

Plant Attribute Extraction

Rosendo Torres

USER INTERFACE

4 December 2022

Table of Contents

1. Subsystem Overview	4
1.1. Subsystem Description	4
1.2. Current State of the Subsystem	4
1.3. Future Improvements	4
2. Void model	4
2.1. Model Requirements	4-5
2.2 Code Structure	5-7
2.2.1 HTML and CSS	5-6
2.2.2 PHP	6-7
2.2.3 JS	7
3. Validation	8
3.1. Validating Multiple File Selection	8
3.1.3. Validation Results	8
3.2. Validation of User Friendliness	9
3.3. Validation of Warnings and Popups	9
3.3.3. Validation Results	10
Appendix A: Acronyms and Abbreviations	11
Appendix B: Definition of Terms	12
Appendix C: Important Links	13

List of Figures

Figure 1: Overview of Code Flow for Website User Interface	5
Figure 2: Dropdown Menus on Website	5
Figure 3: CSS Code for Checkboxes	6
Figure 4: HTML Implementation for Dropdown Menus	6
Figure 5: Orthomosaic.php file used for data population of dropdown menu	7
Figure 6: Platform function for dropdown menu	7
Figure 7: Dropdown menus displaying multiple file selection	8
Figure 8: Different warnings based on failure cases	9

1. Subsystem Overview

1.1. Subsystem Description

The subsystem contains code created in JS and HTML that deals with the display of the data onto the website based on different restrictions and file selections. The system should be user friendly and easy to use in order for the result generation to be as efficient as possible. This is done with many user warnings based on all possible mistakes the user can make and updating them with what issue has happened and how to fix it. I was tasked by the sponsor to have the ability to select multiple Orthomosaics and CHM files, filter the results based on selected files and improve the usability of the website.

1.2. Current State of the Subsystem

Currently all tasks assigned by the sponsor have been completed and the system is fully functional. The website is now super user friendly with many hints as to what files are supposed to be selected and a lot of warnings to let them know how to fix their errors. Before, some of the failure cases wouldn't alert the user at all. In addition, the user can select multiple Orthomosaics and CHM files to generate results from. My partner, Campbell Motter, added many extra features with the data/file manipulation with respect to what files can be selected and what attributes can be generated. He will explain those in his validation report. Depending on what Project, Platform, Sensor and Boundary files are then the respective Orthomosaics and CHM files are displayed.

1.3. Future Improvements

At the moment there is no progress bar or any way for the user to track the generation of results and downloading of files. They could implement this small feature in the future. The website may be improved upon in the user friendly aspect just in general because at the moment it is great but maybe some small features can be implemented.

2. Void model

2.1. Model Requirements

The code is made up of HTML, PHP, Python, and JS code that flows as shown in Figure 1. In Figure 1, the first flow diagram shows just a general overview of how the code links in order for the website to work. The second diagram gives an example of a specific task such as generating results on the website. The overall display of the

website is done in HTML in the index.php file. All the functionality with data generation, filtering and display is done in the JS, PHP and Python.

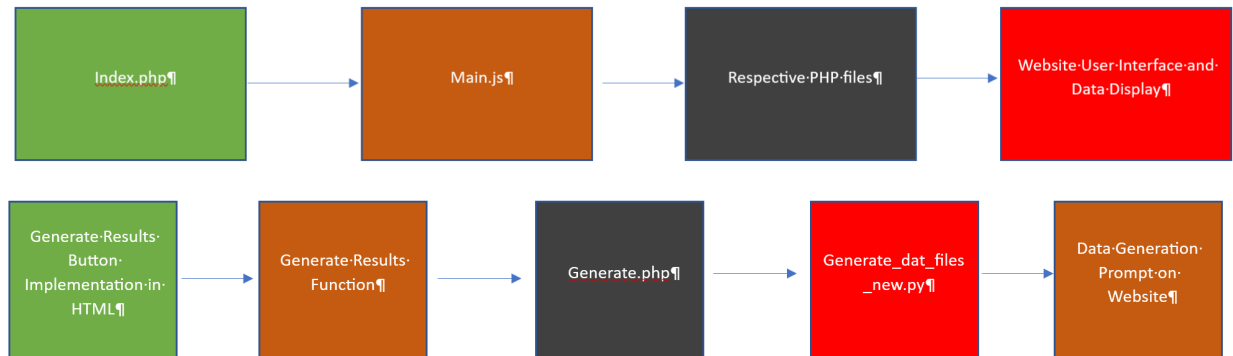


Figure 1: Overview of Code Flow for Website User Interface

2.2. Code Structure

2.2.1. HTML and CSS

For the general display of items on the website, HTML and CSS was used as shown in the figures below. Figure 3 shows CSS code for the checkbox implementation on the website. Figure 4 shows how the implementation is used in order for the dropdown menus to be displayed on the website. Figure 2 shows the menus on the website that the user sees.

<p>Orthomosaic</p> <div> --Select Orthomosaic File-- <div> <input type="checkbox"/> All <input type="checkbox"/> 20220523_cc_p4r_parking_mosaic.tif <input type="checkbox"/> 20220408_cc_p4r_parking_mosaic.tif <input type="checkbox"/> 20220427_cc_p4r_parking_mosaic.tif </div> </div>	<p>* Canopy Height Model</p> <div> --Select a Canopy Height Model File-- <div> <input type="checkbox"/> All <input type="checkbox"/> 20220523_cc_p4r_parking_chm.tif <input type="checkbox"/> 20220427_cc_p4r_parking_chm.tif <input type="checkbox"/> 20220408_cc_p4r_parking_chm.tif </div> </div>
---	--

Figure 2: Dropdown Menus on Website

```

114 .dropdown-check-list {
115     display: inline-block;
116 }
117
118 .dropdown-check-list .anchor {
119     position: relative;
120     cursor: pointer;
121     display: inline-block;
122     padding: 5px 145px 5px 10px;
123     border: 1px solid #ccc;
124     background: white;
125 }
126
127 .dropdown-check-list .anchor:after {
128     position: absolute;
129     content: "";
130     border-left: 2px solid black;
131     border-top: 2px solid black;
132     padding: 3px;
133     right: 10px;
134     top: 30%;
135     transform: rotate(225deg);
136 }
137
138 .dropdown-check-list .anchor:active:after {
139     right: 10px;
140     top: 30%;
141 }
142
143 .dropdown-check-list ul.items {
144     padding: 2px;
145     display: none;
146     margin: 0;
147     border: 1px solid #ccc;
148     border-top: none;
149 }
150
151 .dropdown-check-list.visible .items {
152     display: block;
153     background: white;
154 }

```

Figure 3: CSS Code for
Checkboxes

2.2.2. PHP

The PHP code is used to get the data from the database in order to populate and be able to filter out the data. Figure 5 shows the general structure of a PHP file used in order for data population of the Orthomosaic dropdown menu. All other PHP files have the same structure but change what restrictions and where they get data from. The PHP code is also used to conduct certain actions such as generating results or downloading files.

```

<label>Select Canopy Attributes to Generate:</label>
<div class="col-md-12" id="product-list-wrapper-0" style="margin-top: 15px; max-height: 230px; display: inline-block;">
<table id="product-table-0" class="table table-bordered">
<thead style="">
<tr style="color: white; background: black;">
<th style="border: none;">
<input id="check-all-caTable" class="checkbox" type="checkbox" checked="" onchange="ToggleAllRowData_caTable();" />
</th>
<th style="border: none;">--Select All--</th>
</tr>
</thead>
<tbody id="product-list-0">
<tr>
<td style="">
<input id="ch_cb" name="ch_cb" class="checkbox" type="checkbox" value="CanopyHeight" checked="" onchange="ToggleRowData_caTable();" />
</td>
<td style="">
<span>CanopyHeight</span>
</td>
</tr>
<tr>
<td style="">
<input id="cv_cb" name="cv_cb" class="checkbox" type="checkbox" value="CanopyVolume" checked="" onchange="ToggleRowData_caTable();" />
</td>
<td style="">
<span>CanopyVolume</span>
</td>
</tr>
<tr>
<td style="">
<input id="cc_cb" name="cc_cb" class="checkbox" type="checkbox" value="CanopyCover" checked="" onchange="ToggleRowData_caTable();" />
</td>
<td style="">
<span>CanopyCover</span>
</td>
</tr>
<tr>
<td style="">
<input id="evg_cb" name="evg_cb" class="checkbox" type="checkbox" value="Evg" checked="" onchange="ToggleRowData_caTable();" />
</td>
<td style="">
<span>Evg</span>
</td>
</tr>
</tbody>
</table>
</div>
</div>

```

Figure 4: HTML Implementation for Dropdown Menus

```

/*
Function - GetOrthomosaicList
Description - calls SQL query to fetch a list of Orthomosaic files from the database returning them to main.js in JSON format.
Parameters - con: connection to database
*/
function GetOrthomosaicList($con, $flight_id){
    $sql = "select * from imagery_product where type=1 AND Flight=$flight_id order by FileName"; //SQL query

    $result = mysqli_query($con,$sql); //call query to database

    $OrthomosaicList = array();
    while($row = mysqli_fetch_assoc($result)) { //While each row is returned from the database, add them to an array
        $OrthomosaicList[] = $row;
    }
    echo json_encode($OrthomosaicList); //Send Orthomosaic array in JSON format.
}

//require_once("SetDBConnection.php");
require_once("../resources/database/SetDBConnection.php");

$con = SetDBConnection(); //establish connection to database

if(mysqli_connect_errno())
{
    echo "Failed to connect to database server: ".mysqli_connect_error(); //return an error if we fail to establish a connection.
}
else
{
    $flight_id = $_GET["flightID"]; //get the Flight IDs in regard to what Project, Platform and Sensor is selected.
    GetOrthomosaicList($con, $flight_id); //call GetOrthomosaicList function.
}

mysqli_close($con); //close the connection
?>

```

Figure 5: Orthomosaic.php file used for data population of dropdown menu

2.2.3. JS

The JS is the bridge between the HTML and PHP codes in order for it to all work efficiently. It makes it so that the PHP is able to send the data acquired and set it up in the HTML so the website can be updated. Figure 6 shows the code for the platform menu but every other menu has the same general structure in order to populate the respective dropdown menu.

```

/*
Function - GetPlatformList
Description - makes AJAX call to Platform.PHP which returns Platform
Parameters - none
*/
function GetPlatformList() {
    $.ajax({
        url: "Resources/PHP/Platform.php",
        dataType: 'text',
        data: {
        },
        success: function (response) {
            var data = JSON.parse(response);
            if (data.length > 0) {
                $.each(data, function (index, item) {
                    var platform = "<option value='" + item.ID
                    + "'" + item.Name + "</option>";
                    $("#platform").append(platform);
                });
            }
        }
    });
}

```

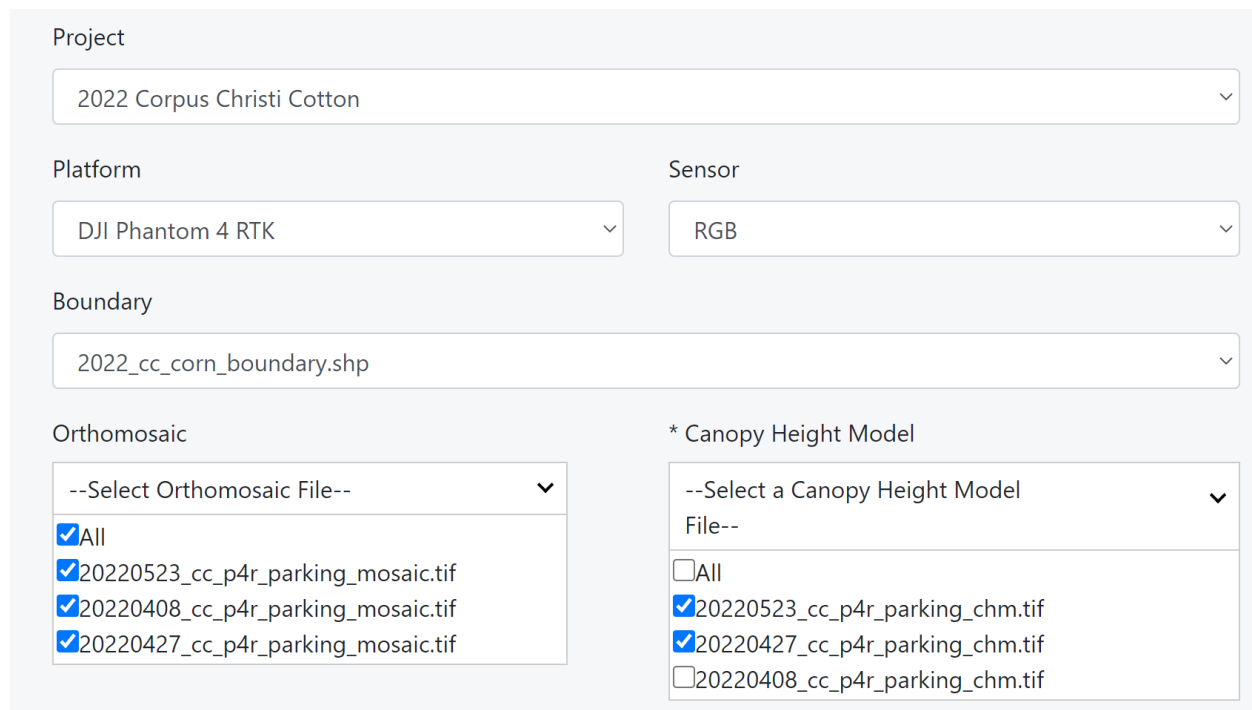
Figure 6: Platform function for dropdown menu

3. Validation

To validate the subsystem, the website had to show any visual updates that were implemented in the code. Not only this but the visual updates had to be correct based on what the sponsor asked for. For all areas of the subsystem, the code created for the project was in working order in conjunction with the code created by other members of the team.

3.1. Validating Multiple File Selection

The dropdown menus had to have the ability to display and be able to select multiple orthomosaic and CHM files at once for generation. Not only this but the names, IDs and file types of the data populated had to be correct in order for the generation to be successful.



The screenshot displays a web form with the following sections:

- Project:** A dropdown menu with the selected value "2022 Corpus Christi Cotton".
- Platform:** A dropdown menu with the selected value "DJI Phantom 4 RTK".
- Sensor:** A dropdown menu with the selected value "RGB".
- Boundary:** A dropdown menu with the selected value "2022_cc_corn_boundary.shp".
- Orthomosaic:** A dropdown menu with the selected value "--Select Orthomosaic File--". Below the dropdown, a list of files is shown with checkboxes:
 - ☒ All
 - ☒ 20220523_cc_p4r_parking_mosaic.tif
 - ☒ 20220408_cc_p4r_parking_mosaic.tif
 - ☒ 20220427_cc_p4r_parking_mosaic.tif
- * Canopy Height Model:** A dropdown menu with the selected value "--Select a Canopy Height Model File--". Below the dropdown, a list of files is shown with checkboxes:
 - ☐ All
 - ☒ 20220523_cc_p4r_parking_chm.tif
 - ☒ 20220427_cc_p4r_parking_chm.tif
 - ☐ 20220408_cc_p4r_parking_chm.tif

Figure 7: Dropdown menus displaying multiple file selection

3.1.3 Validation Results

In Figure 7 we see the dropdown menus working as expected with no issues. In addition, the user is able to select all or just a couple of wanted files for generation. The website also filters out the files based on what Project, Platform, Sensor and Boundary is selected.

3.2 Validation of User Friendliness

The next major requirement is to make sure anyone can use the website even if they are not technologically knowledgeable. The only possible way to validate this was to run tests by having my friends use the website with a rough vague description of what it is for. After a few tests the website was updated based on what my friends had issues with while running through it. There is no way to show results for this section but the updated website has been updated based on the results of the tests.

3.3 Validation of Warnings and Popups

The validation of this section was just going through the website and thinking of every possible error a user can make and making sure an appropriate warning lets the user know what is happening. At the moment there is a warning for every possible failure case and not only that the warning lets the user know how to fix their issue. The warnings work correctly and don't overload the user or stack together.

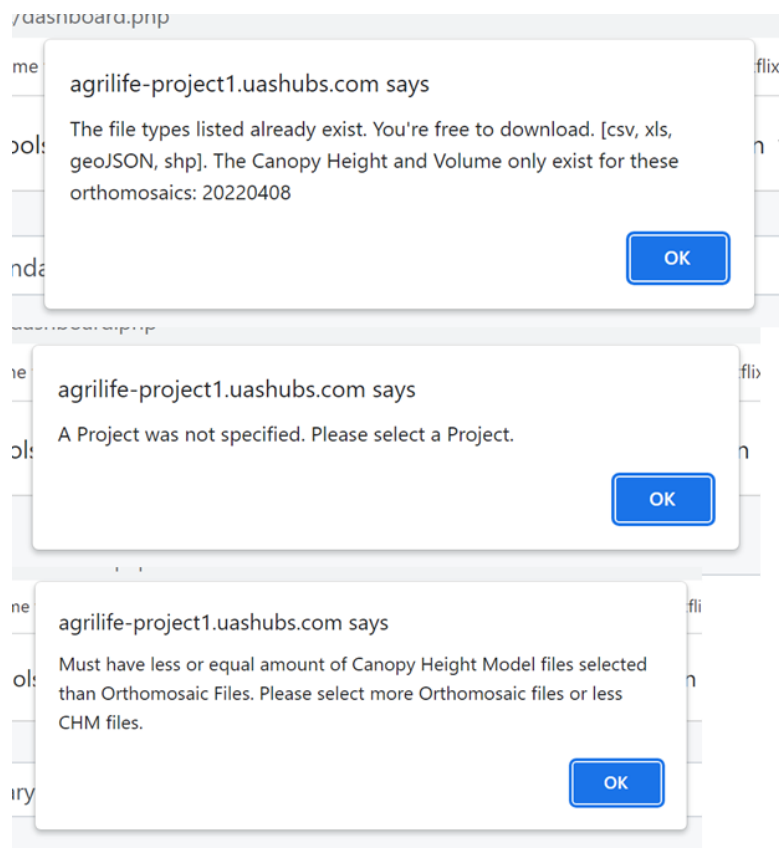


Figure 8: Different warnings based on failure cases

3.3.3 Validation Results

Figure 8 shows just some examples of specific fail cases that are possible with the generation of results. The first popup isn't an error warning but a helpful popup that lets the user know what files specifically are already generated from the ones they have selected for generation. The middle warning is implemented for every dropdown menu and it happens when

there is no selection chosen for the respective dropdown menu. The last warning is a special case that came to exist when we added more features to the website. In this case, you cannot have more CHM files than Orthomosaics because in order to generate results for a specific date you need a matching CHM file.

Appendix A: Acronyms and Abbreviations

JS	JavaScript
PHP	Hypertext Preprocessor
HTML	HyperText Markup Language
CHM	Canopy Height Model
CV	Canopy Volume
CC	Canopy Cover
Exg	Excess Greenness
CH	Canopy Height
CSS	Cascading Style Sheets

Appendix B: Definition of Terms

- Orthomosaic: An image with a high resolution that combines miniature photos to create the eventual image

Appendix C: Important Links

- GitHub Link: https://github.com/RonBatista/capstone_fall_22