INL/RPT-22-67714

Light Water Reactor Sustainability Program

Industry Level Feasibility of LiDAR Data into Fire Modeling Using Fire Risk Investigation in 3D (FRI3D)

June 2022

U.S. Department of Energy

Office of Nuclear Energy

DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

Industry Level Feasibility of LiDAR Data into Fire Modeling Using Fire Risk Investigation in 3D (FRI3D)

Michael Ziccarelli
Steven Prescott
Eric Allen

June 2022

Idaho National Laboratory
Idaho Falls, Idaho 83415
http://www.inl.gov/lwrs

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

ABSTRACT

Many evaluation, assessment, and modeling tasks at nuclear power plants require spatial information. This often requires physical visits to locations within the facility because the 2D or 3D schematics and current models do not contain enough detail or do not capture as-built and real-world conditions. These visits require extensive manual labor for not only the requesting party but support groups such as security. Light detection and ranging (LiDAR) mapping is trying to solve that problem by providing very detailed 3D models for low costs. However, the use of these models can be very limited because either component reference information is missing and too costly to add, or there is no way to extract specific spatial data needed for other tools.

This report presents Idaho National Lab's work with Environmental Intellect (Ei) covering two main efforts. First, to reduce the effort of "tagging" data in large 3D models. By using both existing plant database information and artificial intelligence (AI) to find and read equipment labels. This research explores the ability to provide a simple way for the user to tag items and verify plant data, capturing both the speed of AI and human verification.

The second part of the work is the development of an interface for importing pieces needed for modeling and simulation. Analysis work such as for fire, flood, or physical security all require spatial or 3D models in various levels of detail. This interface will allow for the retrieval of item location or boundaries, enabling the auto-generation of models for varying tools. The application program interface (API) of the Fire Risk Investigation in 3D (FRI3D) was used to test feasibility of exporting the LiDAR tagged spatial information. Outcomes from this work provide preliminary data to determine if the tools and methods could provide substantial industry benefit if fully matured.

ACKNOWLEDGEMENTS

The authors would like to thank our industry collaborators for their contributions in this work.

CONTENTS

FIGURES

ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interface
CAD	Computer-Aided Design
DLL	Dynamically linked library
Ei	Environmental Intellect
FRI3D	Fire Risk Investigation in 3D
INL	Idaho National Laboratory
JSON	JavaScript Object Notation
LiDAR	Light Detection and Ranging
LWRS	Light Water Reactor Sustainability
NPP	Nuclear power plant
NRC	Nuclear Regulatory Commission
OCR	Optical Character Recognition
P&ID	Piping and Instrument Diagram
PDMS	Plant Data Model System
PRA	Probabilistic Risk Assessment
RISA	Risk Informed Safety Analysis
UI	User Interface

1. INTRODUCTION - LIDAR FOR NUCLEAR POWER PLANTS

Light detection and ranging (LiDAR) scans and accompanying software offer many potential uses for the nuclear industry; several facilities have used them for virtual walkdowns and equipment placement planning. Since the raw scan's (i.e., 3D spatial representations with picture overlays) uses are limited, recently, more companies are offering tools that allow adding information to the models, so users can click on items to see names, instructional info, and links to diagrams and videos or to help maintain regulation documents.

These models enable a wide range of applications. However, adding this data to a model can be very time consuming and require someone well-versed in facility designs and operations. This is especially true with nuclear facilities due to the extremely large number of components and various safety systems. This obstacle can be overcome by using plant data already compiled for other tasks such as fire modeling.

To overcome this issue and explore additional ways to use LiDAR information and in order to reduce operation costs and maintain plant safety, Idaho National Laboratory (INL) is working with the 3D LiDAR scanning and model-hosting company Environmental Intellect (Ei). Using an existing plant's plant data model system (PDMS) or FRANX-fire model, we show how to simplify the process of linking the data to spatial locations or items in the 3D model.

Work in the previous year [1] focused on adding existing plant data to the LiDAR model and allowing the user to find this data and link it to locations they manually add. While this drastically decreases the amount of time needed for tagging equipment in the LiDAR model, there is still many manual steps that could be done automatically. This work identified several areas for using artificial intelligence (AI) to assist in these manual steps with the user verify or correct as needed. This would allow someone with just basic knowledge of the facility and equipment to do almost all the work.

There are two main tasks outlined in this document. First, the evaluation of using AI to auto-identify equipment by reading text-based tags and matching them to plant component data with the user verifying the auto-tagged equipment. Second, this report evaluates the processes for retrieving tagged-item information from the LiDAR model to auto-generate a 3D model in the Fire Risk Investigation in 3D (FRI3D) software, the same process could be used by other applications needing spatial information. This research is being done under the Risk Informed Safety Analysis (RISA) pathway of the Light Water Reactor Sustainability (LWRS) program [2]. While the research performed for this project used plant data and LiDAR scans from an existing facility, for proprietary and security purposes, only test images and information are included in this report.

1.1 Scan Data

LiDAR scanning captures high-resolution, spatially accurate representations of real-world conditions. Embedded photogrammetry in the LiDAR scan data captures multiple high-resolution 360-degree photos of all equipment. Combining these two data-capture techniques into a single, seamless data-capture process provides the best-possible digital representation of real-world equipment and conditions. Various hardware devices were developed to capture this data. Early versions used a tripod that was set up in different locations to capture objects from a few perspectives. Newer devices, such as the one shown in Figure 1, allow the user to move and scan at the same time or in intervals. With more scanning angles, there are fewer shadow spots, and better images are produced.

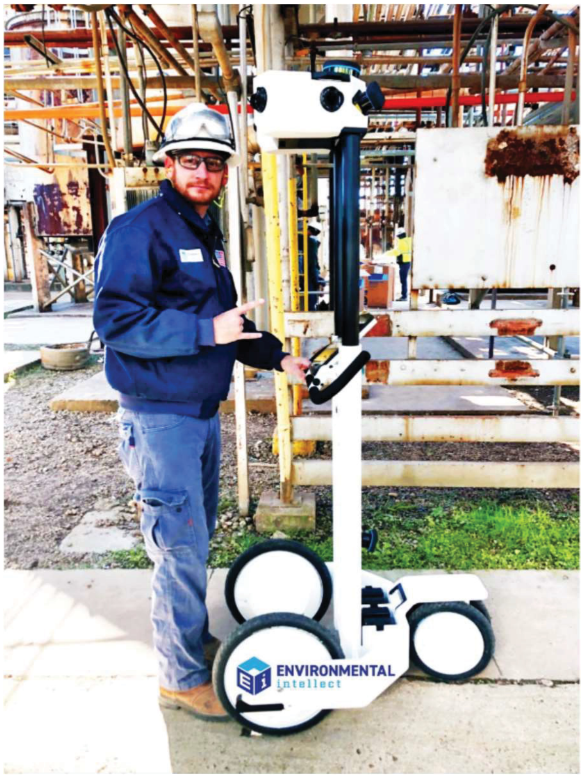


Figure 1. Ei mobile mapping technician with mobile mapping device.

1.2 Visualization

Visualization tools for LiDAR consist of a 3D environment generated by millions to billions of points, depending on the size and resolution of the model. These points construct the shell of objects surrounding areas that were scanned. More recent technologies attach a picture fragment or photogrammetry data to each point that constructs a realistic view of the model.

Typical LiDAR scanning and visualization techniques require the capture and rendering of terabytes of data on a single machine, requiring end users to have powerful computers with extremely high-capacity hard drives. The approach taken by Ei uses a modern and secure web application to render the LiDAR data using a powerful server, allowing the user to view it on a basic computer or tablet with minimal network and processing requirements. The Ei approach also seamlessly integrates photogrammetry data so a user can easily switch between LiDAR and photo views of each piece of equipment as seen in Figure 2.

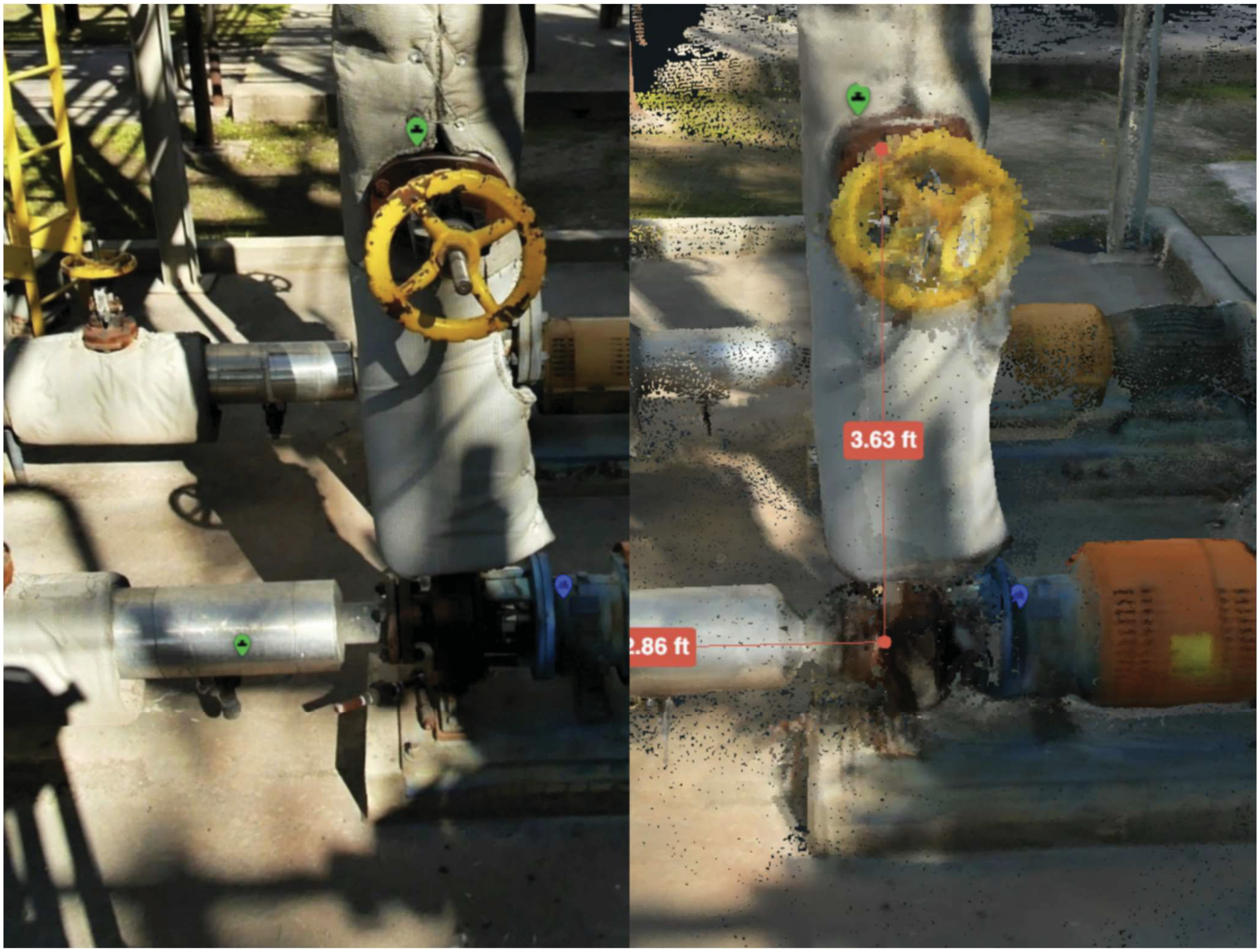


Figure 2. Side-by-side comparison of photogrammetry and LiDAR data collected with mobile mapping and measurement display in the Ei viewer.

1.3 Pins and Linked Information

In addition to the powerful visualization, the Ei viewer enables end users to add their own data points called "pins" in the 3D environment. They are also referred to as "tags" or "points of interest." A main use of these pins is to identify a specific component in the 3D space as shown in Figure 3. Each can also have a bounding box that identifies the space of the component. The user interface allows the pins to be added in a simple and flexible manner from the user entering the data or selecting data that is preloaded into the environment.

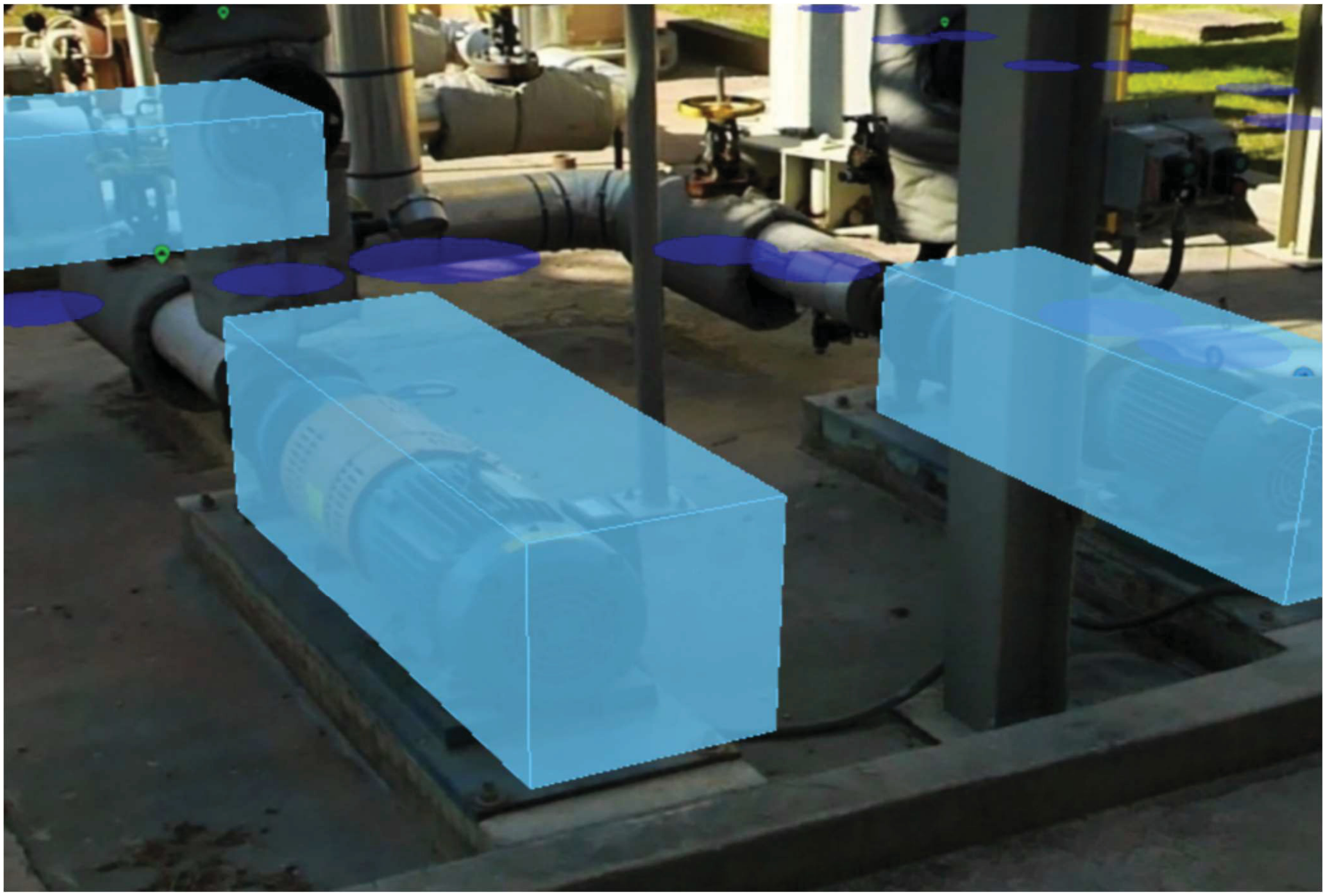


Figure 3. Pins with bounding boxes as shown on the Ei platform..

1.4 Other Features

As stated, different point cloud viewing platforms have varying features. The Ei platform was chosen for this research because it already had many features needed for importing plant data, tagging filters, and initial development of an application programing interface (API) external data retrieval. The Ei platform has many advanced features for its users [3] that may also be useful for the nuclear industry. These features include:

- Computer-aided design (CAD) association with data points—pins can be easily integrated with the site's existing CAD assets (both 2D, 3D, and P&IDs) enabling seamless navigation between real-world views, existing CAD, or engineering drawings. This provides additional context as well as additional engineering information needed to drive various use cases.
- Communication—users can communicate and take notes very easily within the application, enabling collaboration between various stakeholders and the capture of valuable information for future users. The different teams allow for notifications to the correct people.

1.5 Industry Demonstration Model

Through collaborative work with an existing nuclear power plant (NPP), a LiDAR scan was performed on two switch gear rooms and a pump room, using the equipment described in Section 1.1. This work was performed in less than 1 day for a cost of about $10,000, and it is estimated that a scan of one reactor area can be done in 3–5 days depending on accessibility and areas needed. The cost can vary drastically, with an estimate of $50K–200K, depending on the needs of the user and processing of the model.

The raw data collected by the various LiDAR sensors was processed on vendor-provided software processing and registration tools. These tools output industry-standard file formats (such as .pts and .e57). These LiDAR point cloud outputs are then optimized and undesired fragments removed (people, moving items, temporary items, etc.), then transferred to the Ei Everywhere cloud-hosted environment in a web-optimized format.

The PDMS and FRANX models were also provided by this facility for use in linking the plant data to items in the LiDAR scan, described in subsequent sections.

2. LINKING NUCLEAR POWER PLANT DATA

The API used for the initial assisted tagging in the previous year [1] was modified to upload the plant data used by the AI for item matching. This work focused on a tool with a generic or customizable workflow so that it could be used with multiple platforms and a variety of NPP data.

Most nuclear power facilities have databases used for risk analysis such as fire probabilistic risk assessment (PRA). Examples include the FRANX database used for external hazard PRA models or a PDMS. These databases contain components, information about components (e.g., component model, location, etc.), and logical/physical links between items. They also can group items into physical zones. This section explains the process used to pull this data into the LiDAR visualization environment for use in tagging components. The design is intended to allow for customizing a given plant's data and interfacing with any LiDAR visualization software that has or is willing to implement the required features.

2.1 Existing Plant Data

The two database types being used for this research are the PDMS and FRANX databases. Both the PDMS and FRANX databases are Microsoft Access database files, consisting of all the equipment, cables, and raceways with descriptions and other information for these items and other safety-related items inside the plant. The location of each item is associated with a particular physical location boundary or zone.

2.2 API Design

There are two APIs under development to transfer information from a PDMS or FRANX database to Ei. The first is an API that connects directly to the database. When a "GET" request is sent to this API, it will query the database for all equipment, cables, and/or raceways, placing all this information into a list. Since this list can become quite large, it is broken up into smaller lists. These smaller lists are sent to Ei's system via a series of consecutive "POST" requests. After Ei has received all the data, the API sends a response code for the initial GET request.

The second API was created in Ei's system. This API allows objects, which can be displayed on Ei's web UI, to be created through a POST request. This is the POST request mentioned above, to which the lists of data will be sent, as shown in Figure 4.

Since the plant data API can be used at many sites that may have variations in their database systems, the API has been constructed to allow for maximum compatibility with different types of databases. This is done by utilizing the Dapper NuGet library to connect to the database and making the database connection string and relevant table and column names configurable.

The database configuration is stored in a JavaScript Object Notation (JSON) file, which allows for easy updates. An example JSON file is shown in Figure 5 (this has been shortened to only show relevant fields). A path to the database file is needed along with the names of the tables that contain the Equipment ID, Cable ID, Raceway ID, the respective descriptions, and Fire Zone ID. The column names for those items will also be needed. If, however, a FRANX database is being used, it will not be necessary to update the configuration file with table and column names. This is because FRANX databases have a universal format for storing this information, which has been built into this API.

The JSON file shown in Figure 5 outlines the customization fields for retrieving the plant data. For example, if a FRANX database is used, the “DatabaseIsFRANX” value in the “Settings” group should be set to “true.” It also has a place for the Ei API token, “EiApiAccessToken,” which is required to send any data to Ei. All the fields and a step-by-step process will be specified in the documentation, but it is designed for someone with knowledge of the system to be able to set it up in less than a day’s work.

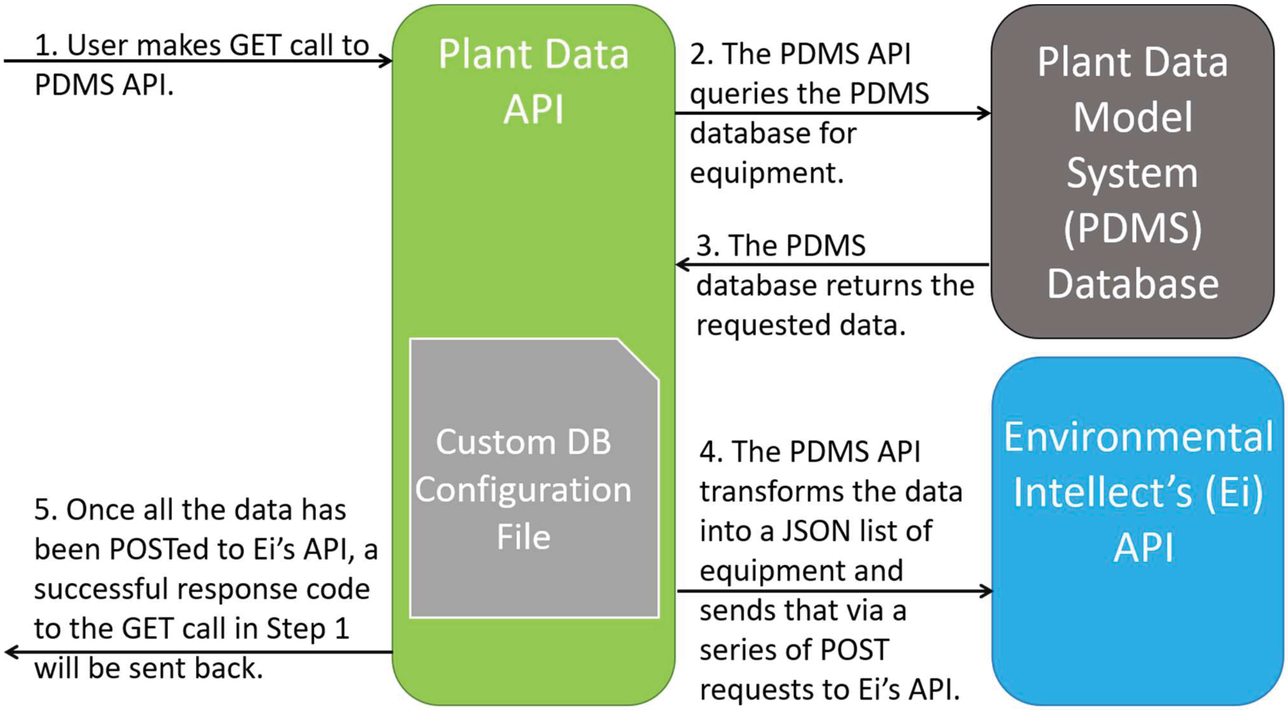
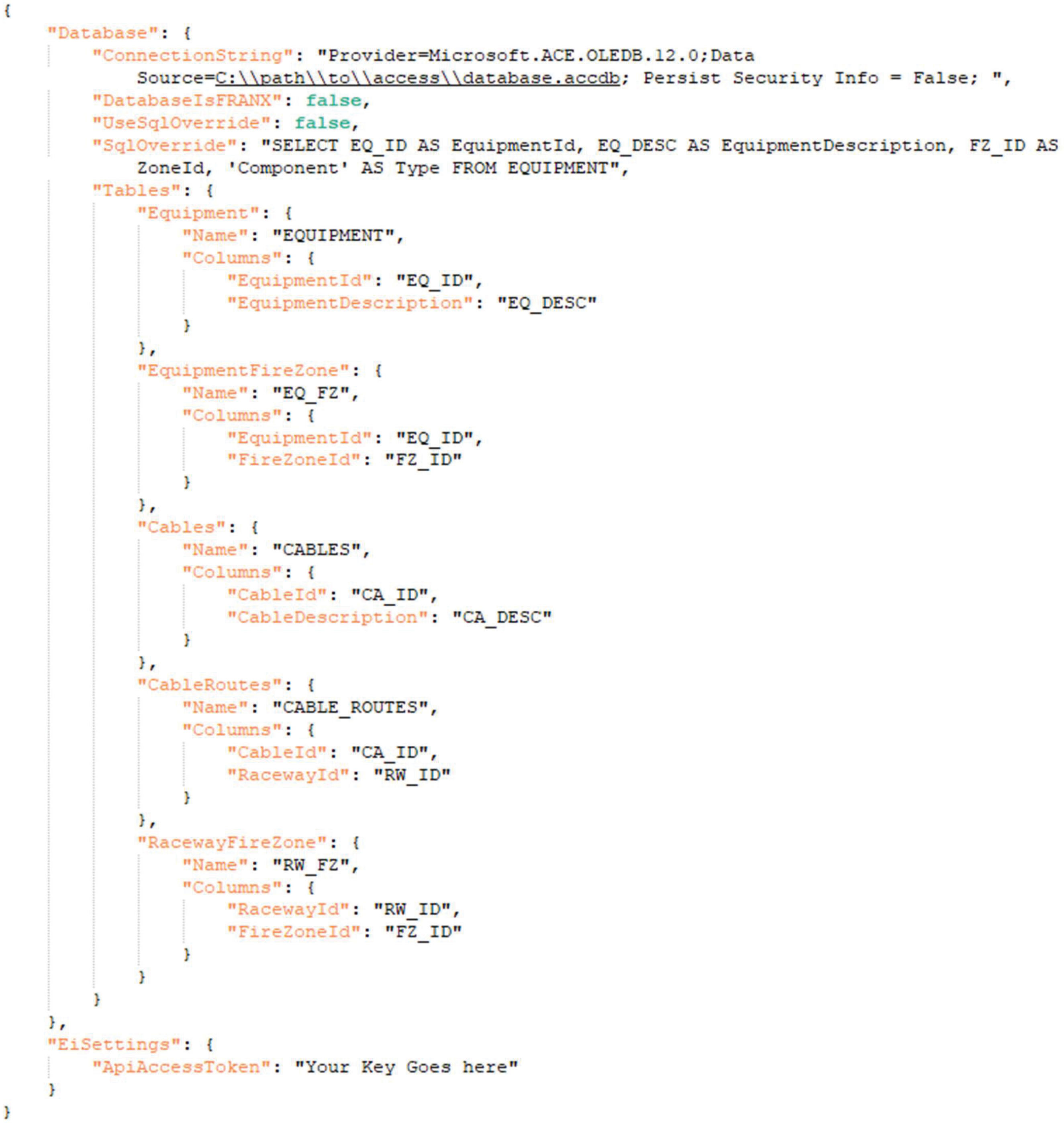

1. User makes GET call to PDMS API.
Plant Data API
2. The PDMS API queries the PDMS database for equipment.
Plant Data Model System (PDMS) Database
3. The PDMS database returns the requested data.
Custom DB Configuration File
4. The PDMS API transforms the data into a JSON list of equipment and sends that via a series of POST requests to Ei’s API.
Environmental Intellect’s (Ei) API
5. Once all the data has been POSTed to Ei’s API, a successful response code to the GET call in Step 1 will be sent back.

Figure 4. Data flow for plant data to Ei’s LiDAR interface.



```
{
    "Database": {
        "ConnectionString": "Provider=Microsoft.ACE.OLEDB.12.0;Data
            Source=C:\\path\\to\\access\\database.accdb; Persist Security Info = False; ",
        "DatabaseIsFRANX": false,
        "UseSqlOverride": false,
        "SqlOverride": "SELECT EQ_ID AS EquipmentId, EQ_DESC AS EquipmentDescription, FZ_ID AS
            ZoneId, 'Component' AS Type FROM EQUIPMENT",
        "Tables": {
            "Equipment": {
                "Name": "EQUIPMENT",
                "Columns": {
                    "EquipmentId": "EQ_ID",
                    "EquipmentDescription": "EQ_DESC"
                }
            },
            "EquipmentFireZone": {
                "Name": "EQ_FZ",
                "Columns": {
                    "EquipmentId": "EQ_ID",
                    "FireZoneId": "FZ_ID"
                }
            },
            "Cables": {
                "Name": "CABLES",
                "Columns": {
                    "CableId": "CA_ID",
                    "CableDescription": "CA_DESC"
                }
            },
            "CableRoutes": {
                "Name": "CABLE_ROUTES",
                "Columns": {
                    "CableId": "CA_ID",
                    "RacewayId": "RW_ID"
                }
            },
            "RacewayFireZone": {
                "Name": "RW_FZ",
                "Columns": {
                    "RacewayId": "RW_ID",
                    "FireZoneId": "FZ_ID"
                }
            }
        }
    },
    "EiSettings": {
        "ApiAccessToken": "Your Key Goes here"
    }
}
```

Figure 5. JSON properties for custom database retrieval.

2.3 Equipment Data Loader Web App

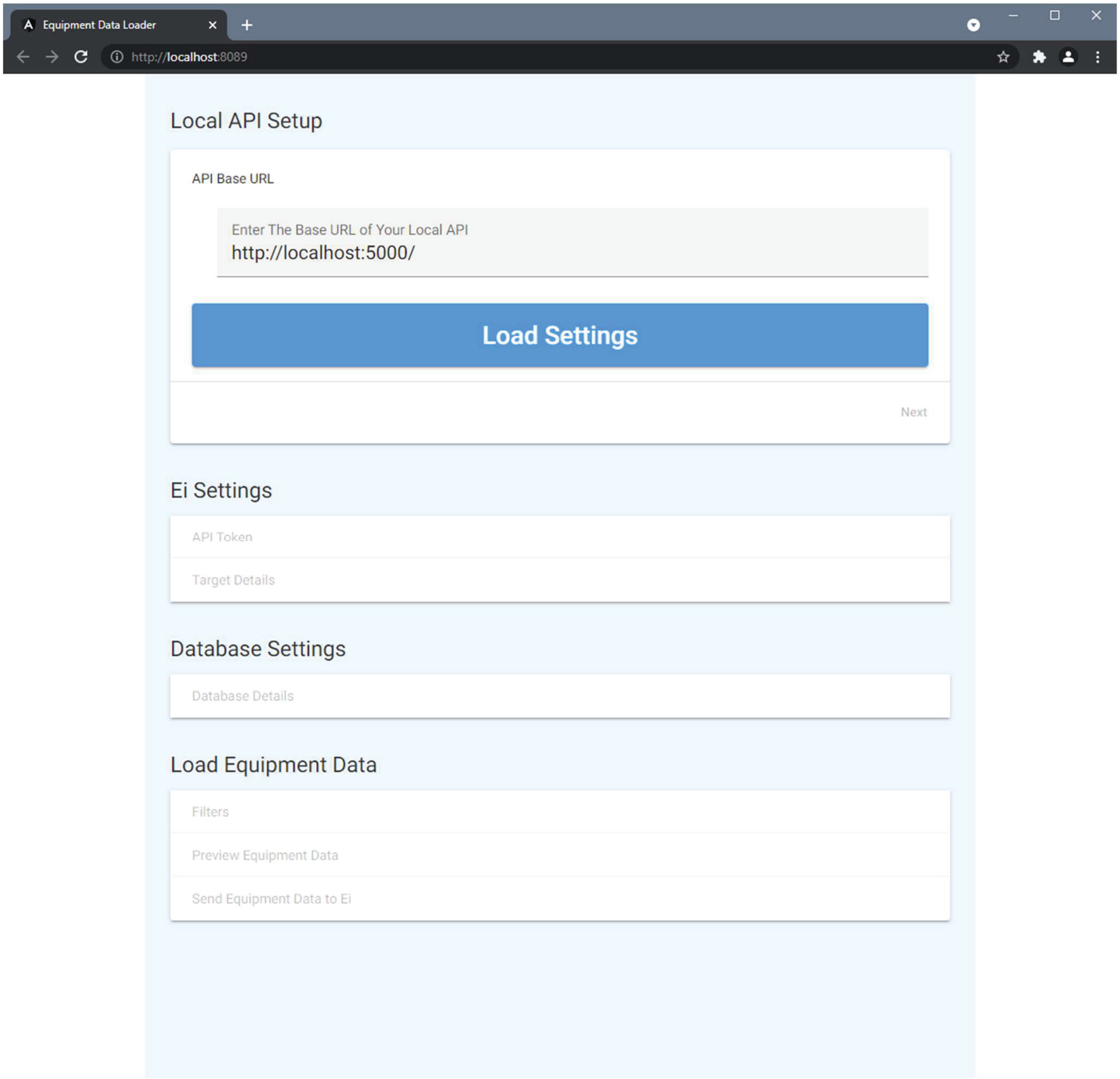
In order to assist with properly modifying the configuration JSON file, a web app has been developed to walk users through the process with an easy-to-use interface. This web app, like the plant data API, will run and be served locally on the user's computer. Figure 6 through Figure 17 show the steps and forms used to uploading the data.

Upon opening the web app, it asks for the base URL of the plant data API. A default value for this is prefilled, as this will be the same in most cases. Once the URL has been provided, the user can click the "Load

Settings” button to load the existing settings defined in the current configuration JSON file and activate the ability to edit them. As a visual cue to the user that steps have been activated, the relevant steps no longer appear as greyed out; Figure 6 and Figure 7 show the web app before and after clicking the “Load Settings” button, respectively.

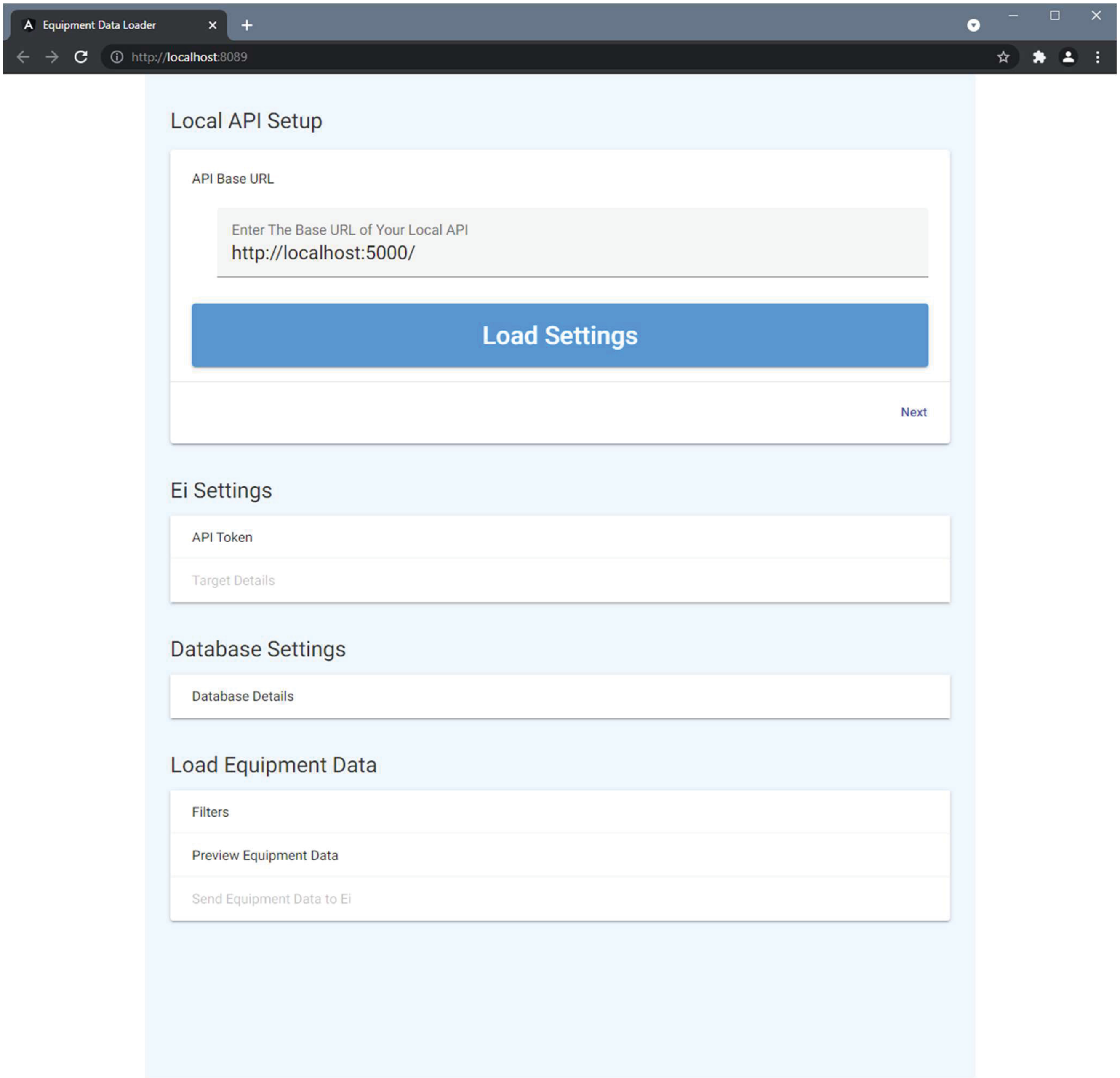
After the current settings have successfully loaded, the user can click the “Next” button to move on to the next step of setting the Ei API access token. Similar to above, after entering the token, the user can click “Validate Token” to test the token works properly and activate the next step. Figure 8 and Figure 9 show screenshots of the web app before and after having the token validated.

When the Ei API access token has been successfully validated, it will make API calls to Ei to populate the next step’s options for choosing the Ei “Map,” “Layer,” “Layer Type,” Fire Zone “Property,” and Equipment Type “Property” as the target destination for the equipment data load. This step is shown in Figure 10.



Enter The Base URL of Your Local API
http://localhost:5000/

Figure 6. Initial view of the equipment data loader web app.



Local API Setup

API Base URL

Enter The Base URL of Your Local API
http://localhost:5000/

Ei Settings

API Token

Target Details

Database Settings

Database Details

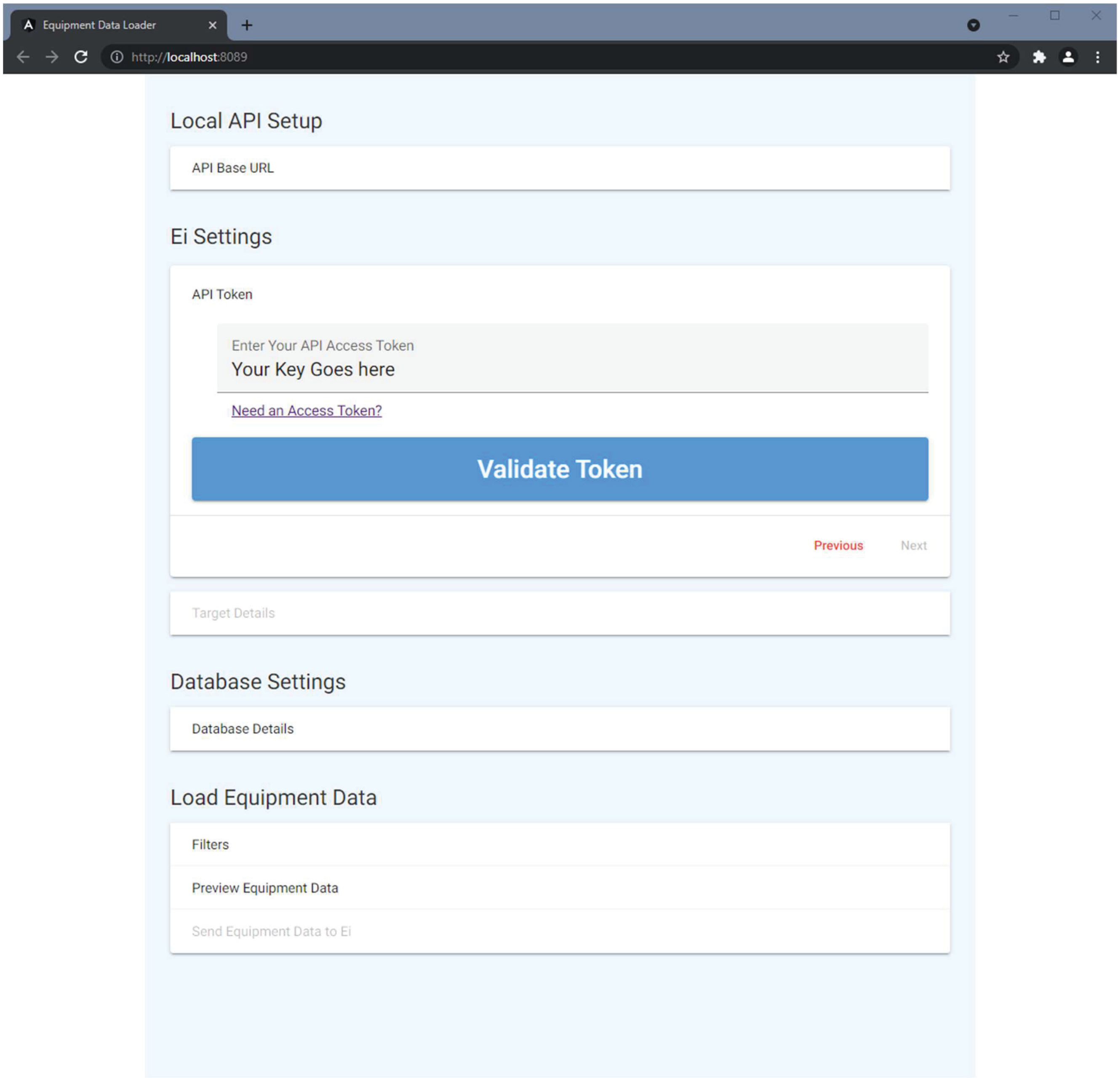
Load Equipment Data

Filters

Preview Equipment Data

Send Equipment Data to Ei

Figure 7. Screenshot of the equipment data loader web app after current settings have been loaded.



Local API Setup

API Base URL

Ei Settings

API Token

Enter Your API Access Token
Your Key Goes here

Need an Access Token?

Target Details

Database Settings

Database Details

Load Equipment Data

Filters

Preview Equipment Data

Send Equipment Data to Ei

Figure 8. Screenshot of the equipment data loader web app before validating the Ei's API Access Token.

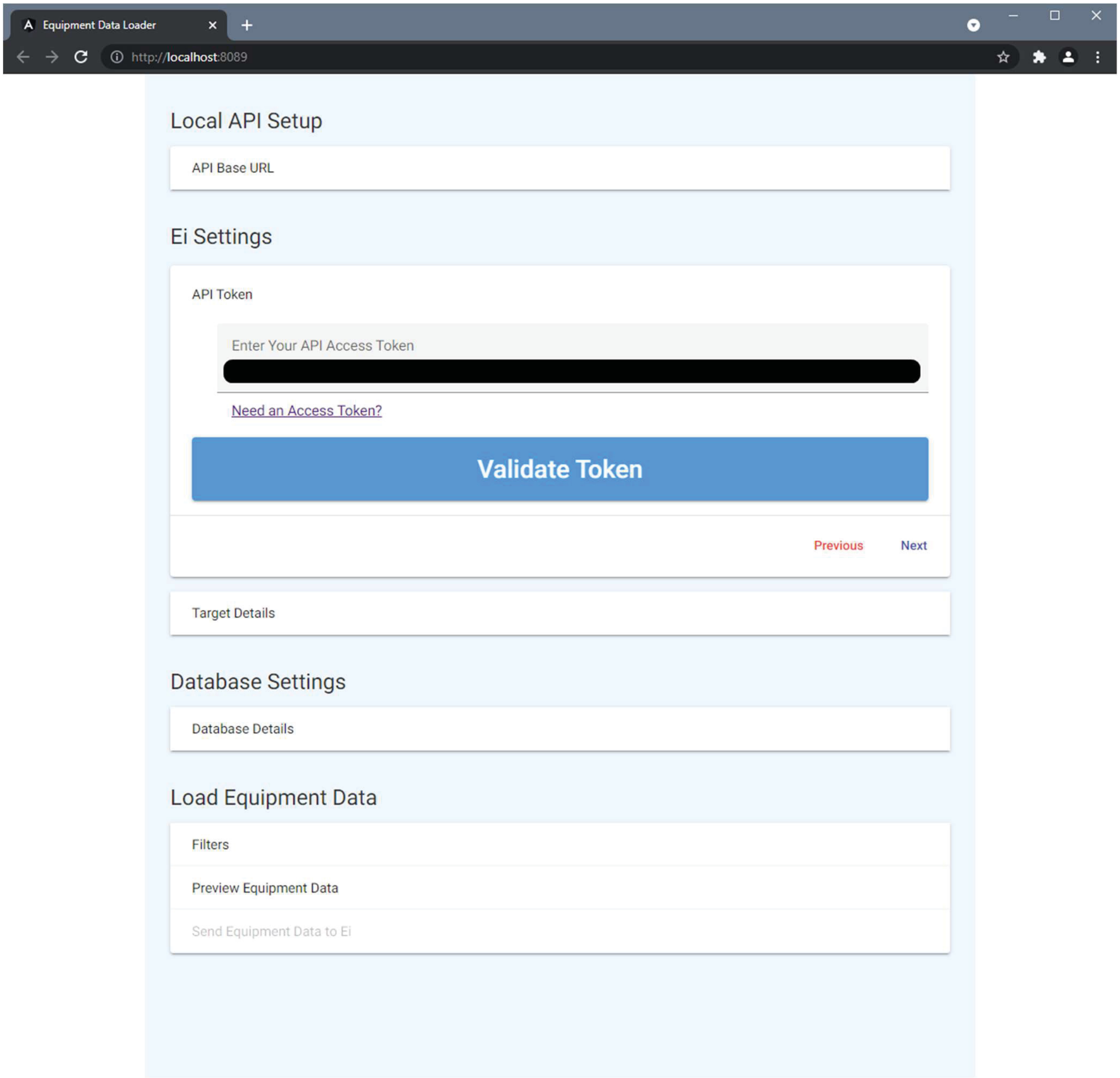
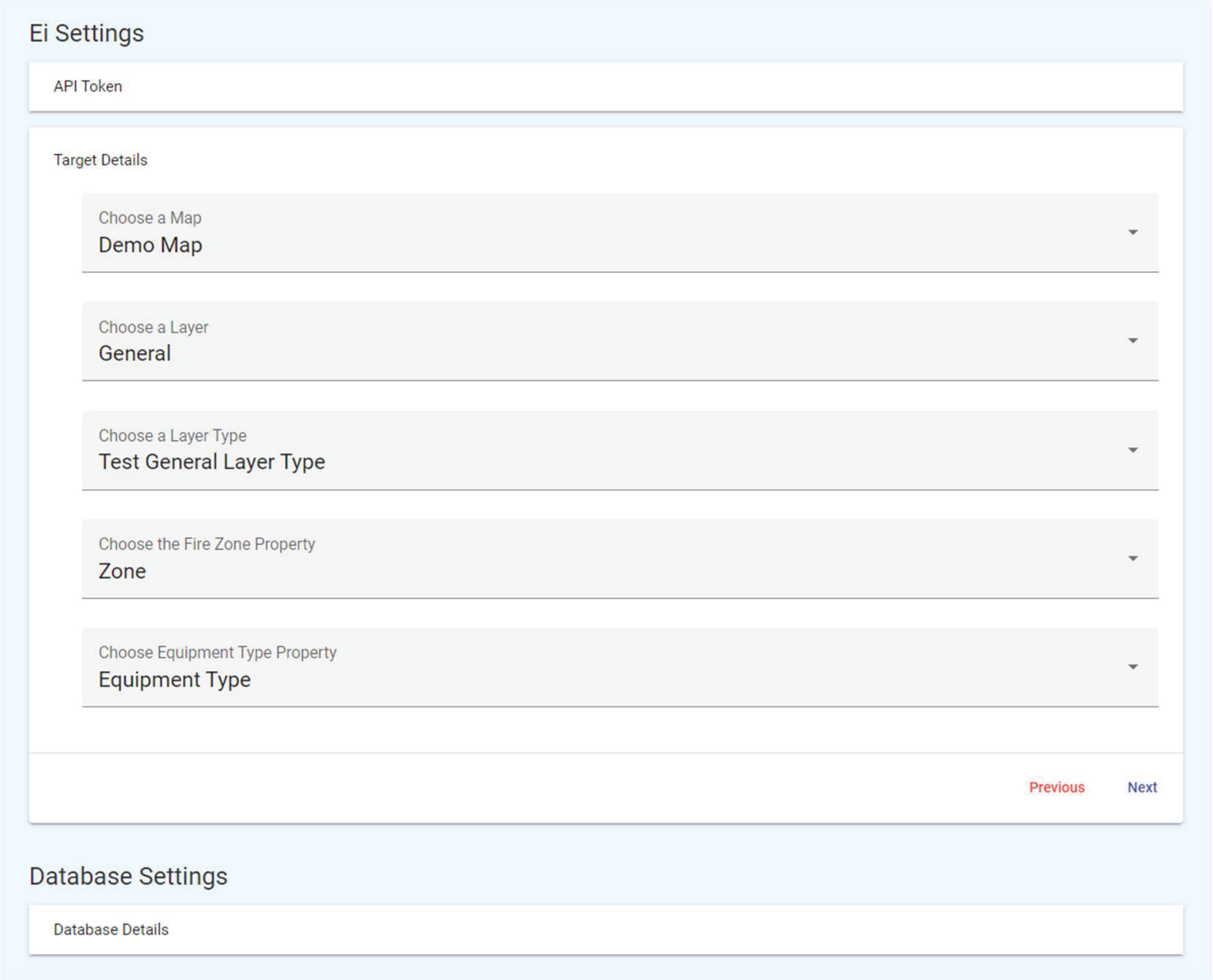


Figure 9. Screenshot of the equipment data loader web app after validating the Ei's API Access Token. For security reasons, the token has been blacked-out.



Ei Settings

API Token

Target Details

Choose a Map
Demo Map

Choose a Layer
General

Choose a Layer Type
Test General Layer Type

Choose the Fire Zone Property
Zone

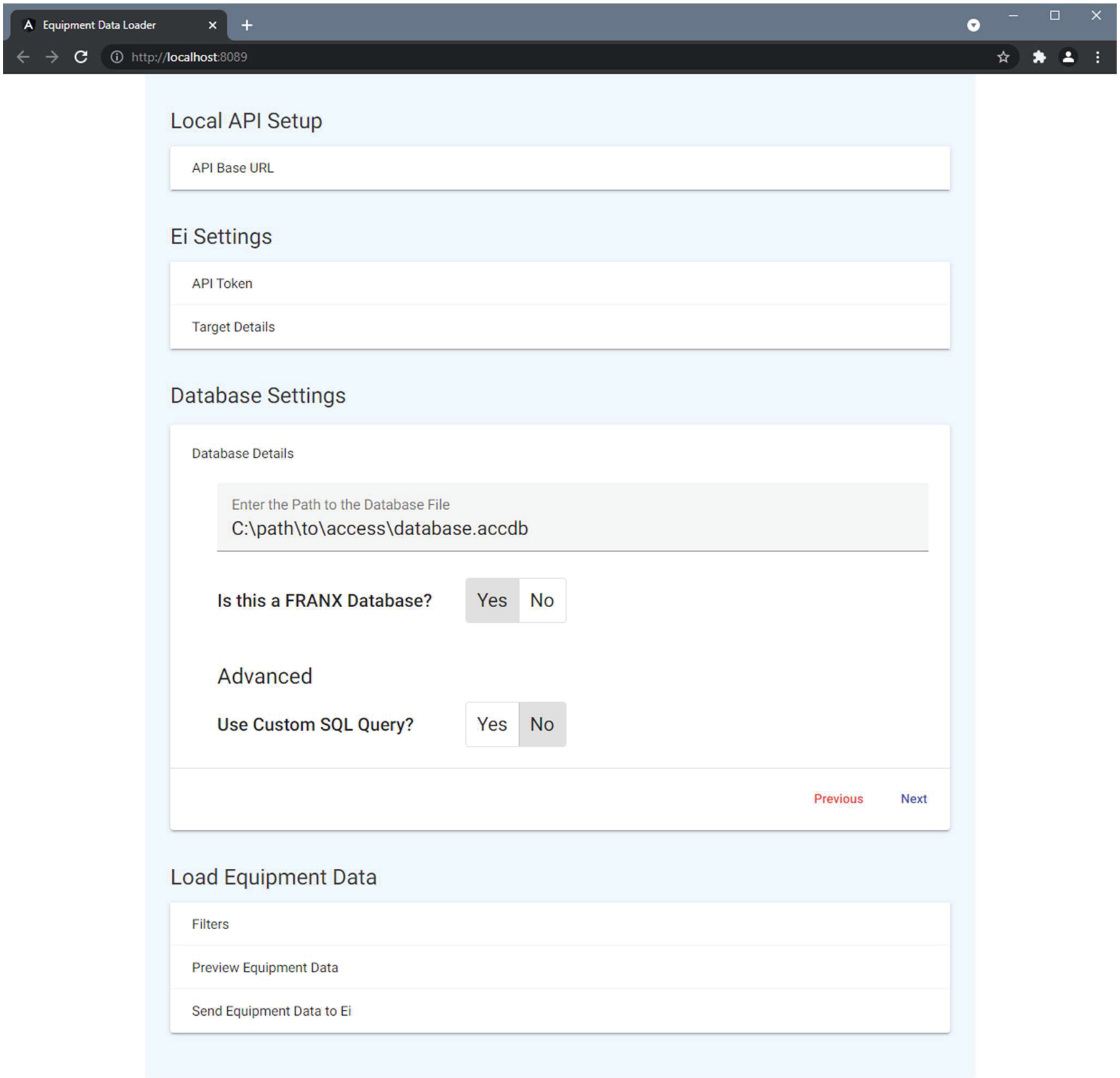
Choose Equipment Type Property
Equipment Type

Database Settings

Database Details

Figure 10. Screenshot of the “Target Details” section of the equipment data loader web app.

The next step in the flow is inputting the database details; these details include the path to the database file, whether the database is a FRANX database, table and column names if it is not a FRANX database, and the option to use a custom SQL query. Screenshots of this section are shown in Figure 11, Figure 12, and Figure 13. The custom SQL option is an advanced option available for use with non-standard databases and should be used with caution; ideally, it should only be used if the database is not a FRANX database and setting the table and column names failed to produce proper results.



Local API Setup

API Base URL

Ei Settings

API Token

Target Details

Database Settings

Database Details

Enter the Path to the Database File
C:\path\to\access\database.accdb

Is this a FRANX Database? [✓ Yes] No

Advanced

Use Custom SQL Query? Yes [✓ No]

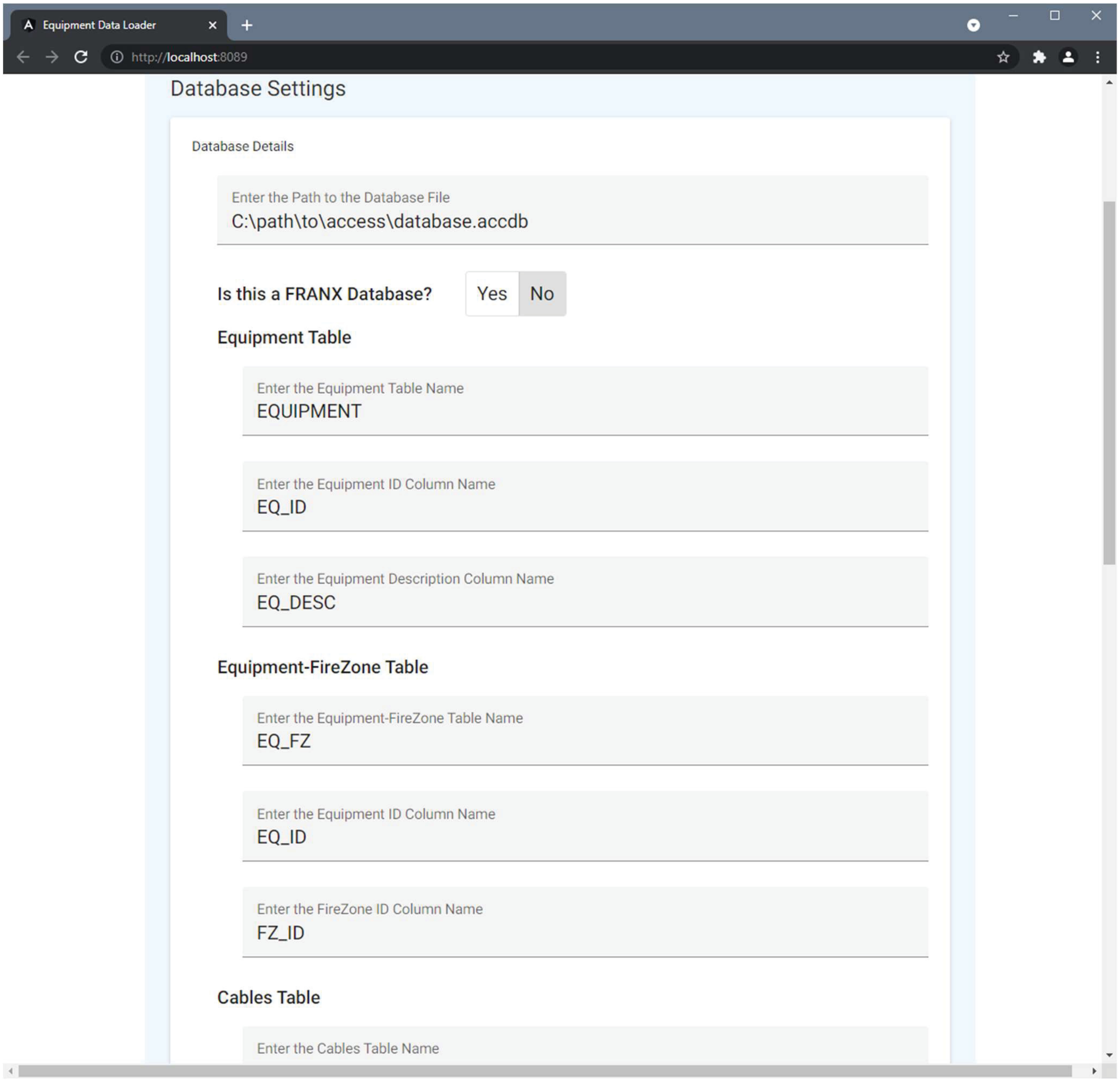
Load Equipment Data

Filters

Preview Equipment Data

Send Equipment Data to Ei

Figure 11. Screenshot of the “Database Details” section of the equipment data loader web app for a FRANX database.



Database Settings

Database Details

Enter the Path to the Database File
C:\path\to\access\database.accdb

Is this a FRANX Database? Yes [✓ No]

Equipment Table

Enter the Equipment Table Name
EQUIPMENT

Enter the Equipment ID Column Name
EQ_ID

Enter the Equipment Description Column Name
EQ_DESC

Equipment-FireZone Table

Enter the Equipment-FireZone Table Name
EQ_FZ

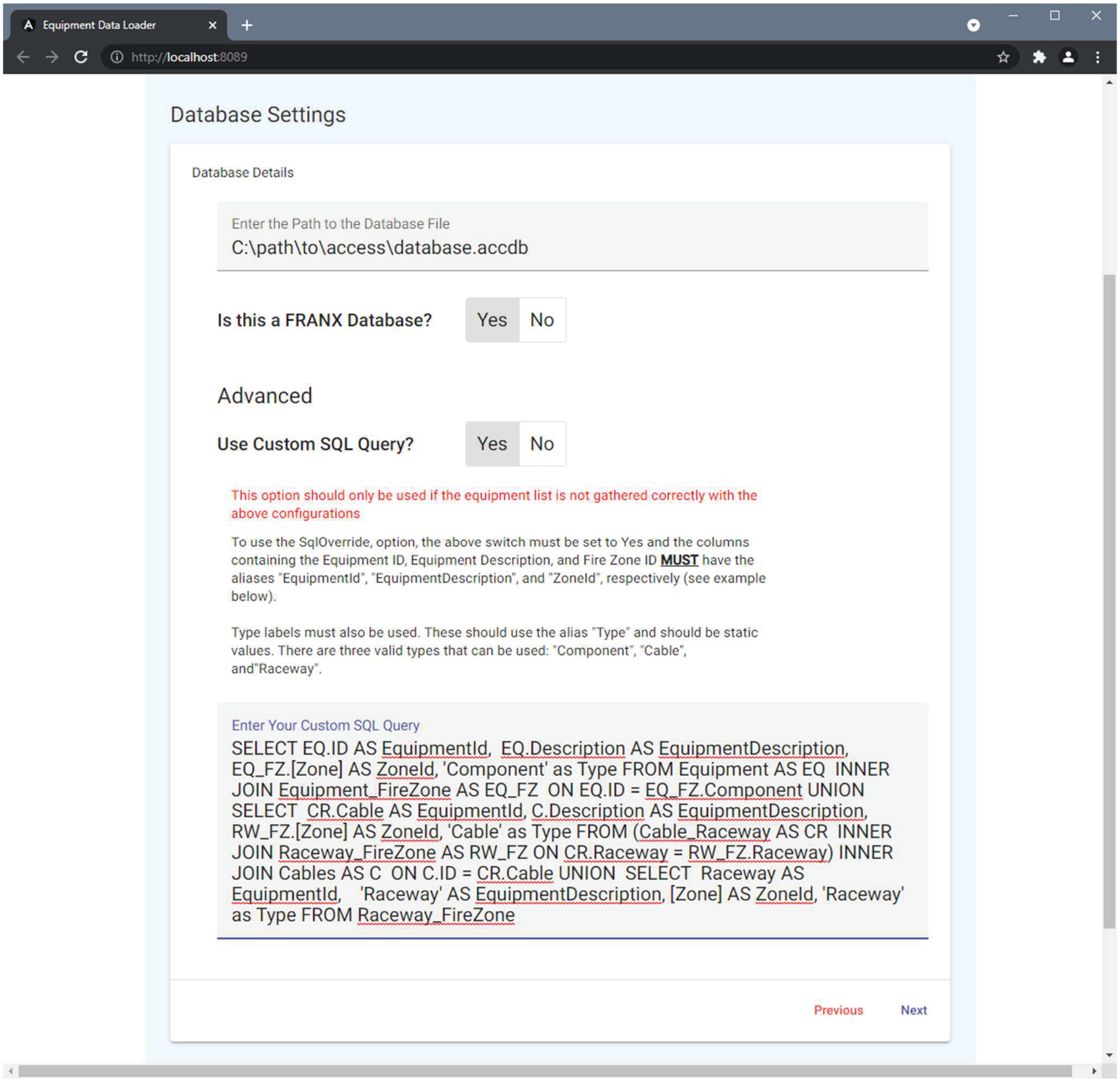
Enter the Equipment ID Column Name
EQ_ID

Enter the FireZone ID Column Name
FZ_ID

Cables Table

Enter the Cables Table Name

Figure 12. Screenshot of the "Database Details" section of the equipment data loader web app for a non-FRANX database.



Database Settings

Database Details

Enter the Path to the Database File
C:\path\to\access\database.accdb

Is this a FRANX Database? [✓ Yes] No

Advanced

Use Custom SQL Query? [✓ Yes] No

This option should only be used if the equipment list is not gathered correctly with the above configurations

To use the SqlOverride, option, the above switch must be set to Yes and the columns containing the Equipment ID, Equipment Description, and Fire Zone ID **MUST** have the aliases "EquipmentId", "EquipmentDescription", and "ZoneId", respectively (see example below).

Type labels must also be used. These should use the alias "Type" and should be static values. There are three valid types that can be used: "Component", "Cable", and"Raceway".

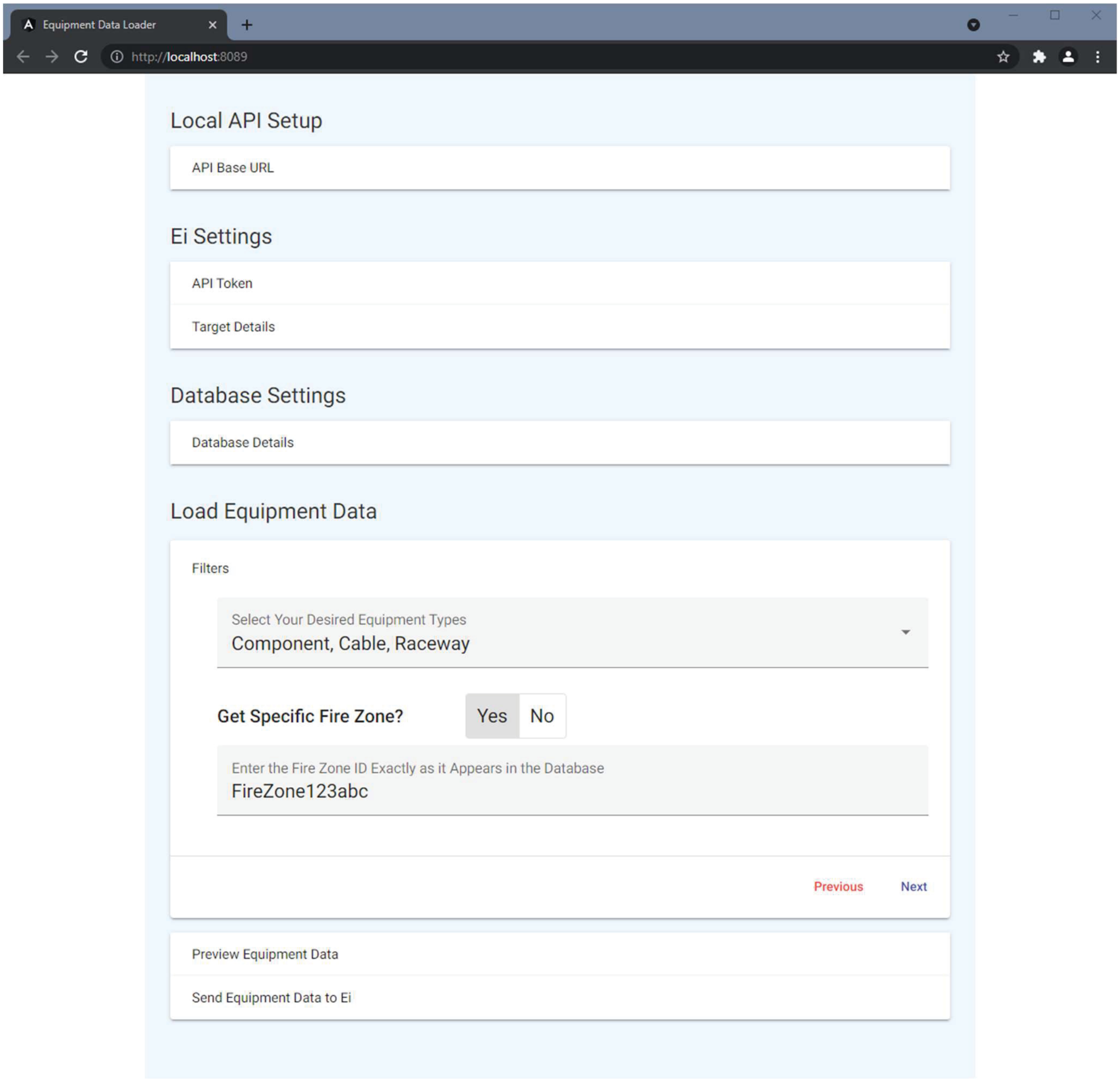
Enter Your Custom SQL Query

```
SELECT EQ.ID AS EquipmentId,  EQ.Description AS EquipmentDescription,
EQ_FZ.[Zone] AS ZoneId, 'Component' as Type FROM Equipment AS EQ  INNER
JOIN Equipment_FireZone AS EQ_FZ  ON EQ.ID = EQ_FZ.Component UNION
SELECT  CR.Cable AS EquipmentId, C.Description AS EquipmentDescription,
RW_FZ.[Zone] AS ZoneId, 'Cable' as Type FROM (Cable_Raceway AS CR  INNER
JOIN Raceway_FireZone AS RW_FZ ON CR.Raceway = RW_FZ.Raceway) INNER
JOIN Cables AS C  ON C.ID = CR.Cable UNION  SELECT  Raceway AS
EquipmentId,    'Raceway' AS EquipmentDescription, [Zone] AS ZoneId, 'Raceway'
as Type FROM Raceway_FireZone
```

Figure *13*. Screenshot of the “Database Details” section of the equipment data loader web app when using a custom SQL query.

When the above steps have been completed, the user is then ready to load the equipment data. Filters can be defined, as shown in Figure 14, and then the data can be previewed before sending it to Ei, as shown in Figure 15. If the previewed data looks correct, the user can click the “Save These Settings” button shown in Figure 15 to generate the proper JSON text that can be copied and pasted into the user’s configuration JSON file, “appsettings.json.” A screenshot of the pop-up box with the generated JSON text is shown in Figure 16.

After the user has verified the previewed data is correct and okay to send to Ei, they can proceed to the final section and click the “Click Here to Send The Equipment Data to Ei” button. This will send the “GET” request described at the beginning of Section 2.1, which will make calls to Ei’s API to upload the data. Since this action may take some time, a loading bar will appear under the button, as shown in Figure 17, and disappear once the data has been successfully uploaded to Ei. If an error occurs during this process, the user should see a message appear on the screen saying so.



Local API Setup

API Base URL

Ei Settings

API Token

Target Details

Database Settings

Database Details

Load Equipment Data

Filters

Select Your Desired Equipment Types: Component, Cable, Raceway

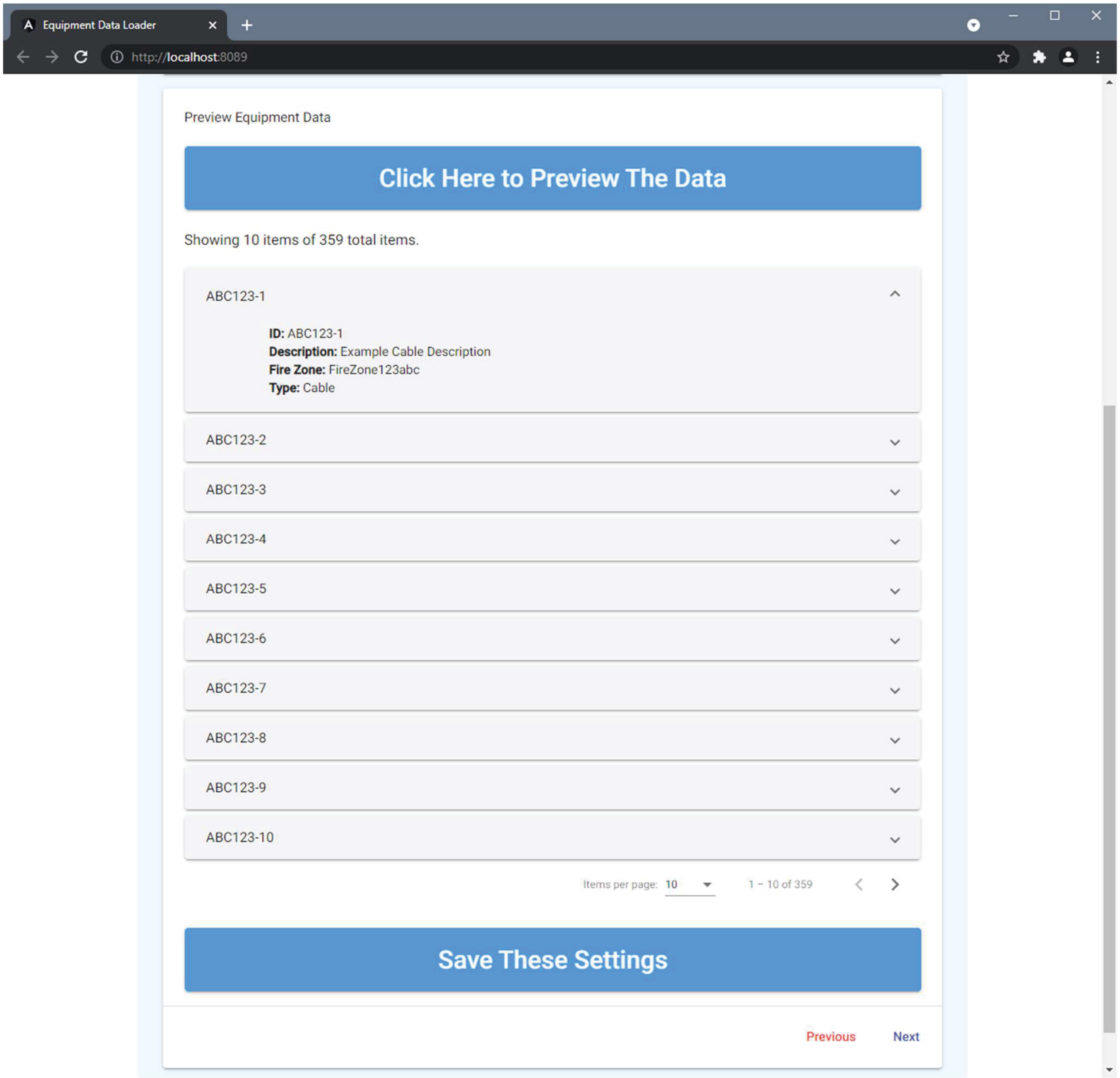
Get Specific Fire Zone? [✓ Yes] No

Enter the Fire Zone ID Exactly as it Appears in the Database: FireZone123abc

Preview Equipment Data

Send Equipment Data to Ei

Figure 14. Screenshot of the “Filters” section of the equipment data loader web app.



Preview Equipment Data

Showing 10 items of 359 total items.

ABC123-1

ID: ABC123-1
Description: Example Cable Description
Fire Zone: FireZone123abc
Type: Cable

ABC123-2

ABC123-3

ABC123-4

ABC123-5

ABC123-6

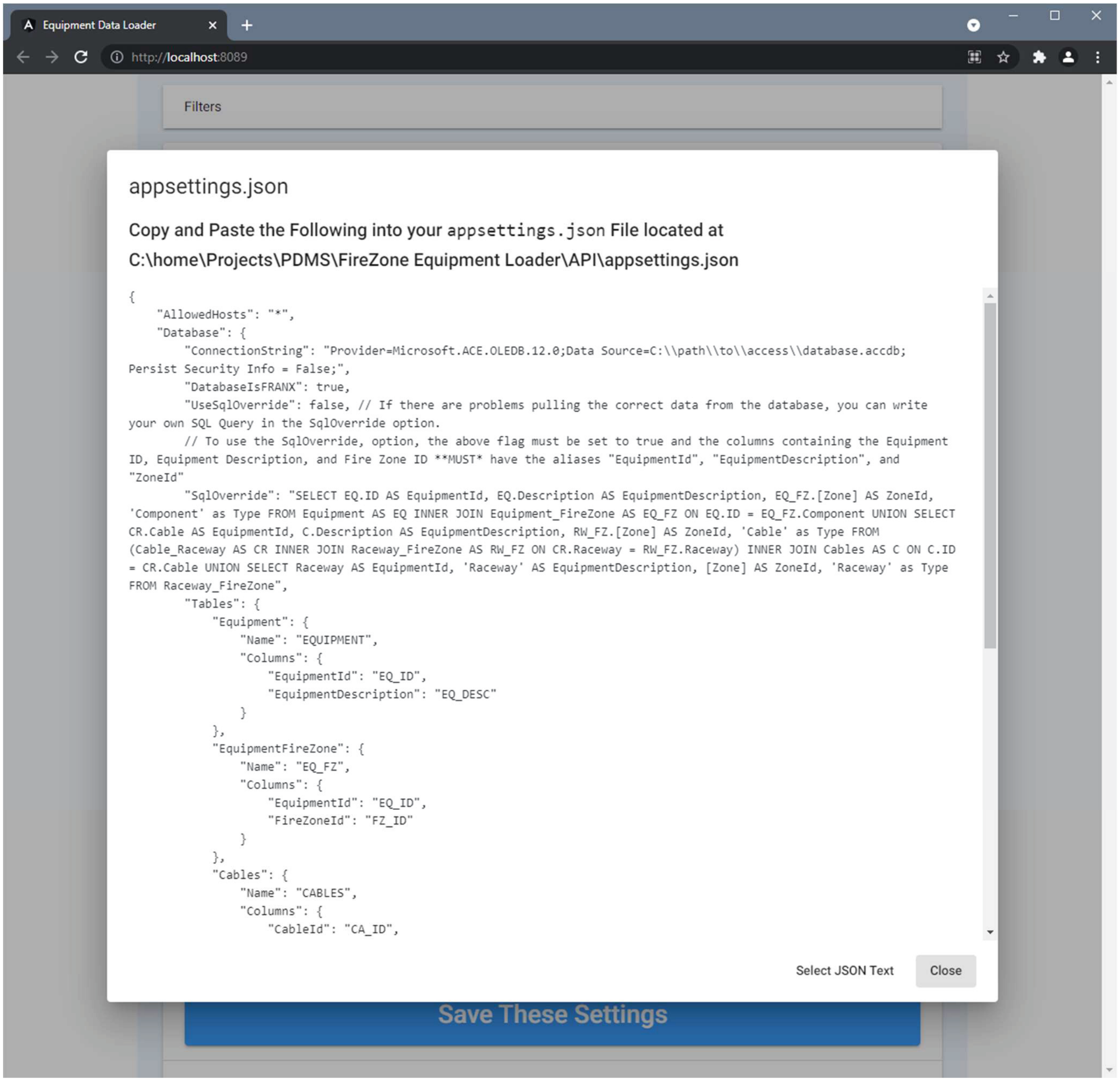
ABC123-7

ABC123-8

ABC123-9

ABC123-10

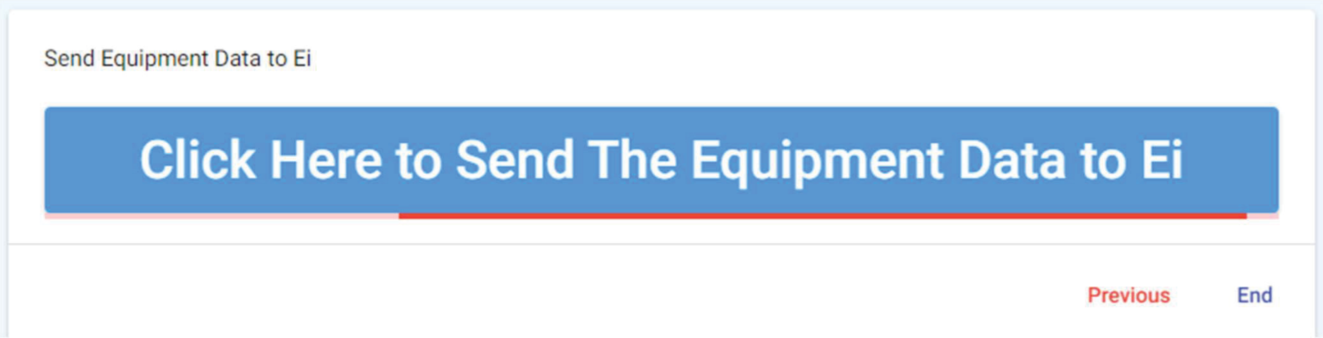
Figure 15. Screenshot of the “Preview Equipment Data” section of the equipment data loader web app.



Copy and Paste the Following into your `appsettings.json` File located at C:\home\Projects\PDMS\FireZone Equipment Loader\API\appsettings.json

```
{
    "AllowedHosts": "*",
    "Database": {
        "ConnectionString": "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\path\\to\\access\\database.accdb;
Persist Security Info = False;",
        "DatabaseIsFRANX": true,
        "UseSqlOverride": false, // If there are problems pulling the correct data from the database, you can write
your own SQL Query in the SqlOverride option.
        // To use the SqlOverride, option, the above flag must be set to true and the columns containing the Equipment
ID, Equipment Description, and Fire Zone ID **MUST* have the aliases "EquipmentId", "EquipmentDescription", and
"ZoneId"
        "SqlOverride": "SELECT EQ.ID AS EquipmentId, EQ.Description AS EquipmentDescription, EQ_FZ.[Zone] AS ZoneId,
'Component' as Type FROM Equipment AS EQ INNER JOIN Equipment_FireZone AS EQ_FZ ON EQ.ID = EQ_FZ.Component UNION SELECT
CR.Cable AS EquipmentId, C.Description AS EquipmentDescription, RW_FZ.[Zone] AS ZoneId, 'Cable' as Type FROM
(Cable_Raceway AS CR INNER JOIN Raceway_FireZone AS RW_FZ ON CR.Raceway = RW_FZ.Raceway) INNER JOIN Cables AS C ON C.ID
= CR.Cable UNION SELECT Raceway AS EquipmentId, 'Raceway' AS EquipmentDescription, [Zone] AS ZoneId, 'Raceway' as Type
FROM Raceway_FireZone",
        "Tables": {
            "Equipment": {
                "Name": "EQUIPMENT",
                "Columns": {
                    "EquipmentId": "EQ_ID",
                    "EquipmentDescription": "EQ_DESC"
                }
            },
            "EquipmentFireZone": {
                "Name": "EQ_FZ",
                "Columns": {
                    "EquipmentId": "EQ_ID",
                    "FireZoneId": "FZ_ID"
                }
            },
            "Cables": {
                "Name": "CABLES",
                "Columns": {
                    "CableId": "CA_ID",
```

Figure 16. Screenshot of the generated JSON text modal of the equipment data loader web app.

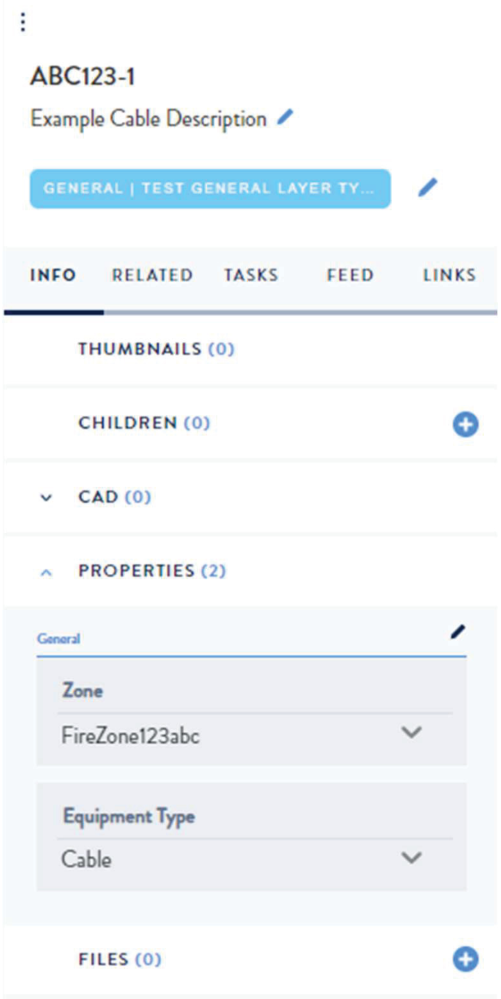


Send Equipment Data to Ei
Click Here to Send The Equipment Data to Ei
Previous
End

Figure 17. Screenshot of the final step of the equipment data loader web app to send data to Ei with loading bar.

2.4 Plant Data in the Ei's System

When the plant data is being loaded into Ei's system, each item will be named using the item's ID and retain its associated description. The items will be assigned to the selected "Map," "Layer," and "Layer Type" (see Figure 10) and will automatically populate the selected "Fire Zone Property" and "Equipment Type Property" with their proper values. The details for the cable featured in Figure 15, after it has been loaded into Ei, are shown in Figure 18.



ABC123-1

Example Cable Description

General

Zone: FireZone123abc

Equipment Type: Cable

Figure 18. Example details of a cable after being loaded into Ei.

2.5 Customization

The plant data API and the equipment data loader web app were built to specifically communicate with Ei. However, since the code is simply a framework using Ei's API, it can be customized to adapt to the APIs of other tools. This would allow for the equipment data to be extracted in any desired format and then imported into any other tool or system.

3. AI IMAGE PROCESSING IN LIDAR

AI is used to describe many methods for automatic processing of digital information. For this research, the AI methods are used to look at LiDAR images and find text on labels, then read and match that text to database information.

3.1 AI Methods

There are many AI methods for processing various types of data, but one of the most well-developed and reliable is image processing. Through the use of training sets, AI algorithms can effectively identify objects in pictures [4], and this is becoming prevalent in day-to-day activities. For those driving a Tesla, they see it as they drive with the shape of nearby vehicles being identified into general categories such as cars, SUVs, trucks, etc. as shown in Figure 19.

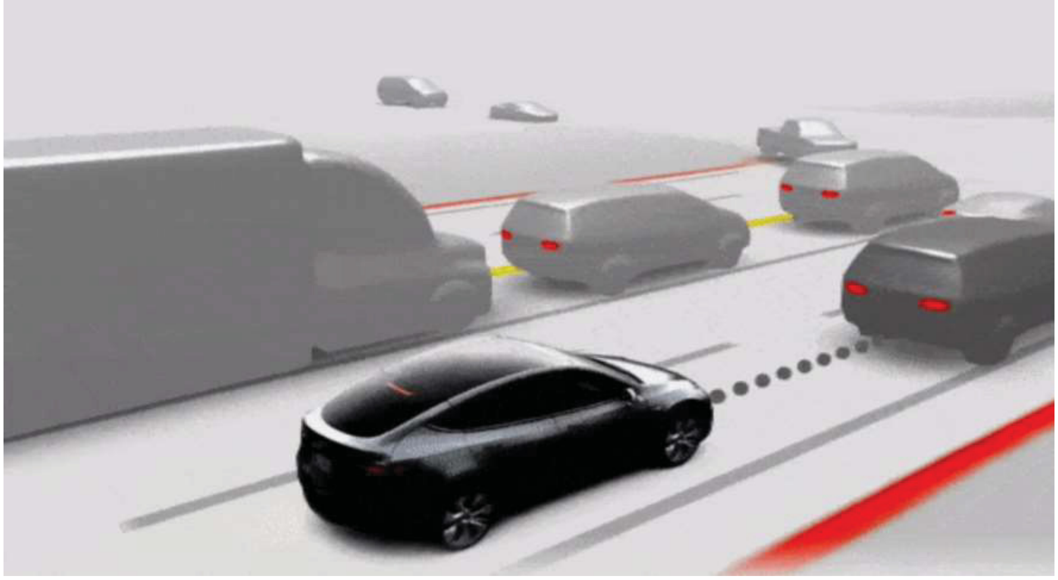


Figure 19. Example of Tesla's real-time vehicle classification.

The accuracy of models to detect objects in LiDAR data (by itself) is lacking. A majority of the domain-expertise in the greater AI/machine vision space is focused on detecting objects in 2D photos, where the algorithms foundationally are based on interpolating results from the 2D matrix of pixels presented in a static image. LiDAR scans typically also take detailed 2D images while scanning, as shown in Figure 20 that can be used for the machine vision. Though 3D LiDAR data has significantly more data (including positional data), the 3-D nature of the data is a unique challenge for AI object detection algorithms. For the best results, we utilized a modified approach which included using tried-and-true machine vision models to make detections on the 2D photos and then used additional positional metadata provided by the LiDAR scans to interpolate accurate 3D positions.

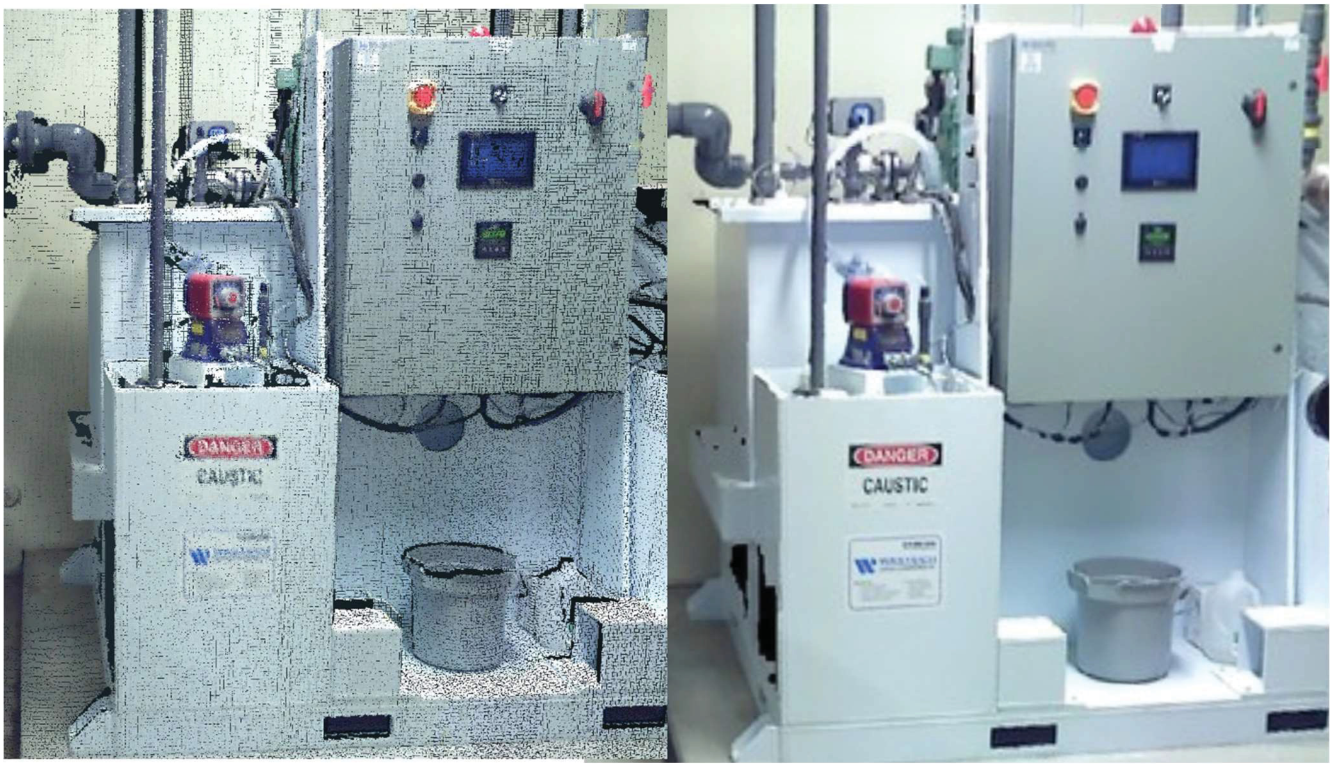


Figure 20. LiDAR 3D point cloud on the left, 2D image taken with the scan.

3.2 Text Identification

Initial work attempted to identify components by having the AI locate tags or placards as shown in Figure 21 [1]. This was quite successful in finding the location of tags and placards, but it became too complex to read the text off the tags to link with the plant database information.

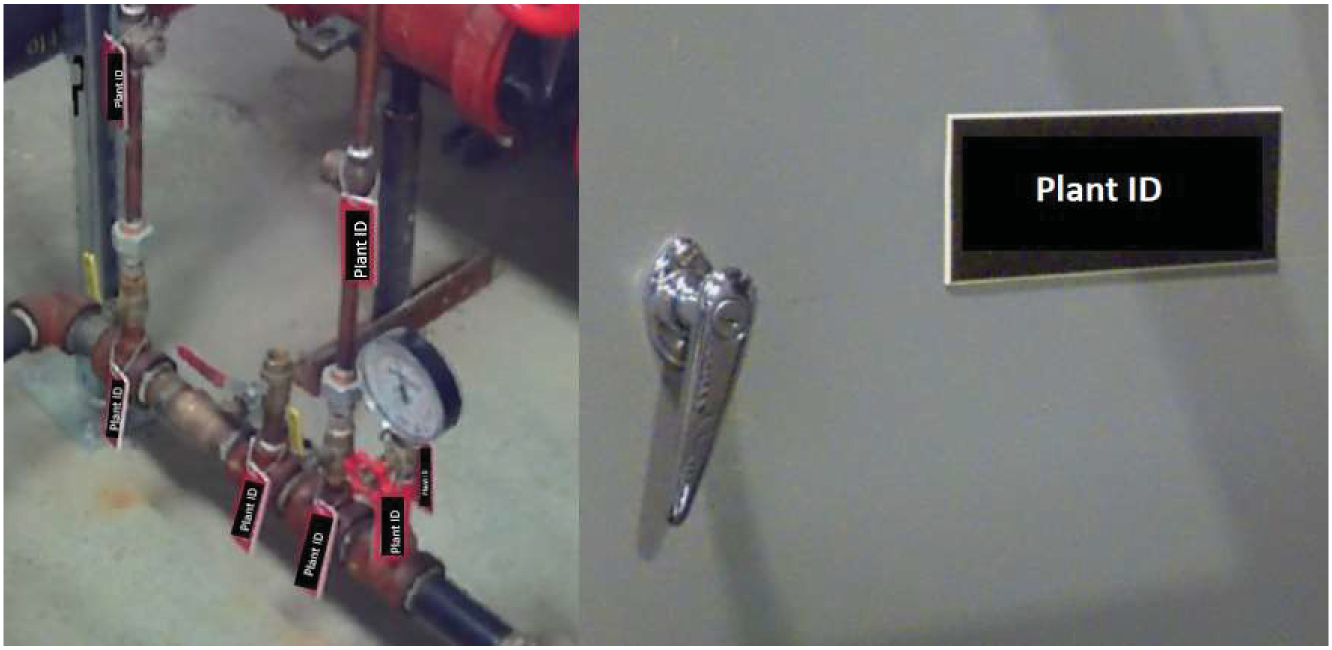


Figure 21. Equipment tags and placards used by nuclear facilities.

To overcome the challenge, a different approach was taken to directly identify text. Ei utilized Google's Vision API [5] to detect all text in the images as shown in Figure 22. The processing of the API is handled in Ei Everywhere, which internally produces 2D bounding boxes for all detected text, along with the text it detected, the vertices for the bounding box, and a probability score as shown in Figure 23.

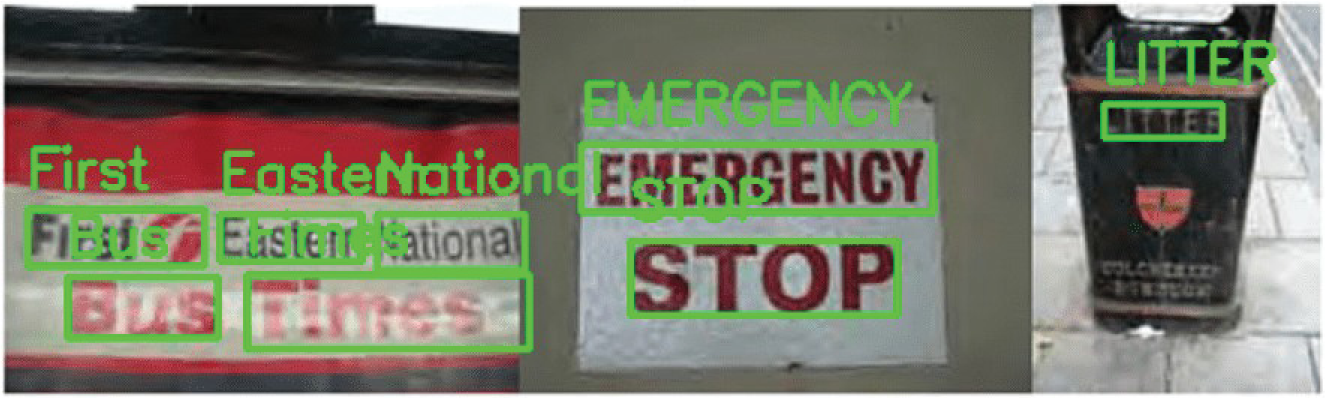
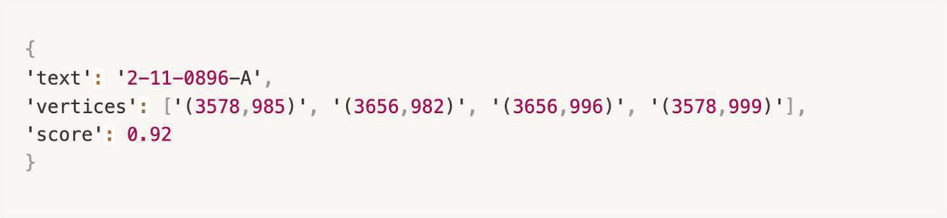


Figure 22. Example of Google Vision identifying and reading text.



```
{
'text': '2-11-0896-A',
'vertices': ['(3578,985)', '(3656,982)', '(3656,996)', '(3578,999)'],
'score': 0.92
}
```

Figure 23. Example data output from Google's Vision API for detected text.

3.3 Ray Casting and Interpolation

Utilizing the text-detection findings from Section 3.2, the Ei Everywhere application and algorithm utilizes information from nearby photos, including their identified text-detection bounding box and known coordinates, to interpolate valid detections and identify known positions. The algorithm mathematically traces rays from the origin of each photo through the bounding boxes identified by the text-detection step. The intersection for at least two (2) rays of similar text from adjacent photos constitutes an ideal "hit," and as such, the application places a suggested 3D pin in that location (associated with other metadata).

3.4 User Tagging

Currently the linking the AI-identified items to the plant data is done in two steps. The first is to validate the accuracy of AI-detected objects, Ei utilizes a "human-in-the-loop" accept/reject workflow, which is built into the Ei Everywhere application. Detected items in the area are shown in a list as shown in Figure 24. This step eliminates all the irrelevant text from signs, information labels, or other similar text matches. The placed pins from the ray casting step are presented to the user in a workflow-like stage where they are presented the options of either approving or rejecting the detections.

Selecting "Approve" on any detection confirms the pin's location, where the user is prompted with a screen to confirm the linkage to the externally loaded data. This validation from the user makes sure all text identified will be either categorized as a specified plant item or marked as irrelevant text, reducing the human error of missing items.

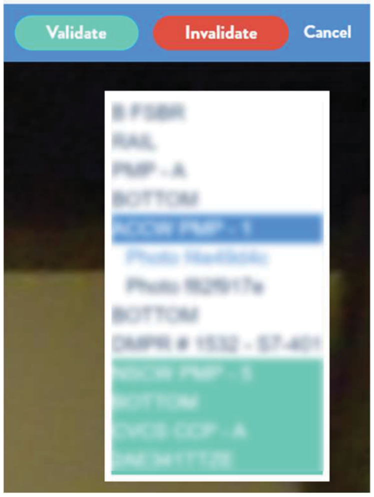


Figure 24. User validation workflow screenshot.

3.5 Name Matching

To complete and validate the linking between the LiDAR pins and the uploaded plant data, a second "human-in-the-loop" process is used. For each item identified in the previous step, Ei performs a regular expression-based matching to identify potential matches to an item uploaded from the plant database. This information is presented to the user in a "search box" format, which highlights the highest ranked matches at the top of the results. The algorithm performs primary weighting on the numerical aspects of the detected text.

After the automated identification, the correct tag is confirmed by the user, forming a link to the external record and the 3D LiDAR data. After this step is done for all the items, the LiDAR model contains valid tags with names and descriptions corresponding directly with the actual facility information. If there are items uploaded from the plant database that did not get linked to identified text, they could also be shown to the user. This would provide a secondary check to verify all required items were identified in the LiDAR model. Further work needs to be done on how to visualize and show discrepancies to the user.

3.6 Item Bounding Boxes

Once the user acceptance steps are completed, the user can adjust the 3D bounding box to fit around the equipment utilizing the Ei Everywhere application. The bounding box by default is centered around the detected text and user adjustable to include the entire equipment's bounding box as shown in Figure 25.

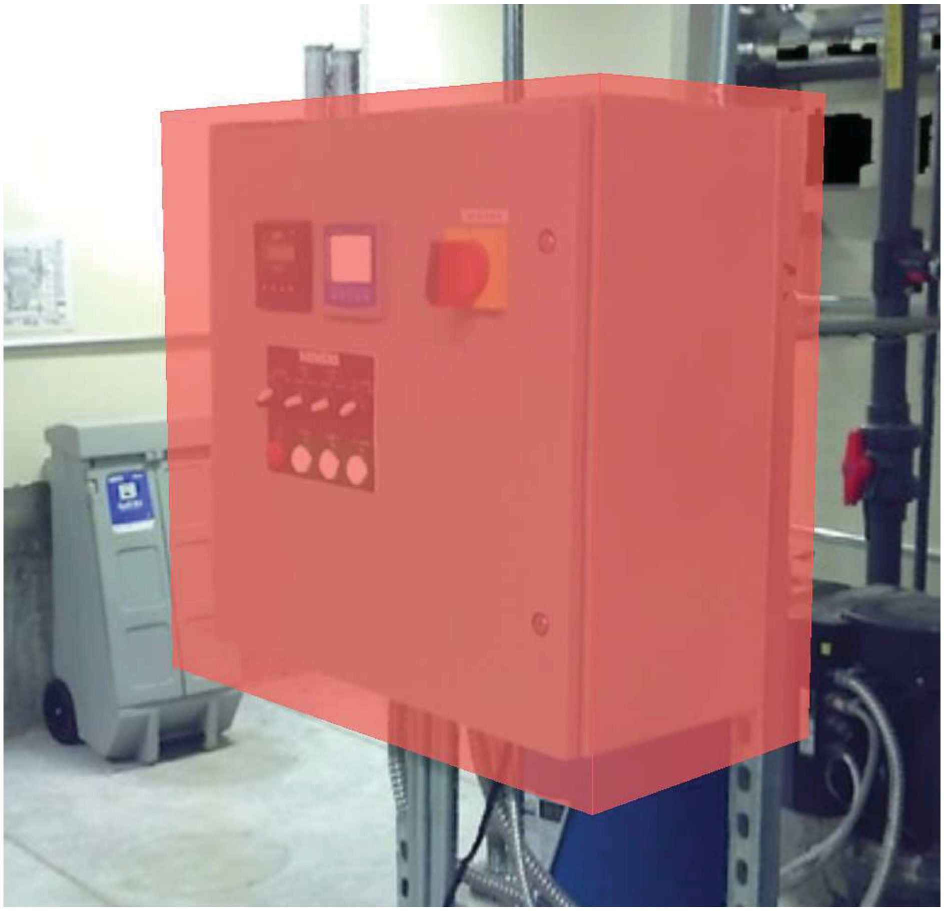


Figure 25. Component bounding box example.

3.7 Initial Testing Results and Issues

Using the scans from the industry switch gear room were used for initial testing. The scans were performed roughly every meter and due to the limited space of walkways, about 0.85 m away from the cabinets. For the electrical cabinets that were spaced about 0.6 m and scans roughly a meter apart. Text matching was highly effective for about 0.5 meter on either side from the center point of the scan.

In the first pass of testing, two matches in Google Vision were used in order to do the ray casting described in Section 3.3. However, only a few full matches were being made. This was due to a couple factors. First, the scan spacing and only having good text recognition overlap of every other cabinet. Second, if the Google Vision from one picture was slightly different from another, then it was not considered a match.

To overcome this, the requirement of two matches was removed. This can make the location information slightly off, but since the user is adjusting this location when making the bounding box, this is not an issue.

4. TAG DATA EXPORT TO FRI3D

Once a LiDAR model has the components tagged and bounding boxes around the items, this information can be used in a variety of use cases needing relative item locations. This research has focused on using the spatial information to reduce manual efforts needed for the detailed fire modeling done in FRI3D.

4.1 FRI3D Background

Detailed fire modeling of NPPs requires many steps, tools, and calculations. The goal of the FRI3D software is to simplify and automate as many of those steps as possible under one platform with a user-friendly interface as shown in Figure 26. FRI3D focuses on following main areas:

1. Import existing FRANX or PDMS, mapping or logic, and component models
2. Model, display, and link 3D spatial models to the imported data
3. Auto-generate fire scenarios from user-specified fire sources using industry fire simulation codes and methods
4. Output and calculate scenario information using industry PRA tools.

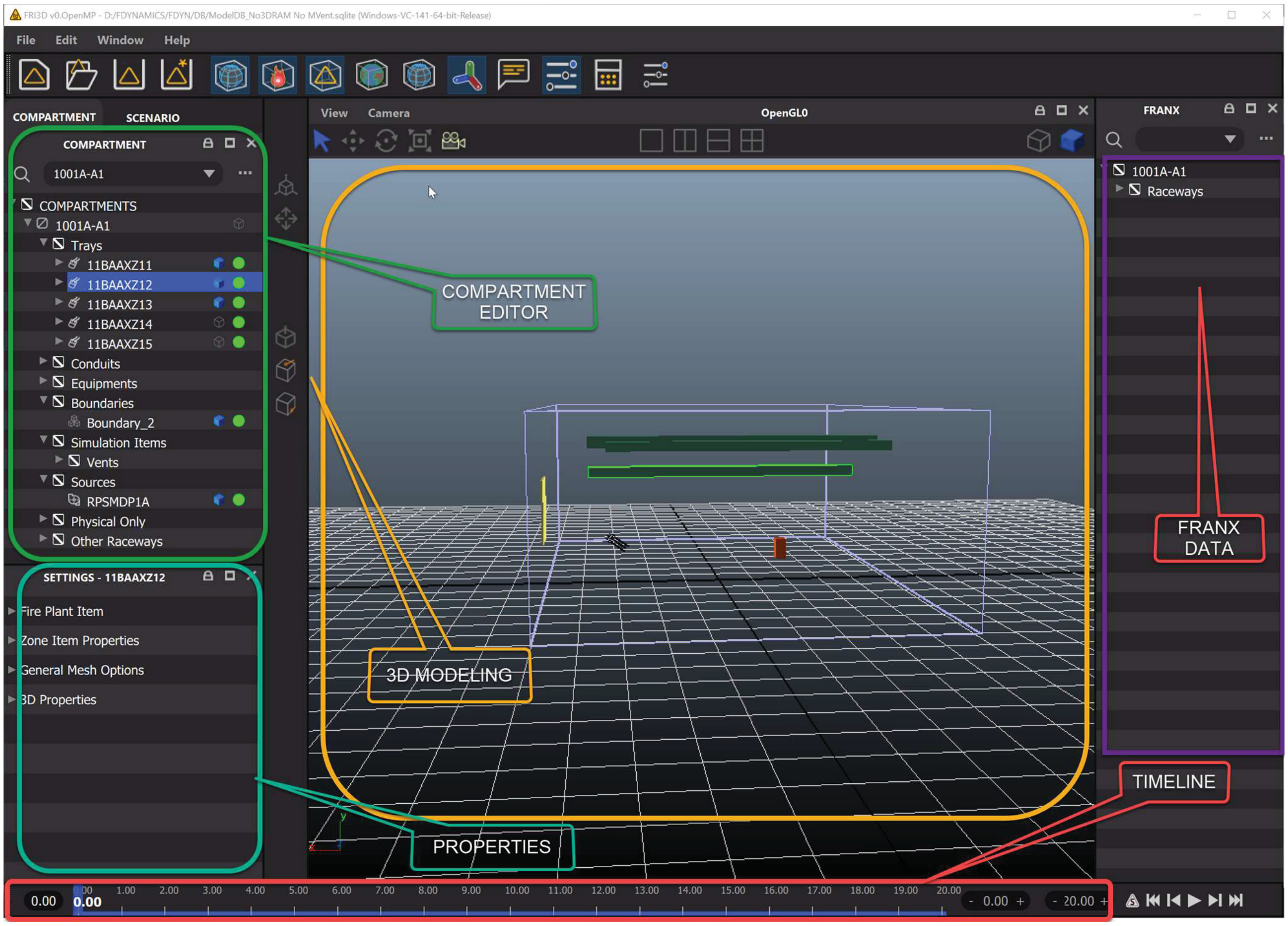


Figure 26. FRI3D graphical user interface layout with data areas labeled.

The 3D area in FRI3D provides both a visual representation of the fire compartment with all the critical components and fire sources and the data needed to run fire simulations. The 3D data is converted into a Consolidated Model of Fire and Smoke Transport (CFAST) fire simulation model and run to generate temperature profiles. This spatial information is also used to perform other fire analyses, such as the evaluation of secondary combustibles. All these areas contribute to the automated process for generating, running, and outputting scenarios as shown in Figure 27.

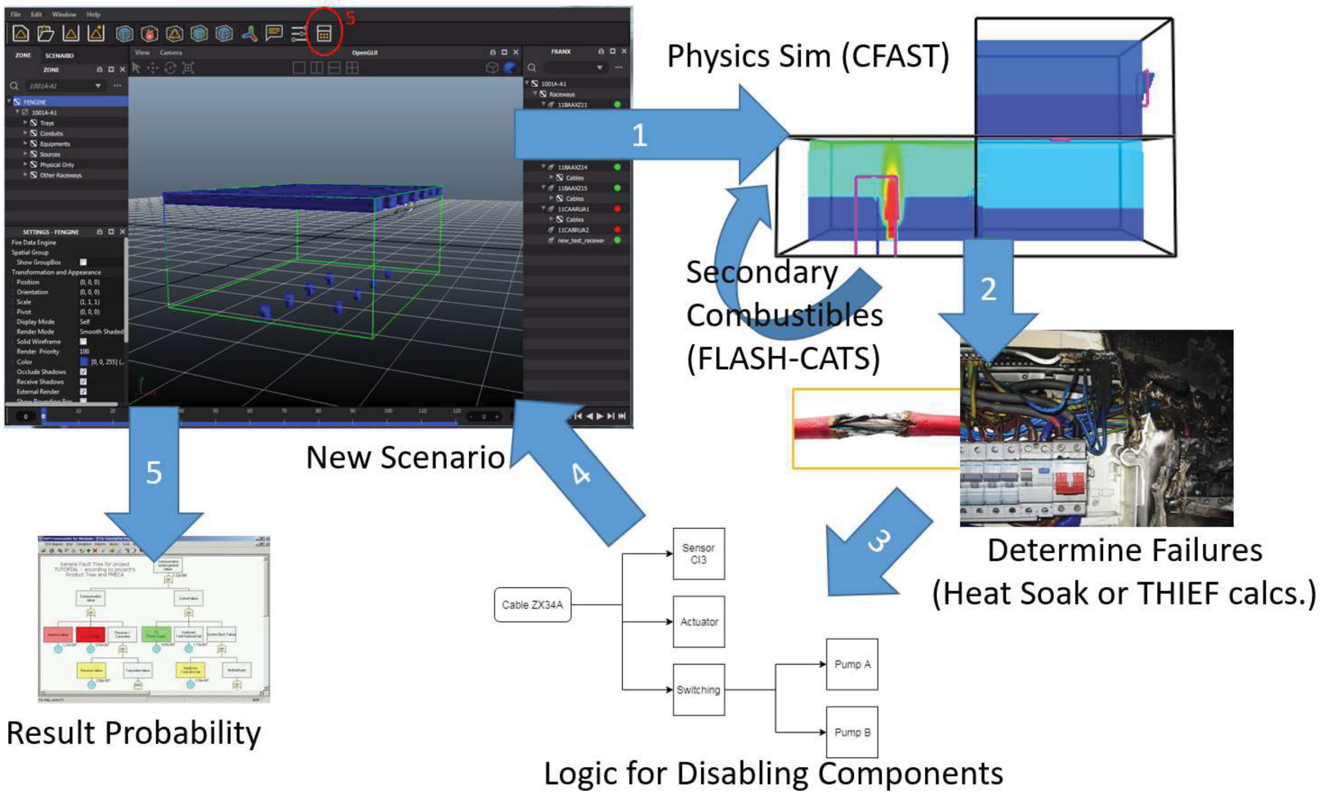


Figure 27. Automated process for generating fire scenarios and viewing results in FRI3D.

Research has shown that FRI3D can reduce time needed for detailed fire modeling scenarios by more than 50% and significantly reduce time for fire reviews required for plant modification [6]. Testing was done using two Nuclear Regulatory Commission (NRC) examples, Appendix A and B of NUREG-1934 [7] and an industry switch gear room (the same plant switch gear room used in this LiDAR scan research).

Even with significant time reductions, one of the major concerns by industry was the need to make accurate 3D models of the fire compartment using architectural drawings and walkdown notes. In the further testing of FRI3D, some modelers had the LiDAR scan available as a virtual walkdown and measurement tool. This simplified some of the modeling tasks and eliminated the need for in-person inspections for those tests. While it is fairly simple to perform the 3D modeling using the FRI3D interface, it is still the most time-consuming process. With ability to import the location and general bounding area of the components, the user can focus on defining fire sources and planning modifications.

4.2 LiDAR API Data

The Ei platform provides a web API to request specific spatial information from the LiDAR model. The normal way to get object data from the Ei API is by making a GET request to the "/v1/objects" endpoint. By default, though, this does not include spatial information; in order to retrieve this information, Ei has implemented the ability to specify optional fields to be included in the returned data. Using the "position" optional field will include spatial information in the results. This is done by adding "optionalFields=position" as a query parameter to the initial endpoint, resulting in "/v1/objects?optionalFields=position." Using other query parameters, Ei also allows for filtering by map and layer.

Including the position field in results does not guarantee that all objects will have position data. Only objects that have been tagged and located within the LiDAR model will have position data, and further, only objects that have had a "volume" added to them will have 3D spatial data, as shown in Figure 28. The spatial data returned by the API call is in a JSON format, providing the location, shape, and size, as shown in Figure 29.

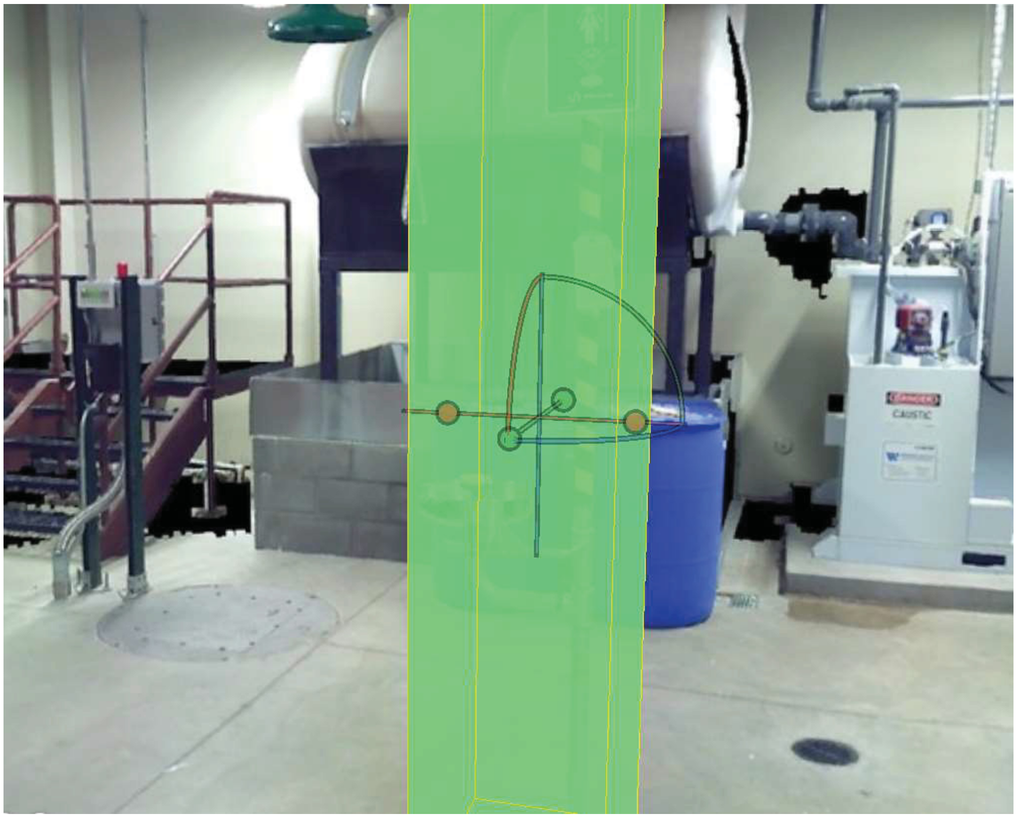
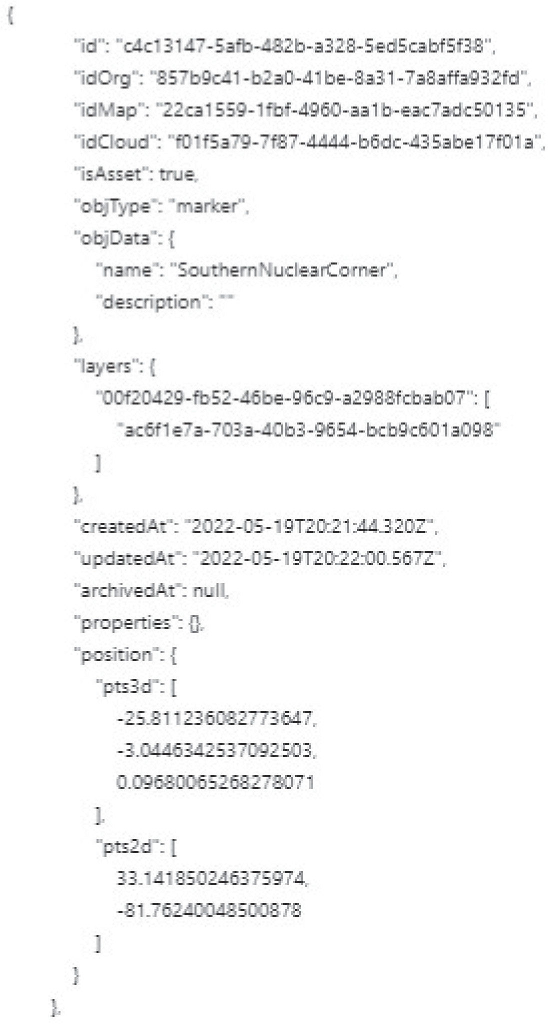


Figure 28. Spatial information for a piece of equipment with a location tag and volume assigned in the Ei platform.



```
{
        "id": "c4c13147-5afb-482b-a328-5ed5cabf5f38",
        "idOrg": "857b9c41-b2a0-41be-8a31-7a8affa932fd",
        "idMap": "22ca1559-1fbf-4960-aa1b-eac7adc50135",
        "idCloud": "f01f5a79-7f87-4444-b6dc-435abe17f01a",
        "isAsset": true,
        "objType": "marker",
        "objData": {
          "name": "SouthernNuclearCorner",
          "description": ""
        },
        "layers": {
          "00f20429-fb52-46be-96c9-a2988fcbab07": [
            "ac6f1e7a-703a-40b3-9654-bcb9c601a098"
          ]
        },
        "createdAt": "2022-05-19T20:21:44.320Z",
        "updatedAt": "2022-05-19T20:22:00.567Z",
        "archivedAt": null,
        "properties": {},
        "position": {
          "pts3d": [
            -25.811236082773647,
            -3.0446342537092503,
            0.09680065268278071
          ],
          "pts2d": [
            33.141850246375974,
            -81.76240048500878
          ]
        }
      },
```

Figure 29. Spatial data returned by Ei in the JSON format.

The spatial data returned by the Ei platform is relative to its own internal markers. In order for this data to be useful to other applications, it needed to be relative to a user-specified location. To do this a relative marker was added to the LiDAR model for the desired zone, the user adjusts this to the center or a corner of the zone, and when processing the data, this marker is used to adjust that data.

Currently, there is no way to retrieve only objects that have spatial data, which means that any filtering is done locally. Additional options are also needed to filter by specific property values and to filter by zone. The zone filtering is also done locally, but it requires more time for returning data from the Ei platform. In order to perform these kinds of filtering locally, it is best to retrieve all objects for the desired map and layer from Ei, which may require multiple requests to the API. After retrieving all objects, they can easily be filtered to produce a list of all objects in a specific zone that also has spatial data.

4.3 FRI3D Module

In order to port the LiDAR data into an application, there must be access to the applications model or methods to modify the model. FRI3D has a modular design, where the backend is a dynamically linked library (DLL) of methods, that the user interface calls to get and set data. This means that other applications can make the same calls as the user interface does. Given this access, a service module for FRI3D porting was added to the same project used for uploading data to the Ei platform. While this research is focused on the feasibility or testing, it is anticipated that software to convert from a LiDAR platform will have a module for each different

LiDAR platform and then a module to modify or output data compatible for each application needing the 3D spatial information.

The FRI3D DLL methods use a JSON format for outputs and input a JSON format for most the calls. The needed functions were defined in DLL import files so they can be called as needed inside of FRI3D. The 3D information is an array of items that defines the position, scale, and pivot, along with other info only relevant to the display and not set by the LiDAR data. A default data file is used for the display data. To port the data from the LiDAR model, the following steps are performed using the DLL:

1. Open the FRI3D model
2. Get all the items from the LiDAR model
3. Filter the items by location (i.e. Compartment or Zon)
4. For each item remaining, get the item from FRI3D
5. If the item exists
    a. If it has 3D data, update with new spatial info from the LiDAR data
    b. If it does not have 3D data, use the default info along with the LiDAR data
    c. Save the item in FRI3D
6. If the item does not exist, skip it.

4.4 FRI3D Model Results

Initial testing was performed by manually adding tags and bounding boxes for equipment in the LiDAR model along with a relative marker tag. Those tags and bounding data where then retrieved and ported into a test FRI3D model. Verification and adjustments were then performed on the resulting FRI3D model. The relative marker was considered as the center point in FRI3D. The 3D positions, and a few shape types were tested and manually verified to correctly transfer to the correct locations in the 3D model. The example shown in Figure 30 demonstrates the results of the FRI3D module where the tagged items with bounding boxes in the top image are converted into and added to the FRI3D model.

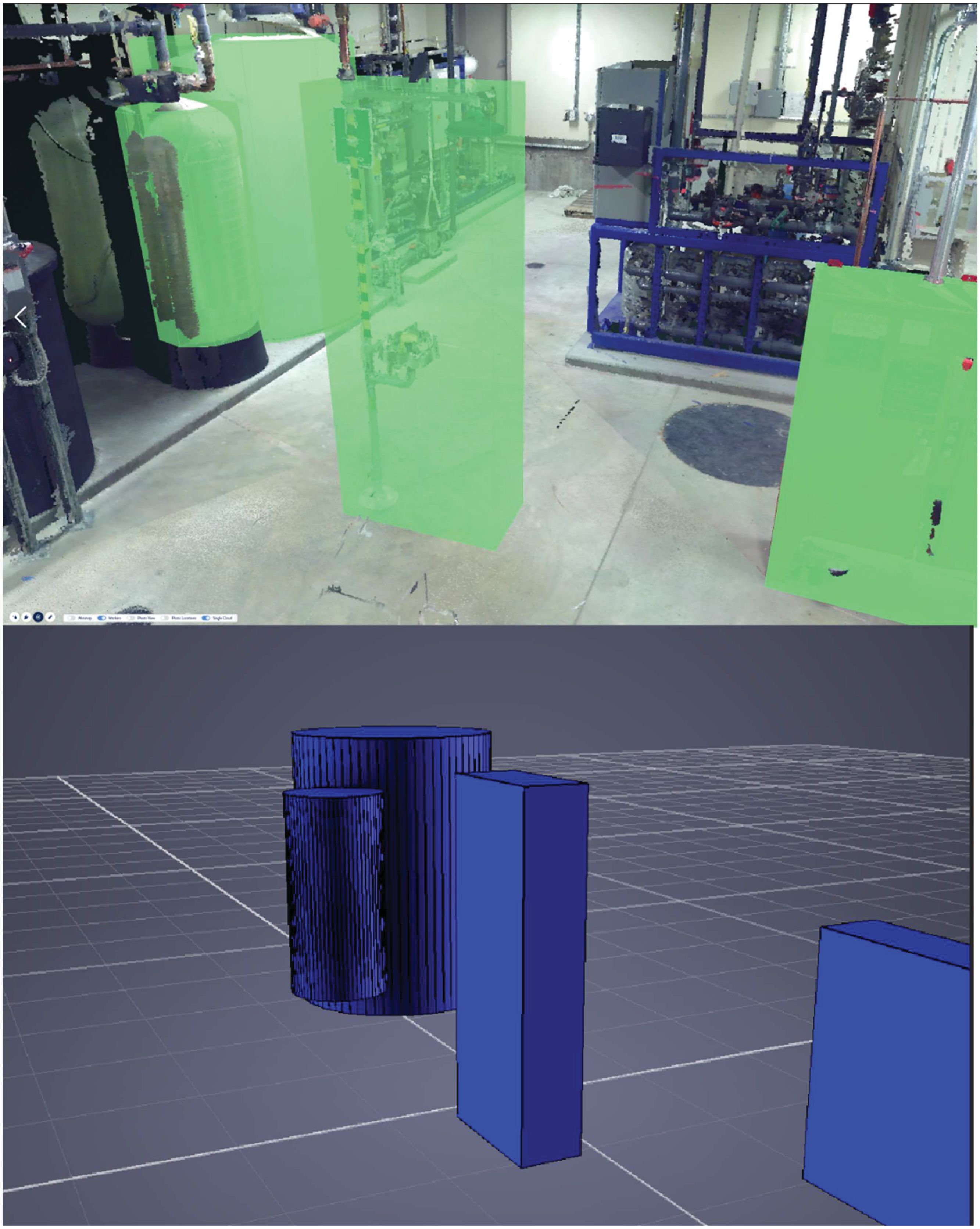


Figure 30. Top - Tagged items in Ei LiDAR model. Bottom - Results from automatically porting tagged 3D data to FRI3D.

4.5 Summary Results

The feasibility of this process was tested for an operating NPP. A switchgear room LiDAR scan as outlined in Section 1.5 was uploaded to the Ei visualization platform. Then, the software described in Section 2 was used to import the components for the same switchgear room from the PDMS database. Next, the AI methods described in Section 3 were used to tag the components with bounding boxes manually adjusted to match the components. Finally, the 3D data for the tagged items was automatically assigned to the corresponding components of the FRI3D compartment.

A manual inspection of the switchgear room verified that Google Vision effectively identified all text except for a few text items blocked by obstacles. There were small variations in the database names vs. what was on the labels, such as underscores instead of dashes or spaces. The regular expression matching helped with this task, providing similar matches from the database.

The results between the LiDAR bounding boxes and the 3D objects automatically generated in FRI3D were qualitatively compared. The center point marker auto-adjusted the location information successfully so the items were correctly positioned in the FRI3D model.

Results from this work show that while LiDAR data and AI are not reliable enough for fully automated component identification, it can be successfully used with a “human-in-the-loop” process to substantially reduce the work and expertise required. The FRI3D auto-generated 3D example shows that once tagging has been performed and bounding boxes added, it is a fairly simple process for data to be used by applications relying on spatial information.

Using LiDAR for nuclear modeling activities and AI processing algorithms will require validation and scanning guidance. Virtual measurement and outlining tools will need to be NQA-1 compliant, verifying they are accurate, for use in nuclear safety modeling. For the AI algorithms to be most effective, guidance for photo quality, scan spacing, and max/min scan distance from targets needs to be established.

5. FUTURE WORK

Initial feasibility testing was successfully performed for the methods outline in this document. The final FY 2022 task of the project is to fully and make bounding boxes for all the items in the industry switch gear model and perform a qualitative comparison with a manually generated model made in FY 2021.

The following are planned areas of future work:

1. **Streamlined User Interface:** This work focused on the feasibility of using AI to identify the labels and matching them to database information. Minimal work has been done on an interface to show matches then validate or modify them. More research and development needs to be performed in order to optimize this process and make it easier for the user with focus on allowing the user to automatically (and in bulk) accept tagging and linkage options based on AI probability and match weighting.
2. **Generation of 3D Bounding Boxes for Equipment Utilizing AI:** Currently, the user still must manually confirm the 3D bounding box around the detected equipment, as the text-detection algorithm only detects the text. To allow the algorithm to automatically suggest 3D bounding boxes around equipment, a more robust equipment detection model may be developed. This equipment detection model could then leverage the already developed ray casting algorithm (utilized for the current text detections). The ray casting algorithm can be modified to not only calculate the center of the intersection of various rays but calculate the effective volume of their intersection (such as the smallest cube or sphere that fits the intersection).

6. REFERENCES

1. Prescott, S., M. Ziccarelli , and E. Allen, 2021, "Investigating Application of Lidar For Nuclear Power Plants," INL/EXT-21-64452, Idaho National Laboratory.
2. Zhang, H., Y.J. Choi, C. Smith, S. Prescott, D. Mandelli, and A. Alfonsi, 2013, "Risk-Informed Systems Analysis (RISA) Pathway Technical Program Plan," INL-EXT-19-55721, Idaho National Laboratory.
3. Ei, 2021, "App," Accessed April 1, 2021, https://www.env-int.com/app.
4. He, K., X. Zhang, S. Ren, and J. Sun, 2015, “Deep residual learning for image recognition,” arXiv:1512.03385, Cornell University, https://doi.org/10.48550/arXiv.1512.03385.
5. Google, n.d., "Vision AI," Accessed June 13, 2022, https://cloud.google.com/vision.
6. Prescott, S., R. Christian, R. Sampath, J. Biersdorf, and S. Lawrence, 2021, "Industry Level Integrated Fire Modeling Using Fire Risk Investigation in 3D (FRI3D)," INL/EXT-21-604079, Idaho National Laboratory.
7. Salley, M. H. and R. Wachowiak, 2012, "Nuclear Power Plant Fire Modeling Analysis Guidelines (NPP FIRE MAG)," NUREG-1934, U.S. Nuclear Regulatory Commission.