PROJECT SILVAFLUX

Design Review 3

Big Data Computing and Interface for Tropical Forest Regeneration

By Team Clean Carbon

Introduction

Frontend team:

Curtis McHone - Team Lead
Justin Stouffer
Jonathan Bloom

Backend team:

Richard McCue
Shayne Sellner

Team Sponsor:

Allie (Alexander) Shenkin

Team Mentor:

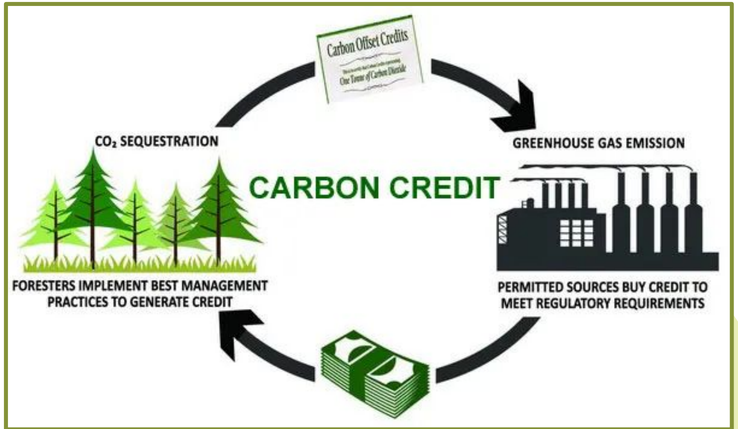
Vahid Nikoonejad Fard

Introduction

- Our sponsor Allie (Alexander) Shenkin and his team have discovered a new climate cooling service that allows for 30% more carbon credits to be sold for a designated plot of land

- Carbon credits are purchasable credits that landowners and project developers can sell. These carbon credits directly correlate to the amount of carbon dioxide a certain plot of land takes in

- Developers and landowners sell these credits to help businesses or corporations offset their carbon footprint



Introduction

- Our team has been asked to create an application that is able to calculate the amount of carbon dioxide a certain area will uptake using this new discovery
- As a team we hope that our final application makes buying and selling carbon credits more profitable and accessible to the average person as well as revolutionize the way carbon credits are sold and predicted

Problem Statement

- There is not currently a way to accurately predict the amount of CO_2 uptake for a plot of land.
- Because of this land developers cannot predict how much money they are going to make or if the project is even going to be worth their time
- Allie's discovery makes investing in reforestation projects much more profitable, helping not only the investor but the planet, but what is the best way to implement this
- The current software prototype that is used to calculate the CO_2 prediction is simply too slow, inefficient, and not user friendly

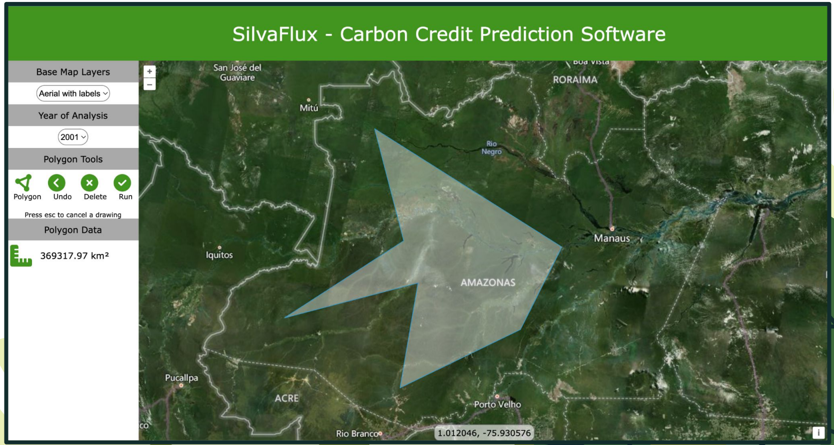
Solution Overview

Front End

- Web interface
  - Django Web Framework
  - Utilizes the OpenLayers javascript library
  - Easy to use map interface
    - Bing Maps API base maps
    - Ability to Zoom, Pan, Scale
- User friendly and responsive
- Ability to draw a polygon
- Measure area
- Sends the polygon to the prediction script

Bing Maps

OpenLayers



Solution Overview

Back End

- High performance linux based server
- Python based prediction system
  - Static raster simulation
- Raster layer storage
- Send results to front end

Database

- PostgreSQL database
- Stores user login information
- User access control
- Expandability for user query storage

Requirement/Architecture Overview

Requirements Acquisition

- Weekly client meetings
- Initial project description and overview
- External research & sponsor recommendations

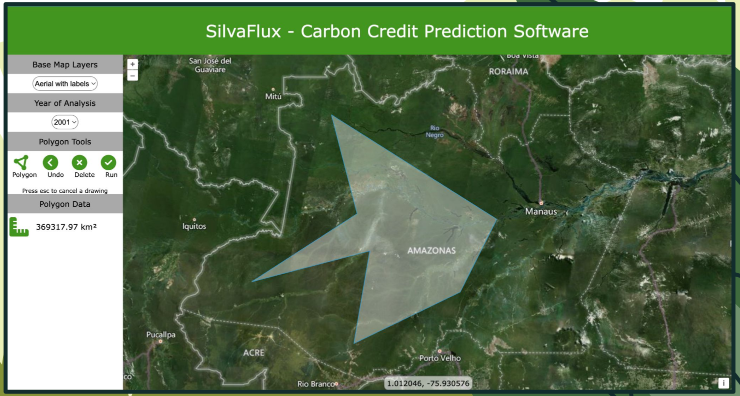
Requirements

- Simple web interface with an interactable map
- Send polygon to backend
- Computationally efficient backend that computes carbon uptake
- Database with encryption to store user account information

Architecture/Implementation Overview

Front End

- Django Framework
- OpenLayers map using bing maps baselayers
- Communicates with Django Postgre database to verify login information
- Polygon tool on map built into OpenLayers
- Calls backend script with polygon coordinates and year selected



Architecture/Implementation Overview

Backend

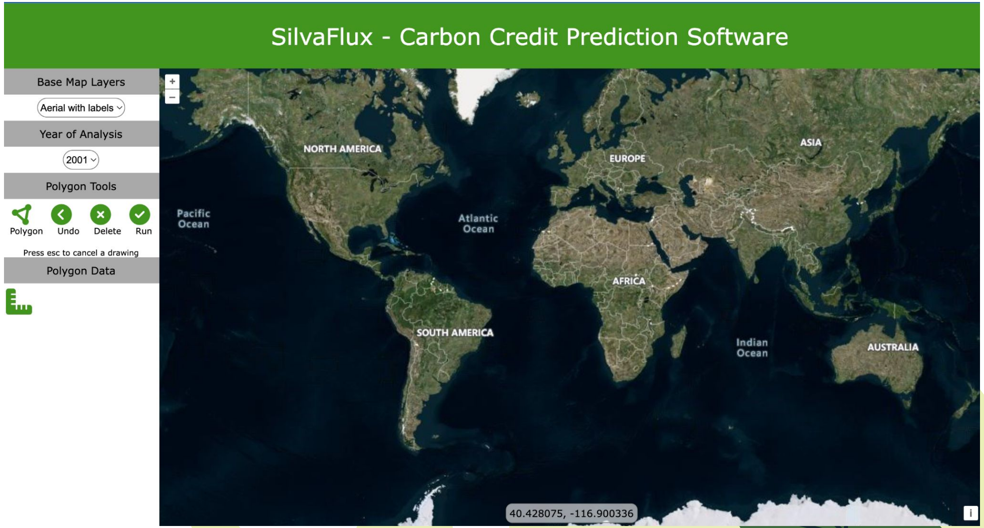
- Precomputed rasters for years 2014-2021 for global prediction
- Python script to compute the carbon uptake on the polygons plot of land
- Return carbon uptake back to front end

Database

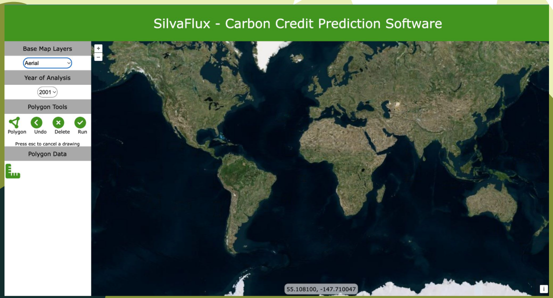
- Stores user login information utilizing encryption
- (Stretch goal) Store user queries for usage based billing

Prototype

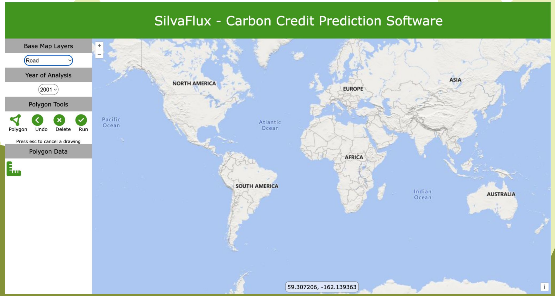
Silvaflux Login
Username
Password
Submit



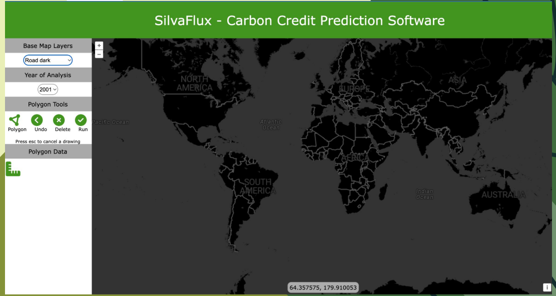
SilvaFlux - Carbon Credit Prediction Software
Base Map Layers
Aerial with labels
Year of Analysis
2001
Polygon Tools
Polygon
Undo
Delete
Run
Press esc to cancel a drawing
Polygon Data
NORTH AMERICA
EUROPE
ASIA
Pacific Ocean
Atlantic Ocean
AFRICA
SOUTH AMERICA
Indian Ocean
AUSTRALIA
40.428075, -116.900336



SilvaFlux - Carbon Credit Prediction Software
Base Map Layers
Aerial
Year of Analysis
2001
Polygon Tools
Polygon
Undo
Delete
Run
Press esc to cancel a drawing
Polygon Data
55.108100, -147.710047

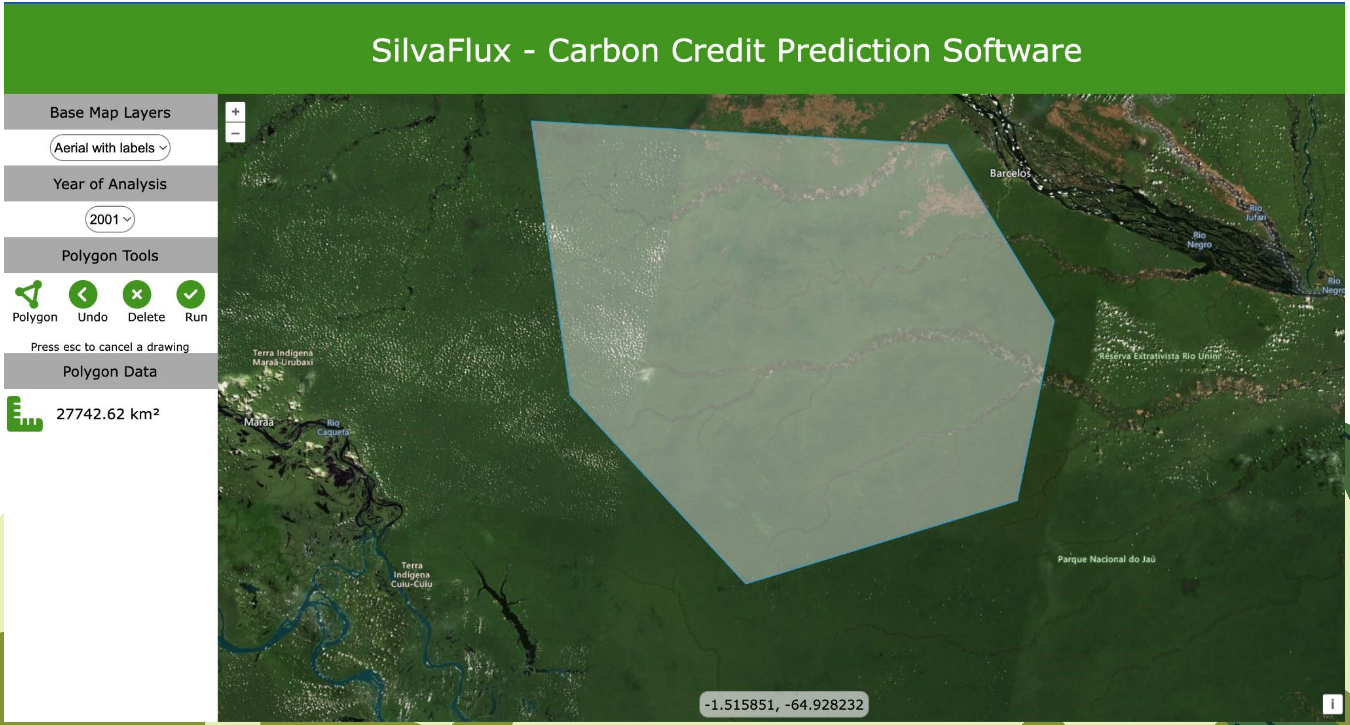


SilvaFlux - Carbon Credit Prediction Software
Base Map Layers
Road
Year of Analysis
2001
Polygon Tools
Polygon
Undo
Delete
Run
Press esc to cancel a drawing
Polygon Data
NORTH AMERICA
EUROPE
ASIA
Pacific Ocean
Atlantic Ocean
AFRICA
SOUTH AMERICA
Indian Ocean
AUSTRALIA
59.307206, -162.139363



SilvaFlux - Carbon Credit Prediction Software
Base Map Layers
Road dark
Year of Analysis
2001
Polygon Tools
Polygon
Undo
Delete
Run
Press esc to cancel a drawing
Polygon Data
64.357575, 179.910053

Prototype



Annual tons CO_2 uptake: -0.323998

Challenges

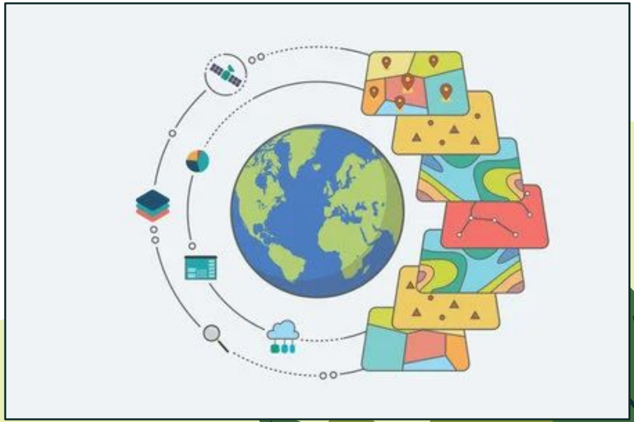
Backend

Raster storage:

- Our rasters are up to ~45GB in size, up to 21 rasters
- 45GB x 21 rasters = ~945GB total storage needed

Runtime:

- Takes very long to read the raster into memory



Frontend

Local CSS:

- CSS for OpenLayers is not working locally

Django:

- Could not start up the web server

Resolutions

Backend

Storage Solutions:

- Increase our AWS storage size
- Support less rasters

Runtime Solutions:

- Reading only sections of the raster into memory

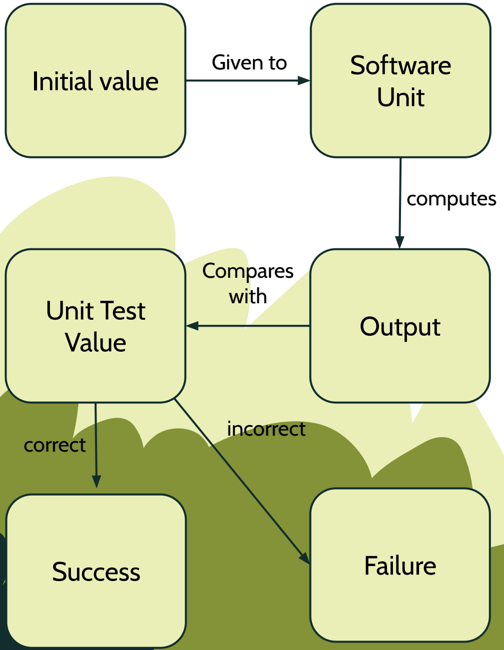
Frontend

CSS and Django Solutions:

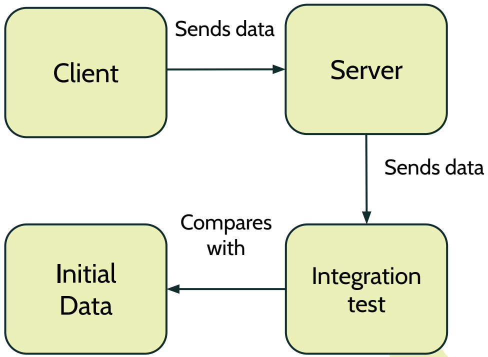
- More research
  - On static CSS files and linking
  - On Django file architecture and server startup

Testing Plan

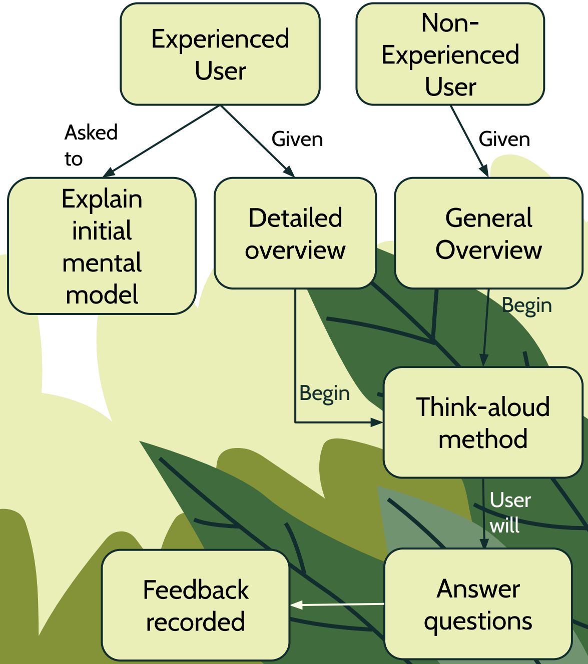
Unit testing



Integration testing



Usability testing



Testing Plan

Unit testing

Ensures our software and code is accurate and reliable
Tested Units:

- Correct coordinates
- Map projection
- CO_2 uptake amount
- Year of prediction selection

Once Unit testing is complete, our team will know exactly which functions/isolated pieces of software work correctly

Integration testing

Four different tests:

- Query results are correctly sent over to postgres from Django
- Logout function restricts access to home page
- Prediction page vulnerabilities
- Frontend coordinates correctly sent and match to backend coordinates

Once Integration testing is complete, our team will know exactly which software systems are correctly communicating with one another

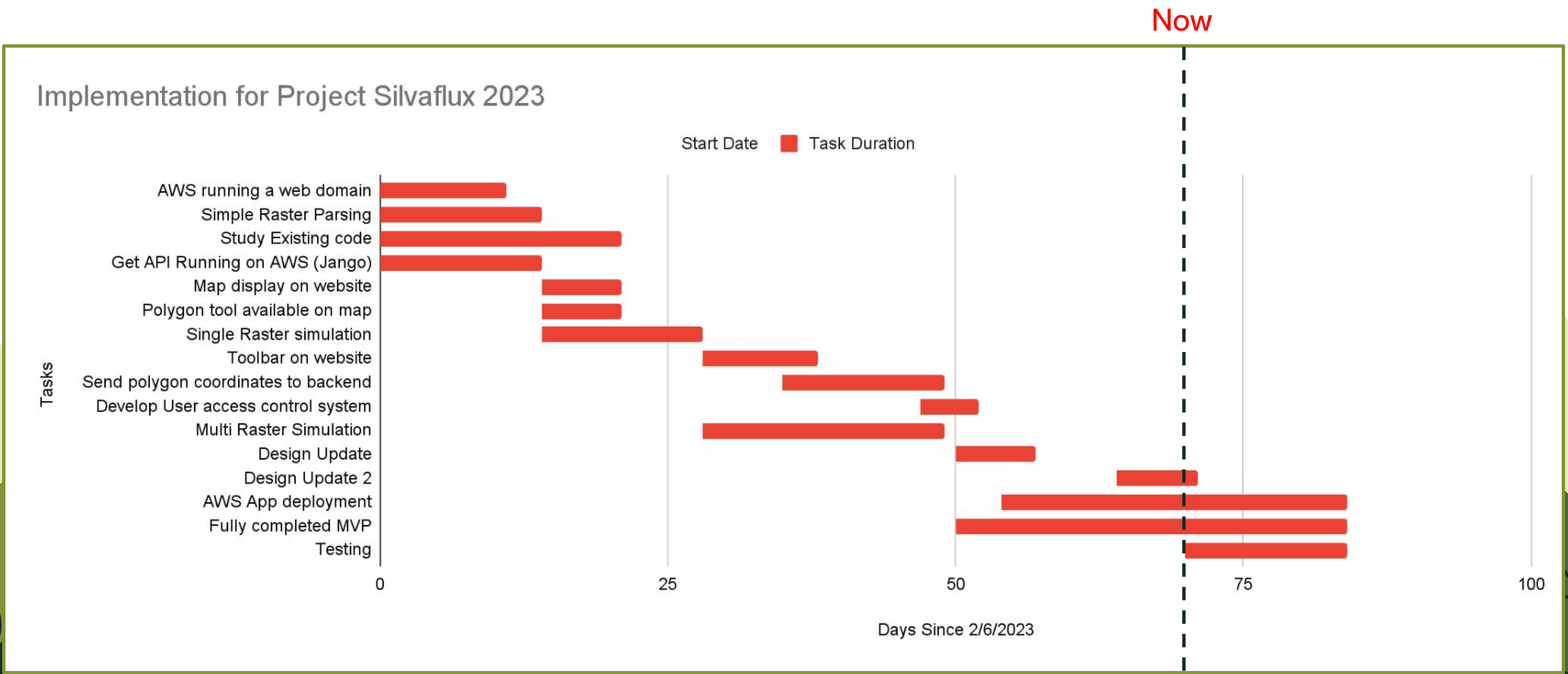
Usability testing

Users will provide feedback on:

- Initial Impressions
- Ease of use, navigation around interface
- Expected functionality of features
- Interpretation of results, useful or redundant data

Once Usability testing is complete, our team will have a better understanding of what users are thinking when using our system, and the different functionalities that they find important

Schedule


Now
Implementation for Project Silvaflux 2023
Start Date
Task Duration
AWS running a web domain
Simple Raster Parsing
Study Existing code
Get API Running on AWS (Jango)
Map display on website
Polygon tool available on map
Single Raster simulation
Toolbar on website
Send polygon coordinates to backend
Develop User access control system
Multi Raster Simulation
Design Update
Design Update 2
AWS App deployment
Fully completed MVP
Testing
Tasks
0
25
50
75
100
Days Since 2/6/2023

Conclusion

Our current implemented solution includes:

- A responsive and convenient web interface front-end with authenticated user access control system (Django)
- Fast and secure data transmission between our web interface and backend (Django)
- Secure database that holds all of the user access information (PostgreSQL)
- Correct data sent back to the user with additional useful information

Functional Requirements:

Simple web interface with a interactable map, with fast communication with backend

Computationally efficient backend that will run the prediction on a specified plot of land

Database with encryption to store user account information

Conclusion

System Performance

- Instantaneous Login/Logout connection with database
- Under 10 sec runtime for a polygon of 1000 hectares

Architecture Implementation

- Prediction computation located on the backend, implemented in Django framework
- High performance AWS backend server
- PostgreSQL database used to store authenticated user information

Challenges/Resolutions:

- Computational performance was accelerated to meet runtime requirements
- AWS storage was upgraded to hold a larger amount of rasters
- Django and Web front end are up and running with working CSS