Small Project

MSD Assets Status Tracker System

By: Alyshia West

Table of contents

Implementation

Asset tracking systems can help MSD improve how their assets are used by providing information on their location and how they are being used. This can help optimize routes and schedules and prevent unauthorized use.

Abstract of project

This project is aimed to develop an MSD Asset Tracker System, which is essential for effective asset management within the organization. The asset tracker will enable the tracking of asset availability, ensuring that all resources are accounted for and utilized efficiently.

Functional specifications of the project

The following is a list of the functionalities of the system.

1. The system should be accessible to MSD Users and multiple users at same time.
2. UI is expected to be interactive and user friendly.
3. Assets shows the result in Single page.

UI Design

Considering the needs and expectations for this project the goal is to create interfaces which users find easy to use and pleasurable. Including informational components, interactive elements, visual elements, and containers.

Mockup

Sno.	Feature N.	Featyre Type		Active Sta	Abandon	Removed	Consolida	Design		Operating	To Be Rem	Inactive
	Manhole	SS		170101	2328							
	Inlet	SW										

Asset Status Tracker

Prototype

Asset Status Information

NO.	ACTIVE	INACTIVE	OPERATING	CONSOLIDATED

Copyright 2024 MSD: Project Clear | Wastewater + Stormwater

Data Flow Diagrams

Context level DFD
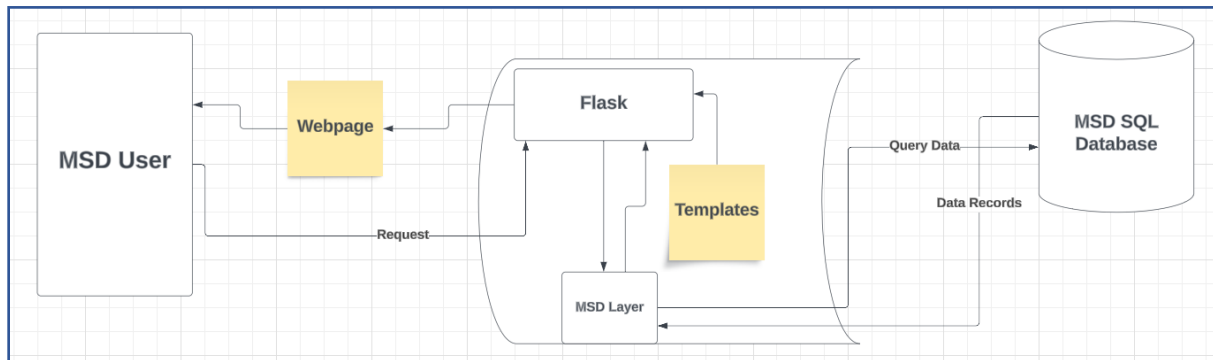
Milestones and Timelines

	Milestone Name	Milestone Description	Timeline	Outcome
1	Technology familiarization	Understanding of the technology needed to implement the project.	Week 1	Advancing my skills and knowledge by creating mini projects that will ultimately converge to form the final product.
2	Database	Decide Database design.	Week 2	Stayed alongside with the database the organization currently uses (SQL developer and Oracle).
3	High-level and Detailed Design	Listing down all possible scenarios like different types of tutorials. Then coming up with flow-charts or pseudocode to handle the scenario.	Week 2	

4	Implementation of the front-end of the system	Implementation of the main screen giving the login, screen that follows the login giving various options, screens for each of the options.	Week 2/3	Used HTML coding in Visual Studio to enhance and maintain the front-end of the system.
5	Integrating the front-end with the database	The front-end developed in the earlier milestone will now be able to update the employee database. In short, the system should be ready for integration testing.	Week 3	Developed code that integrates the database with the front-end layout, ensuring seamless data flow and user experience.
6	Integration Testing	The system should be thoroughly tested by running all the test cases written for the system.	Week 4	
7	Final Review	Issues found during the previous milestone are fixed and the system is ready for the final review.	Week 5	

Technical Requirements

Technology Stack

Frontend: Visual studio code

Backend: Python (Django/Flask)

Database: MySQL, Oracle

Hosting: Google Cloud, Microsoft Edge, GoLive

Code

Python

Imports

```python
from flask import Flask, render_template
import oracledb
```

Oracle database connection

```python
app = Flask(__name__)

dsn = oracledb.makedsn("RJMSDDBT02","1521",service_name="gisdevl")
username="username" #Replace with your username
password="password" #Replace with your password
```

Tables from database

```
ss_table_names = ["ssManhole", "ssInlet", "ssIntakeOutfall", "ssCleanout", "ssNetworkJunction",
                  "ssNetworkStructure","ssValve", "SSP", "ssPump", "ssPumpStation","ssGhostPipe",
                  "ssGravityMain", "ssPressurizedMain", "ssCasing"]

sw_table_names = ["swManhole", "swInlet", "swIntakeOutfall", "swCleanout", "swNetworkJunction",
                  "swNetworkStructure", "swValve", "swPump", "swPumpStation", "swGhostPipe",
                  "swGravityMain", "swPressurizedMain", "swCasing", "swManagementPoint", "swManagementLine"]
```

Query

```python
def fetch_status_counts(table_name, feature_type):
    query = f"""
        SELECT
            COUNT(CASE WHEN STATUS = 1 THEN 1 END) AS Active,
            COUNT(CASE WHEN STATUS = 2 THEN 1 END) AS Abandoned,
            COUNT(CASE WHEN STATUS = 3 THEN 1 END) AS Removed,
            COUNT(CASE WHEN STATUS = 4 THEN 1 END) AS Consolidated,
            COUNT(CASE WHEN STATUS = 5 THEN 1 END) AS Design,
            COUNT(CASE WHEN STATUS = 6 THEN 1 END) AS Operating,
            COUNT(CASE WHEN STATUS = 7 THEN 1 END) AS To_Be_Removed,
            COUNT(CASE WHEN STATUS = 8 THEN 1 END) AS Inactive
        FROM MSD.{table_name}
        """

    with oracledb.connect(user=username, password=password, dsn = dsn) as conn:
        cursor = conn.cursor()
        cursor.execute(query)

        result = cursor.fetchone()

        columns = ["Active", "Abandoned", "Removed", "Consolidated", "Design", "Operating", "To_Be_Removed", "Inactive"]

        data = {
            "Feature_Name": table_name,
            "Feature_Type": feature_type,
            **dict(zip(columns, result))}
    return data
```

Table data

```python
# Routes
@app.route('/')
def index():

    table_data = []

    # SS Tables
    for index, table_name in enumerate(ss_table_names, start = 1):
        status_counts = fetch_status_counts(table_name, "SS")
        status_counts["S_No"] = index
        table_data.append(status_counts)

    # sw Tables
    for index, table_name in enumerate(sw_table_names, start = 1):
        status_counts = fetch_status_counts(table_name, "SW")
        status_counts["S_No"] = index
        table_data.append(status_counts)

    return render_template('index.html', table_data=table_data)

if __name__ == '__main__':
    app.run(debug=True)
```

HTML

Layout designs

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>CSS Template</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
* {
  box-sizing: border-box;
}

body {
  font-family: Garamond, Serif;
}

/* the header */
header {
  background-color: #ffffff;
  padding: 40px;
  text-align: center;
  font-size: 40px;
  color: #003c96;
}

/* the list inside the menu */
nav ul {
  list-style-type: none;
  padding: 0;
}

table {
  float: middle;
  padding: 20px;
  width: 80%;
  background-color: #f1f1f1;
  height: 500px;
}

/*  floats after the columns */
section::after {
  content: "";
  display: table;
  clear: both;
}
```

```css
/*  the footer */
footer {
  background-color: #003c96;
  padding: 20px;
  text-align: center;
  color: white;
}


@media (max-width: 600px) {
  nav, article {
    width: 100%;
    height: auto;
  }
}
```

MSD Image

```html
</style>
</head>
<body>

<img src="https://msdprojectclear.org/wp-content/uploads/2020/03/MSDPC-Logo-WS.jpg" style="width: 10%; float: left; height: 100px;">
```

Header Title

```html
<header>
  <h2>Asset Status Information Tracker</h2>
</header>
```

Table Formats

```css
<section>
<html>
<head>
<style>
table {
  font-family: Garamond, Serif;
  border-collapse: collapse;
  width: 100%;
  height: 100px;

}

td, th {
  border: outset #e79417;
  text-align: left;
  padding: 10px;
}

tr:nth-child(even) {
  background-color: #dddddd;
}
```

```html
</style>
</head>
<body>

<table>
  <tr>
    <th>S. No</th>
    <th>Feature Name</th>
    <th>Active</th>
    <th>Abandoned</th>
    <th>Removed</th>
    <th>Consolidated</th>
    <th>Design</th>
    <th>Operating</th>
    <th>To Be Removed</th>
    <th>Inactive</th>
  </tr>
  {% for row in table_data %}
  <tr>
    <td>{{ row.S_No }} </td>
    <td>{{ row.Feature_Name }} </td>
    <td>{{ row.Active }} </td>
    <td>{{ row.Abandoned }} </td>
    <td>{{ row.Removed }} </td>
    <td>{{ row.Consolidated }} </td>
    <td>{{ row.Design }} </td>
    <td>{{ row.Operating }} </td>
    <td>{{ row.To_Be_Removed }} </td>
    <td>{{ row.Inactive }} </td>
  </tr>
  {% endfor %}

</table>

</section>
```

Footer

```
</section>

<footer>
  <p>Copyright 2024. MSD: Project Clear | Wastewater + Stormwater.</p>
</footer>

</body>
</html>
```

Queries

```
SELECT
    COUNT(CASE WHEN STATUS = 1 THEN 1 END) AS Active,
    COUNT(CASE WHEN STATUS = 2 THEN 1 END) AS Abandoned,
    COUNT(CASE WHEN STATUS = 3 THEN 1 END) AS Removed,
    COUNT(CASE WHEN STATUS = 4 THEN 1 END) AS Consolidated,
    COUNT(CASE WHEN STATUS = 5 THEN 1 END) AS Design,
    COUNT(CASE WHEN STATUS = 6 THEN 1 END) AS Operating,
    COUNT(CASE WHEN STATUS = 7 THEN 1 END) AS To_Be_Removed,
    COUNT(CASE WHEN STATUS = 8 THEN 1 END) AS Inactive
FROM <TABLE>
```

Example code

```
SELECT

    COUNT (CASE WHEN STATUS = 1 THEN 1 END) AS Active,

    COUNT (CASE WHEN STATUS = 2 THEN 1 END) AS Abandoned,

    COUNT (CASE WHEN STATUS = 3 THEN 1 END) AS Removed,

    COUNT (CASE WHEN STATUS = 4 THEN 1 END) AS Consolidated,

    COUNT (CASE WHEN STATUS = 5 THEN 1 END) AS Design,

    COUNT (CASE WHEN STATUS = 6 THEN 1 END) AS Operating,

    COUNT (CASE WHEN STATUS = 7 THEN 1 END) AS To_Be_Removed,

    COUNT (CASE WHEN STATUS = 8 THEN 1 END) AS Inactive

FROM MSD.ssManhole
```

Completion

Asset Status Information Tracker

S. No	Feature Name	Feature Type	Active	Abandoned	Removed	Consolidated	Design	Operating	To Be Removed	Inactive
1	ssManhole	SS	169839	2219	7144	1	8	0	100	29
2	ssInlet	SS	32794	194	5697	0	5	0	63	3
3	ssIntakeOutfall	SS	380	10	922	0	0	0	0	0
4	ssCleanout	SS	2873	50	536	0	0	1	4	0
5	ssNetworkJunction	SS	74	0	1	0	0	0	0	0
6	ssNetworkStructure	SS	753	8	59	0	1	0	0	0
7	ssValve	SS	1244	15	167	0	0	0	0	0
8	SSP	SS	2600	15	155	0	53	0	0	0
9	ssPump	SS	374	11	66	0	0	0	0	0
10	ssPumpStation	SS	288	16	23	0	0	0	0	0
11	ssGhostPipe	SS	109	1	4	0	0	0	0	0
12	ssGravityMain	SS	211315	3892	13514	2489	10	0	173	40
13	ssPressurizedMain	SS	380	73	88	56	2	429	0	0
14	ssCasing	SS	518	2	9	0	0	0	1	0
1	swManhole	SW	44782	65	1245	0	0	0	0	0

10	ssPumpStation	SS	288	16	23	0	0	0	0	0
11	ssGhostPipe	SS	109	1	4	0	0	0	0	0
12	ssGravityMain	SS	211315	3892	13514	2489	10	0	173	40
13	ssPressurizedMain	SS	380	73	88	56	2	429	0	0
14	ssCasing	SS	518	2	9	0	0	0	1	0
1	swManhole	SW	44782	65	1245	0	0	0	0	0
2	swInlet	SW	156433	160	4929	0	1	0	0	3
3	swIntakeOutfall	SW	28215	62	2620	0	0	0	1	0
4	swCleanout	SW	1539	1	22	0	0	0	0	0
5	swNetworkJunction	SW	4070	0	60	0	0	0	0	0
6	swNetworkStructure	SW	415	2	10	0	0	0	0	0
7	swValve	SW	117	0	10	0	0	0	0	0
8	swPump	SW	19	0	1	0	0	0	0	0
9	swPumpStation	SW	4	1	0	0	0	0	0	0
10	swGhostPipe	SW	4798	1	111	0	0	0	0	0
11	swGravityMain	SW	216169	731	6782	202	2	0	0	0
12	swPressurizedMain	SW	13	3	2	0	0	7	0	0
13	swCasing	SW	69	0	3	0	0	0	0	0
14	swManagementPoint	SW	9706	1	289	0	1	0	0	0
15	swManagementLine	SW	103	0	51	0	0	0	0	0

Resources

GitHub

A developer platform that allows developers to create, store, manage and share their code. It uses Git software, which provides distributed version control of access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project.

Links Used

https://github.com/stlmsd/InternProject

https://github.com/github

W3Schools

A website that offers free and paid resources for learning web development and coding online. The website includes the following:

1. Tutorials
2. Exercises and quizzes
3. Code editor
4. Web templates
5. Courses
6. Website builder

Links Used

https://www.w3schools.com/

GeeksforGeeks

A leading platform that provides computer science resources and coding challenges for programmers and technology enthusiasts, along with interview and exam preparations for upcoming aspirants.

Links Used

https://www.geeksforgeeks.org/