Irrigation Web App Documentation

LC Mueller Irrigation Group

David Nam

May 31, 2019

Table of Contents

Back to Top

Project Identification

The Mueller Irrigation Group at Lethbridge College needs a system that helps researchers and farmers gather more soil moisture information of any field of interest. A web application can achieve this need in an affordable and accessible mean for irrigation research and planning.

Project Purpose

The purpose of the web app is to evaluate the shallow and subsurface values of soil moisture in a field and review the trends of the field in the Soil Moisture Active Passive (SMAP) dataset. Similar to the Irrisat-Cloud web app (irrisat-cloud.appspot.com), this web app will have the same features and will be open to future expansion to include Canadian weather data.

System Scope

The following are the main requirements for the web application:

- User login
- Load a map UI
- Draw polygons on the map UI
- Database to save the polygons as a 'field' under the user's account
- Include these Datasets: Landsat 7/8, SMOS, SMAP
    - Optional: Canadian Crop
- Must allow data processing:
    - Select a date to narrow down the datasets
    - Calculating new band values (NDVI, Kc, ETc) from existing datasets
    - Create charts (graphs and tables) from calculated bands
    - Download images and dataset values for NDVI, Kc, and ETc
- Must include a help/quick-start feature

Any additional requests from Mueller Irrigation Group may not be implemented.

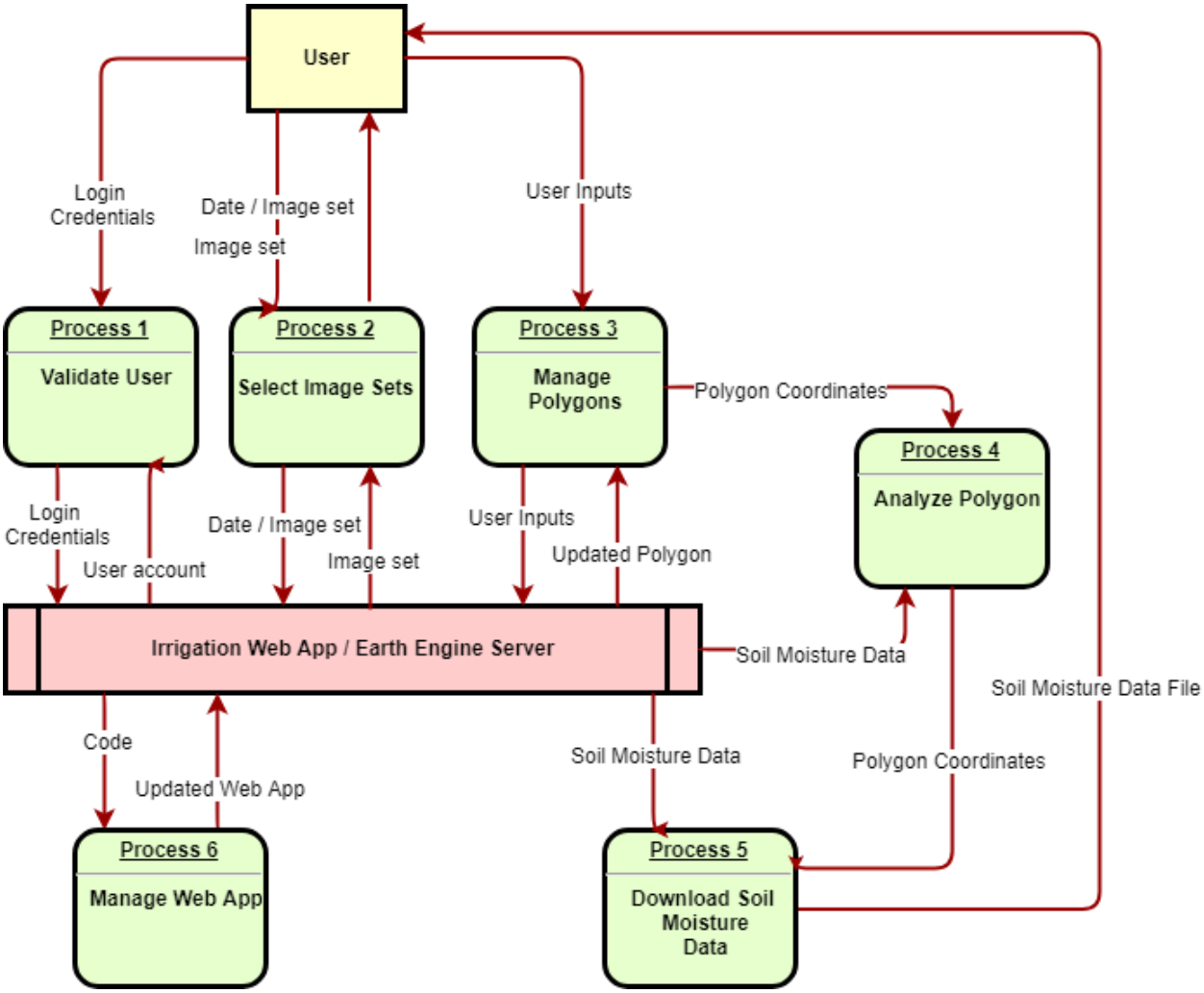
Proposed System Model

Web App Use Cases

See document: Proposed System Use Cases.xlsx

Back to Top

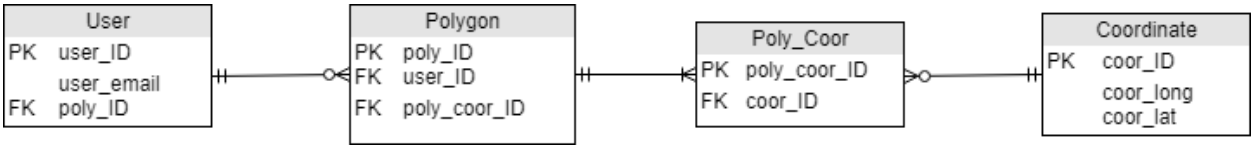
Proposed System Data Flow Diagram



*Note: This is a rough diagram and is susceptible to heavy changes.

Proposed Database & ERD

- A User can have zero or more polygons
- A Polygon has three or more coordinates
- A Coordinate can be in one or more polygons
- A Polygon has one user (multiple users can have the same polygon but those would be saved separately in each of the users' account)



Back to Top

What is Google Earth Engine?

Google Earth Engine is a public archive for satellite imagery and geospatial datasets, as well as, a platform for visualization and analysis of those stored imagery and datasets. Earth Engine (EE) also provides APIs and a code editor to help researchers and developers to analyze these datasets.

More information can be found here: earthengine.google.com

Earth Engine Code Editor

Link: https://code.earthengine.google.com/

The Earth Engine code editor is an online tool provided by Google Earth Engine to develop applications to retrieve and analyze geospatial datasets in Javascript. Although applications made through the code editor can be shared and used by others, the tool is limited to users who have access to the Earth Engine and thus, is not ideal for Mueller Irrigation Group's web application needs.

What is Google App Engine?

The Google App Engine is one of Google's cloud products that allows users to build scalable web applications without the need to manage the server and its configurations. The App Engine supports multiple languages such as Python, PHP, .NET, Go, Java, Node.js, and Ruby.

App Engine Environment: Standard vs. Flexible

Reference: https://cloud.google.com/appengine/docs/the-appengine-environments

- Flexible
    - Requires billing information
    - Supports more programming languages
    - Runs in a docker container that can contain **other programming languages**
    - Access to resources or services in the **compute engine network**
- Standard
    - Free / low cost (Scales with usage)
    - Supports limited programming languages

A standard environment was chosen for the current development of the web app.

Back to Top

Development Environment Setup

Google Earth Engine Account

In order to utilize the Google Earth Engine and all its capabilities, a Google account is needed to sign up via their website, https://earthengine.google.com/. This process may take up to one business day to complete.

Python 2.7.XX

Most, if not all, App Engine demos and projects utilizing Earth Engine are done with Python 2.7. Also, the Geographic Information Systems (GIS) community, as well as, Willemijn Appels are most familiar with Python 2.7.

Download and install the latest version of Python 2.7**:** https://www.python.org/downloads/windows/

Note that Python 2.7's support will be discontinued in 2020 and this project will need to be updated to Python 3 sometime in the future. Google is working on updating their App Engine and documentation to adopt Python 3 as well.

Getting Started with Python by Google: https://developers.google.com/edu/python/

Git

Git is used to access and/or update githubs easily and quickly. It also includes a linux command line call Git Bash which is needed to run build.sh to install Python libraries found in Earth Engine demos.

Git Link: https://git-scm.com/

Creating a project in Google's Cloud Platform with Earth Engine API Access

Reference: https://developers.google.com/earth-engine/app_engine_intro

The following are the steps in creating a project.

1. Create a project here: https://console.cloud.google.com/
2. Create a service account under the IAM & admin > Service accounts of your project
3. Create and Download a key in json format for later use in App Launcher projects.
4. Register the service account for Earth Engine access by filling out this form: https://docs.google.com/forms/d/e/1FAIpQLScFk_pkrrDDF4O8imsEBMaryLDU-Ghf44eHbgujIAl_SXJTJQ/viewform

   Reference: https://developers.google.com/earth-engine/service_account#register-the-service-account-to-use-earth-engine

5. To utilize Google maps, it must be enabled in the project by following the instructions in this link (note that a valid credit card is needed): Google Maps API Setup Link

Back to Top

Reference: https://developers.google.com/maps/documentation/javascript/get-api-key

Windows App Engine Launcher Installation for Local Development

1. Visit: https://cloud.google.com/appengine/docs/standard/python/download
2. Scroll down to "Download and install the original App Engine SDK for Python." and click to expand the link.
3. Download and install App Engine SDK.

How to Run EE Demos in App Engine Launcher

1. Download link for Google Earth Engine API Demos: https://github.com/google/earthengine-api
2. For python demos with the "build.sh" scripts, run it to download the necessary python libraries.
    a. To run .sh scripts on Windows, use Git Bash.
4. Edit the demo's "app.yaml" and add the following lines at the very top of the file if not found.

    application: <any-name-you-want-here>
    version: 1

5. Edit "config.py" to include the following email address to the EE_ACCOUNT or service account line.

    (Note that some demos do not have a config.py file and so find the authentication files in other demo project files and edit the email with the service account email with Earth Engine access.)

    (More notes. Some demos may ask for an OAuth2 account. Those authentication methods are outdated and require more modifications to work with the service account email.)

6. Find the json key from step 3 in "Creating a project in Google's Cloud Platform with Earth Engine API Access" and rename it to privatekey.json.
7. Place the "privatekey.json" file into the same directory as the "config.py" and "app.yaml".
8. In the "index.html" OR "script.js" look for "key=<your-api-key>" and replace the "<your-api-key>" with the following Google Maps API key generated from step 5 in "Creating a project in Google's Cloud Platform with Earth Engine API Access".
9. Launch the Google App Engine Launcher.
10. Click on file > add existing application and locate the directory of the demo.

Other Notes:

- DO NOT CLICK ON DEPLOY or DASHBOARD (this will try to upload the app to the google cloud engine project with the same application name which will fail if not found)
- If the text goes red, the application is missing an "app.yaml" file.
- If the application is not working, you can check the logs by selecting the app in the launcher and clicking the "Logs" button.
- Clicking on "Edit" will only edit the "app.yaml" file.

Back to Top

Prerequisites before Working on the Web App

Learn the Earth Engine API

Best way to learn is through the **Earth Engine Code Editor** by coding and testing out different scripts and examples. Learn how to filter or reduce an image collection, how to customize the visualization settings, how to map an image or image collection over the map UI, how to join two different collections to create a new collection, how to create charts and tables from a collection and much more.

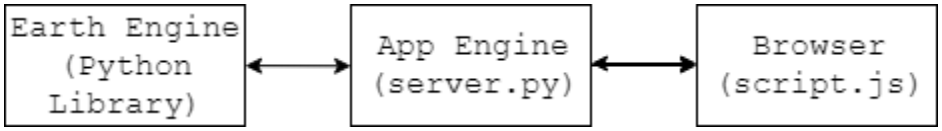
References to get you started:

- Developer's Guide: https://developers.google.com/earth-engine/
- Earth Engine Beginning Curriculum: https://docs.google.com/document/d/1ZxRKMie8dfTvBmUNOO0TFMkd7ELGWf3WjX0JvESZdOE

Learn the layout of an App Engine Project and How it functions

Best way to learn is again, experimenting. Go through the demos and see how to get them working. Missing a library? Check if the library is in the project. Not detecting a directory? Check if the app.yaml included it.

Simplified App Engine Python with Earth Engine Project Diagram



Learn Python and the libraries: Jinja and WebApp2

Learning Python is required to translate the code from the Code Editor to Python. There are a lot more steps needed to get a functioning web app working that code editor has done for the developer such as, adding or removing layers, the communication between the Javascript and Python through WebApp2 and replacing text in HTML through Jinja.

Reference to use JSON requests with WebApp2: https://www.youtube.com/watch?v=XkddK0Rd7nA

Back to Top

Equations for Image Band Processing

NDVI (Normalized Difference Vegetation Index, saturates at LAI ≥ 3)

$$NDVI = \frac{NIR - RED}{NIR + RED}$$

Kc crop coefficient, determined from NDVI. Numbers in this equation may change.

$$K_c = 1.1875 * NDVI + 0.05$$

ETc crop water use, determined from weather data in DayMetV3 and calculated Kc coefficient. Numbers may change, and results will have to be carefully evaluated.

$$ET_{ref} = 0.0135 \left(\frac{srad}{28.94}\right)\left(\frac{tmax - tmin}{2} + 17.8\right)$$

$$ET_c = K_c * ET_{ref}$$

As an intermediate step it may be necessary to show ETref as well as ETc.

Website (HTML, CSS) Development

Bootstrap Studio, not to be confused with the free bootstrap javascript library (https://getbootstrap.com/), was used to develop the base website of the web app which can be found here: https://bootstrapstudio.io/

Since the bootstrap library is free, Bootstrap Studio is not necessary to continue the development of the website, however, for those who want the Bootstrap Studio project files, they can be found here: https://github.com/LC-Mueller-Irrigation/bootstrap-project-files.git

Installing Libraries for the Irrigation Web App

Reference: https://cloud.google.com/appengine/docs/standard/python/tools/using-libraries-python-27

Install the libraries into 'lib' directory by either running the build.sh inside the directory or by using this command in the GitBash command line:

```
pip install -t lib -r requirements.txt
```

Back to Top

Irrigation App in Earth Engine Code Editor

Link: https://code.earthengine.google.com/0da90e0ad7d615c2892c2cf9462afcae

Important Notes:

- Links can be shared with anyone but can only be run by users who have access to the Earth Engine.
- Drawing new polygons while the program is running will not process Kc, ETc data for the new polygon. Requires the program to run again after a polygon is drawn.
- Saving polygons in the app is not possible as there is no database to save the polygons and the users.
    - However, users can save the app into their own script and it will keep record of the polygons they draw assuming that they save the script after each polygon drawn.

Irrigation Web App Git Hub Project

Link: https://github.com/LC-Mueller-Irrigation/Irrigation-Web-App

Back to Top