Guideline Documentation: Building Rangeland Monitoring Dashboard

## Landing Page

* Map: Display the anomaly (z score) for the years 2015-2020 using machine learning classification (unsupervised classification).
* Current Status at the National Level:
  + Total grazing areas
  + Total areas under watch
  + Total areas at risk of degradation/desertification
  + Total number of herder households
  + Total number of livestock
  + Total number of animals
  + Annual Net Primary Production: Line graph showing net primary production for each year for the selected soum.
* Hotspots: Table displaying the top 20 soums in the country at the highest risk for rangeland degradation or desertification watch.

At Soum Level

* Map: Display the anomaly (z score) for the years 2015-2020 using machine learning classification (unsupervised classification) categorized as high-risk, medium-risk, and low-risk.
* Drought: Heat map displaying the drought index (Desertification index) for the selected soum.
* Annual Net Primary Production: Line graph showing net primary production for each year for the selected soum.
* Hotspots: Table displaying the top 20 soums in the country at the highest risk for rangeland degradation or desertification watch.

# Building the Web Application

To create a web application with interactive widgets and display the map on a webpage, we can use a combination of backend and frontend technologies.

### Backend (Python using Flask):

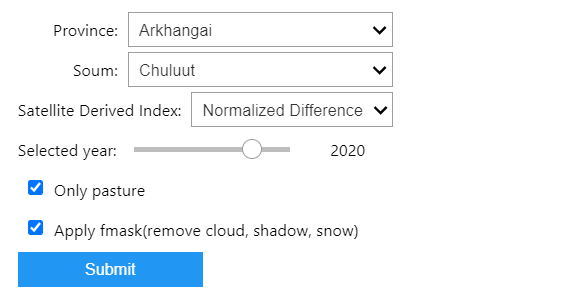
* Set up a Flask web application to handle the backend logic and data processing.
* Define routes to handle user interactions and process the data from the interactive widgets.
* Use the Earth Engine Python API or other tools like Microsoft Planetary Computer for data processing and anomaly calculation.
* Export the final anomaly layer as an image or tileset that can be hosted on the web.

Frontend (HTML, CSS, JavaScript using React):

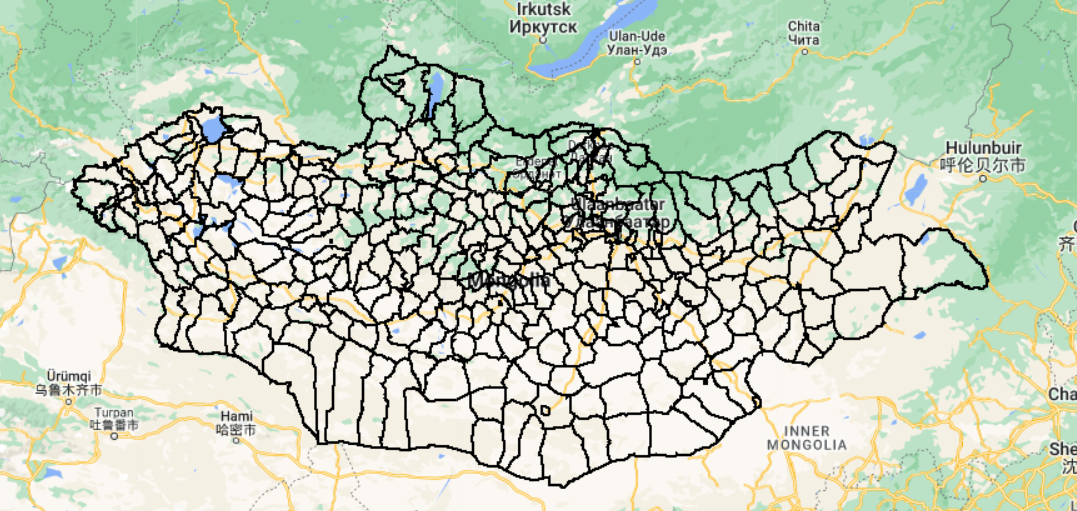
* Use HTML, CSS, and JavaScript to create the frontend of the web application.
* Use React to build interactive components and handle user input.
* Set up a form or interface to capture user input using interactive widgets.
* Use JavaScript to handle user interactions and send the input values to the backend for processing.
* Display the map on the webpage using a mapping library like Leaflet or Mapbox GL JS.
* Connect the backend with the frontend using AJAX requests or fetch API calls to send/receive data between the two.

### Workflow:

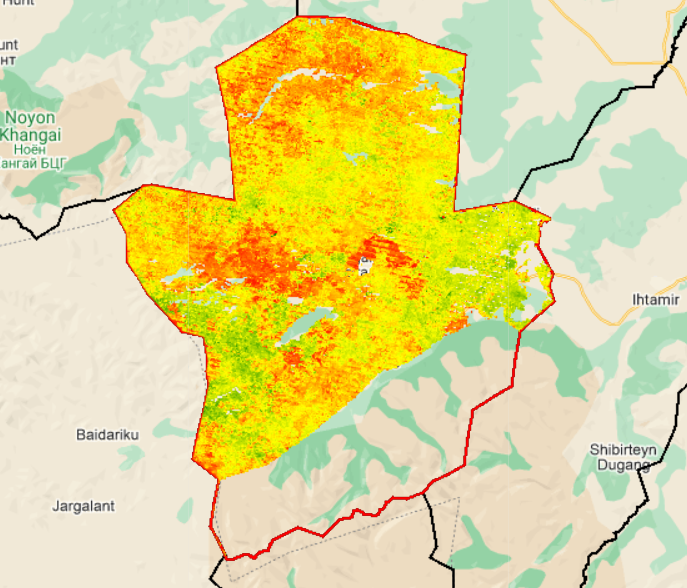
* Design Interactive Widgets:
  + Retrieve selected values from widgets such as province, soum, nd\_indices, year\_widget, pasture\_widget, and fmask\_widget.



* + Filter a feature collection (soum\_aimag) based on the selected province and soum.
  + Add a layer to the map using ee.Image().paint() to highlight the selected soum with a name based on the province and soum.



* + Construct a Landsat collection based on the selected soum and other parameters.
  + Create a summer composite image using the Landsat collection and the selected vegetation index.
  + Apply a pasture mask to the composite image if the "Only pasture" checkbox is selected.
  + Filter the summer composite image for the selected year.
  + Compute the anomaly by subtracting the mean composite from the selected image and dividing it by the standard deviation composite.
  + Define visualization parameters for the anomaly layer.
  + Add the anomaly layer to the map using the visualization parameters and a name based on the selected year and vegetation index.
  + Center the map on the selected soum.



* + Handle exceptions and display an error message if any error occurs during computation.

Similar Websites:

* [CarbonPlan Forest Risks](https://carbonplan.org/research/forest-risks)
* [Landcover.io](https://www.landcover.io/)
* [Living Atlas - Land Cover Explorer](https://livingatlas.arcgis.com/landcoverexplorer/#mapCenter=26.956%2C23.175%2C4&mode=step&timeExtent=2017%2C2022&year=2022&showImageryLayer=true&renderingRule=0)