Data Model

Account (3) -> Price Plan (3) -> Contact (5) -> Reading

Initial the project

1. clone the repo to local system

```
git clone https://github.com/sf-wilson/joyofenergy-apex.git
```

2. dev branch mgmt (need web auth)

```
# new a branch
git branch dev

# push the code from local system to dev branch
git push origin dev

# switch the branch
git checkout dev

# check all the branch on the repo
git branch
```

3. check the metadata structure and modify the package.xml for deploying to org
    - ApexClass
    - LightningComponentBundle
    - CustomObject
    - QuickAction

Extra Setup in SFDC

1. create an app and tabs for the custom part
    - tabs
        - Price Plans
        - Readings
    - app
        - JOI Energy
2. assign FLS for system admin via profile
    - Price Plan
    - Reading
    - Contact
3. import sample data
4. adjust page layout
    - Price Plan
    - Reading

    - Account - Account Layout
    - Account - Contact Layout (also need to add quick action: FetchMockData)
5. go to the one of the contact record, then click the FetchMockData button, then reading records will be initialized.

```
select id,Client__r.name,Name, format(ReadingTime__c), Value__c from Reading__c
order by ReadingTime__c desc
```

6. adjust test code to solve the assert errors in `UsageStubTest`
7. develop recommendPlans lwc & also need to add @AuraEnabled for property & method
8. add the component to contact record page

Run Code in your Scratch Org

1. dev hub -> enable dev hub
2. push source to scratch org
3. import sample data
4. add component to contact record page

Common CLI

```
# set alias
sfdx force:alias:set sp21=wilson@releasesp21.com

# import data
sfdx force:data:tree:import -f data/Account.json -u sp21
sfdx force:data:tree:import -f data/Account.json -u joyofenergyapex

# push code to dev
git status # check changed files
git add . # add all new or changed files
git commit -m "your commit"
git push -u origin dev
```

Ref

1. Comparable: Sorting Objects in Salesforce
2. Comparable Interface
3. Use Lightning Web Components for Quick Action
4. Create Screen Quick Actions
5. Call Apex method from Lightning Web Components

ToDo

1. trial the ref1, 3
2. optimize data import -> just like dreamhouse

3. optimize fetchreading records - refresh to display the reading records after clicing the quick action button

Bug

1. click quick action insert reading -> refresh recommend data
2. edit / delete reading
    - refresh recommend data

Misc.

```
communicate data across components:
1. components with hierarchy -> custom event
    ref: https://salesforcediaries.com/2019/12/10/send-multiple-parameters-in-lwc-events-via-detial-property/
2. components without hierarchy -> Pub-Sub Model
    ref: https://www.sfdcpanther.com/pub-sub-in-lightning-web-component/
3. platform event
    ref: https://inevitableyogendra.blogspot.com/p/how-to-refesh-lightning-datatable-after.html


Lightning message service vs platform events vs pubsub

As of July 2020, c/pubsub has been superseded by the Lightning Message Service.


1. apexrefresh with wire
2. optimize the import data process, and if partial success, how it does.

readingListUpdate

disconnectCallback vs disconnectedCallback1
```

Implementing Comparable

Apex will not let you use the built in sort method for List to sort sObjects by a field inside. To do this, we have to implement our own comparable class to do the sorting for us.