*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Final Deliverable

Project Title: PluMA Graphical User Interface

**Team Members: Rishabh Vaidya, Cesia Bulnes, Bhavyta Chauhan**

**Product Owner(s)**: Trevor Cickovski

**Mentor(s)**: Trevor Cickovski

**Instructor**: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *2016 Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

***Abstract***

*This document presents the information necessary to gain a good understanding of PluMA: Plugin-based Microbiome Analysis. 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 interface. Since plugins are dynamically loaded, you can assemble a pipeline by downloading PluMA and only the plugins that you need from the pool.*

*Currently, PluMA can be intricate for those who don’t have coding experience or have never used the terminal. With the GUI, users can create the same pipeline formation and install plugins with a very easy user experience that does not require any coding whatsoever.*

**Table of Contents**

**Introduction** ..………………………………………………………………………………..……….....7

Current System .……………..……....……………………………………………………..…..7

Purpose of New System …....…..………………………………..…………………………....8

**User Stories**

Implemented User Stories ...…………………………….………………………….………..15

Pending User Stories ...………………………………………...………………………..…...16

**Project Plan**

Hardware and Software Resources ………………………………...…………………...….17

Sprints Plan ……………………………………………..……………………………..……....18

*Sprint 1* …..………………………………………………………………………………..18

*Sprint 2*…………………………………………………………..………………………...19

*Sprint 3*…………………………………………………………………………………….20

*Sprint 4.*……………………………………………………………………………………21

*Sprint 5....*………………………………………………………………………………… 22

*Sprint 6*…………………………………………………………………………………… 23

*Sprint 7*.……………………………………………………………………………………24

**System Design**

Architectural Patterns ……….…..………………………………………………………………….25 System and Subsystem Decomposition…………...……...…………………………………….…26

Deployment Diagram..………………………………………………………….……………...…….26

Design Patterns…….…………………………………………………………………….….............27

**System Validation.**………..…………………………………………………………………………...28

**Appendix.**…………………………………………………………………………………………….....31

Appendix A - UML Diagrams....……....…………………………………………………………….31

Appendix C - Sprint Review Reports.……………………………………………………...……....34

Appendix C - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents..…………………………..……………………………………………………..35

**References** ..…………………………………………………………………………………………....35

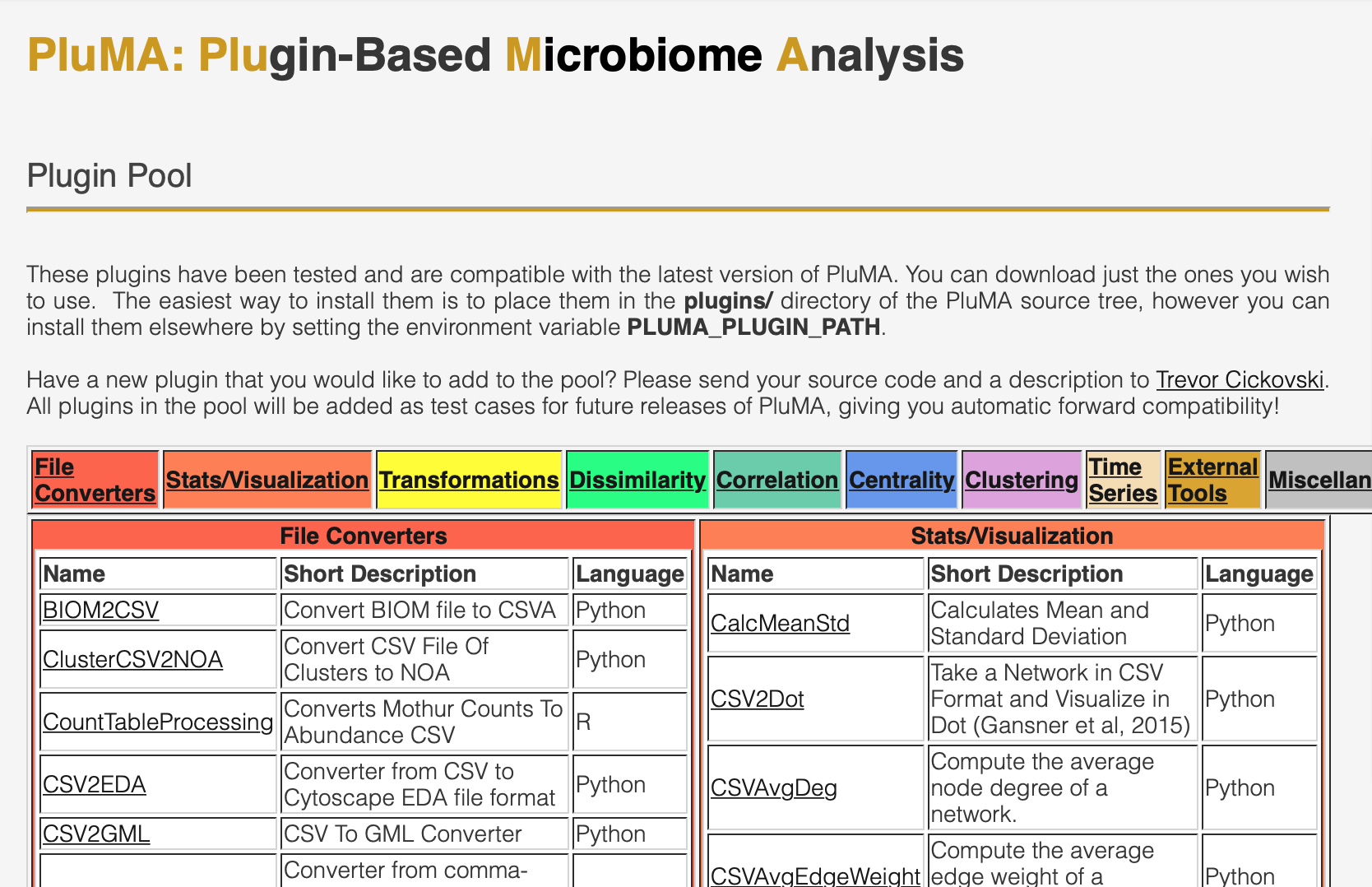
# **Introduction**

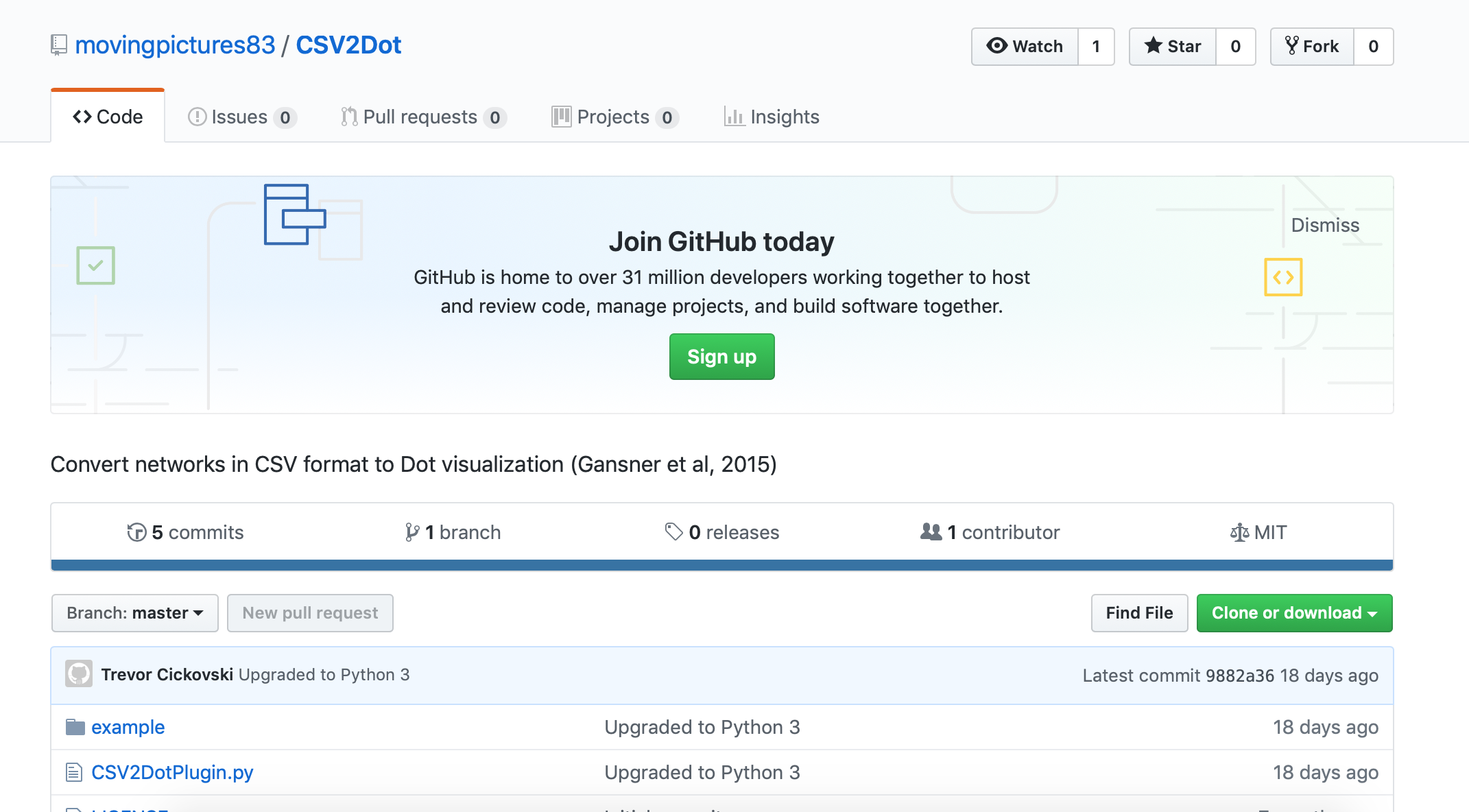
*Through PluMA: Plugin-based Microbiome Analysis, plugins can be developed standalone using either the scripted or computational (compiled) interfaces to PLUMA. The scripted interface currently supports Python, Perl and R, and the compiled interface supports C++ for the CPU and CUDA for the GPU.*

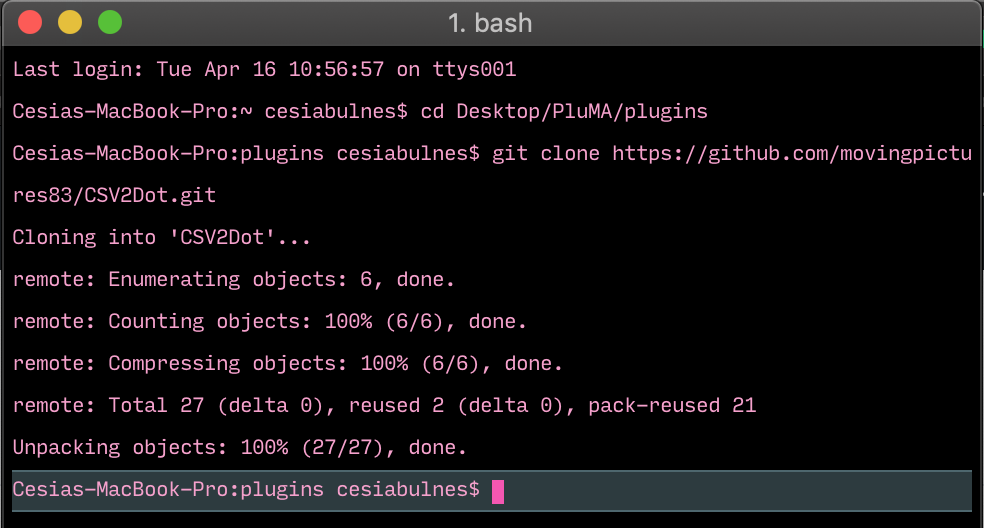
*PluMA-Gui facilitates the cloning of each plugin into the plugins folder and the formation of pipelines by letting the user drag a plugin into the pipeline, with an input file and an output file.*

## **Current System**

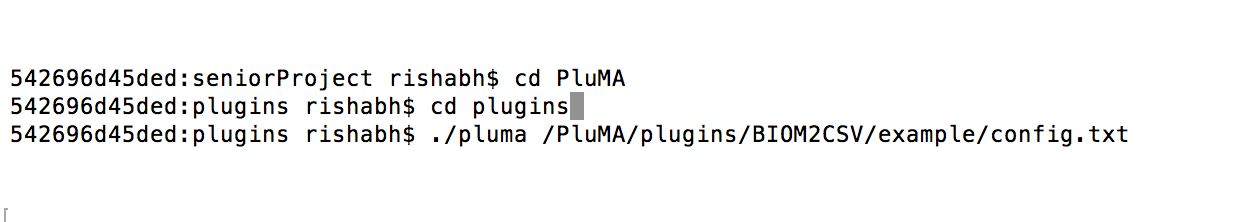
The current system has the plugin pool, when you click any of the plugins it takes you to the github.





Then you have to clone it into the plugins folder using your terminal.

Along with cloning the folder, the user also has to install PluMA in order to be able to run it. Additionally, the user has to download scons which is a tool that enables the user to run various types of program files such as Python, C, Perl and more by typing scons and the filename in the terminal. The PluMA software uses scons in order to run the different language files in the plugin directories. Once the folder is cloned, the user can run the program file in that folder by typing ./pluma along with the plugin directory that includes the path to the config.txt file in the command prompt.



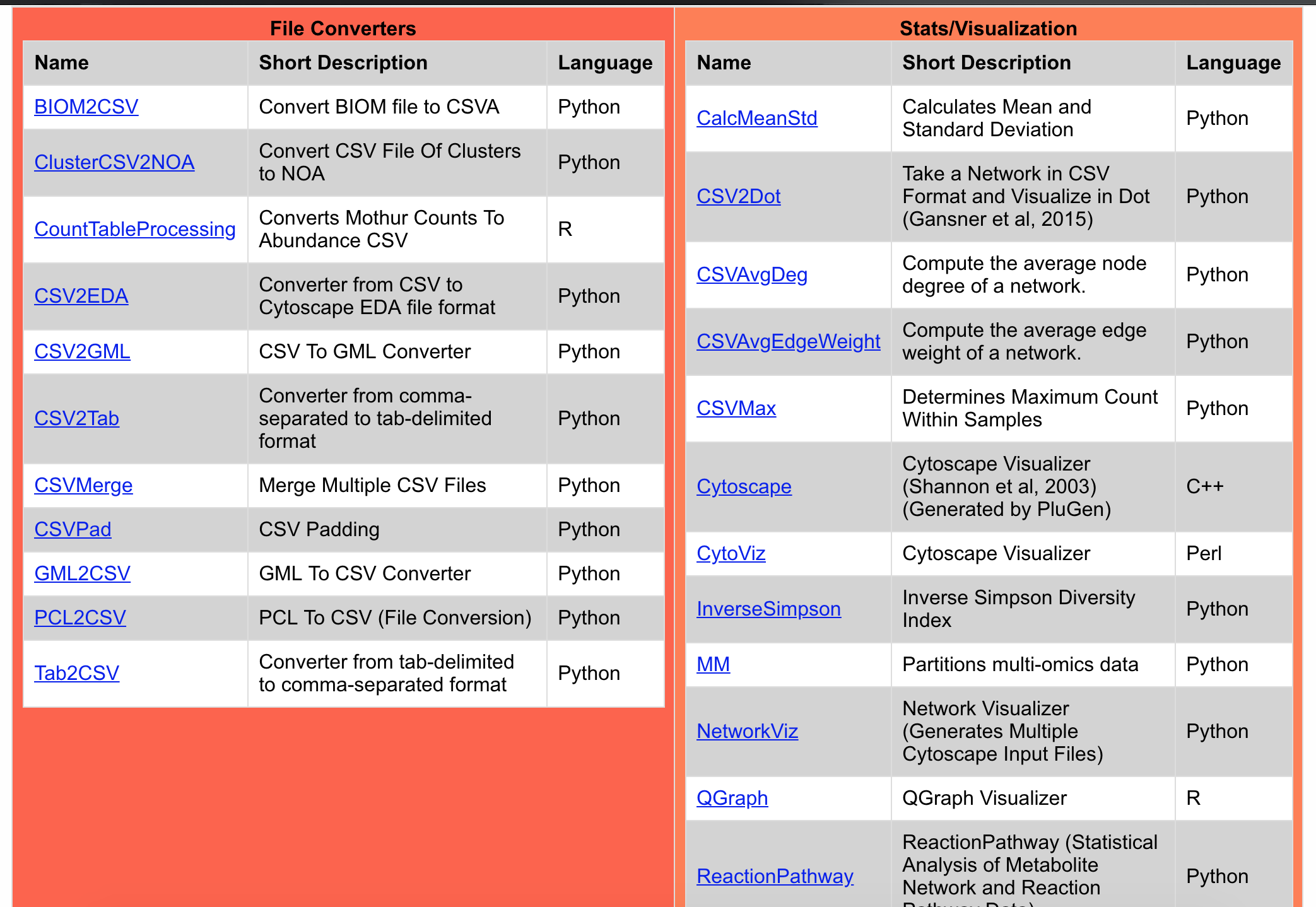
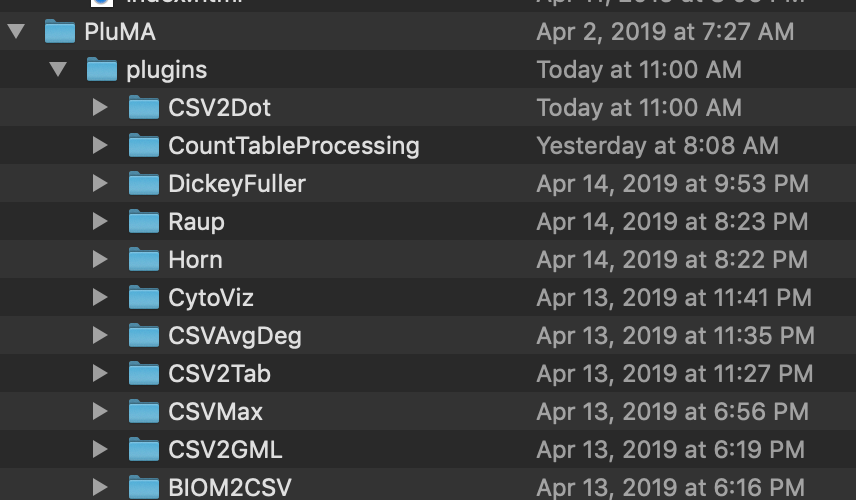
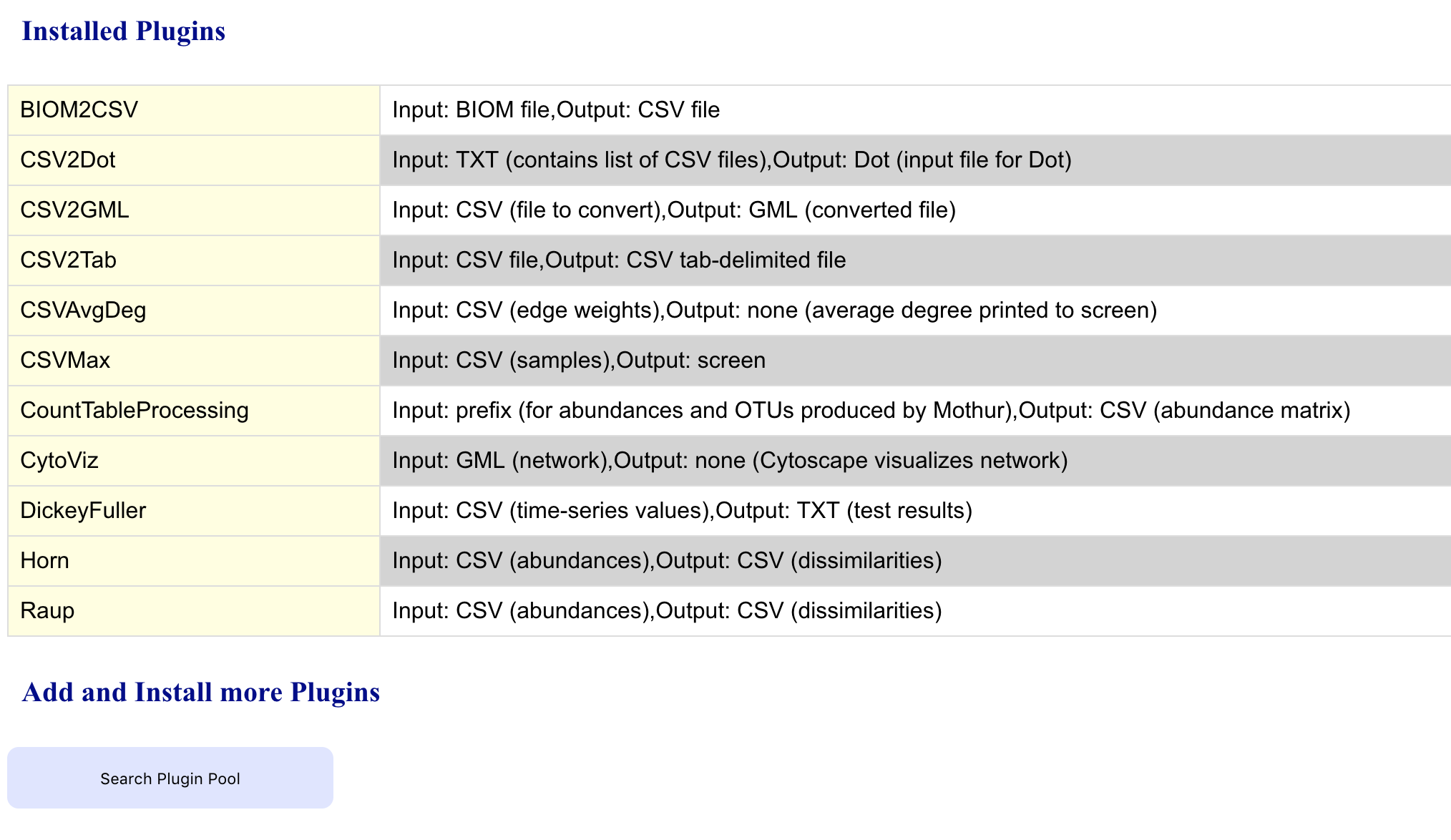
## **Purpose of New System**

The new system includes a Graphical User Interface that makes it easier for users without a technical background to use the PluMA software. Such users may include biologists who have no prior experience with running commands on the terminal/ command line or cloning repositories from GitHub.

The interface is a desktop application that only needs to be installed on the user’s computer and is very straightforward to use. It helps in assembling pipelines through such operations as searching the online plugin pool, downloading plugins, assembling pipelines by dragging and dropping installed plugins, uploading input/ output files associated with a plugin, and saving the configuration file of an assembled pipeline.

**Pipeline Cloning, Installation and Plugin Pool**

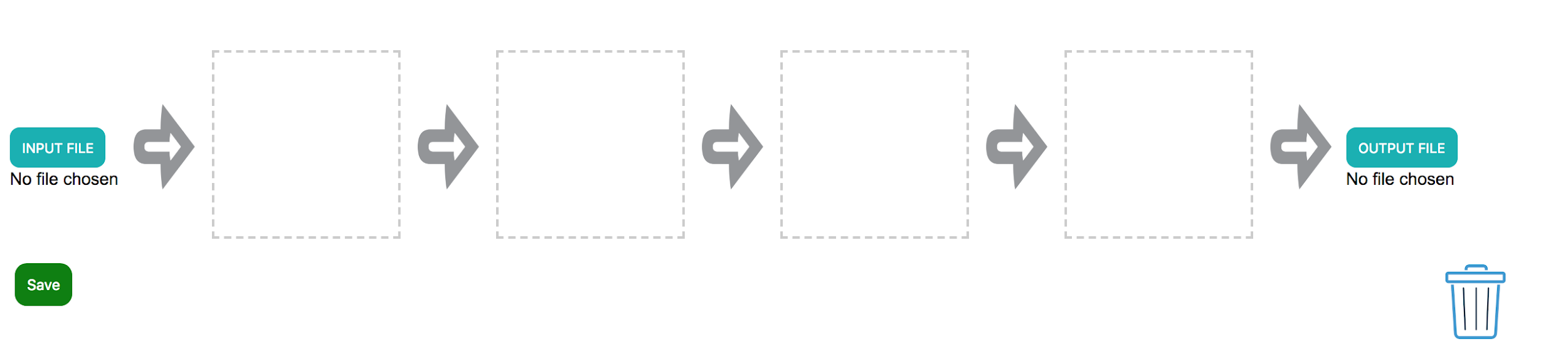
Pipelines that have been cloned, will appear under the installed plugins. The software dynamically reads all the plugins in the plugins folder of PluMA. Apart from that, when you click on Search Plugin Pool, the bioOrg website of the plugin pool is scraped, using cheerio. On click on any of the plugins, the plugin will be cloned into the plugins folder of the user.



After the plugins are cloned into the user’s plugin folder, the user can drag and drop the displayed plugins into the pipeline.

**Pipeline Assembly Area**

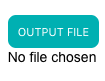
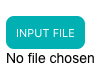
The pipeline assembly area includes boxed areas where plugins can be dragged and dropped from the list of Installed Plugins. The user also has the option to delete unnecessary/unwanted plugins from the pipeline by dragging plugins into the trash can shown below.



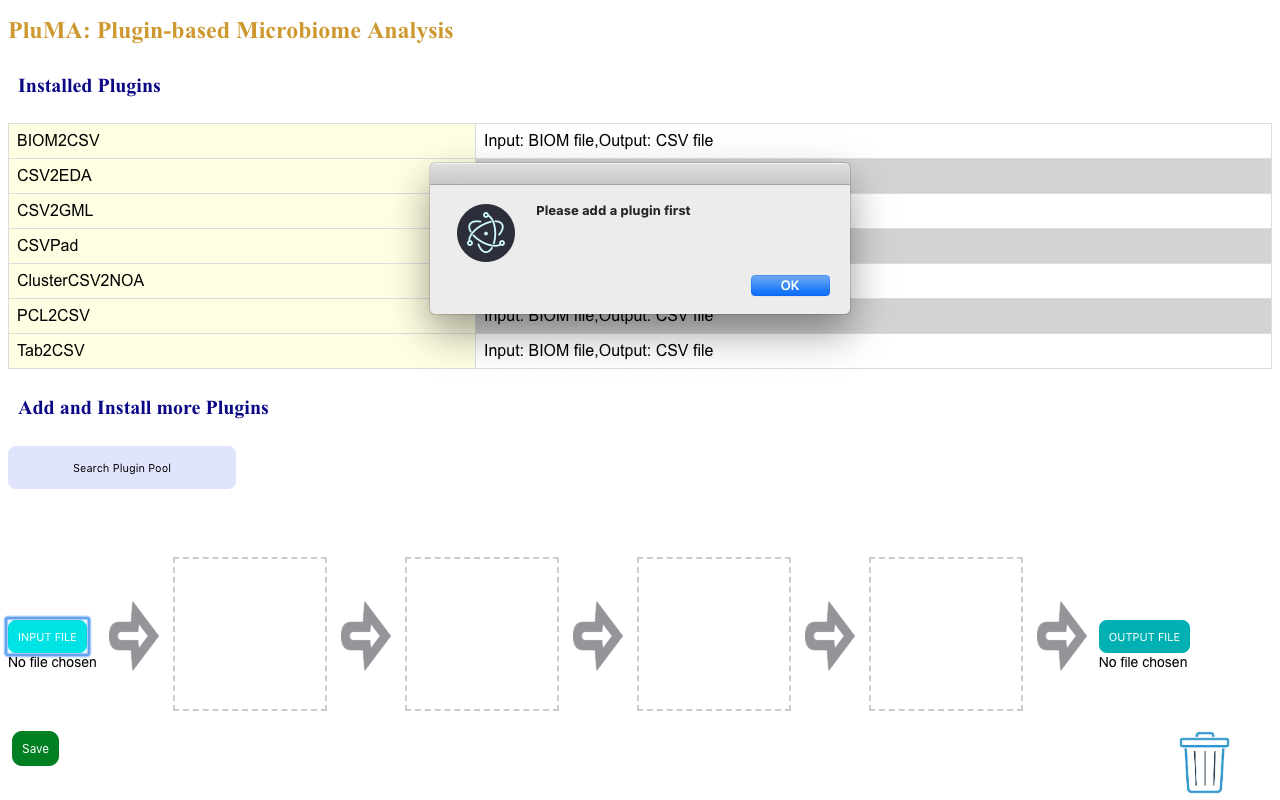
**Uploading Input/ Output Files**

Each plugin has an input and output file associated with it. For the PluMA software, it is possible for the user to provide input and output files for the intermediate plugins in the pipeline since it is not necessary that the pipeline operates linearly.

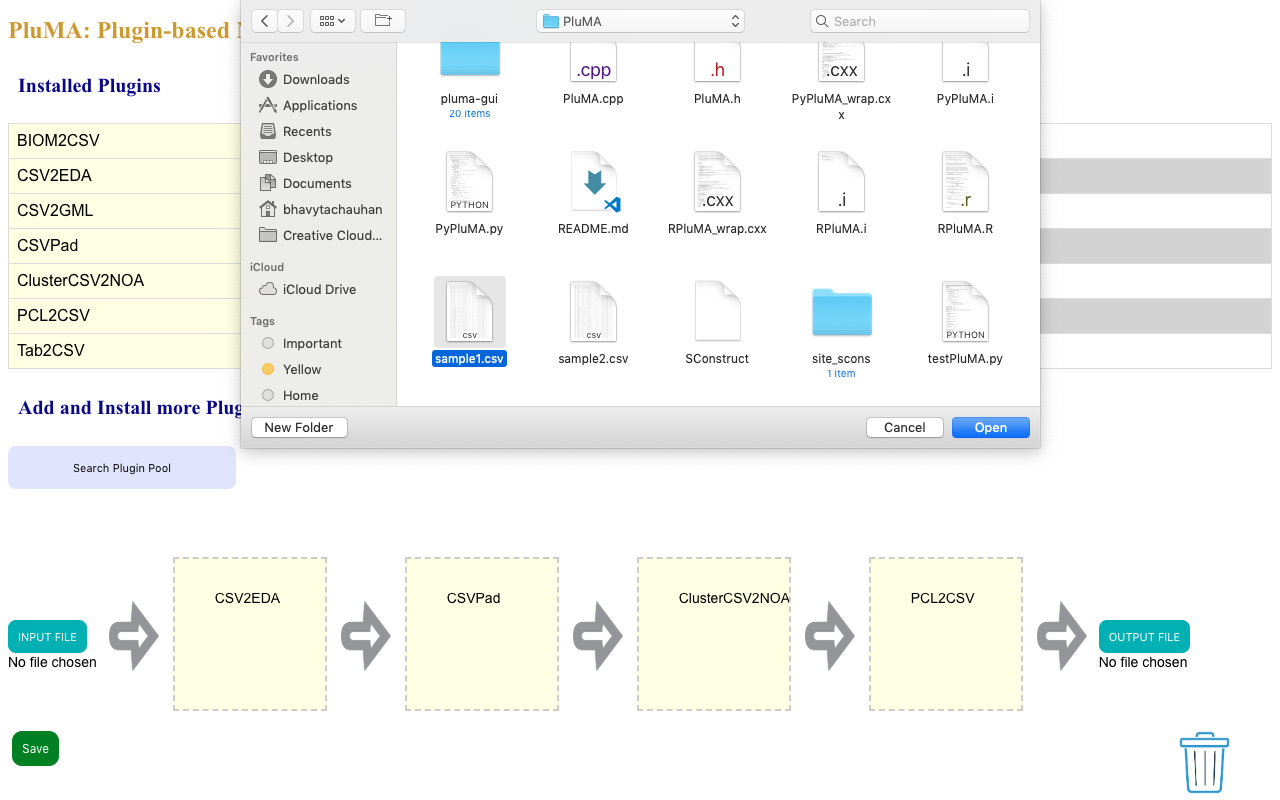
Based on the current implementation of the interface however, it is assumed that only linear pipelines are allowed. Therefore, the user only needs to upload an input file for the first plugin and an output file for the last plugin. This is done through the input and output file upload buttons (shown below) at the beginning and end of the pipeline.

****

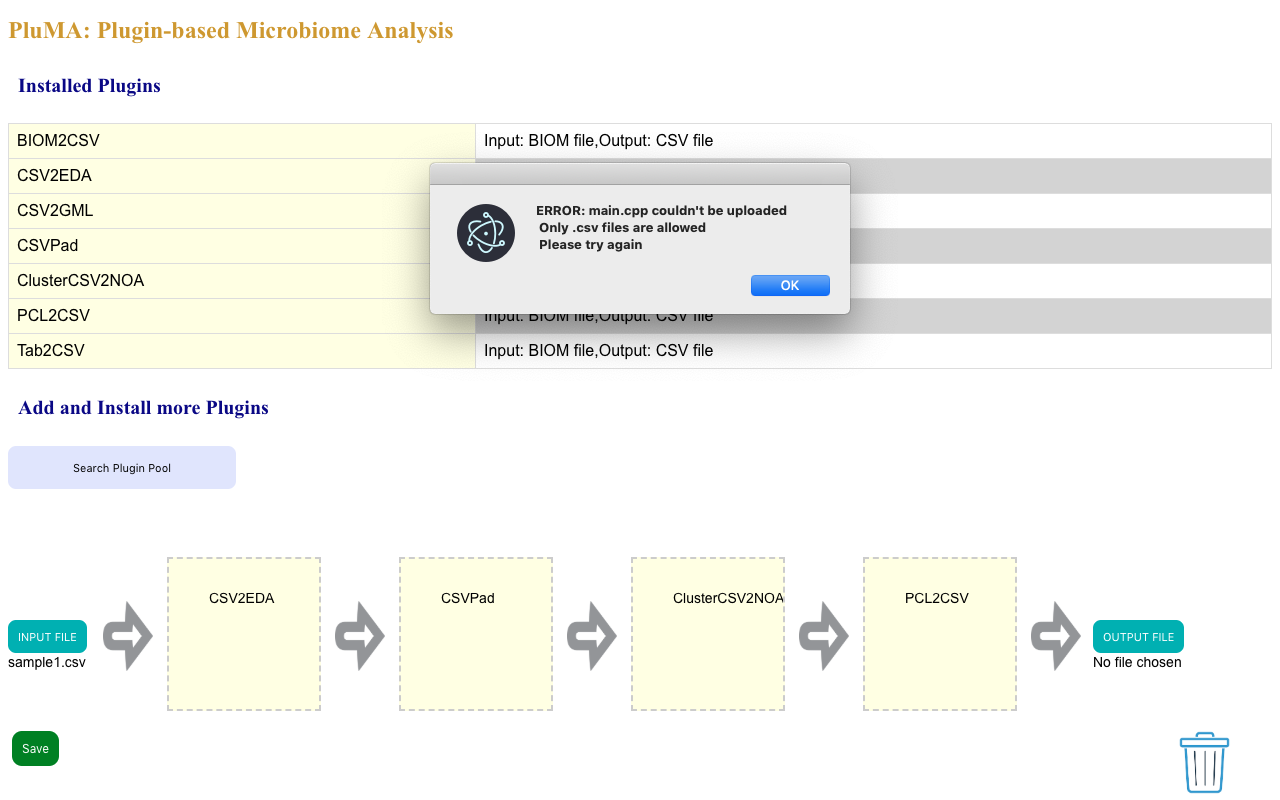
The user may not upload a file unless the associated plugin has been dragged to the respective boxed area. An alert is displayed if the user tries to do so.

****

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.

****

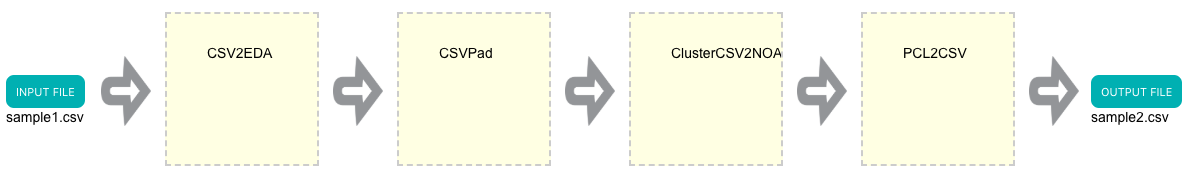
If the type (extension) of the chosen file is incorrect, the upload is canceled and an alert is displayed informing the user that the upload failed due to incorrect file type and the correct file type that is allowed for upload.

****

If the upload is successful, the name of the uploaded file appears under the button.

****

Here is a view of a completely assembled pipeline:

****

Once the user adds their respective plugins and input/ output files associated with the first and last plugin, he/she can save the pipeline by clicking on the save button. This generates a config.txt file which contains the input filename, the name of all the plugins added, temporary output filenames for each plugin box, and the output file name.



**User Stories**

The following section provides the detailed user stories that were implemented in this iteration of the PluMA: Graphical User Interface project. These user stories served as the basis for the implementation of the project’s features. This section also shows the user stories that are to be considered for future development.

## **Implemented User Stories**

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

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

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

## **Pending User Stories**

All user stories have been completed

# **Project Plan**

We focused on creating a user-friendly graphical interface for PluMA: a plugin-based microbiome analysis program. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

## **Hardware and Software Resources**

The following is a list of all hardware and software resources that were used in this project:

1. Electron
2. Chromium
3. Bitbucket
4. Git
5. HTML
6. CSS
7. Javascript
8. Cheerio

## **Sprints Plan**

### **Sprint 1**

**20190122 : Sprint 1 Planning Meeting Minutes**

Attendees: Trevor Cickovski, Cesia Bulnes, Rishabh Vaidya, Bhavyta Chauhan

Start time: 6:30 pm

End time: 7:30 pm

After discussion, the velocity of the team was estimated to be 76 hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story #1 PLMA-1[~~PLMA-1~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-1) - As a developer, I want to ensure that the electron app is cross-platform **DONE**
* User Story #2 PLMA-2 [~~PLMA-2~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-2) - Build a visual prototype **DONE**
* User Story #3 PLMA-3 [~~PLMA-3~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-3) - Build a visual prototype **DONE**
* User Story #4 PLMA-4 [~~PLMA-4~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-4) - Review frameworks and technologies **DONE**
* User Story #5 PLMA-5 [~~PLMA-5~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-5) - Review frameworks and technologies **DONE**
* User Story #6 PLMA-6 [~~PLMA-6~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-6) - Review frameworks and technologies **DONE**
* User Story #7 PLMA-7 [~~PLMA-7~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-7) - Setup static front page **DONE**
* User Story #8 PLMA-8 [~~PLMA-8~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-8) - Setup static front page **DONE**
* User Story #9 PLMA-9 [~~PLMA-9~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-9) - Setup static front page **DONE**

The team members indicated their willingness to work on the following user stories.

* Bhavyta Chauhan
  + [~~PLMA-3~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-3) - Build a visual prototype **DONE**
  + [~~PLMA-5~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-5) - Review frameworks and technologies **DONE**
  + [~~PLMA-9~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-9) - Setup static front page **DONE**
* Rishabh Vaidya
  + [~~PLMA-1~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-1) - As a developer, I want to ensure that the electron app is cross-platform **DONE**
  + [~~PLMA-4~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-4) - Review frameworks and technologies **DONE**
  + [~~PLMA-7~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-7) - Setup static front page **DONE**
* Cesia Bulnes
  + [~~PLMA-2~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-2) - Build a visual prototype **DONE**
  + [~~PLMA-6~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-6) - Review frameworks and technologies **DONE**
  + [~~PLMA-8~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-8) - Setup static front page **DONE**

### **Sprint 2**

**20190204 : Sprint 2 Planning Meeting Minutes**

Attendees: Trevor Cickovski, Cesia Bulnes, Rishabh Vaidya, Bhavyta Chauhan

Start time: 10:00 am

End time: 11:00 am

After discussion, the velocity of the team was estimated to be 84 hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User story #1 PLMA-20 [~~PLMA-20~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-20) - As a developer, I want to be able to code in a main page **DONE**
* User Story #2 PLMA-7 [~~PLMA-7~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-7) - Setup static front page **DONE**
* User Story #3 PLMA-8 [~~PLMA-8~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-8) - Setup static front page **DONE**
* User Story #4 PLMA-9 [~~PLMA-9~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-9) - Setup static front page **DONE**
* User Story #5 PLMA-16 [~~PLMA-16~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-16) - As a user, I want to be able to access plugins from the plugins folder in order to setup pipelines**DONE**
* User Story #6 PLMA-17[~~PLMA-17~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-17) - As a user, I want to be able to drag and drop the plugins in order to assemble a pipeline **DONE**
* User Story #7 PLMA-18[~~PLMA-18~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-18) - As a user, I want to be able to send input and output file names to each plugin to assemble a pipeline **DONE**
* User Story #8 PLMA-19[~~PLMA-19~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-19) - As a user, I want to be able to save the config file in order to be able to run PluMA **DONE**

The team members indicated their willingness to work on the following user stories.

* Bhavyta Chauhan
  + [~~PLMA-9~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-9) - Setup static front page **DONE**
  + [~~PLMA-18~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-18) - As a user, I want to be able to send input and output file names to each plugin to assemble a pipeline **DONE**
* Rishabh Vaidya
  + [~~PLMA-7~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-7) - Setup static front page **DONE**
  + [~~PLMA-17~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-17) - As a user, I want to be able to drag and drop the plugins in order to assemble a pipeline **DONE**
* Cesia Bulnes
  + [~~PLMA-19~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-19) - As a user, I want to be able to save the config file in order to be able to run PluMA **DONE**
  + [~~PLMA-8~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-8) - Setup static front page **DONE**
  + [~~PLMA-16~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-16) - As a user, I want to be able to access plugins from the plugins folder in order to setup pipelines **DONE**

### 

### **Sprint 3**

**20190219 : Sprint 3 Planning Meeting Minutes**

Attendees: Trevor Cickovski, [Cesia Bulnes](https://sp-jira.cis.fiu.edu:8453/display/~cbuln004) ,[Rishabh Vaidya](https://sp-jira.cis.fiu.edu:8453/display/~rvaid004), [Bhavyta Chauhan](https://sp-jira.cis.fiu.edu:8453/display/~bchau004)

Start time: 10:00 am

End time: 11:00 am

After discussion, the velocity of the team was estimated to be 84 hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story #1 PLMA-22 [~~PLMA-22~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-22) - As a user, I should be able to drag and drop installed plugins into the pipeline **DONE**
* User Story #2 PLMA-23 [~~PLMA-23~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-23) - As a developer, I want to implement a function to parse through the readme file of a plugin**DONE**
* User Story #3 PLMA-24 [~~PLMA-24~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-24) - 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 **DONE**
* User Story #4 PLMA-25 [~~PLMA-25~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-25) - As a user, I want to be able to view the input file type, plugin name, output file type **DONE**
* User Story #5 PLMA-16 [~~PLMA-16~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-16) - As a user, I want to be able to access plugins from the plugins folder in order to setup pipelines**DONE**
* User Story #6 PLMA-18 [~~PLMA-18~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-18) - As a user, I want to be able to send input and output file names to each plugin to assemble a pipeline **DONE**
* User Story #7 PLMA-19 [~~PLMA-19~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-19) - As a user, I want to be able to save the config file in order to be able to run PluMA **DONE**
* User Story #8 PLMA-26 [~~PLMA-26~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-26) - As a user, I want to be able to click on a button to access the plugin pool **DONE**
* User Story #9 PLMA-27 [~~PLMA-27~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-27) - As a user, I should able to view all plugins from the pool in a popup window **DONE**

The team members indicated their willingness to work on the following user stories.

* [Bhavyta Chauhan](https://sp-jira.cis.fiu.edu:8453/display/~bchau004)   
  + [~~PLMA-18~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-18) - As a user, I want to be able to send input and output file names to each plugin to assemble a pipeline **DONE**
  + [~~PLMA-23~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-23) - As a developer, I want to implement a function to parse through the readme file of a plugin **DONE**
  + [~~PLMA-24~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-24) - 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 **DONE**
* [Rishabh Vaidya](https://sp-jira.cis.fiu.edu:8453/display/~rvaid004)
  + [~~PLMA-19~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-19) - As a user, I want to be able to save the config file in order to be able to run PluMA **DONE**
  + [~~PLMA-22~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-22) - As a user, I should be able to drag and drop installed plugins into the pipeline **DONE**
* [Cesia Bulnes](https://sp-jira.cis.fiu.edu:8453/display/~cbuln004)
  + [~~PLMA-16~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-16) - As a user, I want to be able to access plugins from the plugins folder in order to setup pipelines **DONE**
  + [~~PLMA-25~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-25) - As a user, I want to be able to view the input file type, plugin name, output file type**DONE**
  + [~~PLMA-26~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-26) - As a user, I want to be able to click on a button to access the plugin pool **DONE**
  + [~~PLMA-27~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-27) - As a user, I should able to view all plugins from the pool in a popup window **DONE**

### **Sprint 4**

**20190304 : Sprint 4 Planning Meeting Minutes**

Attendees: Trevor Cickovski, [Cesia Bulnes](https://sp-jira.cis.fiu.edu:8453/display/~cbuln004) ,[Rishabh Vaidya](https://sp-jira.cis.fiu.edu:8453/display/~rvaid004), [Bhavyta Chauhan](https://sp-jira.cis.fiu.edu:8453/display/~bchau004)

Start time: 10:00 am

End time: 11:00 am

After discussion, the velocity of the team was estimated to be 65 hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story #1 PLMA-24 [~~PLMA-24~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-24) - 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 **DONE**
* User Story #2 PLMA-18 [~~PLMA-18~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-18) - As a user, I want to be able to send input and output file names to each plugin to assemble a pipeline **DONE**
* User Story #3 PLMA-27 [~~PLMA-27~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-27) - As a user, I should able to view all plugins from the pool in a popup window **DONE**
* User Story #4 PLMA-29 [~~PLMA-29~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-29) - As a developer, I want to save temporary input/output files to the config file **DONE**

The team members indicated their willingness to work on the following user stories.

* [Bhavyta Chauhan](https://sp-jira.cis.fiu.edu:8453/display/~bchau004)   
  + [~~PLMA-18~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-18) - As a user, I want to be able to send input and output file names to each plugin to assemble a pipeline **DONE**
  + [~~PLMA-24~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-24) - 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 **DONE**
* [Rishabh Vaidya](https://sp-jira.cis.fiu.edu:8453/display/~rvaid004)
  + [~~PLMA-29~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-29) - As a developer, I want to save temporary input/output files to the config file **DONE**
* [Cesia Bulnes](https://sp-jira.cis.fiu.edu:8453/display/~cbuln004)
  + [~~PLMA-27~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-27) - As a user, I should able to view all plugins from the pool in a popup window **DONE**

### 

### **Sprint 5**

**20190318 : Sprint 5 Planning Meeting Minutes**

Attendees: Trevor Cickovski, Cesia Bulnes, Rishabh Vaidya, Bhavyta Chauhan

Start time: 11:00 am

End time: 12:00 pm

After discussion, the velocity of the team were estimated to be 84 hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story #1 PLMA-18[~~PLMA-18~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-18) - As a user, I want to be able to send input and output file names to each plugin to assemble a pipeline **DONE**
* User Story #2 PLMA-18[~~PLMA-24~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-24) - 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 **DONE**
* User Story #3 PLMA-18[~~PLMA-30~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-30) - As a user, I want to git clone the repo of a selected plugin inside my plugin folder in the background of my webscraping. **DONE**
* User Story #4 PLMA-18[~~PLMA-29~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-29) - As a developer, I want to save temporary input/output files to the config file **DONE**
* User Story #5 PLMA-18[~~PLMA-35~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-35) - 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 **DONE**
* User Story #6 PLMA-18[~~PLMA-33~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-33) - As a developer, I want to be able to cooperate with my team member to send input/output files to each plugin **DONE**
* User Story #7 PLMA-18[~~PLMA-34~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-34) - 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 **DONE**

The team members indicated their willingness to work on the following user stories.

* [Bhavyta Chauhan](https://sp-jira.cis.fiu.edu:8453/display/~bchau004)
  + [~~PLMA-18~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-18) - As a user, I want to be able to send input and output file names to each plugin to assemble a pipeline **DONE**
  + [~~PLMA-24~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-24) - 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 **DONE**
  + [~~PLMA-34~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-34) - 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 **DONE**
  + [~~PLMA-35~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-35) - 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 **DONE**
* [Cesia Bulnes](https://sp-jira.cis.fiu.edu:8453/display/~cbuln004)
  + [~~PLMA-30~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-30) - As a user, I want to git clone the repo of a selected plugin inside my plugin folder in the background of my webscraping. **DONE**
* [Rishabh Vaidya](https://sp-jira.cis.fiu.edu:8453/display/~rvaid004)   
  + [~~PLMA-29~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-29) - As a developer, I want to save temporary input/output files to the config file **DONE**
  + [~~PLMA-33~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-33) - As a developer, I want to be able to cooperate with my team member to send input/output files to each plugin **DONE**

### **Sprint 6**

**20190401 : Sprint 6 Planning Meeting Minutes**

Attendees: Bhavyta Chauhan, Rishabh Vaidya, Cesia Bulnes

Start time: 10 am

End time: 11 am

After discussion, the velocity of the team were estimated to be 84 hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story #1 PLMA-24[~~PLMA-24~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-24) - 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 **DONE**
* User Story #2 PLMA-37[~~PLMA-37~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-37) - As a developer, I want to be able to hash random values for temporary file names **DONE**
* User Story #3 PLMA-38[~~PLMA-38~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-38) - 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! **DONE**
* User Story #4 PLMA-39[~~PLMA-39~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-39) - As a user I want to be able to add or delete boxes in order to drag plugins as much as I want**DONE**
* User Story #5 PLMA-34[~~PLMA-34~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-34) - 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 **DONE**

The team members indicated their willingness to work on the following user stories.

* [Bhavyta Chauhan](https://sp-jira.cis.fiu.edu:8453/display/~bchau004)
  + [~~PLMA-24~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-24) - 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 **DONE**
  + [~~PLMA-34~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-34) - 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 **DONE**
* [Rishabh Vaidya](https://sp-jira.cis.fiu.edu:8453/display/~rvaid004)
  + [~~PLMA-37~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-37) - As a developer, I want to be able to hash random values for temporary file names **DONE**
* [Cesia Bulnes](https://sp-jira.cis.fiu.edu:8453/display/~cbuln004)   
  + [~~PLMA-38~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-38) - 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! **DONE**
  + [~~PLMA-39~~](https://sp-jira.cis.fiu.edu:8443/browse/PLMA-39) - As a user I want to be able to add or delete boxes in order to drag plugins as much as I want **DONE**

### **Sprint 7**

There are no development-related user stories left. Therefore we are now working on our Final Document, powerpoints and posters for this Sprint.

**System Design**

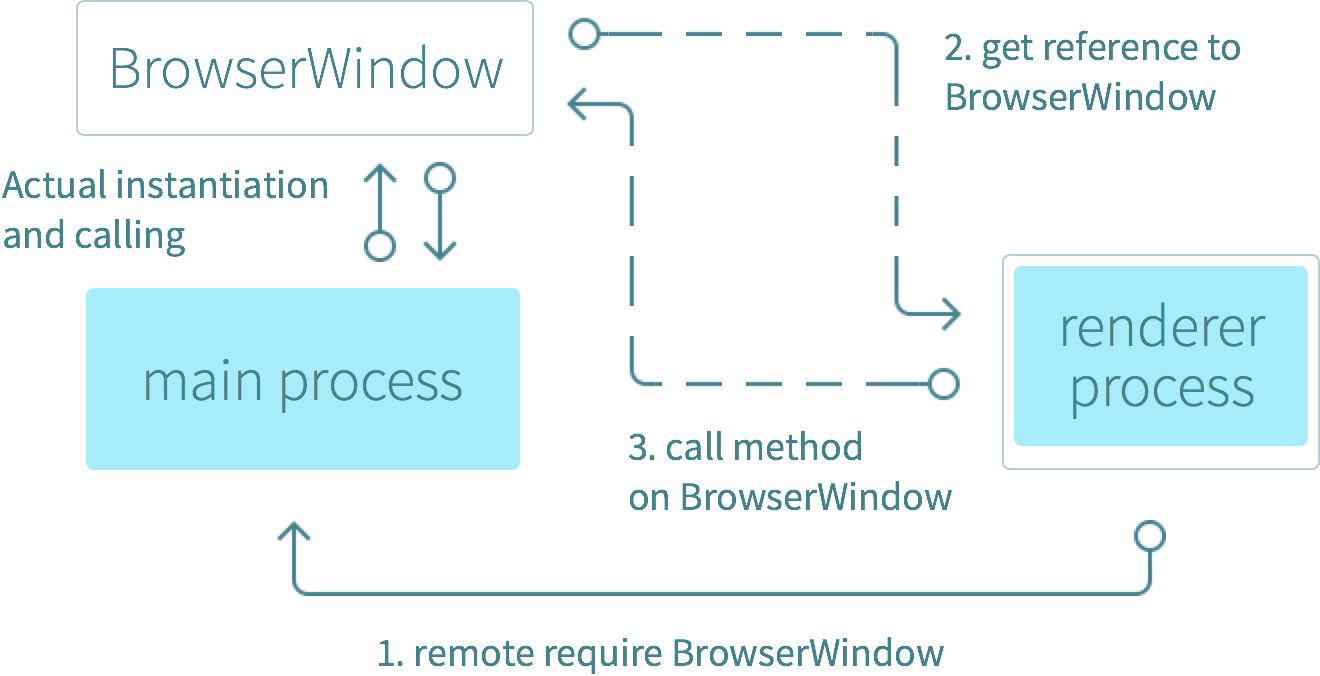
Electron is a framework for creating native applications with web technologies like JavaScript, HTML, and CSS. It takes care of the hard parts so you can focus on the core of your application. Electron allows developers to create cross-platform applications. Some of the most popular applications built on electron are Skype, Github Desktop, Atom, Visual Studio code and many more.

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

**Architectural Patterns**

The process that runs the main script in Electron is called the main process. There is only one main process, never more than one. This process can display a Graphical User Interface by creating web pages. Electron creates web pages using Chromium by using its multi-process architecture. The processes that run the web pages are called rendering processes.

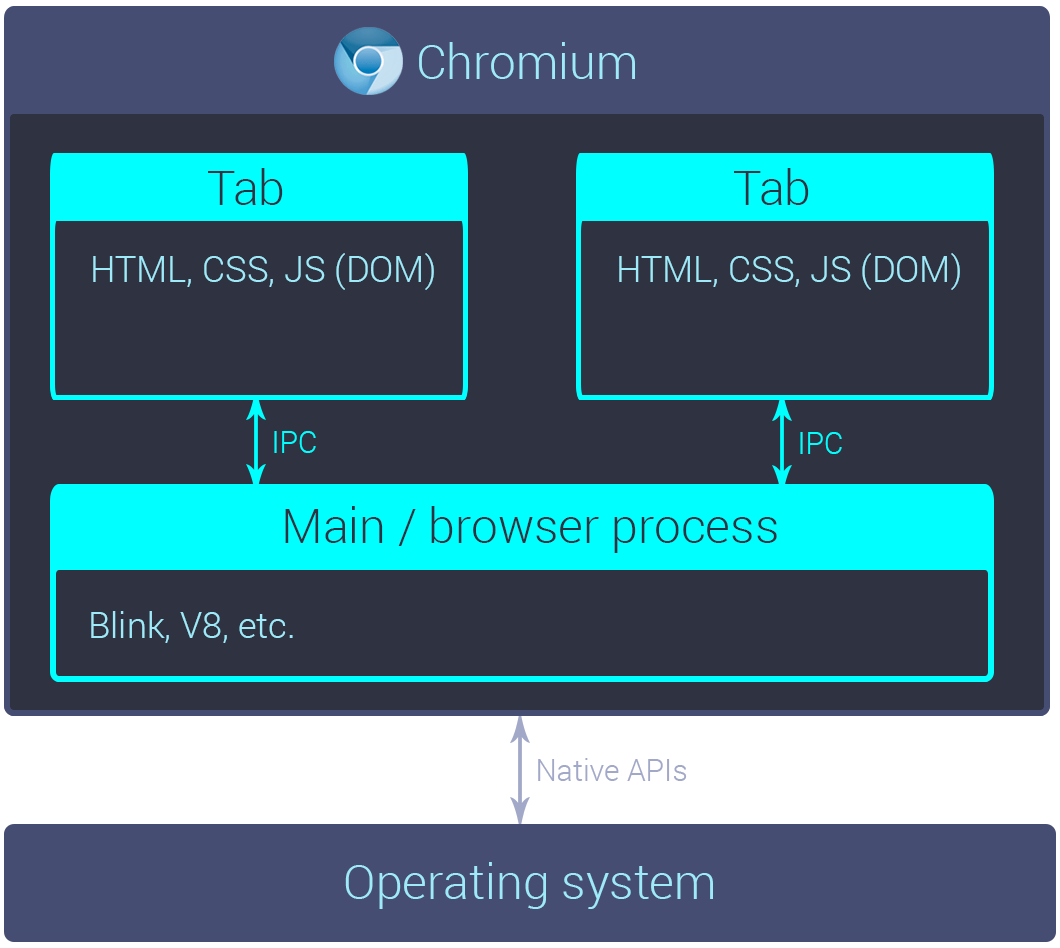
Electron uses Node.js API’s in web pages allowing lower level operating system interactions. The main process creates web pages by creating BrowserWindow instances. Each BrowserWindow instance runs the web page in its own renderer process. When a BrowserWindow instance is destroyed, the corresponding renderer process is also terminated. The main process manages all web pages and their corresponding renderer processes. Each renderer process is isolated and only cares about the web page running in it.



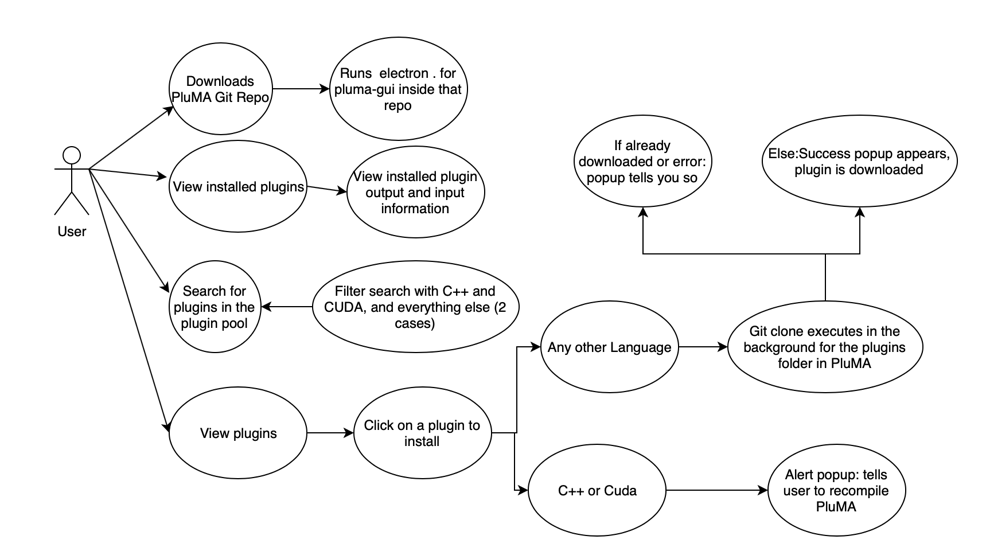
**System and Subsystem Decomposition**

Electron is an open source library developed by GitHub for building cross-platform desktop applications with HTML, CSS, and JavaScript. Electron accomplishes this by combining Chromium and Node.js into a single runtime and apps can be packaged for Mac, Windows, and Linux.

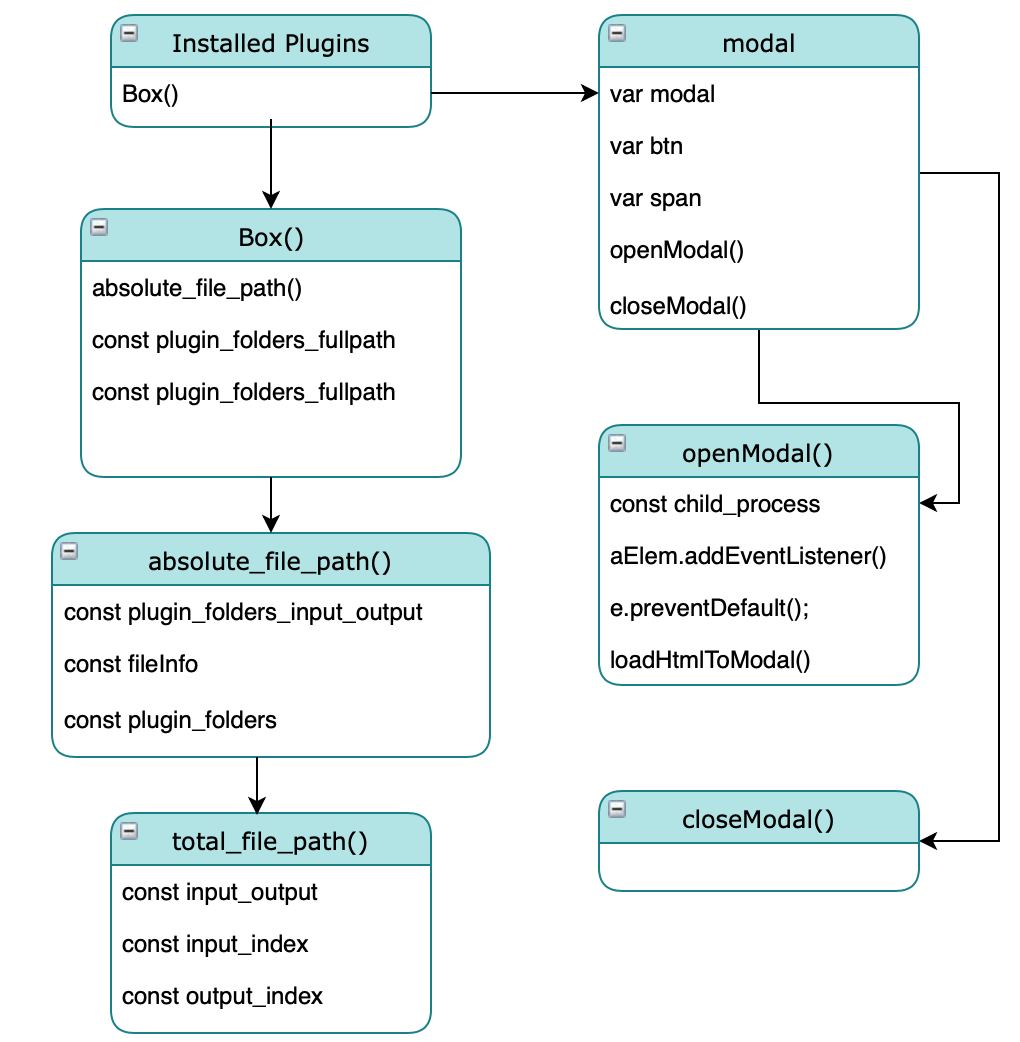
Chromium is a Webkit based web browser with the V8 javascript engine. It supports all the usual browser and DOM APIs and thus is good for making web pages and not good at interacting with the underlying system. Node.js was built by striping out the V8 engine, making a headless command line application, and adding extensive APIs to access the file system. Electron replaces the V8 engine used in Chromium with the new more general purpose oriented one of Node.js. It exposes a few extra APIs to node.js to allow for opening chromium windows.



**Deployment Diagram & Design Patterns**



The following is the deployment diagram for accessing the installed plugins and accessing the scraping of the plugin pool, and cloning of the plugins.

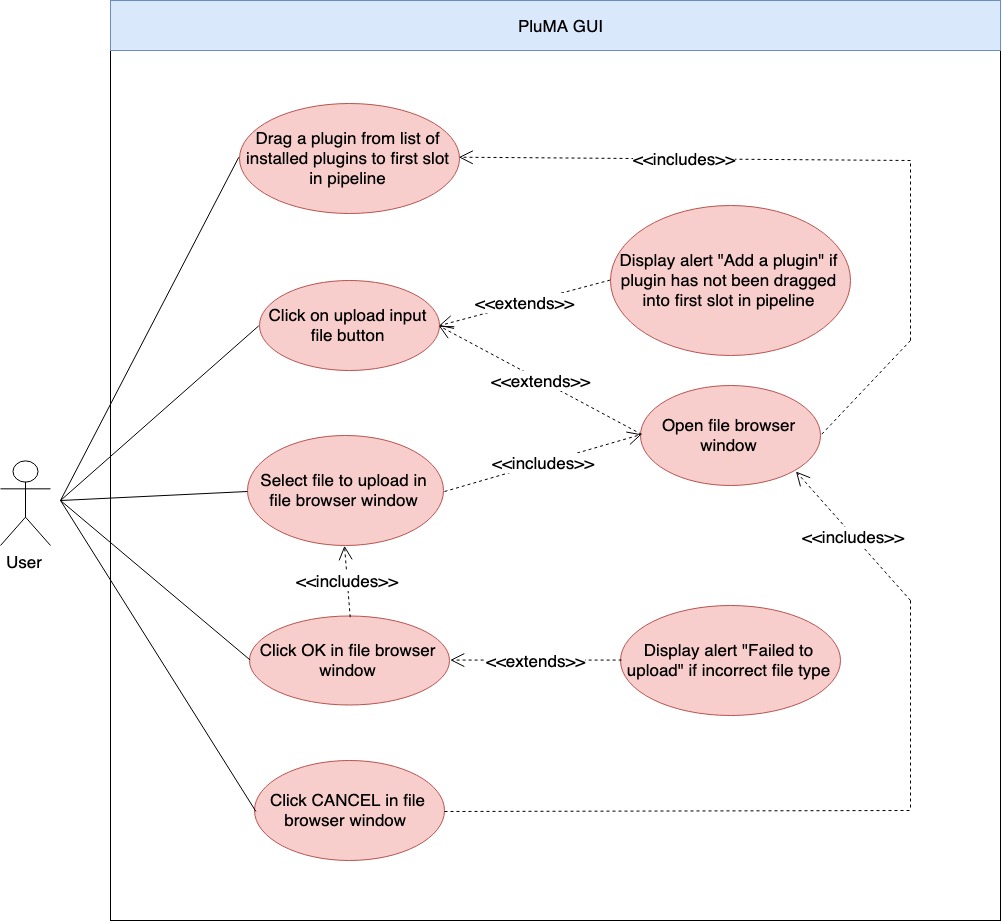


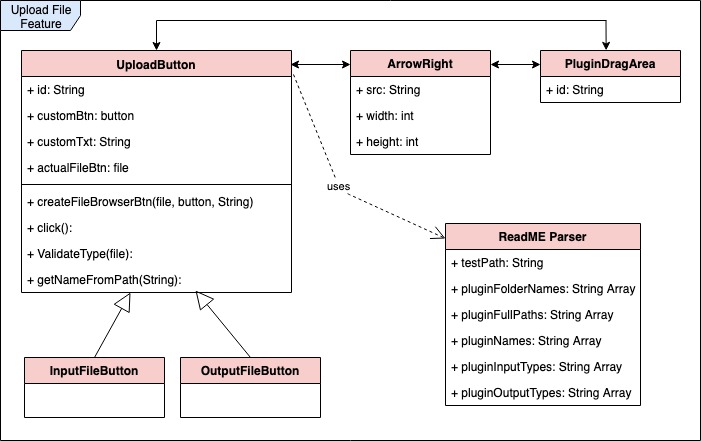
The following are the deployment diagrams for dragging and dropping the plugins into the pipeline, removing unwanted plugins, and saving the config file.

## 

## 

The following are the deployment diagrams for the upload file feature.





# **System Validation**

The following is the validation diagram for accessing the installed plugins and accessing the scraping of the plugin pool, and cloning of the plugins.

# 

# 

# 

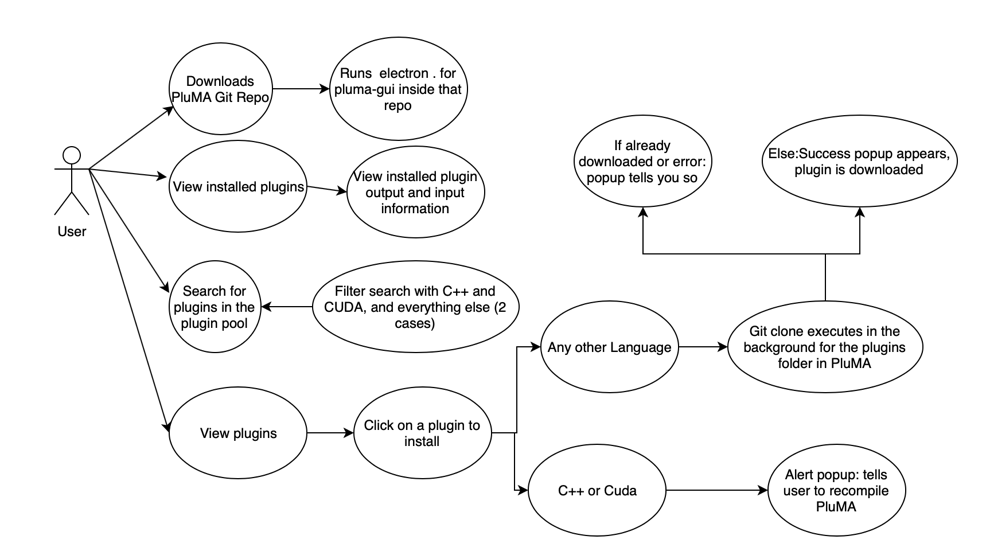
# 

# **Appendix**

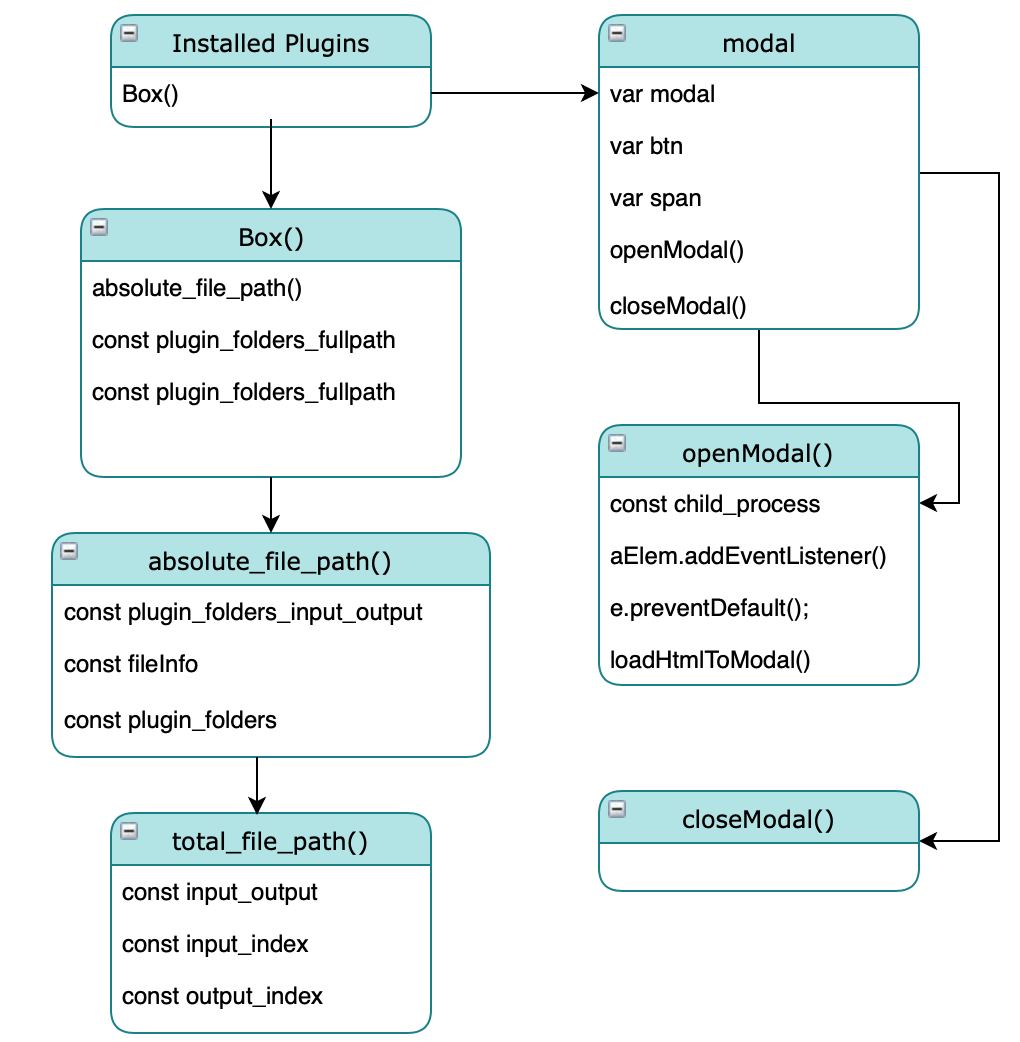
## **Appendix A - UML Diagrams**

**Pipeline Cloning, Installation and Plugin Pool Diagrams:**

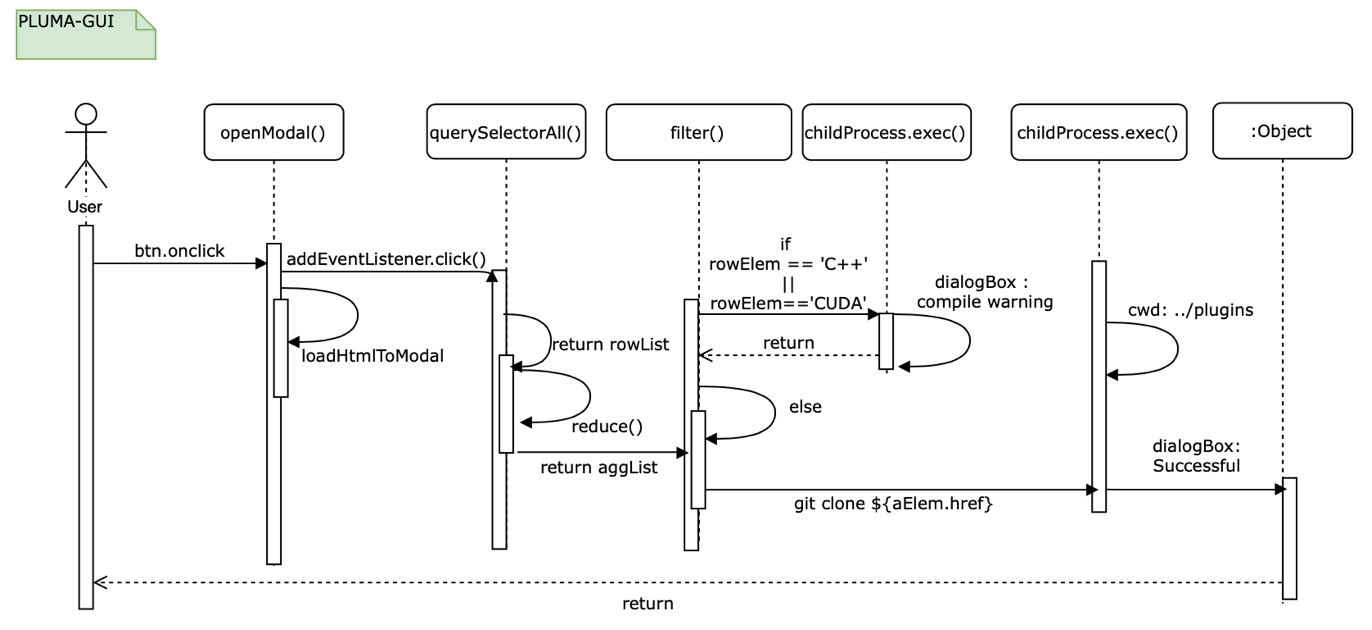
Use Case Diagram:



Class Diagram:

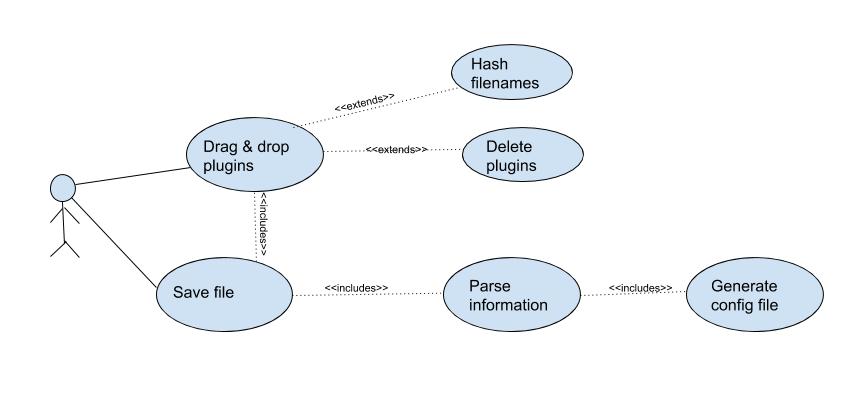


Sequence Diagram:

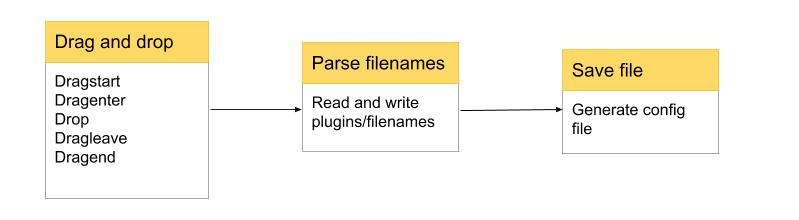


**Drag and Drop Diagrams:**

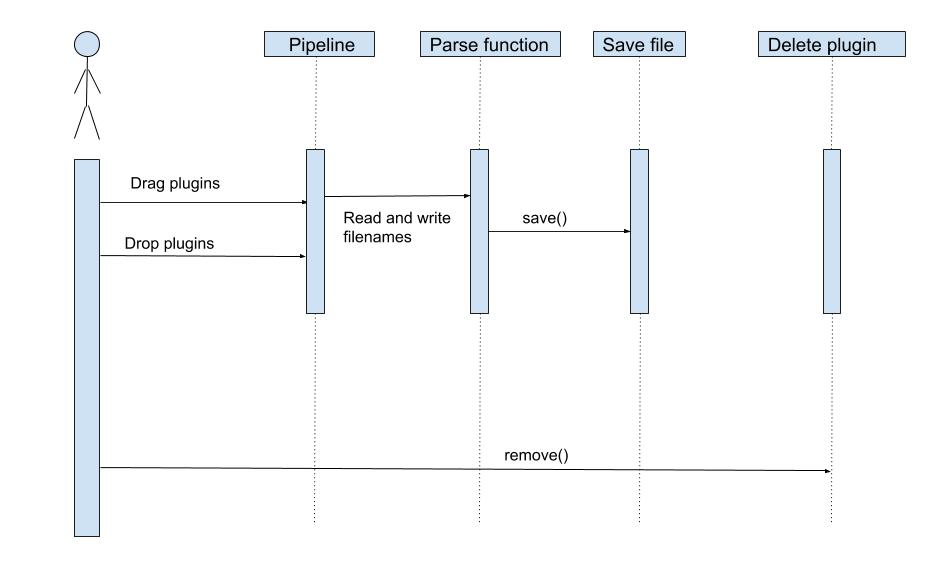
Use Case Diagram:



Class Diagram:

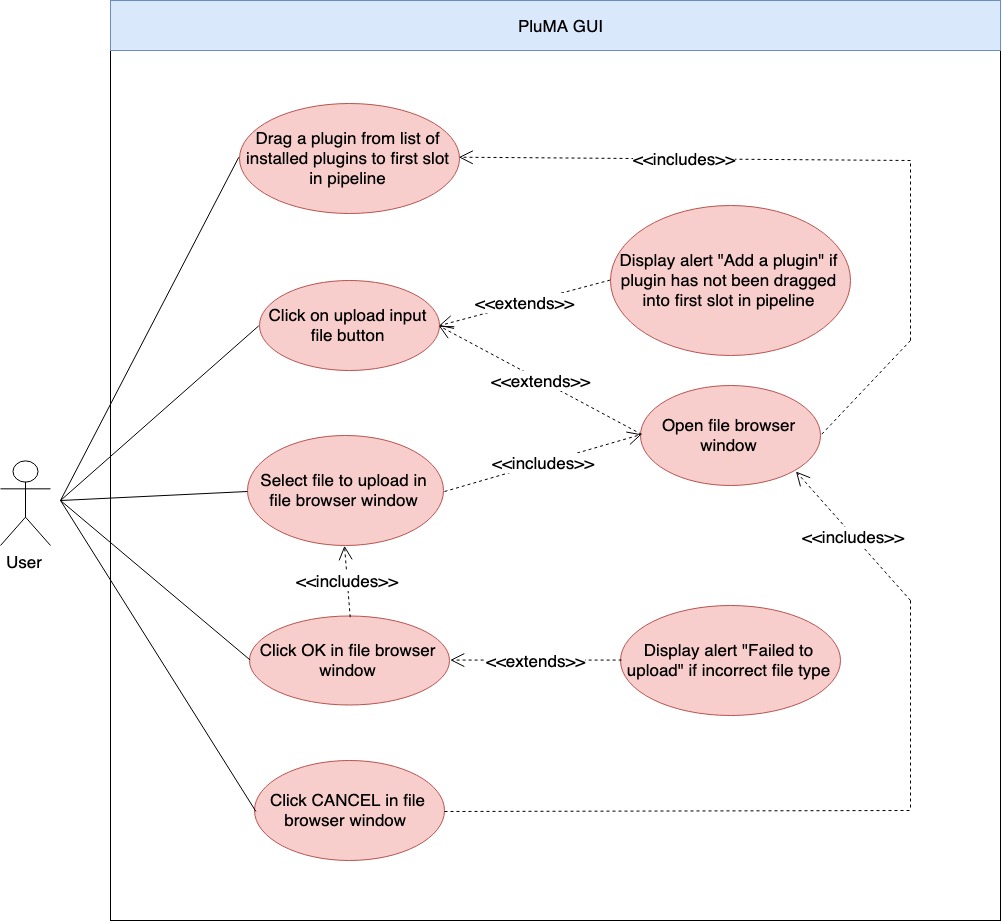


Sequence Diagram:

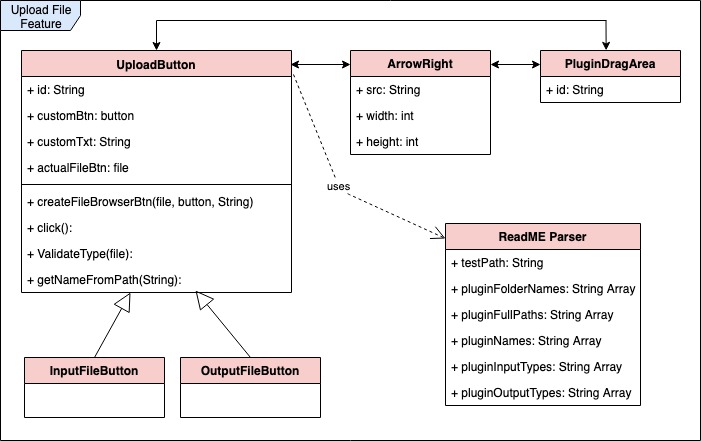


**Upload File Diagrams:**

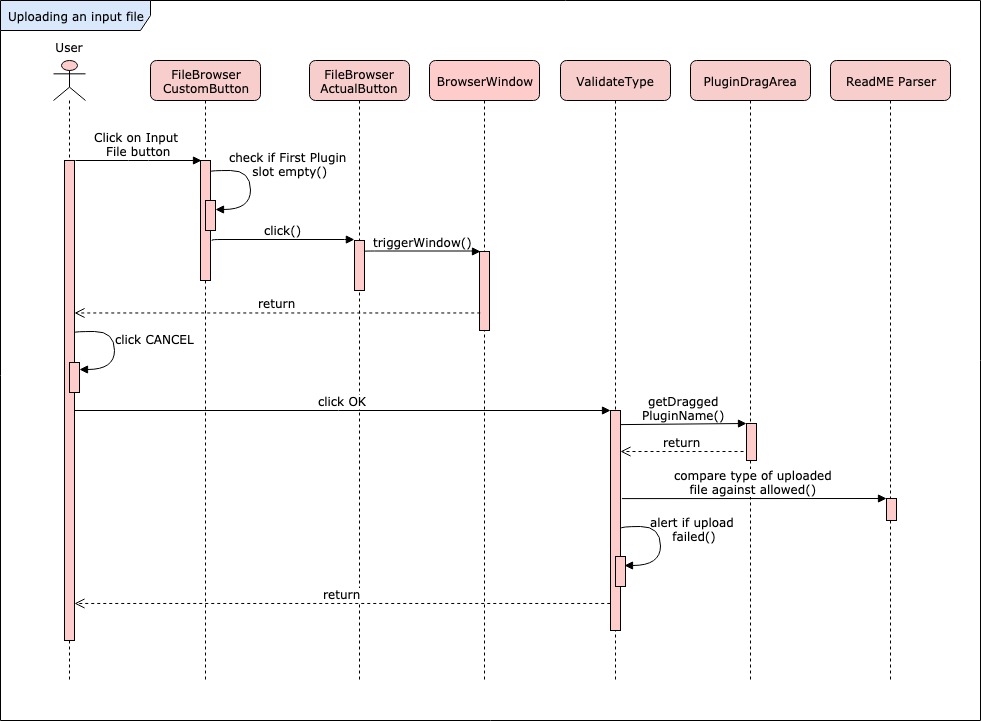
Use Case Diagram:



Class Diagram:



Sequence Diagram:



## 

## 

## 

## 

## 

## 

**Appendix B - Sprint Review Reports**

**Sprint 1:**

20190204 : Sprint 1 Review Meeting Minutes

Attendees: Trevor Cickovski, Cesia Bulnes, Rishabh Vaidya, Bhavyta Chauhan

Start time: 9:30 am

End time: 10:00 am

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* + User Story #1 PLMA-1
  + User Story #2 PLMA-2 
  + User Story #3 PLMA-3 
  + User Story #4 PLMA-4 
  + User Story #5 PLMA-5
  + User Story #6 PLMA-6
  + User Story #7 PLMA-7
  + User Story #8 PLMA-8
  + User Story #9 PLMA-9

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

N/A

**Sprint 2:**

20190219 : Sprint 2 Review Meeting Minutes

Attendees: Rishabh Vaidya, Cesia Bulnes, Bhavyta Chauhan

Start time: 10:00 am

End time: 11:30 am

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story 
* User Story 

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* User Story
* User Story 

**Sprint 3:**

20190304 : Sprint 3 Review Meeting Minutes

Attendees: Bhavyta Chauhan, Rishabh Vaidya, Cesia Bulnes

Start time: 9:30 am

End time: 10:00 am

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story #1 PLMA-16 
* User Story #2 PLMA-19 
* User Story #3 PLMA-22 
* User Story #4 PLMA-23 
* User Story #5 PLMA-25 
* User Story #6 PLMA-26 

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* User Story #1 PLMA-24 
* User Story #2 PLMA-18 
* User Story #3 PLMA-27 

**Sprint 4:**

20190318 : Sprint 4 Review Meeting Minutes

Attendees: Bhavyta Chauhan, Rishabh Vaidya, Cesia Bulnes

Start time: 10:00 am

End time: 11:00 am

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story #1 PLMA-16 

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* User Story #1 PLMA-24 
* User Story #2 PLMA-18 
* User Story #3 PLMA-27 

**Sprint 5:**

20190401 : Sprint 5 Review Meeting Minutes

Attendees: Bhavyta Chauhan, Rishabh Vaidya, Cesia Bulnes

Start time: 10:00 am

End time: 11:00 am

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story #1 PLMA-29 
* User Story #2 PLMA-36 
* User Story #3 PLMA-31 
* User Story #4 PLMA-35 
* User Story #5 PLMA-33 
* User Story #6 PLMA-18 
* User Story #7 PLMA-30 

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* User Story #1 PLMA-24 
* User Story #2 PLMA-34 

**Sprint 6:**

20190415 : Sprint 6 Review Meeting Minutes

Attendees: Bhavyta Chauhan, Rishabh Vaidya, Cesia Bulnes

Start time: 10:00 am

End time: 11:00 am

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story #1 PLMA-29 
* User Story #2 PLMA-36 
* User Story #3 PLMA-31 

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* All user stories have been completed

## 

## 

## 

## 

## 

## **Appendix C - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents**

Installation Guide:

<https://docs.google.com/document/d/1_enogK3s_XLCtxuUZgs7ClcUUhPtUhyd0VC3nq4wHsU/edit?usp=sharing>

User Manuals:

<https://docs.google.com/document/d/1yfVIy-VDmLbUYwTlKqml-tWt8laaFO0VU4OPIt0lDgE/edit?usp=sharing>

Shortcomings/Wishlist Document:

<https://docs.google.com/document/d/1z6931TCC92PefLydhIVCMHDrUM-jNfeNSsZQrYB0dhI/edit?usp=sharing>

Readme:

<https://docs.google.com/document/d/1ysyEigyWXrSbzIr07EA16mg-x3iddQQOP-Yu9N54N5Q/edit?usp=sharing>

# 

# 

# 

# **References**

https://electronjs.org/