Senior Project Final Presentation
April 18, 2019

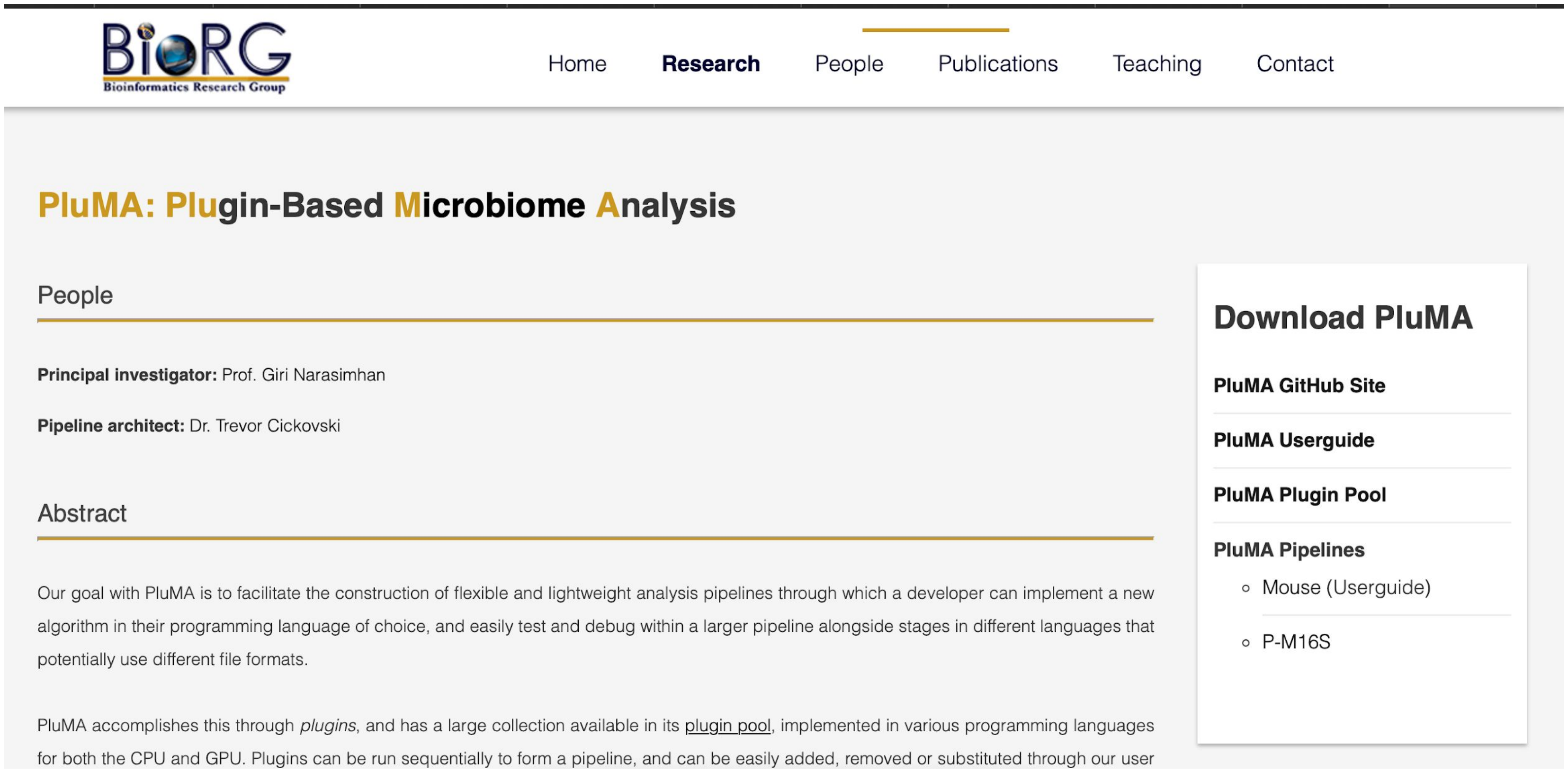
Spring 2019

GUI for PluMA: Plugin-Based Microbiome Analysis

School of Computing and Information Sciences
Florida International University

Team Members:
Rishabh Vaidya
Cesia Bulnes
Bhavyta Chauhan
Product Owner:
Trevor Cickovski
Professor:
Masoud Sadjadi

What it is currently



BioRG Bioinformatics Research Group

Home Research People Publications Teaching Contact

PluMA: Plugin-Based Microbiome Analysis

People

Principal investigator: Prof. Giri Narasimhan

Pipeline architect: Dr. Trevor Cickovski

Abstract

Our goal with PluMA is to facilitate the construction of flexible and lightweight analysis pipelines through which a developer can implement a new algorithm in their programming language of choice, and easily test and debug within a larger pipeline alongside stages in different languages that potentially use different file formats.

PluMA accomplishes this through *plugins*, and has a large collection available in its plugin pool, implemented in various programming languages for both the CPU and GPU. Plugins can be run sequentially to form a pipeline, and can be easily added, removed or substituted through our user

Download PluMA

PluMA GitHub Site

PluMA Userguide

PluMA Plugin Pool

PluMA Pipelines

- Mouse (Userguide)
- P-M16S

Problem definition

- Whole Project:
  - PluMA facilitates the construction of flexible analysis pipelines through which a developer can implement a new algorithm in their programming language of choice. Our goal with PluMA-GUI is to create a user-friendly experience when using PluMA.

- Team Members' Individual Roles:
  - Cesia's Part: I implemented the plugin folder display, the webscraping of the Biorg PluMA plugin pool and the installation of plugins in the background
  - Rishabh's Part: I implemented the drag and drop feature, the save feature to generate the config file, and the remove plugins feature.
  - Bhavyta's Part: I implemented the upload file feature, a ReadME parser, and designed and incorporated the arrows displayed between plugins in the pipeline.

Requirements: User Stories

- List of the user stories we worked on:
    - User Story 1: As a developer, I want to ensure that the electron app is cross-platform
    - User Story 2 & 3: Build a visual prototype
    - User Story 4, 5, & 6: Review frameworks and technologies
    - User Story 7, 8 & 9: Setup static front page
    - User Story 10: As a user, I want to be able to access plugins from the plugins folder in order to setup pipelines
    - User Story 11: As a user, I want to be able to drag and drop the plugins in order to assemble a pipeline
    - User Story 12: As a user, I want to be able to send input and output file names to each plugin to assemble a pipeline
    - User Story 13: As a user, I want to be able to save the config file in order to be able to run PluMA
    - User Story 14: As a developer, I want to be able to code in a main page
    - User Story 15: As a user, I want to be able to choose input/output files and add them to the pipeline
    - User Story 16: As a user, I should be able to drag and drop installed plugins into the pipeline
    - User Story 17: As a developer, I want to implement a function to parse through the readme file of a plugin

Requirements: User Stories

- User Stories
  - User Story 18: As a developer, I want to restrict the input/output file types that a user can upload so that the correct file types are used for each plugin
  - User Story 19: As a user, I want to be able to view the input file type, plugin name, output file type
  - User Story 20: As a user, I want to be able to click on a button to access the plugin pool
  - User Story 21: As a user, I should able to view all plugins from the pool in a popup window
  - User Story 22: As a developer, I want to modify the background and add a logo to the page
  - User Story 23: As a developer, I want to save temporary input/output files to the config file
  - User Story 24: As a user, I want to git clone the repo of a selected plugin inside my plugin folder in the background of my web scraping
  - User Story 25: As a user, I want to be able to collaborate my feature with the other features to ensure proper functionality
  - User Story 26: As a developer, I want to be able to cooperate with my team member to send input/output files to each plugin
  - User Story 27: As a user, I want to see arrows connecting each plugin in an assembled pipeline so that the visual structure better communicates how the pipeline works

Requirements: User Stories

- User Stories
    - User Story 28: As a developer, I want to have placeholder files for those plugins that don’t require user to upload input/output files so that it is easier to generate the config file
    - User Story 29: As a user, I want to be able to drop unwanted plugins in the trash
    - User Story 30: As a developer, I want to be able to hash random values for temporary file names
    - User Story 31: As a user I want to be able to see a warning box when i want to download and install a new plugin in C++ or Cuda, since I would have to recompile through Pluma!
    - User Story 32: As a user I want to be able to add or delete boxes in order to drag plugins as much as I want

- Prioritized User Stories
    - Rishabh: User Story #13, #16, #29

      As a user, I want to be able to save the config file in order to be able to run PluMA
      As a user, I should be able to drag and drop installed plugins into the pipeline

      As a user, I want to be able to drop unwanted plugins in the trash
    - Cesia: User Story #19, #27 & #30

      As a user, I want to be able to view the input file type, plugin name, output file type
      As a user, I should able to view all plugins from the pool in a popup window
      As a user, I want to git clone the repo of a selected plugin inside my plugin folder in the background of my webscraping.

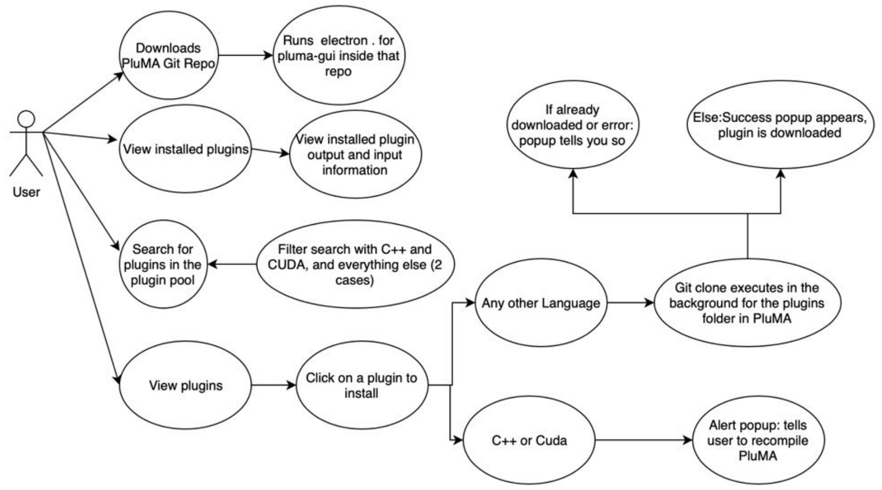
Requirements: Use Cases

- Cesia's Use Case Diagram

User Story Diagram for User Story #27 & #30

- As a user, I should able to view all plugins from the pool in a popup window
- As a user, I want to git clone the repo of a selected plugin inside my plugin folder in the background of my webscraping.

USER STORY DIAGRAM

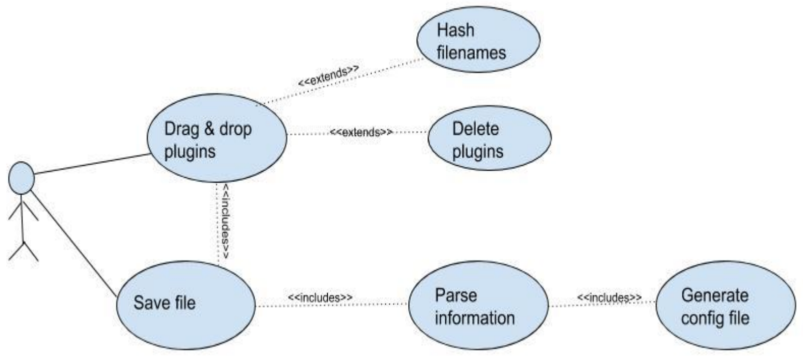

User
Downloads PluMA Git Repo
Runs electron . for pluma-gui inside that repo
View installed plugins
View installed plugin output and input information
Search for plugins in the plugin pool
Filter search with C++ and CUDA, and everything else (2 cases)
View plugins
Click on a plugin to install
Any other Language
Git clone executes in the background for the plugins folder in PluMA
If already downloaded or error: popup tells you so
Else:Success popup appears, plugin is downloaded
C++ or Cuda
Alert popup: tells user to recompile PluMA

- My most important user stories are displayed. I had to parse the information from the README files in order to display the plugins in the 'View installed plugins' table, i also displayed the input and output for each plugin.

  I also was able to scrape the current BioOrg plugin pool into our GUI, and from there, when users would click on any given plugin, it would be installed using child processes to script in the background.

Requirements: Use Cases

- Rishabh's Use Case Diagram

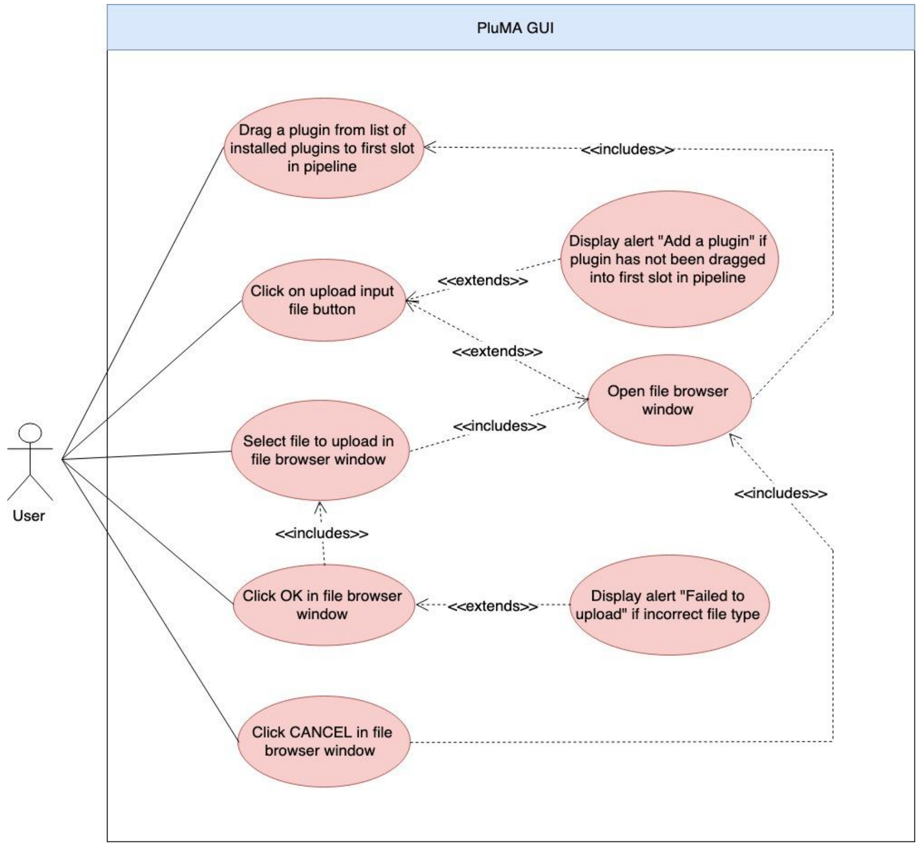


- The most important use cases for my part are dragging and dropping the plugins, parsing the plugin names in order to save the information, and generating the config file. I was able to allow the user to drag and drop plugins into the pipeline.
- As soon as a plugin is dropped, temporary hashed output filenames are generated. Once all plugins are added, the user can save the information and generate a config.txt file.

- The most significant use case is saving the config file because PluMA cannot execute if there is no config file present

Requirements: Use Cases

- Bhavyta’s Use Case Diagram

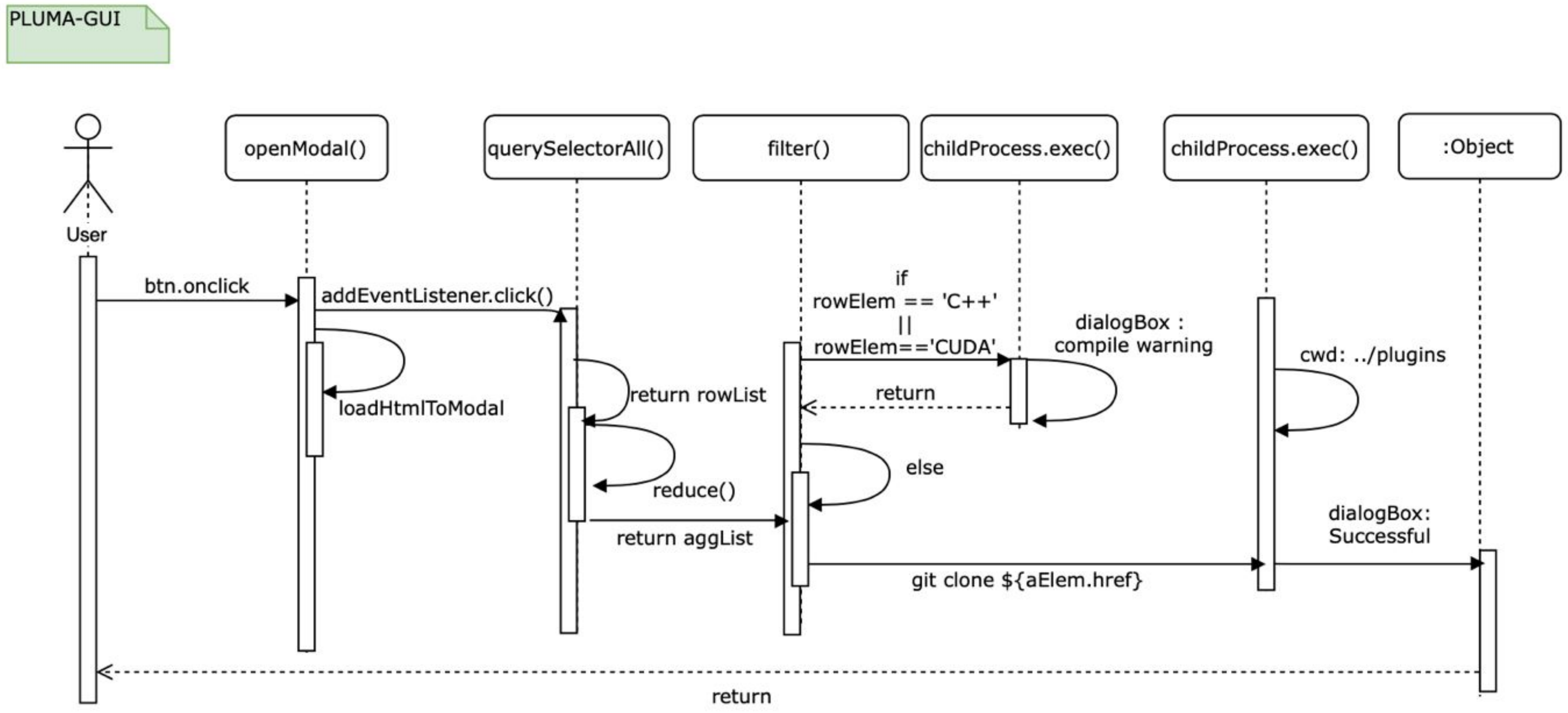

PluMA GUI
User
Drag a plugin from list of installed plugins to first slot in pipeline
Display alert "Add a plugin" if plugin has not been dragged into first slot in pipeline
Click on upload input file button
Open file browser window
Select file to upload in file browser window
Click OK in file browser window
Display alert "Failed to upload" if incorrect file type
Click CANCEL in file browser window
<<includes>>
<<extends>>
<<extends>>
<<includes>>
<<includes>>
<<includes>>
<<extends>>

- Based on the current implementation of the interface, the user only needs to upload an input file for the first plugin and an output file for the last plugin.
- The user may not upload a file unless the associated plugin has been dragged to the respective boxed area. An alert is displayed if the user tries to do so.
- After adding the desired plugins and clicking on the upload file button, a local file browser window is displayed from where the user can select the desired file for upload.

- If the type (extension) of the chosen file is incorrect, the upload is canceled and an alert is displayed informing the user that the upload failed due to incorrect file type and the correct file type that is allowed for upload. If the upload is successful, the name of the uploaded file appears under the button.

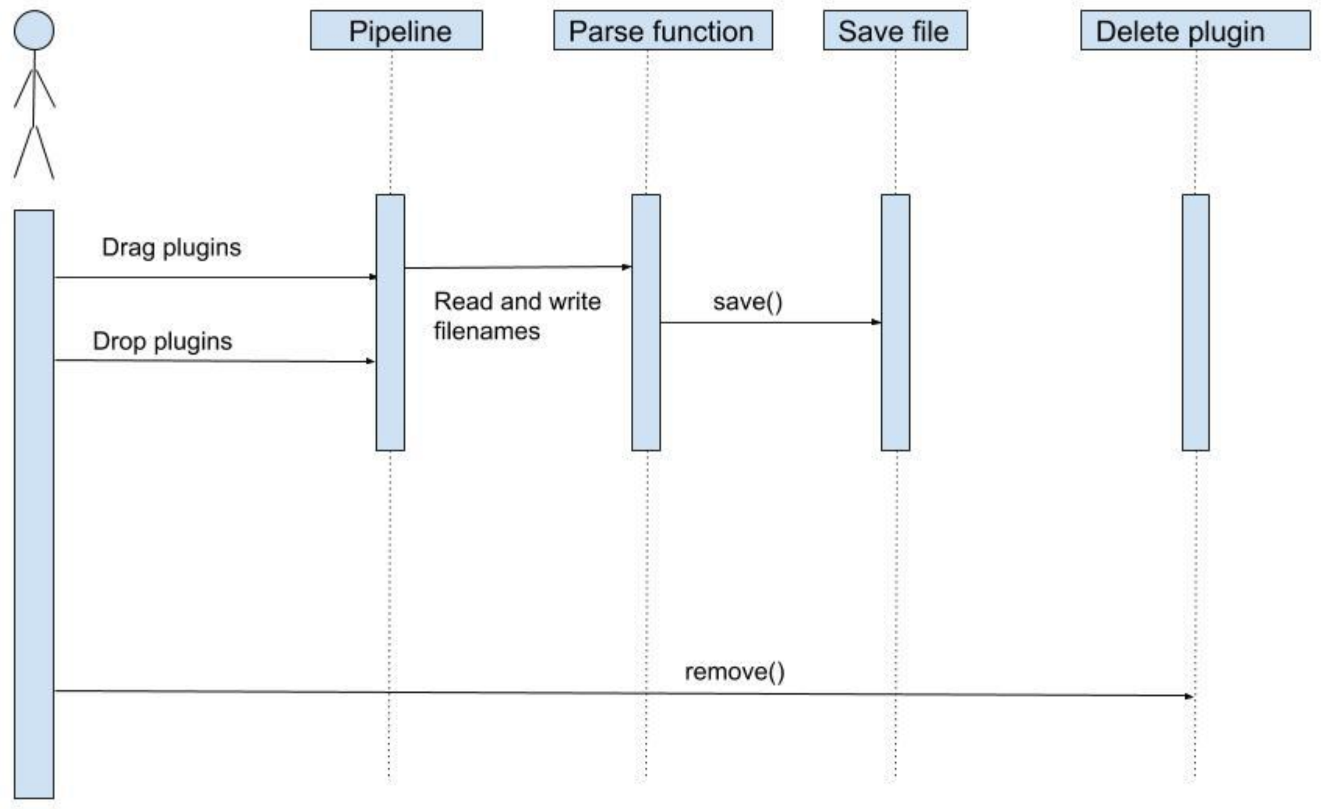
Requirements: Sequence Diagrams

- Cesia's Sequence Diagram:



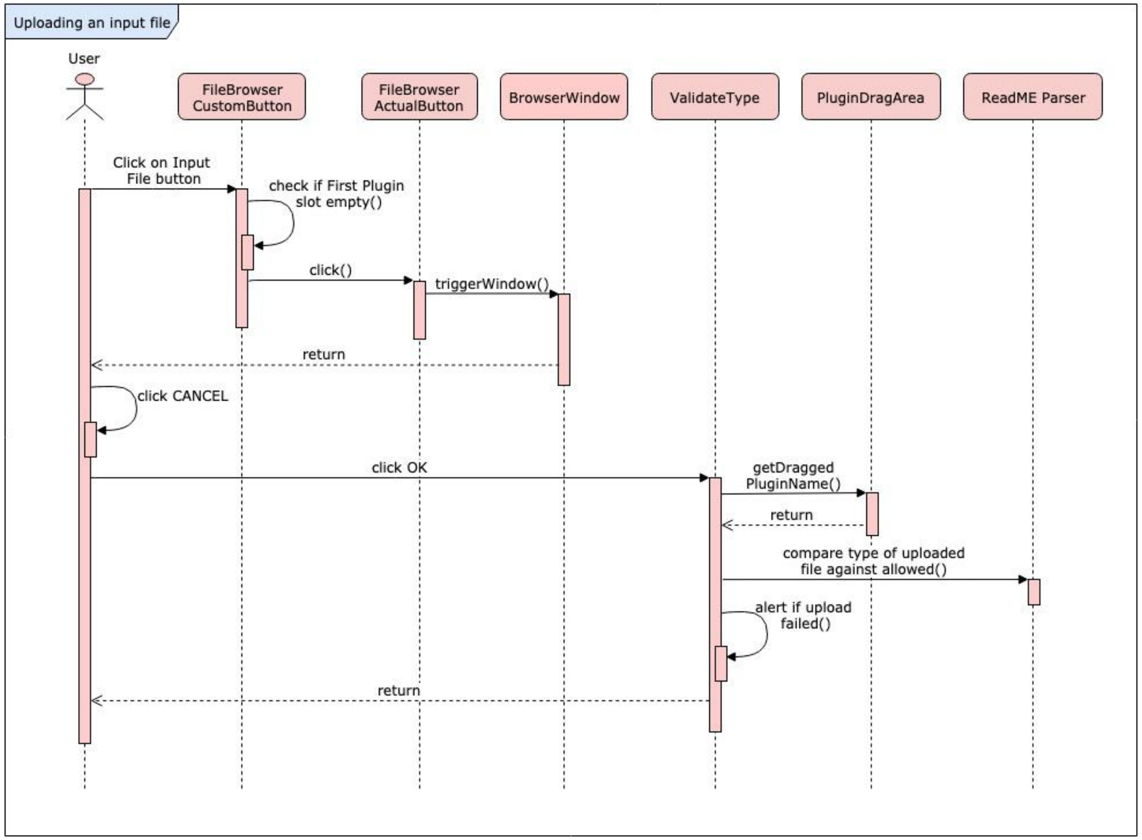
Requirements: Sequence Diagrams

- Rishabh’s Sequence Diagram:

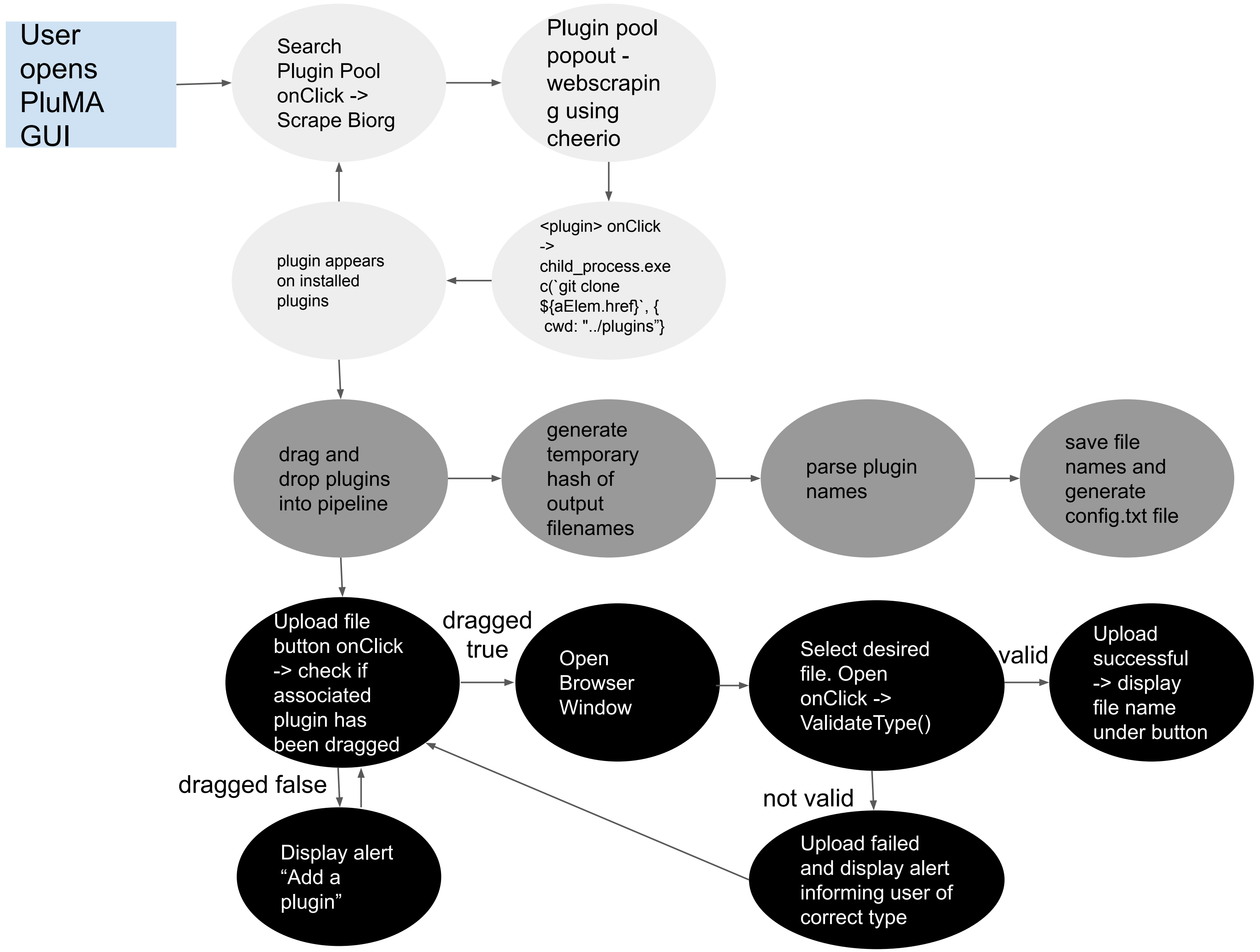


Requirements: Sequence Diagrams

- Bhavyta’s Sequence Diagram

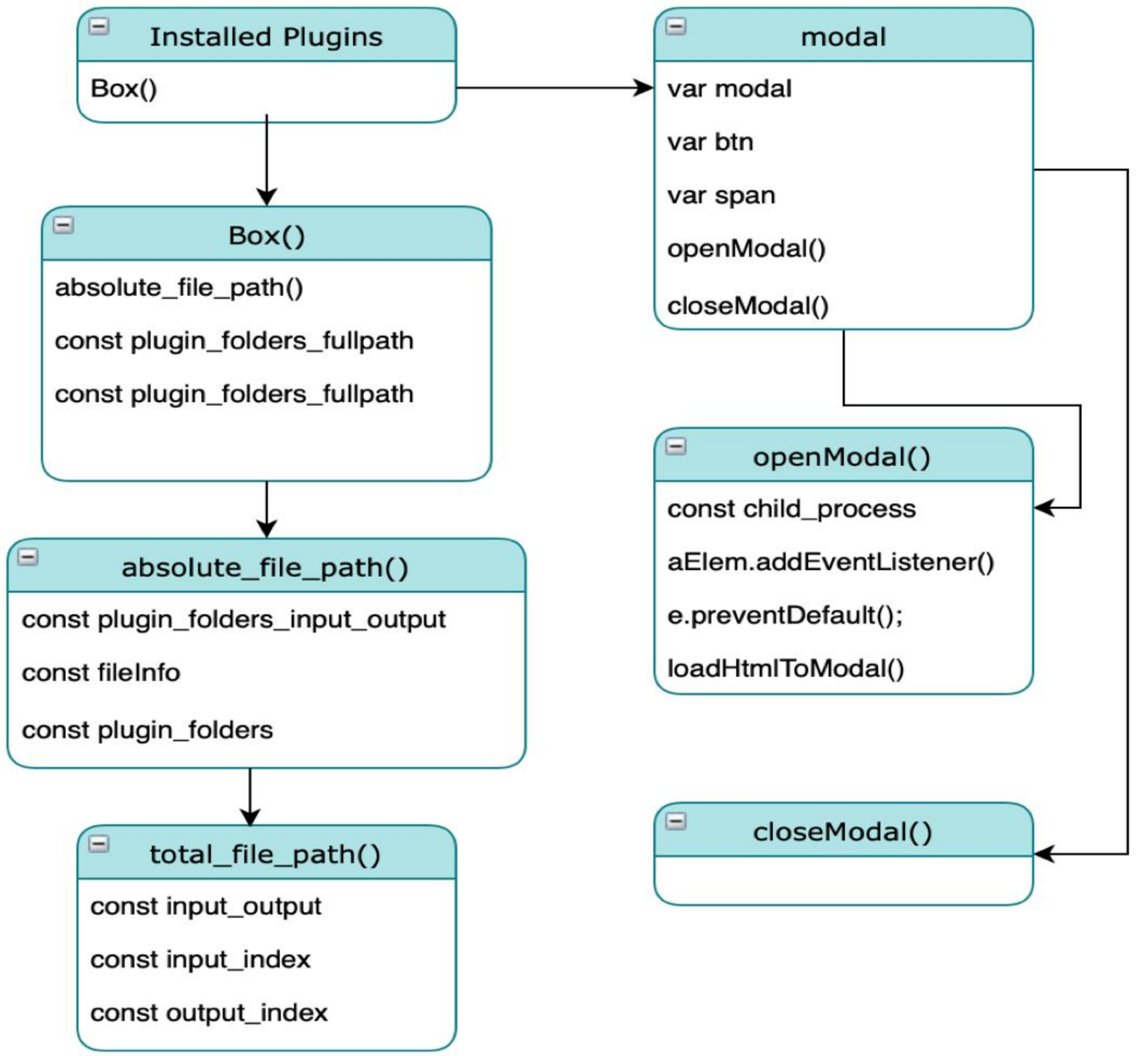


System Design


User opens PluMA GUI
Search Plugin Pool onClick -> Scrape Biorg
Plugin pool popout - webscraping using cheerio
<plugin> onClick -> child_process.exec(`git clone ${aElem.href}`, { cwd: "../plugins"}
plugin appears on installed plugins
drag and drop plugins into pipeline
generate temporary hash of output filenames
parse plugin names
save file names and generate config.txt file
Upload file button onClick -> check if associated plugin has been dragged
dragged true
Open Browser Window
Select desired file. Open onClick -> ValidateType()
valid
Upload successful -> display file name under button
dragged false
Display alert "Add a plugin"
not valid
Upload failed and display alert informing user of correct type

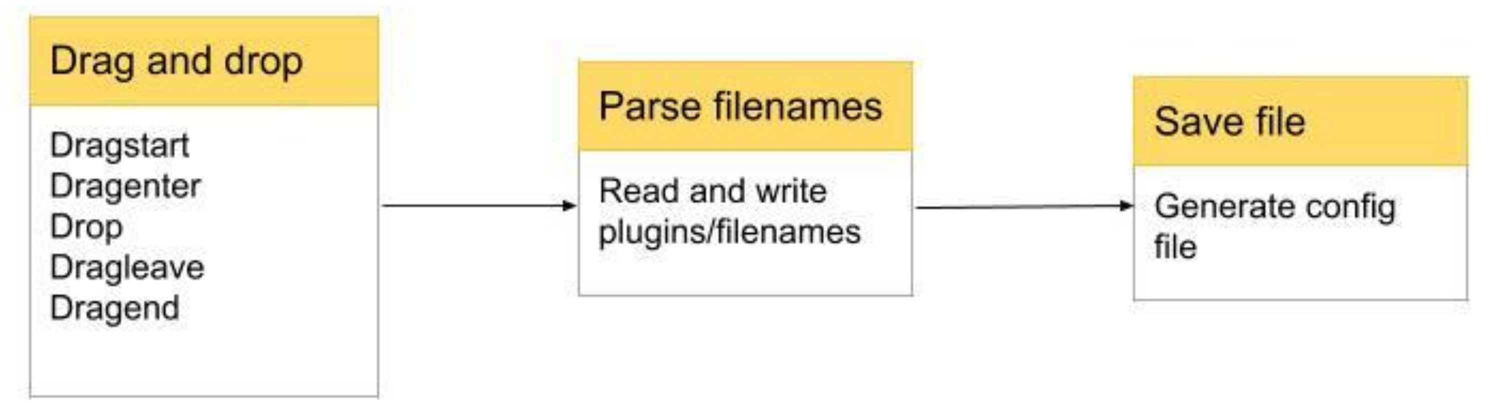
Minimal Class Diagram

- Cesia Bulnes



Minimal Class Diagram

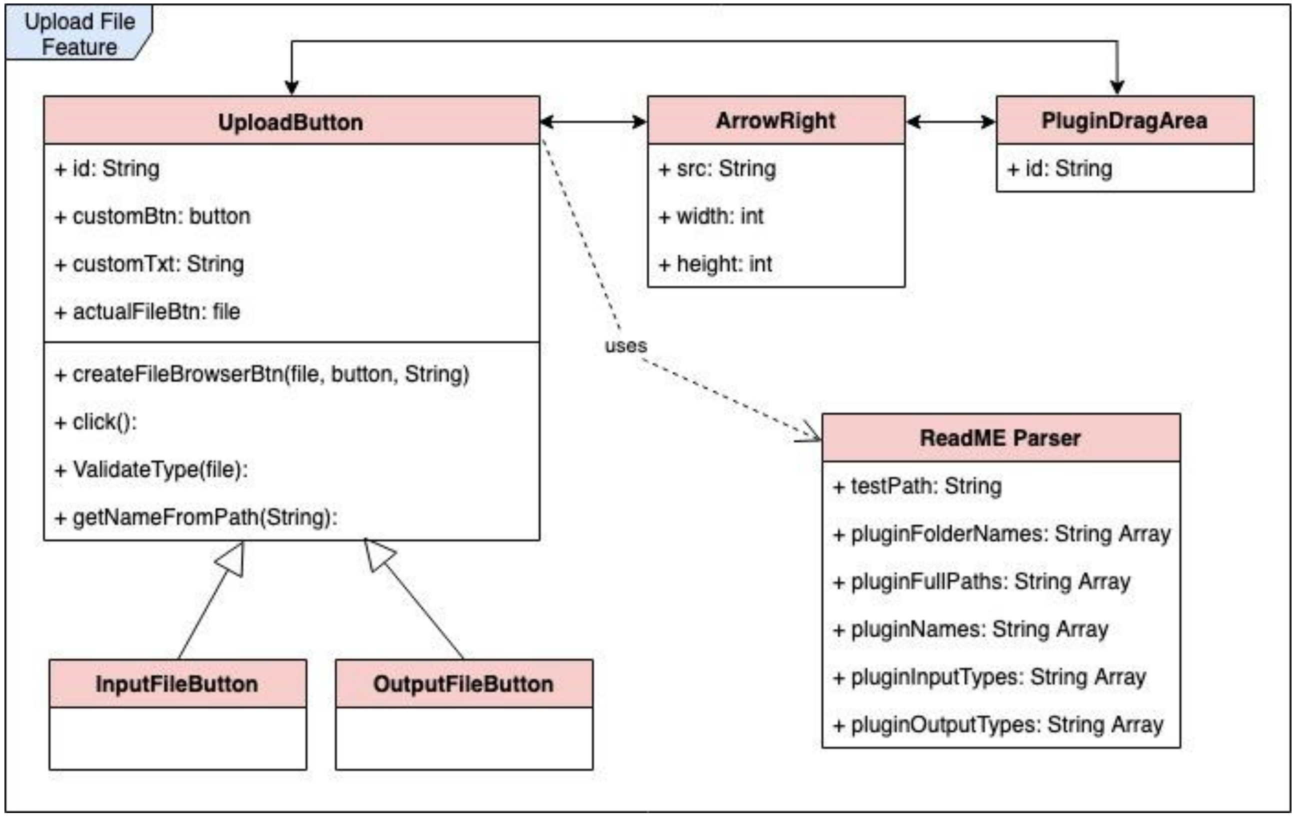
- Rishabh Vaidya



The class diagram shows the process the program goes through when building a pipeline. It also demonstrates how the config.txt file is generated.

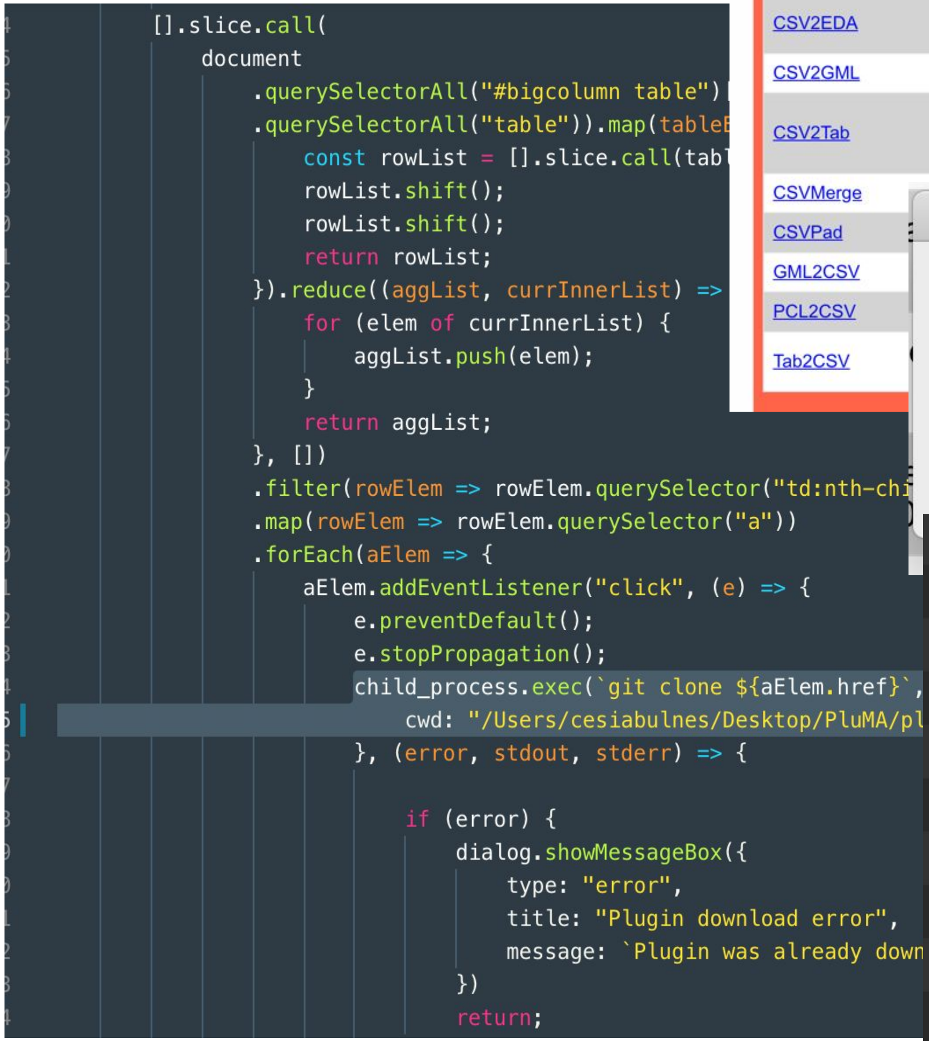
Minimal Class Diagram

- Bhavyta Chauhan

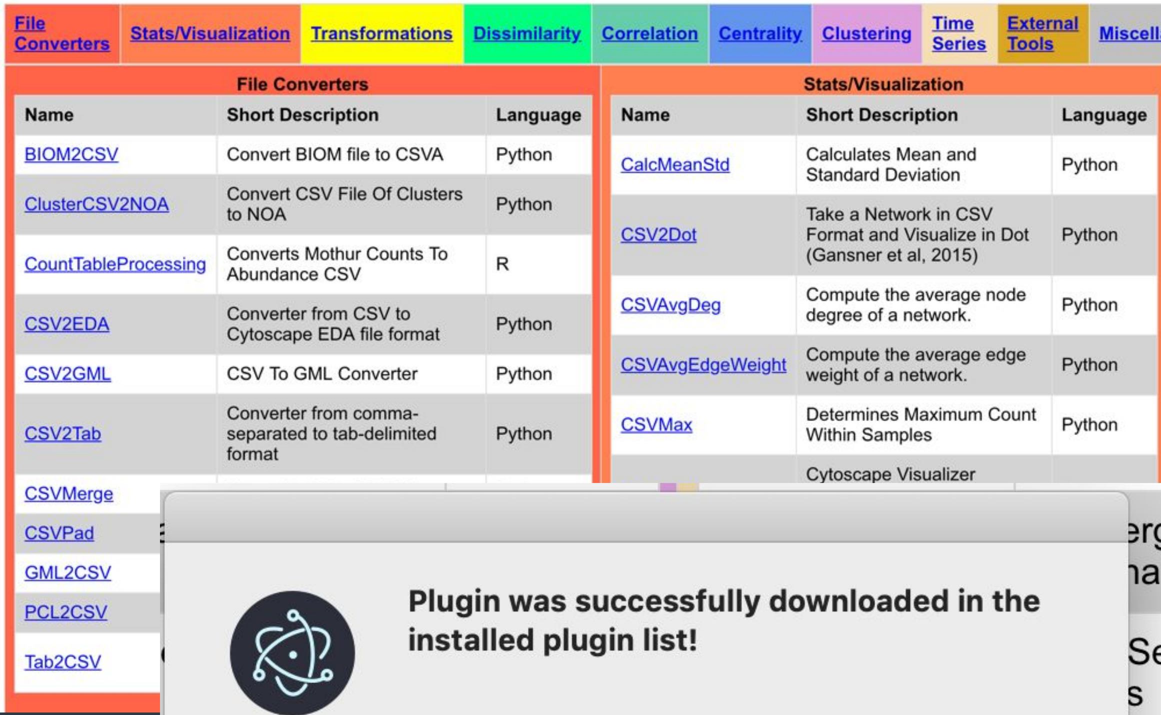


Main algorithm

- Cesia's feature:



```
[].slice.call(
    document
        .querySelectorAll("#bigcolumn table")
        .querySelectorAll("table")).map(tableE
            const rowList = [].slice.call(tabl
            rowList.shift();
            rowList.shift();
            return rowList;
        }).reduce((aggList, currInnerList) =>
            for (elem of currInnerList) {
                aggList.push(elem);
            }
            return aggList;
        }, [])
        .filter(rowElem => rowElem.querySelector("td:nth-chi
        .map(rowElem => rowElem.querySelector("a"))
        .forEach(aElem => {
            aElem.addEventListener("click", (e) => {
                e.preventDefault();
                e.stopPropagation();
                child_process.exec(`git clone ${aElem.href}`,
                    cwd: "/Users/cesiabulnes/Desktop/PluMA/pl
                }, (error, stdout, stderr) => {

                    if (error) {
                        dialog.showMessageBox({
                            type: "error",
                            title: "Plugin download error",
                            message: `Plugin was already down
                        })
                        return;
```

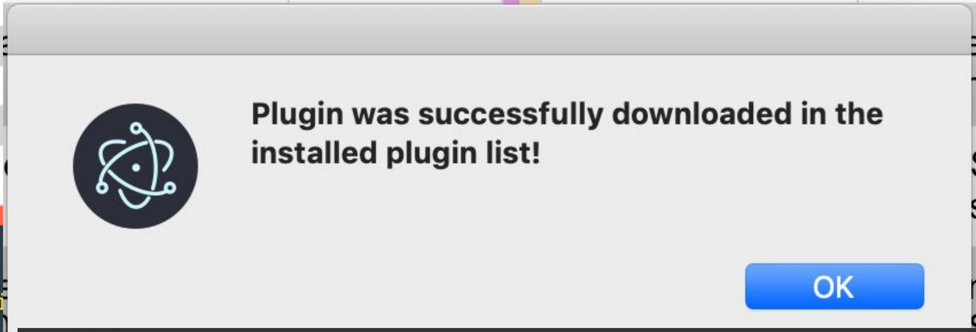


File Converters

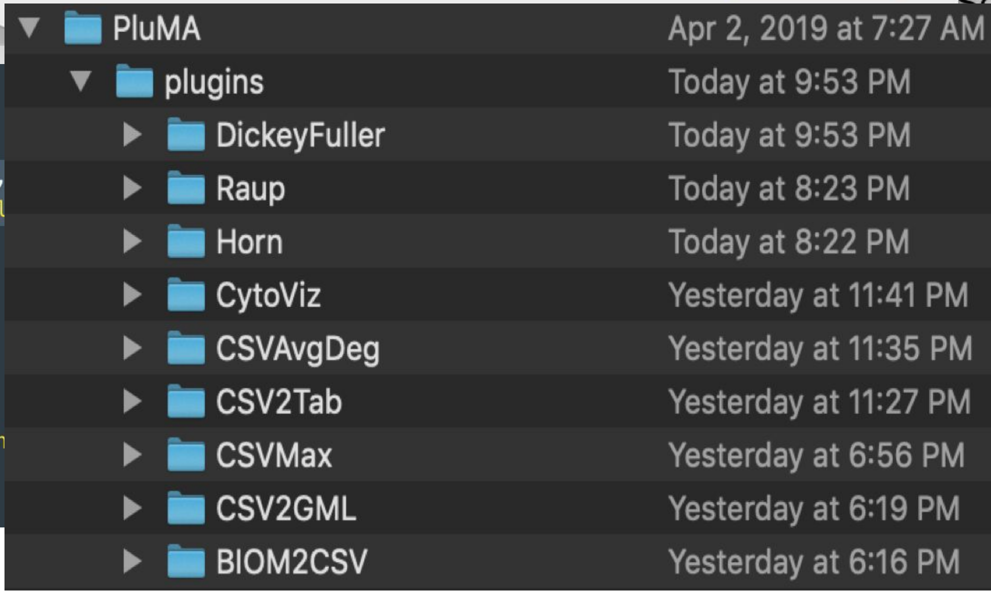
Name	Short Description	Language
BIOM2CSV	Convert BIOM file to CSVA	Python
ClusterCSV2NOA	Convert CSV File Of Clusters to NOA	Python
CountTableProcessing	Converts Mothur Counts To Abundance CSV	R
CSV2EDA	Converter from CSV to Cytoscape EDA file format	Python
CSV2GML	CSV To GML Converter	Python
CSV2Tab	Converter from comma-separated to tab-delimited format	Python
CSVMerge		
CSVPad		
GML2CSV		
PCL2CSV		
Tab2CSV		

Stats/Visualization

Name	Short Description	Language
CalcMeanStd	Calculates Mean and Standard Deviation	Python
CSV2Dot	Take a Network in CSV Format and Visualize in Dot (Gansner et al, 2015)	Python
CSVAvgDeg	Compute the average node degree of a network.	Python
CSVAvgEdgeWeight	Compute the average edge weight of a network.	Python
CSVMax	Determines Maximum Count Within Samples	Python
	Cytoscape Visualizer	



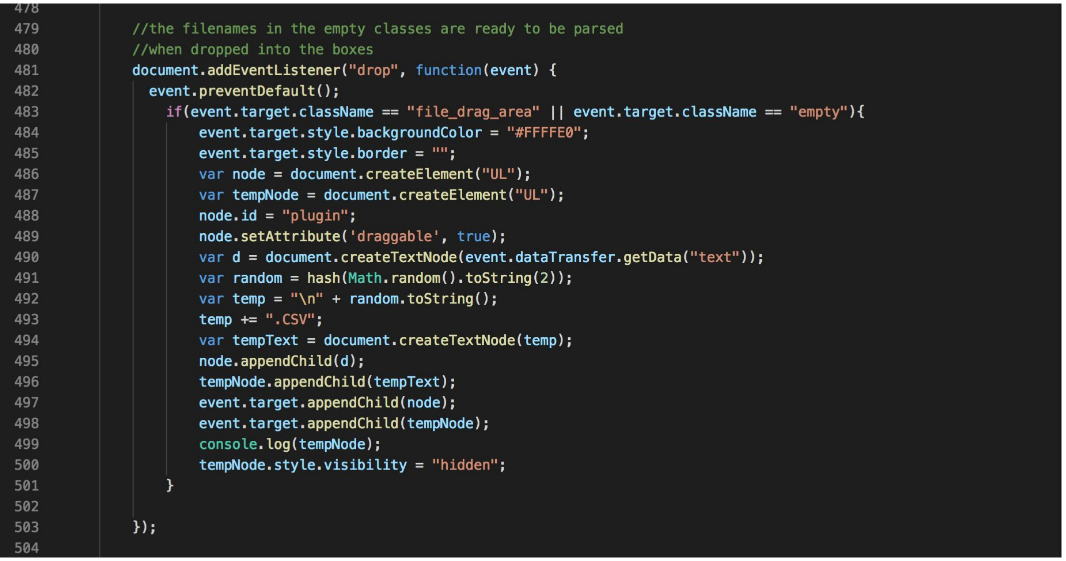
Plugin was successfully downloaded in the installed plugin list!



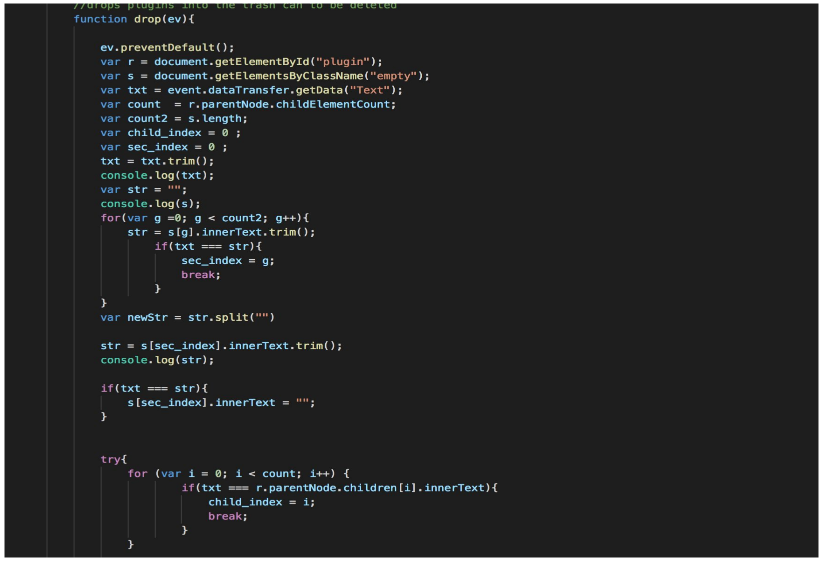
▼ PluMA	Apr 2, 2019 at 7:27 AM
▼ plugins	Today at 9:53 PM
▶ DickeyFuller	Today at 9:53 PM
▶ Raup	Today at 8:23 PM
▶ Horn	Today at 8:22 PM
▶ CytoViz	Yesterday at 11:41 PM
▶ CSVAvgDeg	Yesterday at 11:35 PM
▶ CSV2Tab	Yesterday at 11:27 PM
▶ CSVMax	Yesterday at 6:56 PM
▶ CSV2GML	Yesterday at 6:19 PM
▶ BIOM2CSV	Yesterday at 6:16 PM

Main algorithm

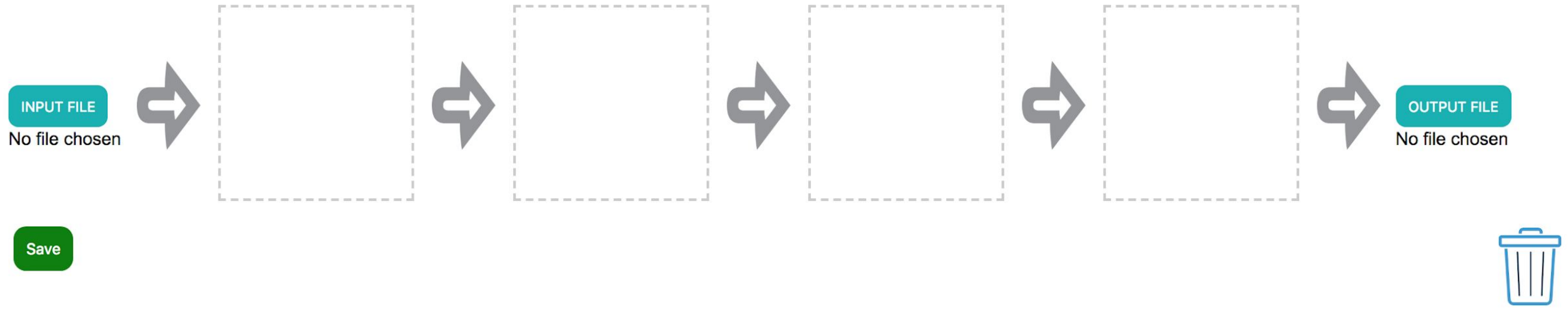
- Rishabh’s feature:



```
//the filenames in the empty classes are ready to be parsed
//when dropped into the boxes
document.addEventListener("drop", function(event) {
  event.preventDefault();
    if(event.target.className == "file_drag_area" || event.target.className == "empty"){
        event.target.style.backgroundColor = "#FFFFE0";
        event.target.style.border = "";
        var node = document.createElement("UL");
        var tempNode = document.createElement("UL");
        node.id = "plugin";
        node.setAttribute('draggable', true);
        var d = document.createTextNode(event.dataTransfer.getData("text"));
        var random = hash(Math.random().toString(2));
        var temp = "\n" + random.toString();
        temp += ".CSV";
        var tempText = document.createTextNode(temp);
        node.appendChild(d);
        tempNode.appendChild(tempText);
        event.target.appendChild(node);
        event.target.appendChild(tempNode);
        console.log(tempNode);
        tempNode.style.visibility = "hidden";
    }

});
```

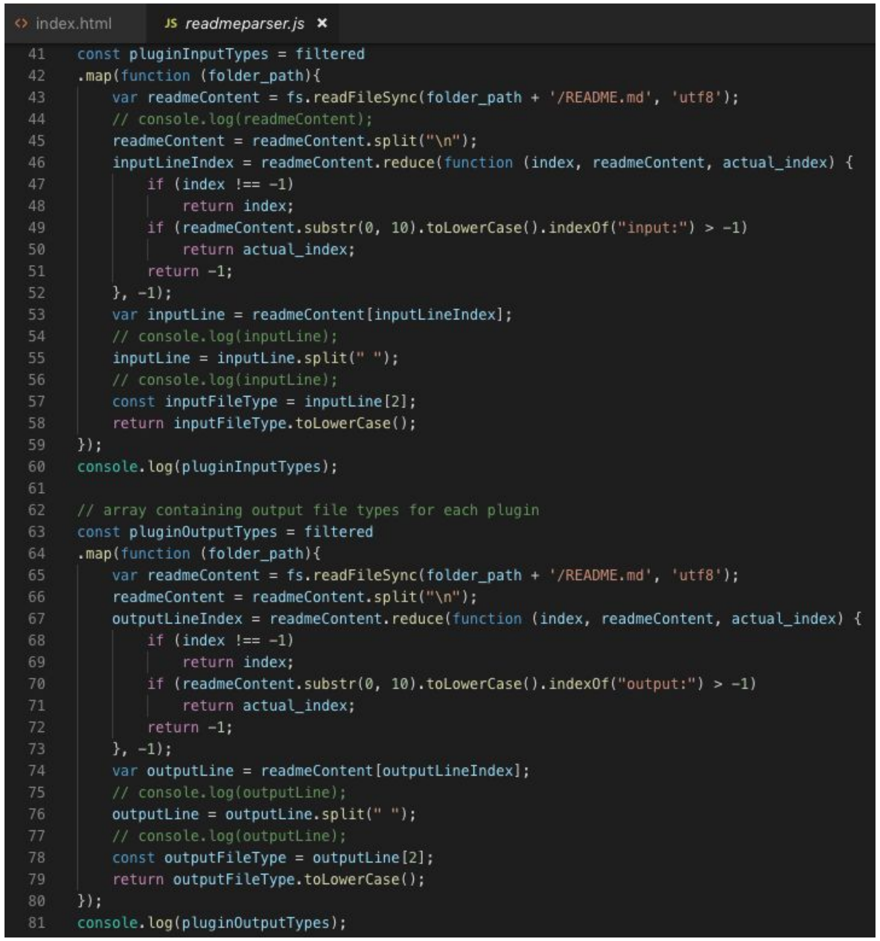


```
//drops plugins into the trash can to be deleted
function drop(ev){

    ev.preventDefault();
    var r = document.getElementById("plugin");
    var s = document.getElementsByClassName("empty");
    var txt = event.dataTransfer.getData("Text");
    var count  = r.parentNode.childElementCount;
    var count2 = s.length;
    var child_index = 0 ;
    var sec_index = 0 ;
    txt = txt.trim();
    console.log(txt);
    var str = "";
    console.log(s);
    for(var g =0; g < count2; g++){
        str = s[g].innerText.trim();
            if(txt === str){
                sec_index = g;
                break;
            }
    }
    var newStr = str.split("")

    str = s[sec_index].innerText.trim();
    console.log(str);

    if(txt === str){
        s[sec_index].innerText = "";
    }


    try{
        for (var i = 0; i < count; i++) {
                if(txt === r.parentNode.children[i].innerText){
                    child_index = i;
                    break;
                }
        }
```

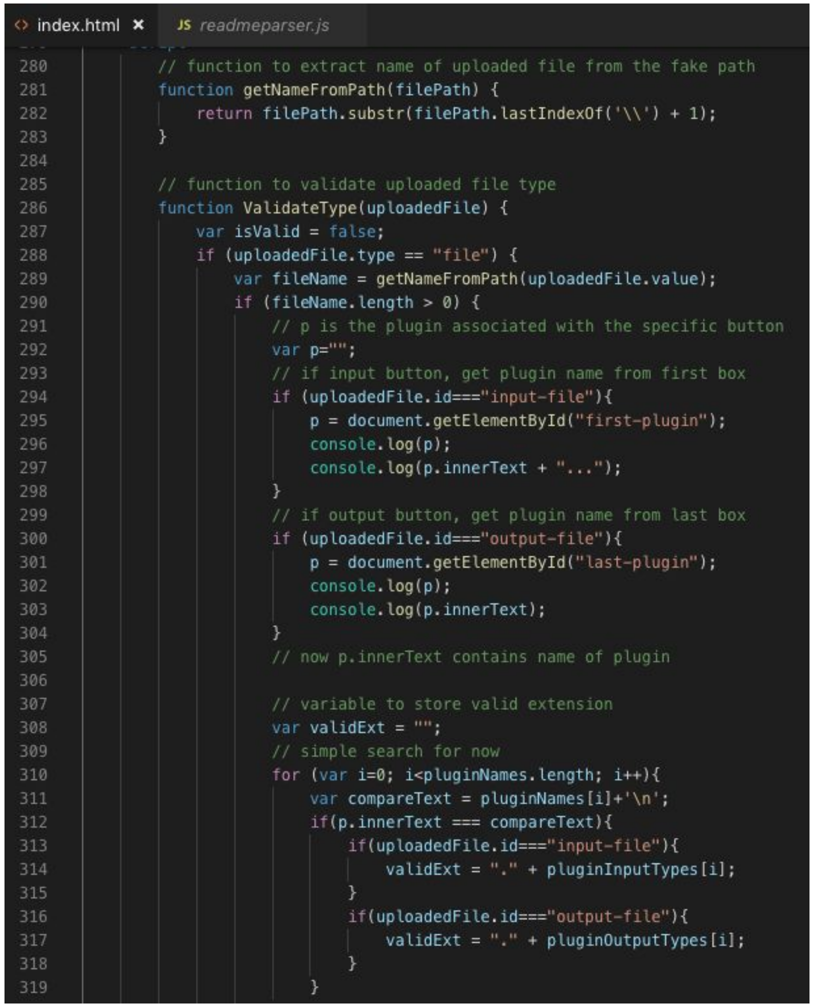


Main algorithm

- Bhavyta’s feature:



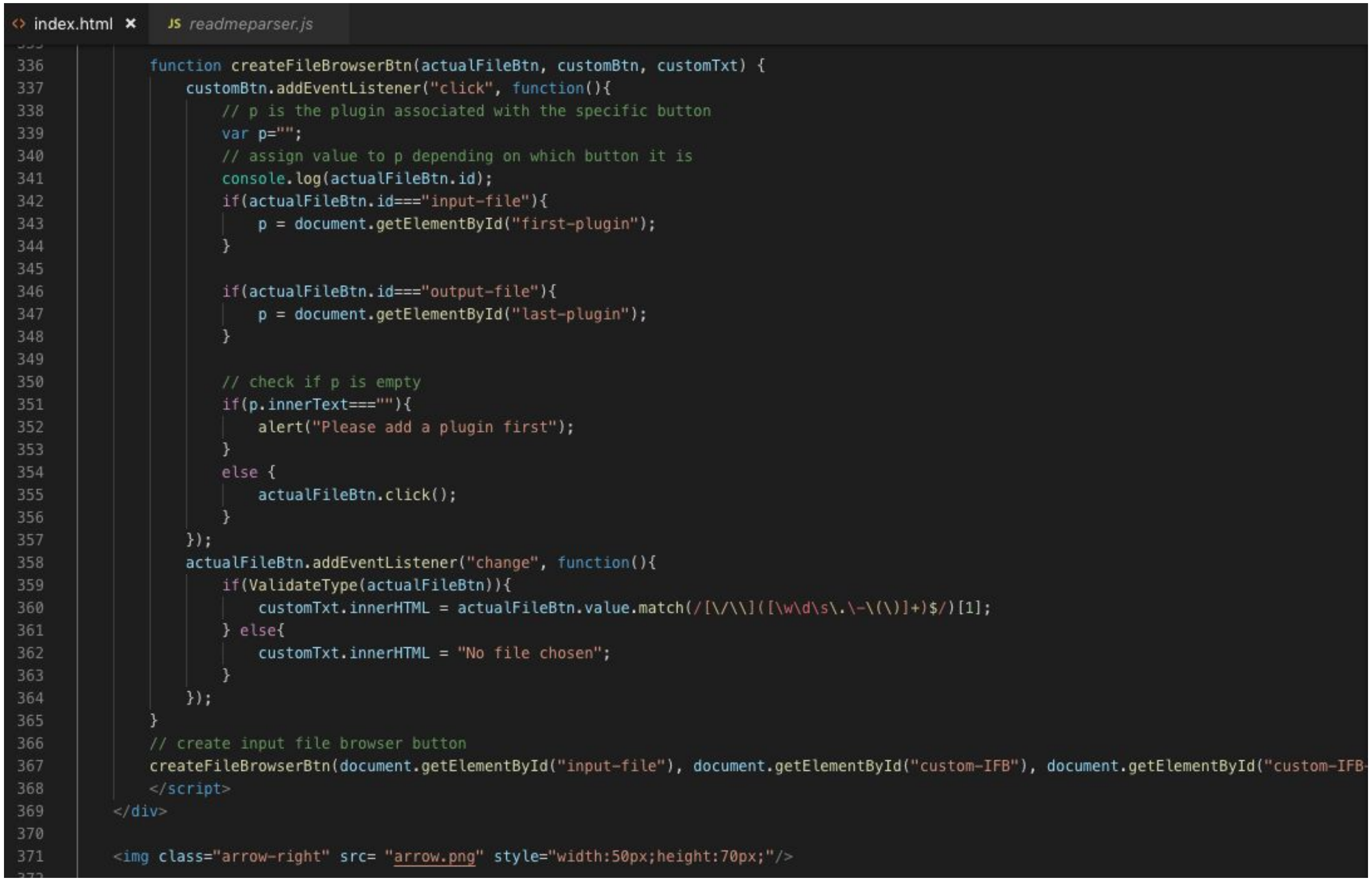
```javascript
const pluginInputTypes = filtered
.map(function (folder_path){
    var readmeContent = fs.readFileSync(folder_path + '/README.md', 'utf8');
    // console.log(readmeContent);
    readmeContent = readmeContent.split("\n");
    inputLineIndex = readmeContent.reduce(function (index, readmeContent, actual_index) {
        if (index !== -1)
            return index;
        if (readmeContent.substr(0, 10).toLowerCase().indexOf("input:") > -1)
            return actual_index;
        return -1;
    }, -1);
    var inputLine = readmeContent[inputLineIndex];
    // console.log(inputLine);
    inputLine = inputLine.split(" ");
    // console.log(inputLine);
    const inputFileType = inputLine[2];
    return inputFileType.toLowerCase();
});
console.log(pluginInputTypes);

// array containing output file types for each plugin
const pluginOutputTypes = filtered
.map(function (folder_path){
    var readmeContent = fs.readFileSync(folder_path + '/README.md', 'utf8');
    readmeContent = readmeContent.split("\n");
    outputLineIndex = readmeContent.reduce(function (index, readmeContent, actual_index) {
        if (index !== -1)
            return index;
        if (readmeContent.substr(0, 10).toLowerCase().indexOf("output:") > -1)
            return actual_index;
        return -1;
    }, -1);
    var outputLine = readmeContent[outputLineIndex];
    // console.log(outputLine);
    outputLine = outputLine.split(" ");
    // console.log(outputLine);
    const outputFileType = outputLine[2];
    return outputFileType.toLowerCase();
});
console.log(pluginOutputTypes);
```



```javascript
// function to extract name of uploaded file from the fake path
function getNameFromPath(filePath) {
    return filePath.substr(filePath.lastIndexOf('\\') + 1);
}

// function to validate uploaded file type
function ValidateType(uploadedFile) {
    var isValid = false;
    if (uploadedFile.type == "file") {
        var fileName = getNameFromPath(uploadedFile.value);
        if (fileName.length > 0) {
            // p is the plugin associated with the specific button
            var p="";
            // if input button, get plugin name from first box
            if (uploadedFile.id==="input-file"){
                p = document.getElementById("first-plugin");
                console.log(p);
                console.log(p.innerText + "...");
            }
            // if output button, get plugin name from last box
            if (uploadedFile.id==="output-file"){
                p = document.getElementById("last-plugin");
                console.log(p);
                console.log(p.innerText);
            }
            // now p.innerText contains name of plugin

            // variable to store valid extension
            var validExt = "";
            // simple search for now
            for (var i=0; i<pluginNames.length; i++){
                var compareText = pluginNames[i]+'\n';
                if(p.innerText === compareText){
                    if(uploadedFile.id==="input-file"){
                        validExt = "." + pluginInputTypes[i];
                    }
                    if(uploadedFile.id==="output-file"){
                        validExt = "." + pluginOutputTypes[i];
                    }
                }
```

Main algorithm

- Bhavyta’s feature (continued):

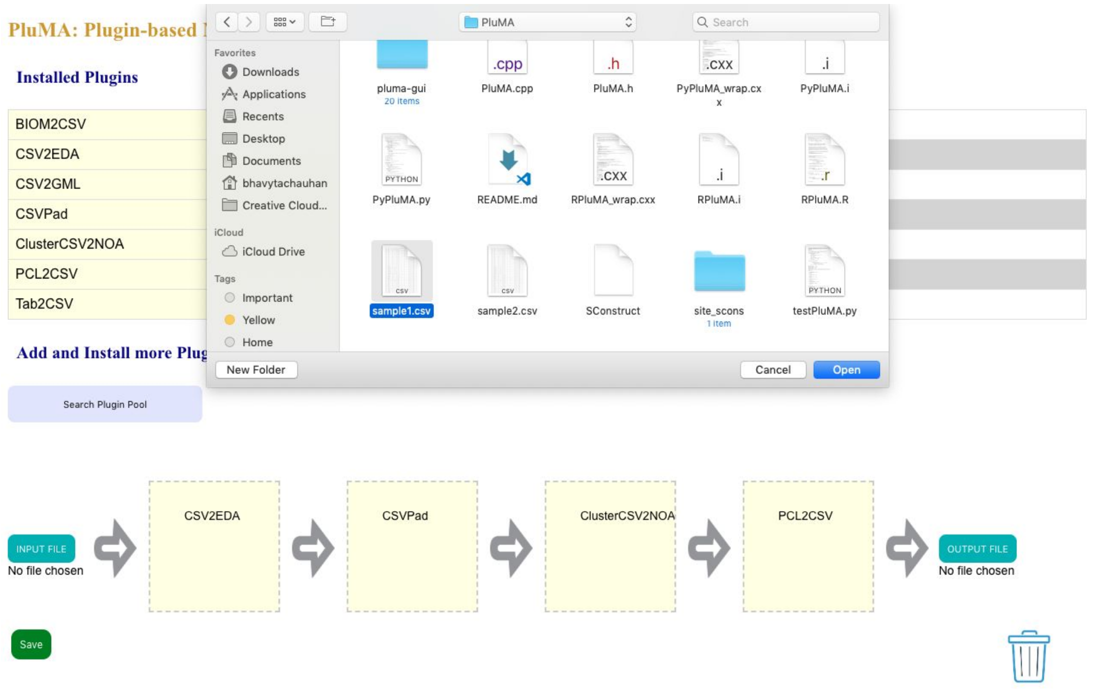


index.html ✕ JS readmeparser.js

```
    function createFileBrowserBtn(actualFileBtn, customBtn, customTxt) {
        customBtn.addEventListener("click", function(){
            // p is the plugin associated with the specific button
            var p="";
            // assign value to p depending on which button it is
            console.log(actualFileBtn.id);
            if(actualFileBtn.id==="input-file"){
                p = document.getElementById("first-plugin");
            }

            if(actualFileBtn.id==="output-file"){
                p = document.getElementById("last-plugin");
            }

            // check if p is empty
            if(p.innerText===""){
                alert("Please add a plugin first");
            }
            else {
                actualFileBtn.click();
            }
        });
        actualFileBtn.addEventListener("change", function(){
            if(ValidateType(actualFileBtn)){
                customTxt.innerHTML = actualFileBtn.value.match(/[\/\\]([\w\d\s\.\-\(\)]+)$/)[1];
            } else{
                customTxt.innerHTML = "No file chosen";
            }
        });
    }
    // create input file browser button
    createFileBrowserBtn(document.getElementById("input-file"), document.getElementById("custom-IFB"), document.getElementById("custom-IFB
    </script>
</div>

<img class="arrow-right" src= "arrow.png" style="width:50px;height:70px;"/>
```

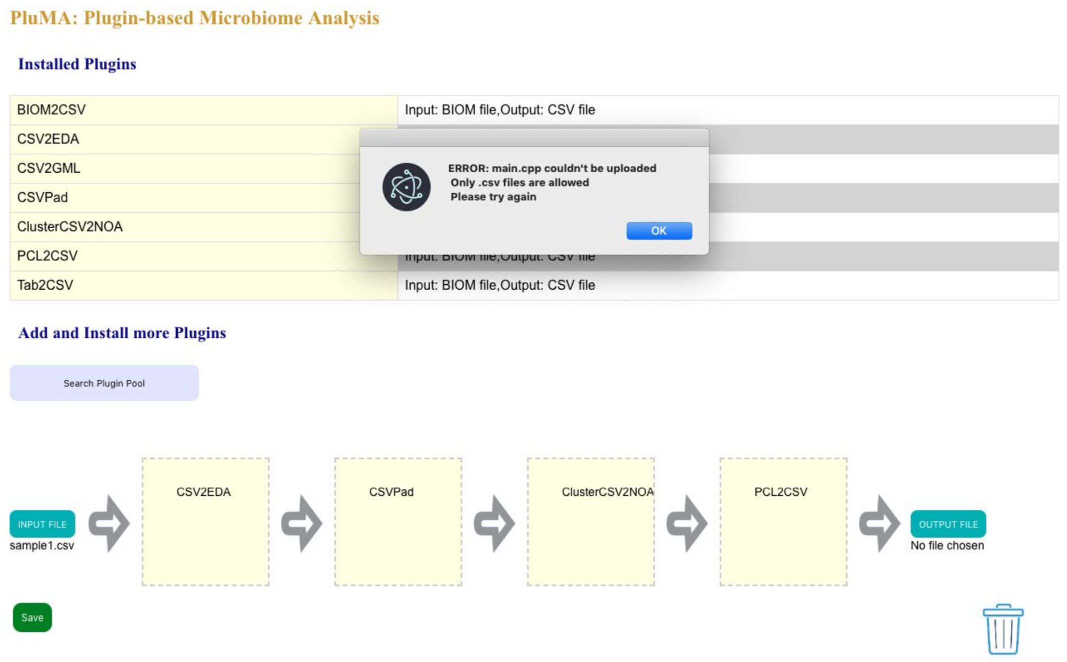
Main algorithm

- Bhavyta’s feature (continued):



Main algorithm

- Bhavyta’s feature (continued):

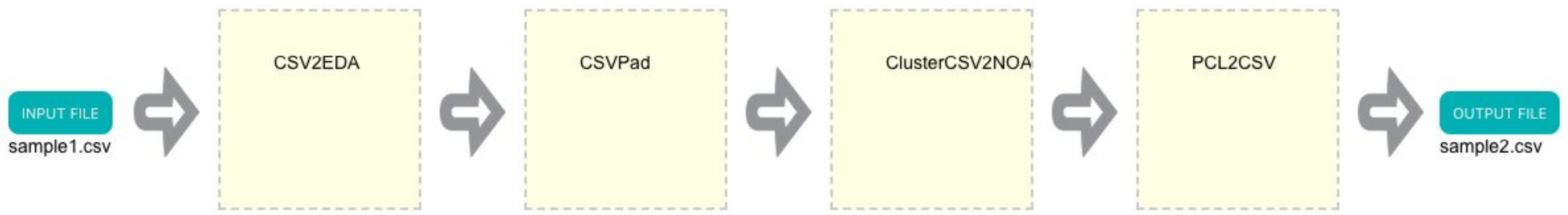


PluMA: Plugin-based Microbiome Analysis

Installed Plugins

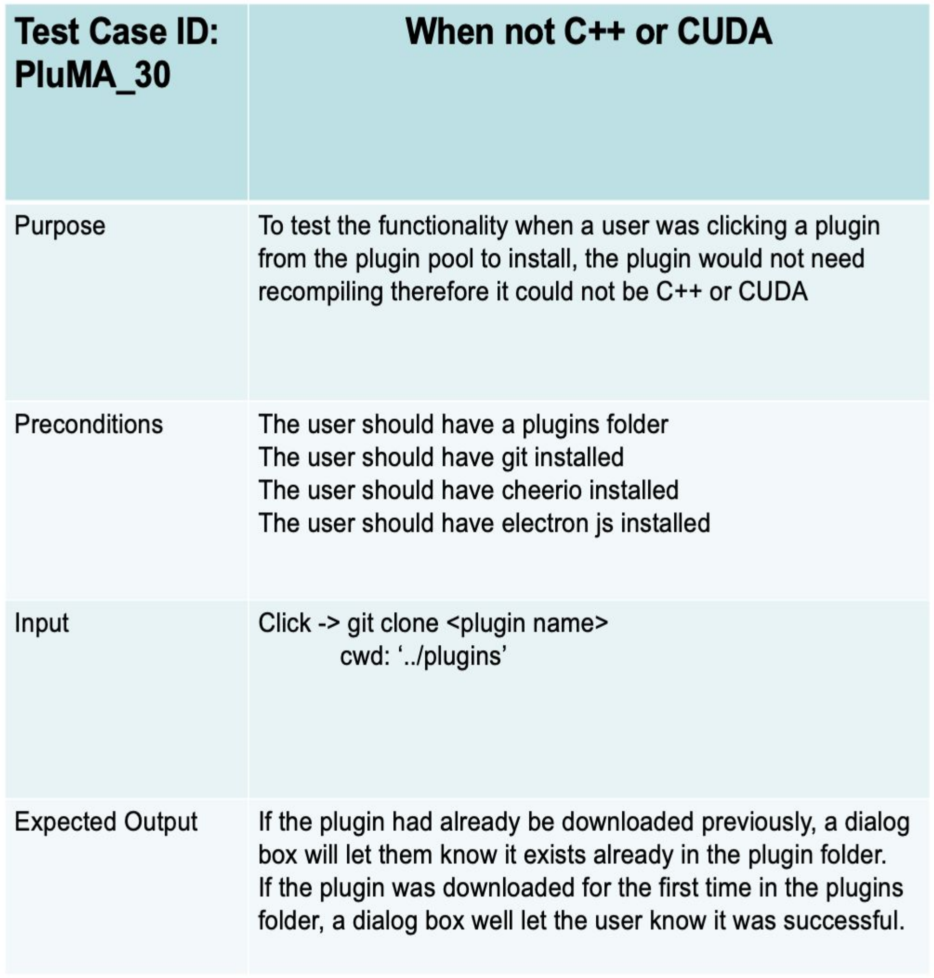
BIOM2CSV	Input: BIOM file,Output: CSV file
CSV2EDA	
CSV2GML	
CSVPad	
ClusterCSV2NOA	
PCL2CSV	Input: BIOM file,Output: CSV file
Tab2CSV	Input: BIOM file,Output: CSV file

ERROR: main.cpp couldn't be uploaded
Only .csv files are allowed
Please try again



Test Suites and Test Cases

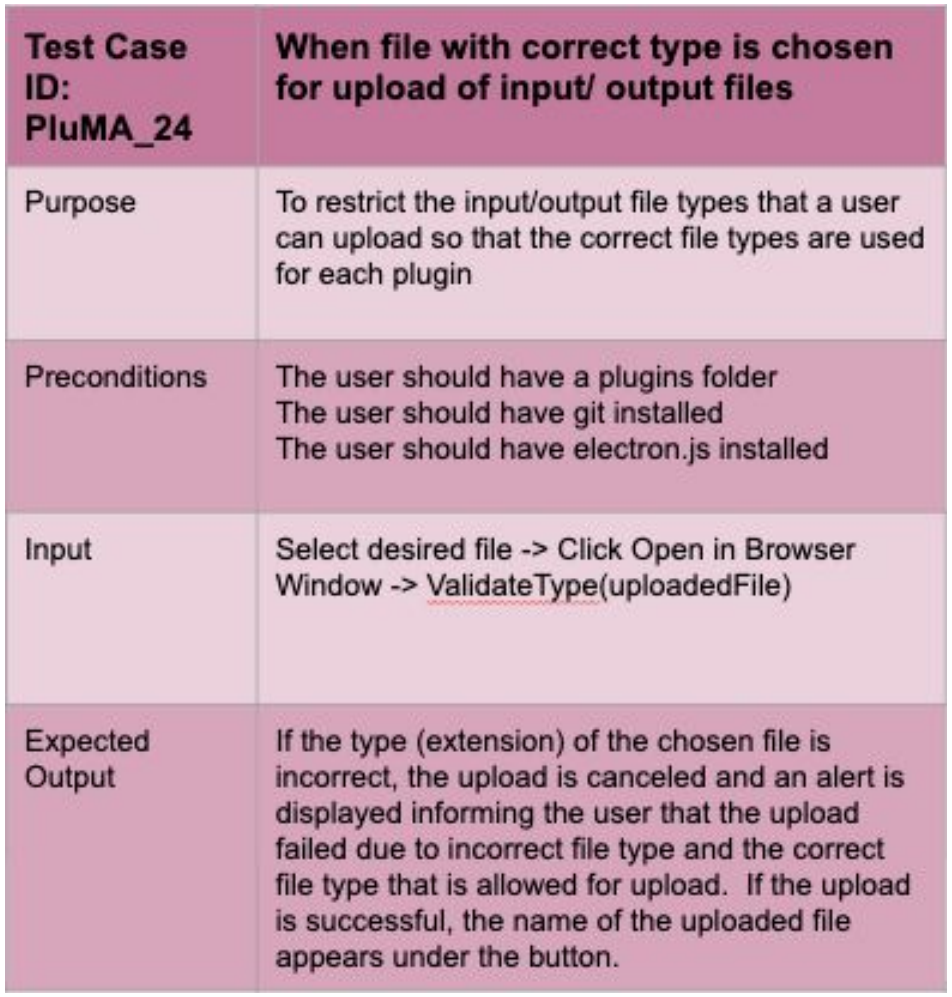
- Cesia’s Test Case



Test Case ID: PluMA_30	**When not C++ or CUDA**
Purpose	To test the functionality when a user was clicking a plugin from the plugin pool to install, the plugin would not need recompiling therefore it could not be C++ or CUDA
Preconditions	The user should have a plugins folder The user should have git installed The user should have cheerio installed The user should have electron js installed
Input	Click -> git clone <plugin name> cwd: ‘../plugins’
Expected Output	If the plugin had already be downloaded previously, a dialog box will let them know it exists already in the plugin folder. If the plugin was downloaded for the first time in the plugins folder, a dialog box well let the user know it was successful.

Test Suites and Test Cases

- Bhavyta’s Test Case



Test Case ID: PluMA_24	**When file with correct type is chosen for upload of input/ output files**
Purpose	To restrict the input/output file types that a user can upload so that the correct file types are used for each plugin
Preconditions	The user should have a plugins folder The user should have git installed The user should have electron.js installed
Input	Select desired file -> Click Open in Browser Window -> ValidateType(uploadedFile)
Expected Output	If the type (extension) of the chosen file is incorrect, the upload is canceled and an alert is displayed informing the user that the upload failed due to incorrect file type and the correct file type that is allowed for upload. If the upload is successful, the name of the uploaded file appears under the button.

Summary

- By making the GUI for PluMA, users that have no coding background will be able to install plugins without running any command on the terminal, drag and drop plugins in order to execute a pipeline of their choice and verify whether or not their input and output files would work with the current system.
- Cesia Bulnes, cbuln004@fiu.edu, 7865371605
- Rishabh Vaidya, rvaid004@fiu.edu, 6023589167
- Bhavyta Chauhan, bchau004@fiu.edu, 7867817138
- Questions?
- Thank You!