
                                    console.log('Message sending to 
map.html:', data);
                                    var mapFrame = 
document.getElementById('mapFrame');
                                    mapFrame.contentWindow.postMessage('coordl
ocation', '*');
                                }
                                else {

                                    localStorage.setItem('shapetype', 'pin')
                                    console.log(data.circle_coords);
                                    localStorage.setItem('circlecoords', 
JSON.stringify(data.circle_coords));
                                    downloadbut.style.display = 'block';
                                    generatingnotify.style.display = 'none'
                                    const coordtextbox = 
document.getElementById('coordtextbox');
                                    coordtextbox.style.display = 'block';;
                                    showNotification(data.notify, true)
                                    console.log('Message sending to 
map.html:', data);
                                    var mapFrame = 
document.getElementById('mapFrame');
                                    mapFrame.contentWindow.postMessage('coordl
ocation', '*');
                                }
                            }
                            else {
                                localStorage.setItem('shapetype', 'circle')
                                localStorage.setItem('coordradius', 
coordradius);
```

```
                                console.log(data.circle_coords);
                                localStorage.setItem('circlecoords',
JSON.stringify(data.circle_coords));
                                downloadbut.style.display = 'block';
                                generatingnotify.style.display = 'none'
                                const coordtextbox =
document.getElementById('coordtextbox');
                                coordtextbox.style.display = 'block';;
                                showNotification(data.notify, true)
                                console.log('Message sending to map.html:',
data);
                                var mapFrame =
document.getElementById('mapFrame');
                                mapFrame.contentWindow.postMessage('coordlocat
ion', '*');
                            }
```

Create Circle

map.html (line 333) (visualizeemployee)

```
if (radiuslist && !radius) {
                    number = 1;
                    circlecoords.forEach((coords, index) => {
                        var circle = L.circle([coords[1], coords[0]], {
                            radius: radiuslist[index],
                        }).addTo(map_9a5fe2b4c0a85f46e0a2eaef8761d87c);
                        circle.bindPopup("Circle " + number).on('click',
function (e) {
                            console.log('clicked')
                            localStorage.setItem('clickedpin',
circle.getPopup().getContent());
                            localStorage.setItem('status', event.data);
                        });
                        circle_array.push(circle);
                        number++;
                        console.log(number);
                    })

                }
                else {
                    number = 1;
                    circlecoords.forEach(coords => {
                        var circle = L.circle([coords[1], coords[0]], {
                            radius: radius,
                        }).addTo(map_9a5fe2b4c0a85f46e0a2eaef8761d87c);
                        circle.bindPopup("Circle " + number).on('click',
function (e) {
```

```
                            console.log('clicked')
                            localStorage.setItem('clickedpin', 
circle.getPopup().getContent());
                            localStorage.setItem('status', event.data);
                        });
                        circle_array.push(circle);
                        number++;
                        console.log(number);
                    });
                }
```

Create Marker

map.html (line 370) (visualizeemployee

```
if (localStorage['shapetype'] === 'pin') {
            let number = 1;
            circlecoords.forEach(coords => {
                var pin = L.marker([coords[1], coords[0]], {
                    radius: radius,
                }).addTo(map_9a5fe2b4c0a85f46e0a2eaef8761d87c);
                pin.bindPopup('Pin ' + number).on('click', function (e) {
                    console.log('clicked')
                    localStorage.setItem('clickedpin', 
pin.getPopup().getContent());
                    localStorage.setItem('status', event.data);
                });
                coordmarkers.push(pin);
                number++;

            });
        }
```

<u>Download Output</u>

- Circle Generator download

Creating of download button

```
<button id="downloadbutton" class="downloadbut" style="display: 
none;">Download</button>
```

Script

index.html (line 917)

```
document.getElementById('downloadbutton').addEventListener('click', function 
(event) { //to download the tempfile which is turned into a url
                fetch('/send_tempfile', { method: 'POST' })
```

Creates txt file for the output in the container.

app.py (line 471)

```
txt_file = open(os.path.join(OUTPUT_FOLDER.name, 'output.txt'), 'w',
encoding='utf-8') #It is the output in the output container
    if searchcontent != '':
        txt_file.write(searchcontent)
    elif replacecontent != '':
        txt_file.write(replacecontent)
    else:
        print(content)
        txt_file.write(content)
    txt_file.close()
```

Creates an excel sheet for the coordinates that were generated

app.py (line 508)

```
xlsx_file_path = os.path.join(OUTPUT_FOLDER.name, f"generated_coords.xlsx")
with open(xlsx_file_path, 'wb') as xlsx_file:
            if circlegenerateddf.empty is not True:
                circlegenerateddf.to_excel(xlsx_file, index=False)
zipfilepath = os.path.join(ZIP_FOLDER.name, f'generated.zip')
with ZipFile(zipfilepath, 'w', zipfile.ZIP_DEFLATED) as zip_file:
            for root, dirs, files in os.walk(OUTPUT_FOLDER.name):
                for file in files:
                    filepath = os.path.join(root, file)
                    zip_file.write(filepath, os.path.relpath(filepath,
OUTPUT_FOLDER.name))
response = send_file(zipfilepath, as_attachment=True,
download_name=f'generated.zip')

return response
```

index.html will create a link for the zip file and click on it to download the zip file for the user

index.html (line 918)

```
.then(response => {
                        //extract filename from response headers
                        const contentDisposition =
response.headers.get('Content-Disposition');
                        let filename = 'output.zip';
                        if (contentDisposition) {
                            const filenameRegex =
/filename[^;=\n]*=((['"]).*?\2|[^;\n]*)/;
```

```
              const matches =
filenameRegex.exec(contentDisposition);
              if (matches != null && matches[1]) {
                filename = matches[1].replace(/['"]/g, '');
              }
            }
            console.log(filename)
            localStorage.setItem('filename', filename)
            //return the blob object
            return response.blob();
          })

          .then(blob => {

            const url = URL.createObjectURL(blob);
            const link = document.createElement('a');
            link.href = url;
            link.download = localStorage['filename'];
            link.click();

            URL.revokeObjectURL(url);

            showNotification('Map updated and result downloaded!',
true);
          })
          .catch(error => {
            console.log(error)
            showNotification('An error occurred when downloading
the file', false);
          });
```

- Clock Record download

Goes through the same process but at app.py, it takes the filename that was uploaded and then names the zip as that. It also takes the dictionary that has the employee status turn into csv and the output from the output container same as app.py (line 471).

app.py (line 506)

```
if employeedict != {} and projectplace != []: #to store employee details in
the csv file
    print(employeedict)
    csv_file = open(os.path.join(OUTPUT_FOLDER.name,
f"{filename.rstrip('.xlsx')}_out.csv"), 'w', newline='')
    fields = ['BadgeID', 'EmpName', 'ClockDate', 'ClockTime', 'Latitude',
'Longitude', 'ProjectCode', 'Status']
    writer = csv.DictWriter(csv_file, fieldnames=fields)
    writer.writeheader()
    writer.writerows(employeedict)
```

```
        csv_file.close()

        zipfilepath = os.path.join(ZIP_FOLDER.name,
f'{filename.rstrip(".xlsx")}.zip')
        print(zipfilepath)

        with ZipFile(zipfilepath, 'w', zipfile.ZIP_DEFLATED) as zip_file:
            for root, dirs, files in os.walk(OUTPUT_FOLDER.name):
                for file in files:
                    filepath = os.path.join(root, file)
                    zip_file.write(filepath, os.path.relpath(filepath,
OUTPUT_FOLDER.name))

        response = send_file(zipfilepath, as_attachment=True,
download_name=f'{filename.rstrip(".xlsx")}.zip')
```

- Other Excel download

It will just give the output text file which is the same as app.py (line 471).

Converting to an EXE application

When converting for the code to be suitable for EXE applications. It is required to make the file path of what EXE application would refer to.

Comment this out (line 96)

```
#excel_file = 'Project Sites.xlsx' #<= use this when you are running in vscode
```

Uncomment this (line 98)

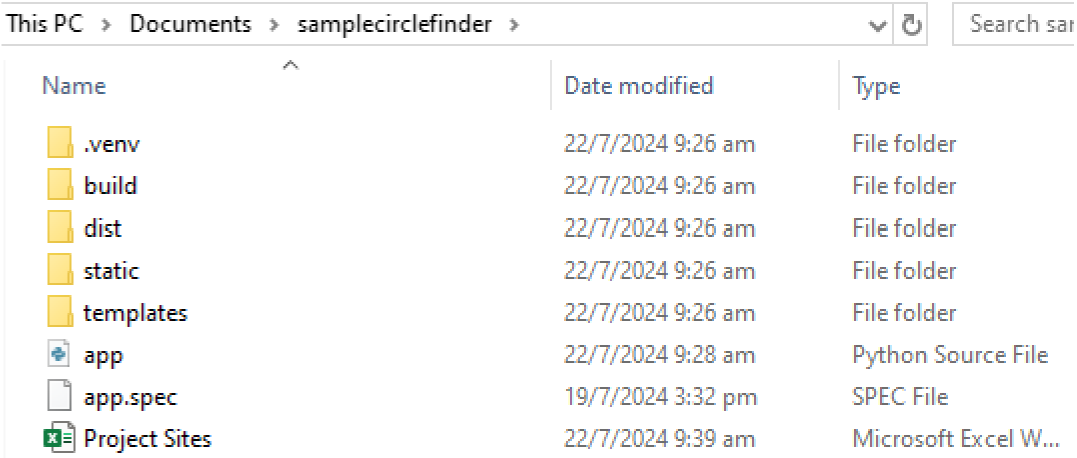
```
excel_file = os.path.join(sys._MEIPASS, 'Project Sites.xlsx') #this is when
you are converting into a .exe file using pyinstaller
```

Then you can turn it into an exe application in the terminal

```
pyinstaller --onefile --add-data "static:static" --add-data "templates:templates" --
add-data "Geo-location Of All SCS Construction Sites.xlsx;." app.py
```

How to maintain the application

If you need to change the project site file, just to the directory of the python script and replace the current one like this



Name	Date modified	Type
.venv	22/7/2024 9:26 am	File folder
build	22/7/2024 9:26 am	File folder
dist	22/7/2024 9:26 am	File folder
static	22/7/2024 9:26 am	File folder
templates	22/7/2024 9:26 am	File folder
app	22/7/2024 9:28 am	Python Source File
app.spec	19/7/2024 3:32 pm	SPEC File
Project Sites	22/7/2024 9:39 am	Microsoft Excel W...

Then in app.py, change the name of the excel sheet

This code (when it comes to testing in VScode)

```
excel_file = '__________________.xlsx' <- in the string change the file
```

and this (when it comes to putting into the exe application)

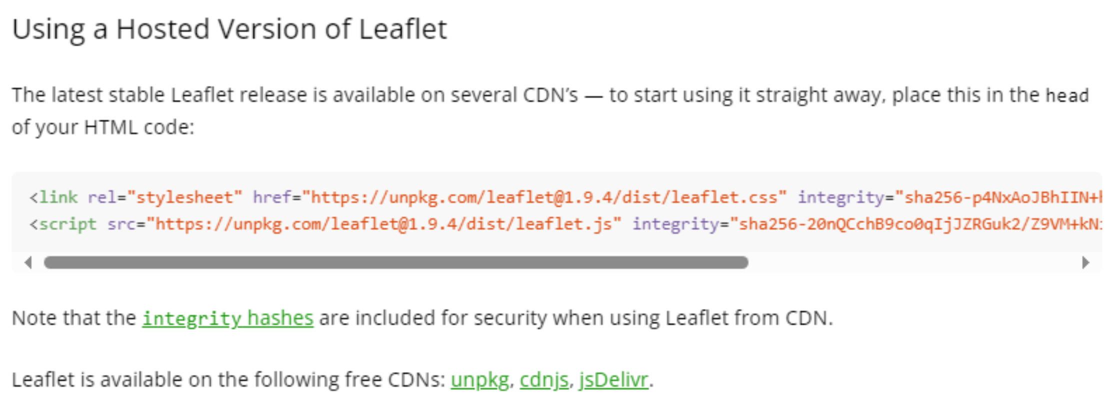
```
excel_file = os.path.join(sys._MEIPASS, '___________.xlsx')
```

OpenStreet map will not update itself so you can check the latest versions in this website (https://leafletjs.com/download.html).

Then remove this in map.html (line 37)

```
<script
src="https://cdn.jsdelivr.net/npm/leaflet@1.9.4/dist/leaflet.min.js"></script>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/leaflet@1.9.4/dist/leaflet.min.css"
rel="stylesheet">
```

And paste the new one from here



Using a Hosted Version of Leaflet

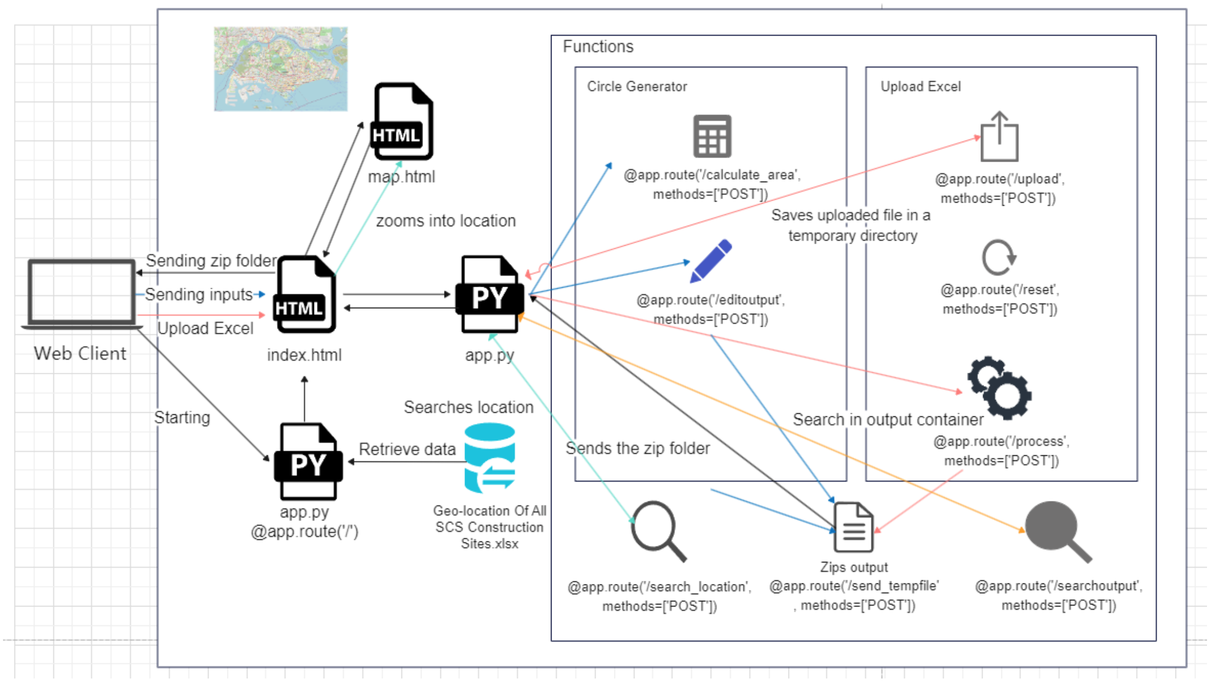
The latest stable Leaflet release is available on several CDN's — to start using it straight away, place this in the `head` of your HTML code:

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" integrity="sha256-p4NxAoJBhIIN+l
<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js" integrity="sha256-20nQCchB9co0qIjJZRGuk2/Z9VM+kN:
```

Note that the integrity hashes are included for security when using Leaflet from CDN.

Leaflet is available on the following free CDNs: unpkg, cdnjs, jsDelivr.

Architecture Diagram


Functions
Circle Generator
Upload Excel
map.html
zooms into location
Sending zip folder
Sending inputs
Upload Excel
Web Client
index.html
app.py
Starting
app.py
@app.route('/')
Searches location
Retrieve data
Geo-location Of All SCS Construction Sites.xlsx
@app.route('/calculate_area', methods=['POST'])
@app.route('/upload', methods=['POST'])
Saves uploaded file in a temporary directory
@app.route('/editoutput', methods=['POST'])
@app.route('/reset', methods=['POST'])
Search in output container
@app.route('/process', methods=['POST'])
Sends the zip folder
Zips output
@app.route('/search_location', methods=['POST'])
@app.route('/send_tempfile' , methods=['POST'])
@app.route('/searchoutput', methods=['POST'])