Cesium Certification Project

Project Name : KML Visualization and Metadata Management

Developed By : Developers of Digile Sdn Bhd

Table of Content

Overview

This project is designed to meet the requirements for becoming a Cesium-certified developer, as outlined in the Cesium Certification Requirements. The application demonstrates proficiency in CesiumJS by integrating core functionalities, including KML data visualization, interactive asset management, and advanced user interactions.

Project Goal

To develop a web-based application that visualizes KML file data using Cesium.js, allowing users to interact with geospatial features, dynamically style map elements based on metadata, and manage feature data through a tabular interface linked to a database.

Key Features:

Integration with Cesium.js

1. Utilize the Cesium.js library for rendering 3D geospatial data.
2. Enable dynamic visualization of KML data on an interactive Cesium map.

KML File Upload

1. Implement a functionality to upload KML files containing geospatial data.

Fly-to Camera Functionality

1. Upon uploading a KML file, automatically adjust the map view to focus on the relevant geospatial features using the "Fly-to" camera function.

Dynamic Metadata Styling

1. Link KML lot data attribute with their corresponding metadata values.
2. Use metadata to assign different colors to map features based on their type or category, enabling intuitive visualization.

Feature Linking and Metadata Update

1. Update the list of data features to dynamically associate them with existing lot IDs extracted from the KML file.

Tabular Data Display

1. Parse KML data and present feature information in a tabular format for better visualization and interaction.

Database Integration

1. Store KML feature metadata in a database.
2. Enable editing and updating of metadata through the application interface, ensuring data consistency and usability.

Technical Specifications

Frontend

- **Cesium.js**: For 3D geospatial visualization.
- **HTML/CSS/JavaScript/jQuery**: For building the user interface.

Backend

- **Database**: MSSQL for storing metadata and feature associations.
- **PHP**: handle **POST** and **GET** requests for interacting with the database.

API Integration

- Fly-to Camera: Leverage Cesium's `viewer.camera.flyTo` function.
- Metadata Styling: Apply Cesium entities' styling options based on data attributes.
- Info box for data attributes.

Diagram

Design

The diagram below shows the process from uploading a KML file containing geospatial data via frontend interface, process and store the data in database, displays the geospatial features on Cesium.js.

Then, linked the data to lot attribute in metadata and styled (color-coded) based on their type. Users can also update the specific attributes directly within a tabular interface to update or correct the metadata.

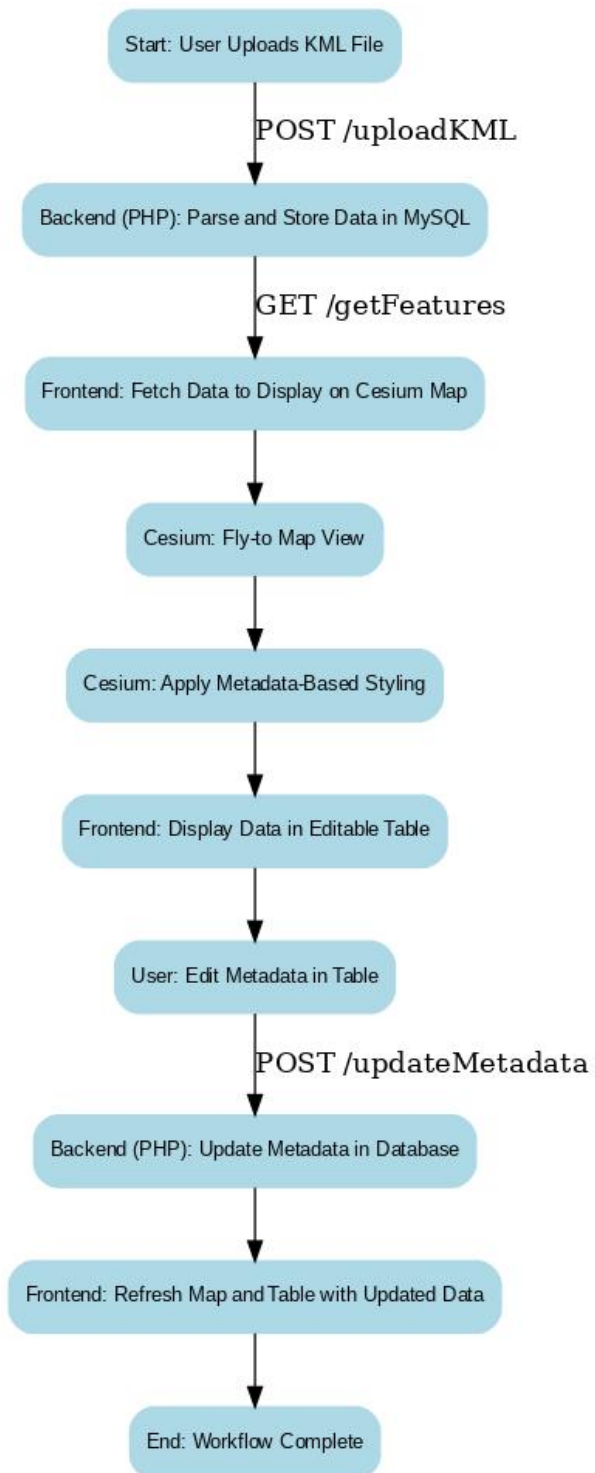Image 1 : System Visualization.

Flow Diagram

Image 2 : Flowchart Diagram.

User Interface

Upload Features

User can upload KML file by using this interface. Currently, this form only can accept KML/KMZ file.
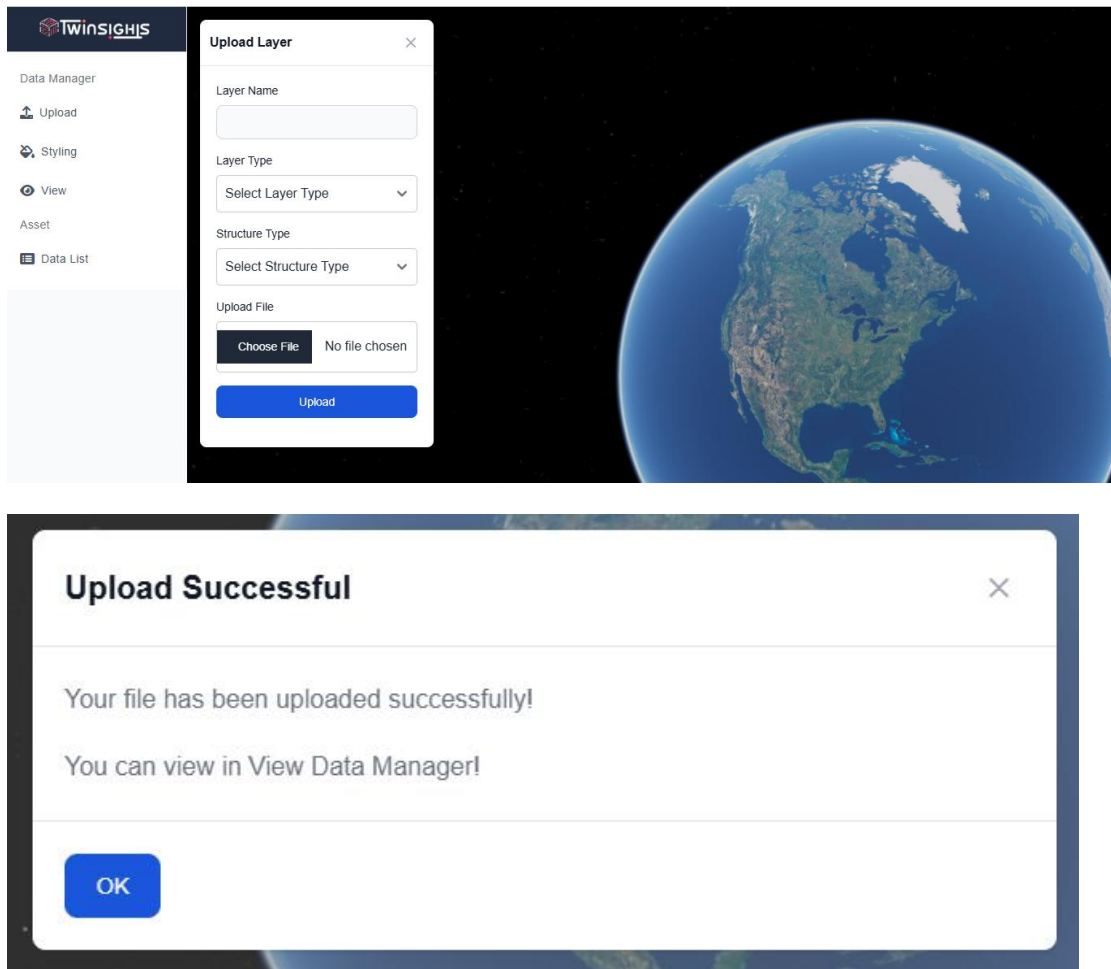

Image 3 : Main User Interface.

Metadata Styling

The interface below displays the data attributes from the KML file, where the lots are color-coded based on the attributes submitted in the land data.
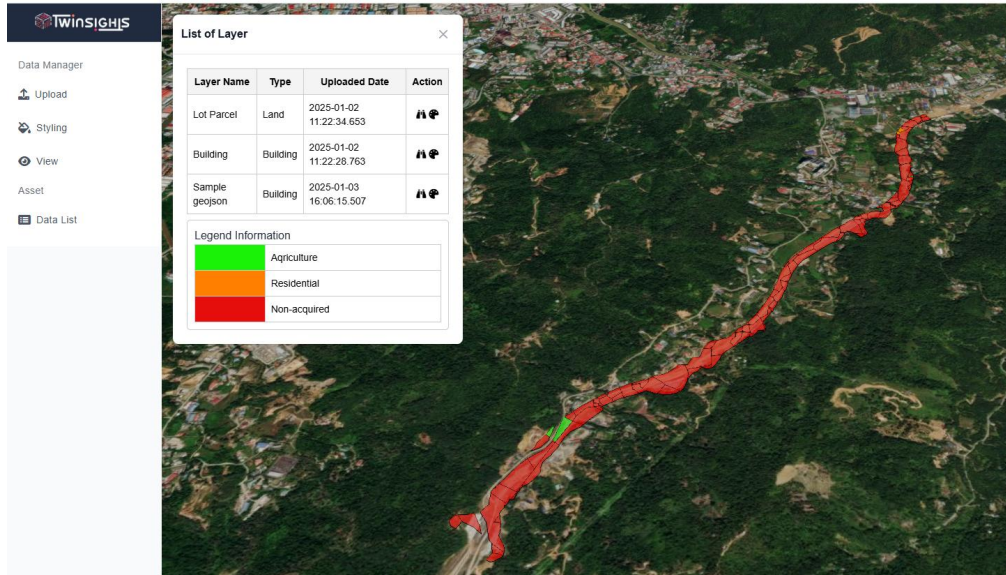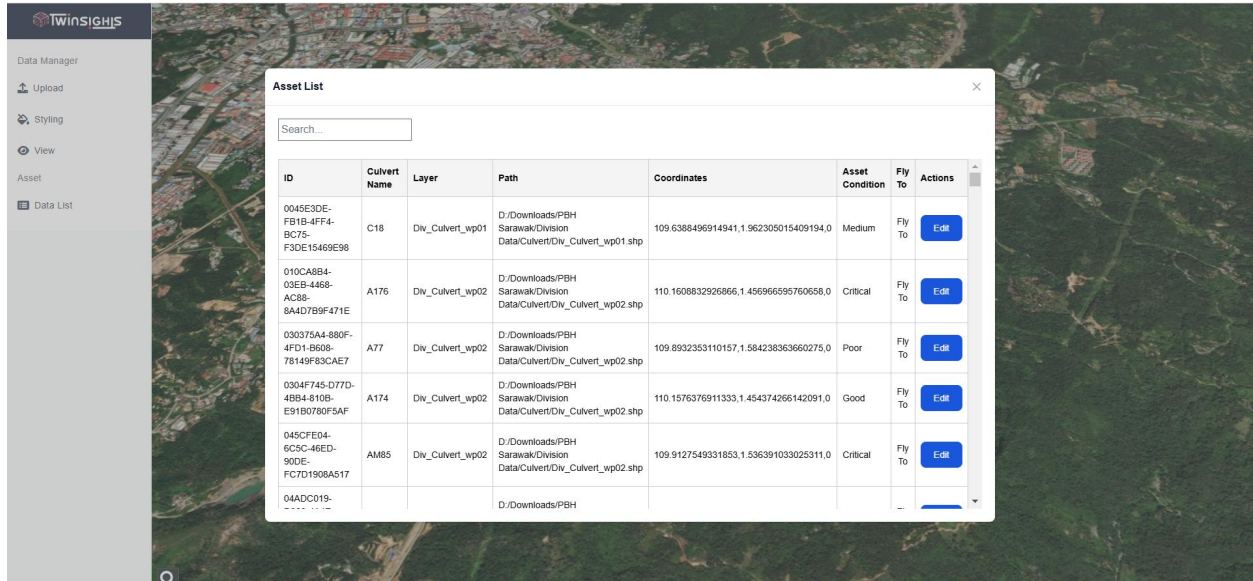

Image 4 : Metadata Styling.

Tabular View of KML Data & Update Metadata

The interface below shows the list of data attributes fetch from KML file that is stored in database table. The asset condition can be edited by using form in Image 6.

Image 5 : Tabular Data Asset List.

Image 6 : Edit Metadata Form.

Challenges

Parsing and Managing KML Files

Parsing KML files can be tricky, especially for large files or those with complex structures (e.g., nested elements, extended data).

Handling Large Datasets

Large KML files with numerous features might cause performance bottlenecks, such as slow rendering on the Cesium map or delayed database operations.

Dynamic Metadata Styling

Applying styles dynamically based on metadata (e.g., assigning colors) might result in visual clutter or incorrect mapping if metadata is inconsistent or missing.

User Experience

Providing an intuitive interface for uploading files, editing metadata, and viewing data can be challenging for non-technical users.

Next Step/Future Work

Advanced Data Visualization

Add **heatmaps**, **3D extrusions**, or other visualization layers (e.g., terrain overlays) in Cesium.js to provide more insights from the KML data which can improves the user's ability to analyze spatial data effectively.

Role-Based Access Control

Implement user authentication with different roles (e.g., Admin, Editor, Viewer) to control access to features like editing metadata or uploading files. This will enhances security by restricting sensitive actions to authorized users.

Improve Performance for Large Datasets

Use **tiling techniques** for Cesium to improve rendering performance and implement **lazy loading** for table data to reduce load times. Ensures smooth performance, even with large datasets.

Extended Asset Management Features

Allow bulk editing of asset conditions via table interface. Allow user to redirect the app to the Google Map to allow them to navigate their route to the asset selected.

Final Results

The successful result of the system include the following key points :

- **Interactive 3D Map Visualization**: KML data will be dynamically rendered on an interactive Cesium map, allowing users to explore geospatial features in 3D.

- **Fly-to Camera Functionality**: The map view will automatically adjust to focus on the relevant geospatial features upon KML file upload, providing a smooth, intuitive user experience.

- **Dynamic Metadata Styling**: Features on the map will be dynamically styled (e.g., different colors) based on metadata attributes, enabling easy differentiation of feature types.

- **Tabular Data Display**: KML feature metadata will be presented in an interactive table, allowing users to sort, filter, and interact with the data.

- **Database Integration**: Metadata and feature associations will be stored in an MSSQL database, ensuring data persistence and enabling real-time updates.

- **Cesium Certification Readiness**: The application demonstrates proficiency in Cesium.js and meets key Cesium certification requirements.

This combination of features will result in a seamless, interactive system for visualizing and managing geospatial data with data integration and dynamic user interactions.

Areas of Contribution

Cesium JS Integration and Map Rendering & Tabular Data Display and Database Integration

Items :

- Focus on integrating the Cesium.js library into the application.
- Implement the basic setup for rendering 3D geospatial data on the Cesium map.
- Ensure the interactive map is working as expected, including navigation and basic layer control.
- Implement a tabular display to present feature information extracted from the KML data.
- Set up database integration to store metadata and feature information for later retrieval and updates.
- Develop functionality for users to update metadata through the interface, ensuring data consistency.

Developers Involved :

- Zainal Safwan bin Zainal Abidin (zainal.safwan@digile.com)

- Rayleen Joy Eulogio (rayleenjoy.eulogio@digile.com)

Fly-to Camera Functionalities, Dynamic Metadata Styling & Visualization and KML File Upload & Parsing

Items :

- Implement the "Fly-to" camera functionality that automatically adjusts the map view based on the uploaded KML file's geospatial features.
- Ensure smooth camera transitions and accurate zoom levels when focusing on relevant features.
- Develop a system for linking KML data attributes with their corresponding metadata values.
- Implement dynamic styling based on metadata (e.g., different colors for map features based on category/type).
- Make the metadata-driven styling flexible and customizable.
- Implement functionality for users to upload KML files.
- Parse the uploaded KML files and extract relevant geospatial data (coordinates, features, metadata).
- Ensure the parsing process is efficient and can handle various KML structures.
- Design the User Interface.

Developers Involved :

- Allyana Anasuhah binti Abd. Ghani (allyana.anasuhah@digile.com)
- Nurul Nafisah binti Jamal (nurul.nafisah@digile.com)
- Myril Riomalos (myril.riomalos@digile.com)