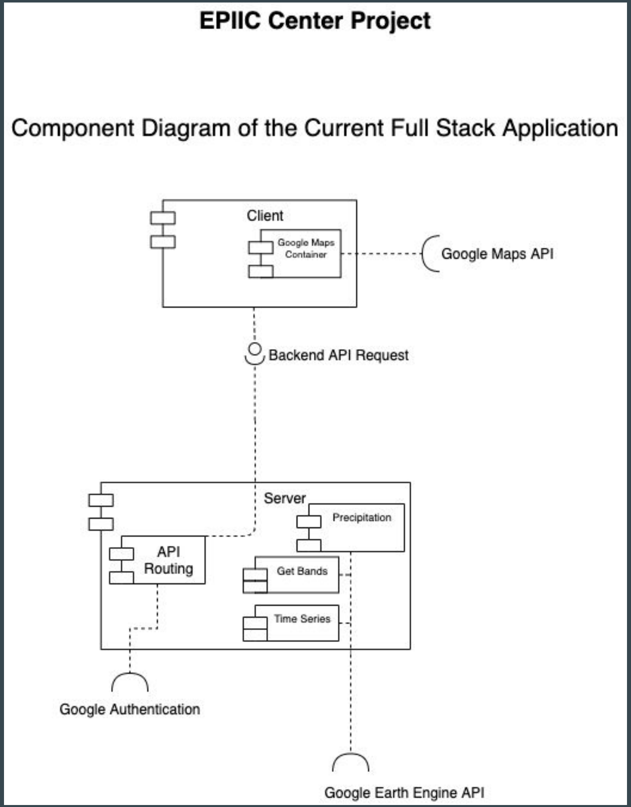
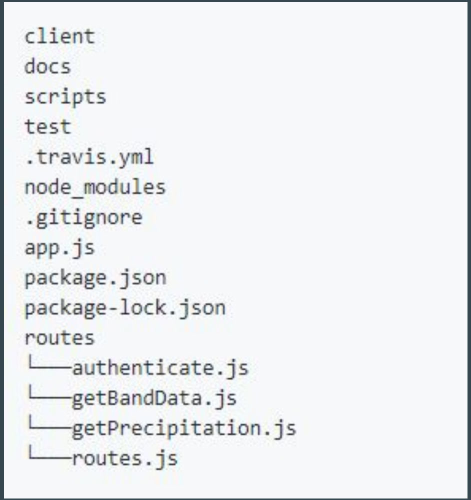
EPIIC Project

Michaela Lightner, Julian Rechsteiner, Norman Tang

System Architecture

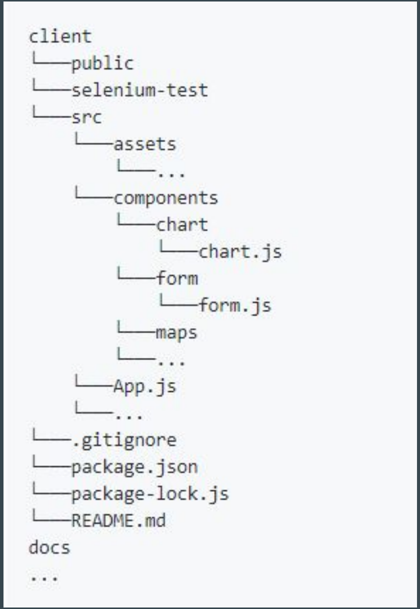


Backend Structure



```
client
docs
scripts
test
.travis.yml
node_modules
.gitignore
app.js
package.json
package-lock.json
routes
└───authenticate.js
└───getBandData.js
└───getPrecipitation.js
└───routes.js
```

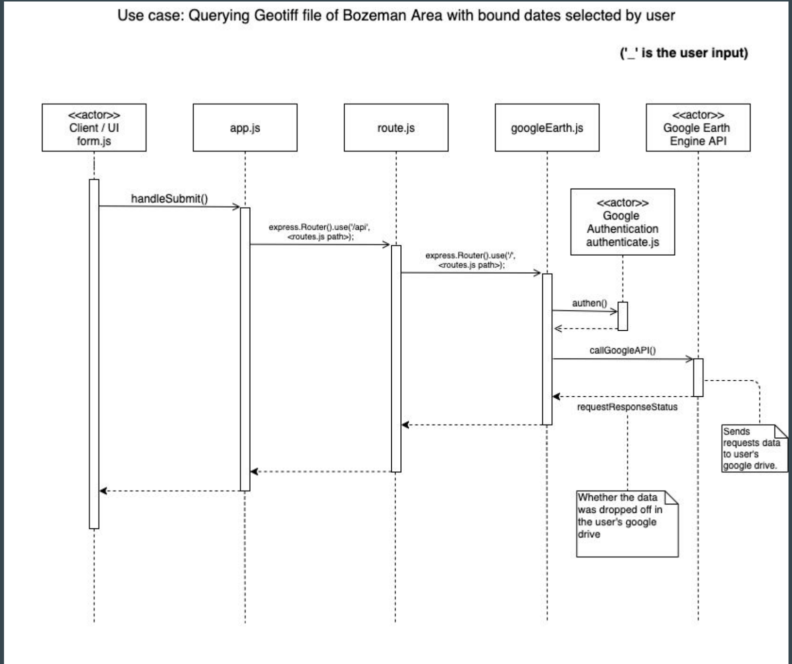
Frontend Structure



```
client
└───public
└───selenium-test
└───src
    └───assets
        └───...
    └───components
        └───chart
            └───chart.js
        └───form
            └───form.js
        └───maps
        └───...
    └───App.js
    └───...
└───.gitignore
└───package.json
└───package-lock.js
└───README.md
docs
...
```

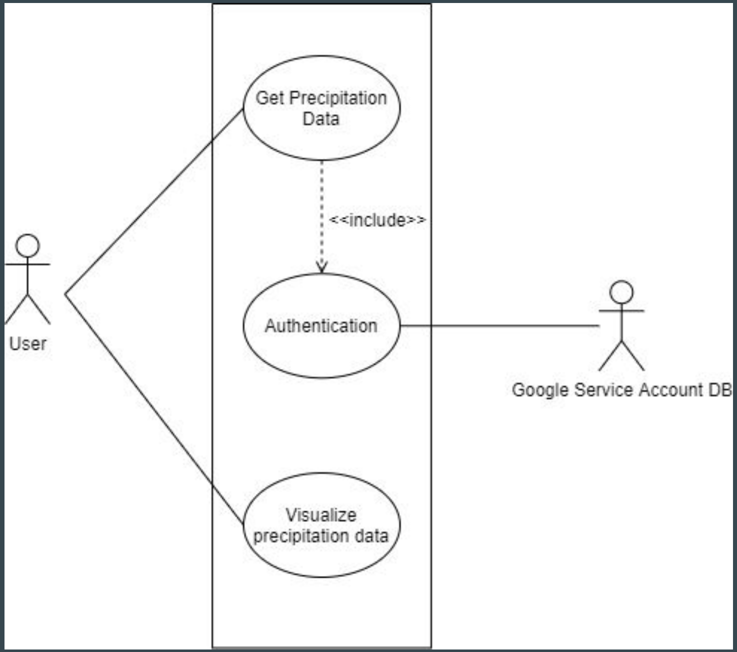
System Design

Backend RESTful API Request from Frontend User Interface


Use case: Querying Geotiff file of Bozeman Area with bound dates selected by user
('_' is the user input)
<<actor>> Client / UI form.js
app.js
route.js
googleEarth.js
<<actor>> Google Earth Engine API
<<actor>> Google Authentication authenticate.js
handleSubmit()
express.Router().use('/api', <routes.js path>);
express.Router().use('/', <routes.js path>);
authen()
callGoogleAPI()
requestResponseStatus
Sends requests data to user's google drive.
Whether the data was dropped off in the user's google drive

Design choices: Why have an intermediate route.js when calling the backend API?
Extendability was taken into consideration when designing the backend REST API system. Keeping all routes in a single file was a decision based on route organization. If more API routes were to be added, a simple line would suffice in the route.js file. During extension, no backend configuration (app.js) should be touched unless necessary, and keeping these two aspects separate was a critical design decision.

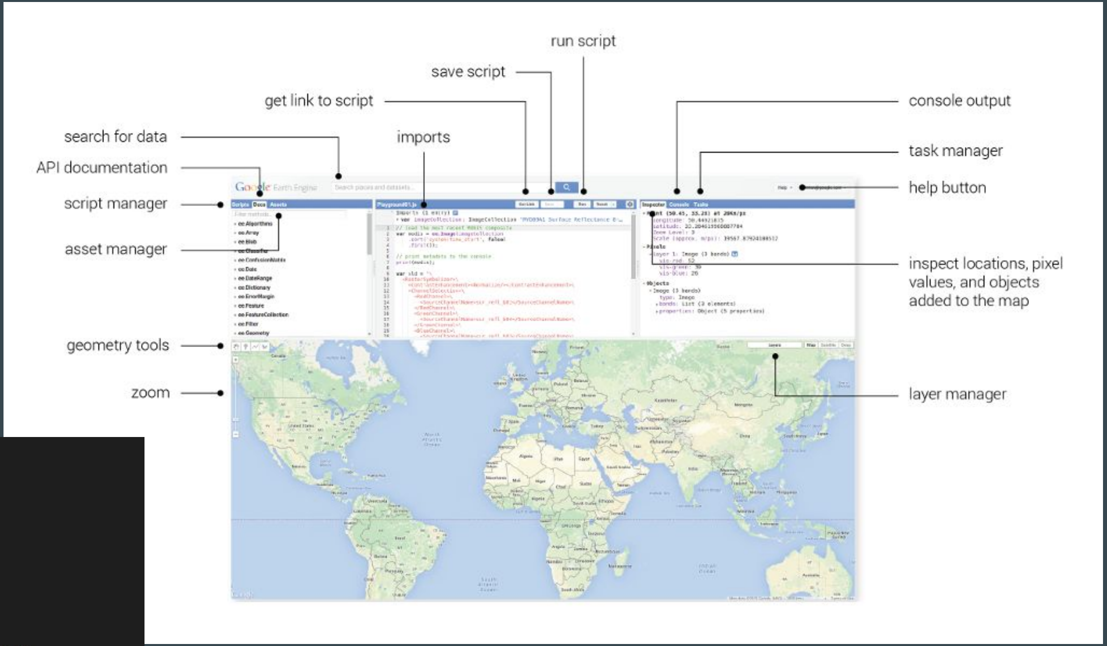
Use Case Diagram for Website User

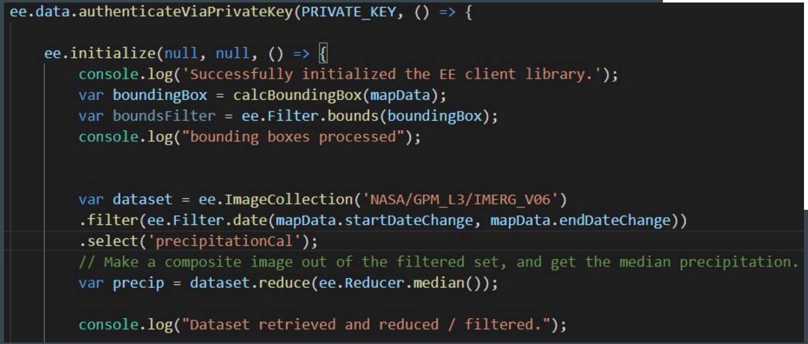


Google Earth Engine

Code Editor

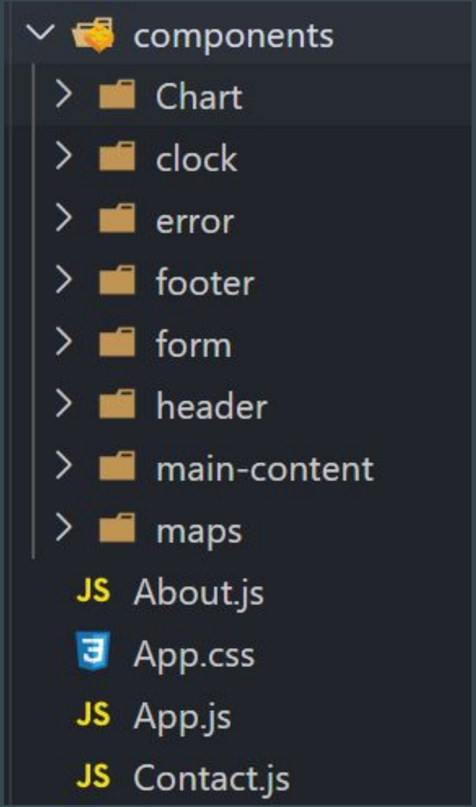
Client Libraries



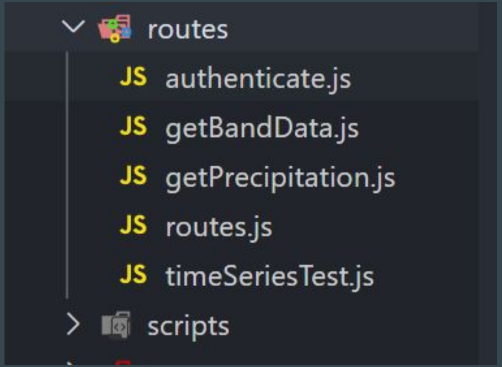


```
ee.data.authenticateViaPrivateKey(PRIVATE_KEY, () => {

    ee.initialize(null, null, () => {
        console.log('Successfully initialized the EE client library.');
        var boundingBox = calcBoundingBox(mapData);
        var boundsFilter = ee.Filter.bounds(boundingBox);
        console.log("bounding boxes processed");


        var dataset = ee.ImageCollection('NASA/GPM_L3/IMERG_V06')
        .filter(ee.Filter.date(mapData.startDateChange, mapData.endDateChange))
        .select('precipitationCal');
        // Make a composite image out of the filtered set, and get the median precipitation.
        var precip = dataset.reduce(ee.Reducer.median());

        console.log("Dataset retrieved and reduced / filtered.");
```

Maintainable/Extensible



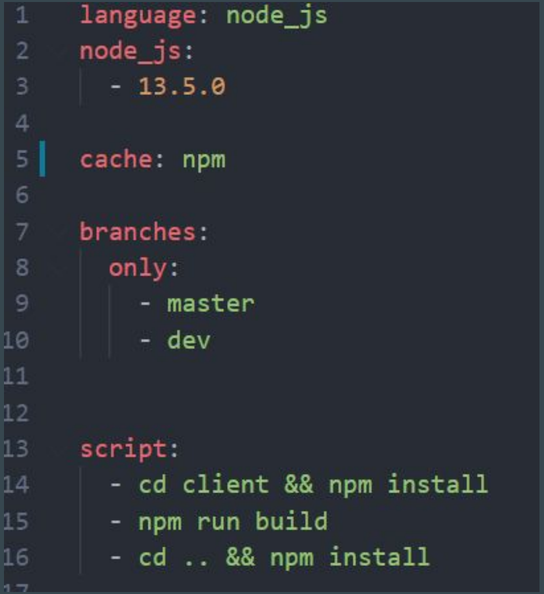
```
components
  Chart
  clock
  error
  footer
  form
  header
  main-content
  maps
  About.js
  App.css
  App.js
  Contact.js
```



```
routes
  authenticate.js
  getBandData.js
  getPrecipitation.js
  routes.js
  timeSeriesTest.js
scripts
```

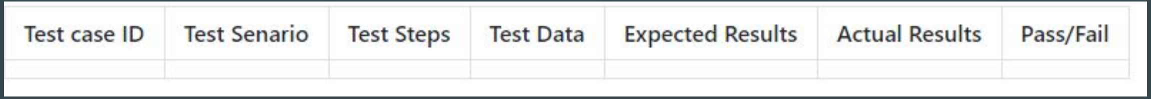
We also have a lot of documentation! Very easy to extend.

Quality Assurance



```
language: node_js
node_js:
  - 13.5.0

cache: npm

branches:
  only:
    - master
    - dev


script:
  - cd client && npm install
  - npm run build
  - cd .. && npm install
```

{REST API}



Test case ID	Test Senario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail

Limitations

@react-google-maps/api

Frontend:

- Form Functionality
  - Correct Datasets
  - Getting Correct Bands
- Chart Integration
- Form Integration with Earth Engine
  - Get Precipitation Data

Backend:

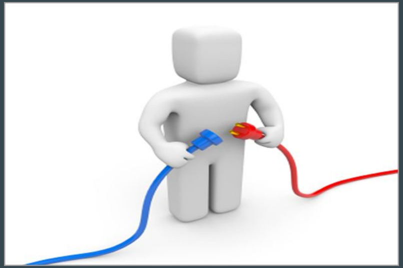
Using a third-party Google Maps React library.

- @react-google-maps/api
- @google-maps-react

Functions in the Google Earth Engine API:

- Export.image.toDrive()
- dataset.getInfo()

Choose Dataset: AMSR-E



DEMO

EPIIC Center

Earth Engine Online Playground

Playground