**BRIEF DESCRIPTION OF SYSTEM**

Client: Company which is using the system

User: Employee of the client company which has login credentials to the system

Plot: Unit of management comprising one or more polygons of coordinates

AOI: Single polygon of coordinates. One or more AOI make a plot

Asset: One satellite image

Job: Celery task

Alert: Records of alerts

AlertConfig: Settings of what will trigger an Alert.

**BRIEF DESCRIPTION OF SYSTEM**

AD HOC

1. User will provide Plots in geoJSON form
2. Each Area of Interest may comprise multiple polygons, but satellite data API can only ship contiguous polygons
3. Django system keep track of the Areas of Interest (AOI) polygons as well as the characteristics of each of them (code, planted date etc)

EACH DAY

1. The system should go to fetch new data from the Planet.com server for each AOI. The data we need is:
   1. PSScene4Band at all cloud cover
2. When the geotiffs are ready, download them and store them in file storage (Amazon S3)
3. After the geotiffs have been downloaded, run the various calculations of NDVI, EVI, DVI etc etc and save the calculated raster results into CSV files in File Storage (Amazon S3)
4. We will then run an analysis script to set a benchmark for comparable plots. (PoC code is in APP Trial Data Preparation.ipynb)
5. Each plot will then be compared against this benchmark. (PoC code is in APP Trial Analysis.ipynb)
6. If it goes above set thresholds, save an alert record in Django, attaching the comments and geojson (not yet implemented)
7. User should be able to log into system to look for the plot/date and download the associated raw geotiff and calculated indices CSV whenever they wish. (not yet implemented)

EACH WEEK

1. An email is sent to a list of users subscribed to alerts from a list of Plots. Here, they can directly download the geojson of the alert area. They can also feedback that the alert was not accurate. (not yet implemented)

**RESOURCES**

Planet – Satellite Imagery

You can sign up for a 14-day trial for planet at: <https://www.planet.com/trial/>

API documentation: <https://developers.planet.com/docs/api/>

API Key is at plantedgewd/settings.py

**PROJECT STRUCTURE**

1. Core -> The main function and building block of the project. (Gaia.py for downloading asset & interact with planet; Athena.py for calculation; Theia.py for presentation: colormap, graph, etc)

2. models.py -> The representation of the database model.

3. admin.py -> Django Admin

4. Facade -> Where you build products out of the building block. Implement business logic and flow here.

**NEED TO DO**

1. Need to create table Plots. When we started the project, we thought each area will comprise only 1 polygon, but that turned out to be false. Therefore, we need to introduce the concept of Plot, which groups multiple AOIs into one analytical block. Fields:
   1. ID (primary key)
   2. client\_plot\_ID: free-text
   3. client\_plot\_description: free-text
   4. name: free-text
   5. status: free-text
   6. description: free-text
   7. constituentAOIs: Foreign Key (from AOI)
   8. client: Foreign Key
   9. weed\_ enable: boolean
   10. forest\_health\_enable: boolean
2. Edit AOI table to add:
   1. variant: free text
   2. date\_planted: datetime
3. Create Alert Table. Stores list of alerts. *Please log which user edited any record in this table* and at what time, and what did they edit.
   1. ID (primary key)
   2. plot: foreign key
   3. alert\_date: datetime
   4. type: weed/forest\_health
   5. status: active/verified/false\_alarm
   6. notes: freetext
   7. area: geojson text

{

"type": "Feature",

"geometry": {

"type": "Polygon",

"coordinates": [

[

[100.0, 0.0], [101.0, 0.0], [101.0, 1.0],[100.0, 1.0], [100.0, 0.0]

]

]

},

"properties": {

“alert\_type": “weed",

“variant\_code": “AC”

“plant\_date”:”06\_2017"

}

}

1. Send via email summary of alerts to all users in a client for all alerts. This is the format

**Alert Date: 22 July 2018 (Sunday)**

**NEW ALERTS**

**Alert Type GPS Geojson Wrong? Let us Know!**

Weed -2.9200, 105.6893 Download Report False Positive

Weed -2.9403, 105.8832 Download Report False Positive

Weed -3.2040, 105.6896 Download Report False Positive

Forest Health -2.8202, 105.6990 Download Report False Positive

Forest Health -2.9209, 102.3322 Download Report False Positive

Forest Health -2.9504, 103.0880 Download Report False Positive

Forest Health -2.8200, 105.6895 Download Report False Positive

**ACTIVE ALERTS FROM PREVIOUS WEEKS**

**Alert Type GPS Geojson Wrong? Let us Know!**

Weed -2.9200, 105.6893 Download Report False Positive

Weed -2.9403, 105.8832 Download Report False Positive

Weed -3.2040, 105.6896 Download Report False Positive

Forest Health -2.8202, 105.6990 Download Report False Positive

Forest Health -2.9209, 102.3322 Download Report False Positive

Forest Health -2.9504, 103.0880 Download Report False Positive

Forest Health -2.8200, 105.6895 Download Report False Positive

1. Create client’s admin backend (UI in the process, but for now base it off Django default UI)
   1. Create new plot: screen for user to upload a geojson. Form fields
      1. Geojson file upload
      2. Name
      3. Status
      4. Description
      5. Variant (Choose from Epel/Acra)
      6. Plot ID:
      7. Planted Date:
      8. Monitor Weed: Yes/No
      9. Monitor Forest Health: Yes/No
      10. Result: Create new plot and new AOIs with corresponding name, status, description linked to the new plot
   2. Edit plot:
      1. Change name
      2. Change status
      3. Change description
      4. Change Variant
      5. Change PlotID
      6. Change Planted Date
      7. Change Monitor Weed
      8. Change Monitor Forest Health
      9. Cannot change geojson – this needs to be done be deactivating this plot and creating a new plot.
      10. Other changes need to update individual AOIs too
   3. Download Asset
      1. Search by client\_plot\_ID
      2. Filter by date
   4. View alert history
      1. Search by client\_plot\_ID
      2. Filter by date
      3. Download geojson
      4. Download associated Asset files
2. Set tasks for:
   1. When create a new AOI, pull data from the last year
   2. Pull data everyday for all satellite images from the day before and run calculations
   3. Run benchmarking everyday
   4. Run analysis on benchmark everyday
      1. Save plots to each folder
      2. Save colormaps to each folder
      3. Create alerts
3. Convert

**OTHER NOTES**

**Executing**

To run Django server

python manage.py runserver 0.0.0.0:8000 &

To run the product on façade, run the celery worker (the command are in the readme)

celery -A plantedge.tasks worker --loglevel=ERROR --concurrency=10 &

**Note: If SHP file is provided:**

1. Convert shp file to geojson (saved into file ). i use <https://ogre.adc4gis.com/>
2. Then depending on how client structure the shp (it usually different everytime), write script to read the geojson and insert through Django model. I dont include it in the repo since usually every shp have their own format.