Front-End Development for the MPC in the Cloud Sprint-4

QingXing Li, Yicun Hou, Haoyu Xu
Mentors: Joseph Caiani, Máirín Duffy

Project Overview

- ChRIS(ChRIS Research Integration System) is an open source framework that utilizes cloud technologies to democratize medical analytics application development and enables healthcare organizations to keep owning their data while benefiting from public cloud processing capabilities.
- Goal: Design and test the front-end functions using TypeScript and deploy the ChRIS Store UI to the MOC
- Stretch Goal: Deploy the ChRIS Store backend to MOC and implement a tool to track the website traffic

Previous sprints

Sprint1: Set up the ChRIS Store backend in Docker and rewrited ChRIS Store UI with Redux

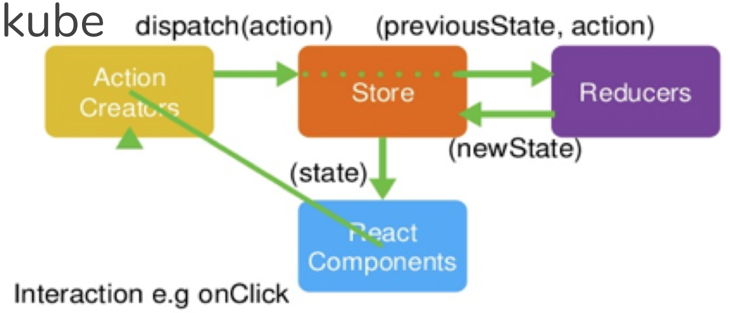
Sprint2: Write the declaration file of all the APIs by using Typescript, and installed the backend of the Chris UI

Sprint3: Debug the declaration file of all the APIs; Test and Debug Actions & Reducer of all the APIs by using Jest

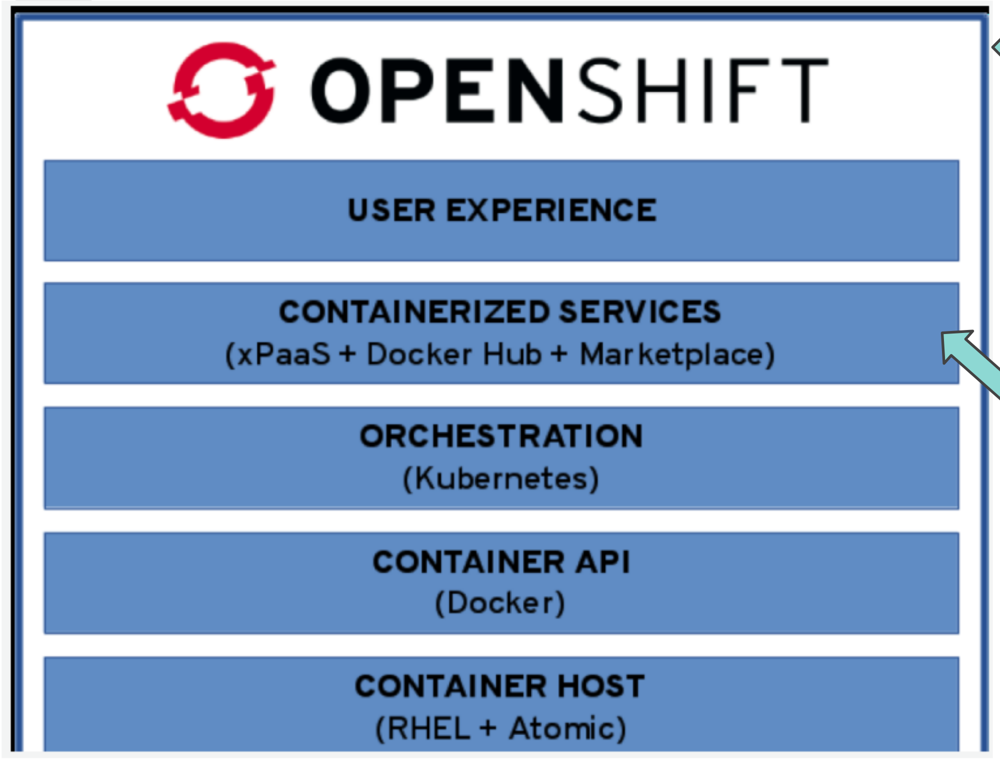
Sprint 4- What we did

- Testing actions & reducers, optimizing testing code and debugging
- Deploy the ChRIS Store UI **(MVP)**
  - **Main task:** Deploy to MOC using Openshift
  - Bonus: Deploy to local machine using Minikube
  - Bonus: Deploy to IBM cloud kubernetes clusters

Redux Flow



Three ways to deploy ChRIS Store UI



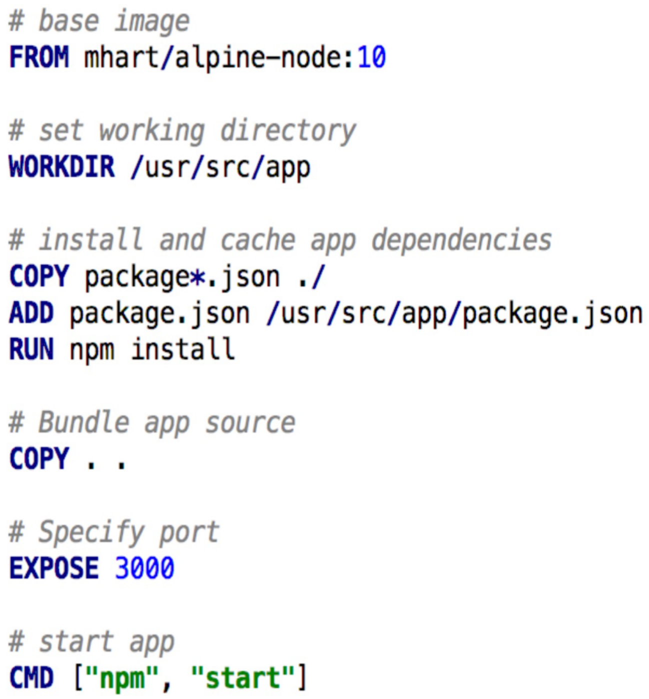
First, we use Openshift to deploy our project

Then, we try to use Minikube, which is a method for creating a local kubernetes cluster, to deploy our project to kubernetes cluster locally

Finally, we use IBM cloud to create our kubernetes cluster service and deploy our project to cloud cluster

Containerize-Docker

Dockerfile



```
# base image
FROM mhart/alpine-node:10

# set working directory
WORKDIR /usr/src/app

# install and cache app dependencies
COPY package*.json ./
ADD package.json /usr/src/app/package.json
RUN npm install

# Bundle app source
COPY . .

# Specify port
EXPOSE 3000

# start app
CMD ["npm", "start"]
```

Build the image:

docker run -p 3000:3000 -d chris/api

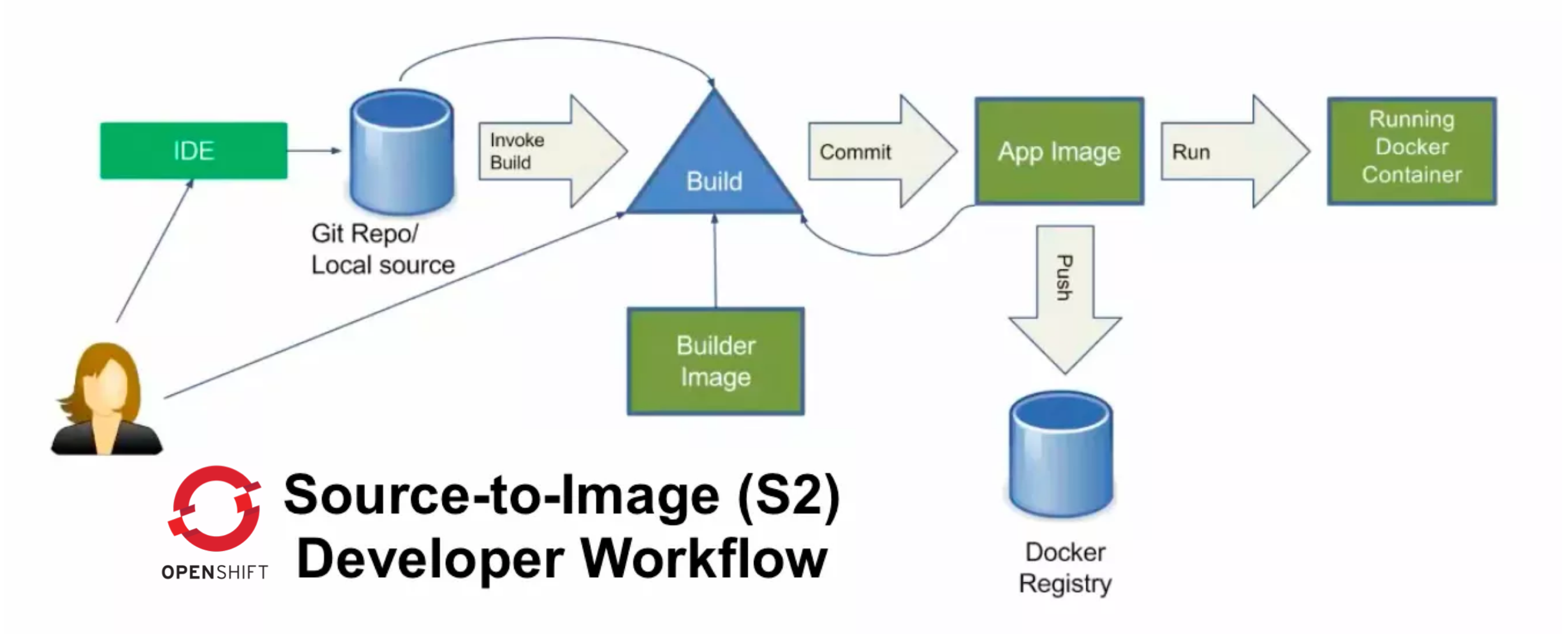
Run the image:

docker build -t chris/api .

You can now access the application at http://localhost:3000

Containerize to openshift -Source-to-Image (S2I)

Source-to-Image (S2I) is a framework that makes it easy to write images that take application source code as an input and produce a new image that runs the assembled application as

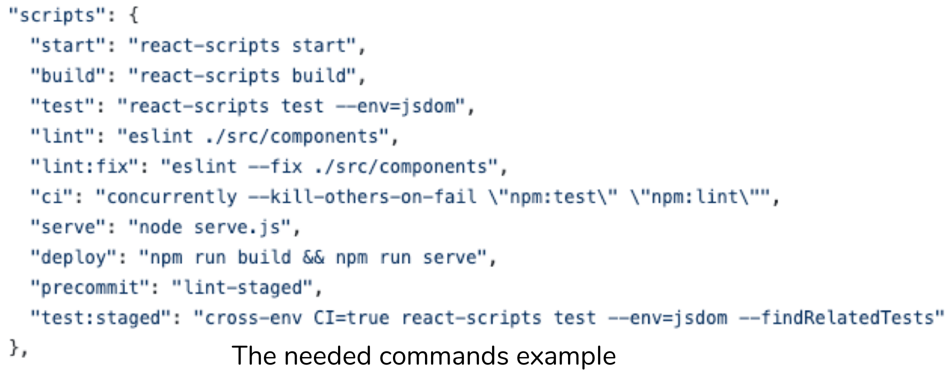


Containerize to openshift - Our case

1. Define all the needed commands to build and deploy a web application just like in local IDE
2. Specify the git location to the Openshift
3. Openshift will clone the source code from git location and run the predefined commands
4. Then push the app image to the Docker Registry and run the application in the Docker container

Ex. Install packages:

- Local: npm install
- Openshift: npm install -s only == productions
  (This command will install all the dependencies)

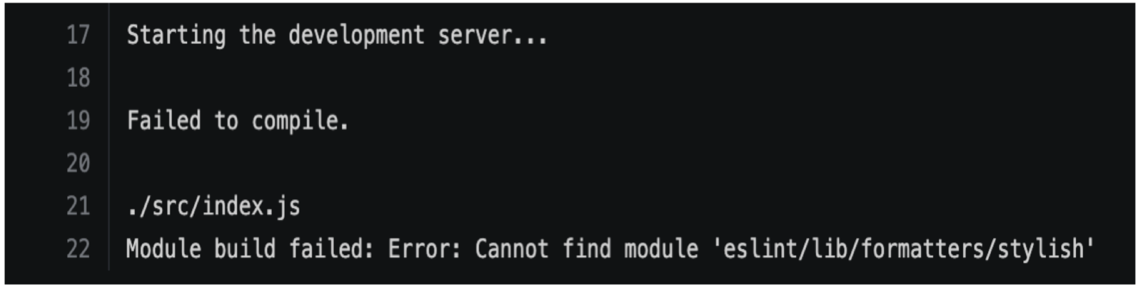


```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test --env=jsdom",
  "lint": "eslint ./src/components",
  "lint:fix": "eslint --fix ./src/components",
  "ci": "concurrently --kill-others-on-fail \"npm:test\" \"npm:lint\"",
  "serve": "node serve.js",
  "deploy": "npm run build && npm run serve",
  "precommit": "lint-staged",
  "test:staged": "cross-env CI=true react-scripts test --env=jsdom --findRelatedTests"
},
```

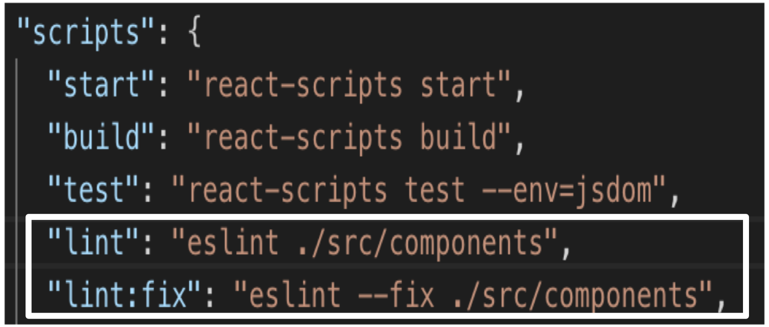
The needed commands example

Containerize to penshift
- Problems and solutions:

- Building: failed to install dependencies on openshift
  - Reasons: the eslint only exist in devdependencies, it will not be installed.
  - Solution: add description of eslint in dependencies in package.json file



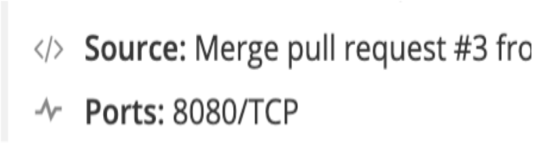
```
Starting the development server...

Failed to compile.

./src/index.js
Module build failed: Error: Cannot find module 'eslint/lib/formatters/stylish'
```



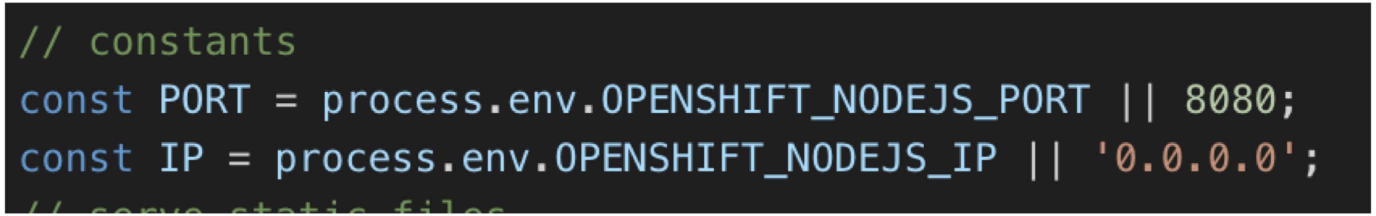
```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test --env=jsdom",
  "lint": "eslint ./src/components",
  "lint:fix": "eslint --fix ./src/components",
```

Containerize to Openshift -Problems and solutions:

- Deploying: failed to visit the website
  - Reasons:
    - The default port for npm project is 3000, but the port of openshift using is 8080.
    - Chrome do not support http, we have to use safari
  - Solution: Change the default port and use safari



Source: Merge pull request #3 fro
Ports: 8080/TCP

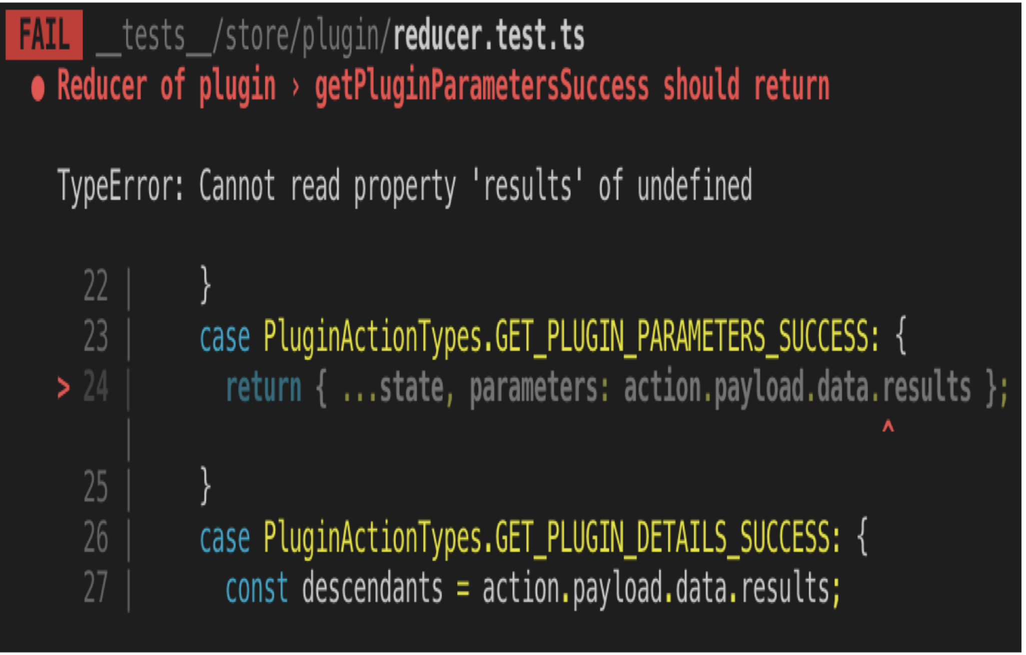


```
// constants
const PORT = process.env.OPENSHIFT_NODEJS_PORT || 8080;
const IP = process.env.OPENSHIFT_NODEJS_IP || '0.0.0.0';
```

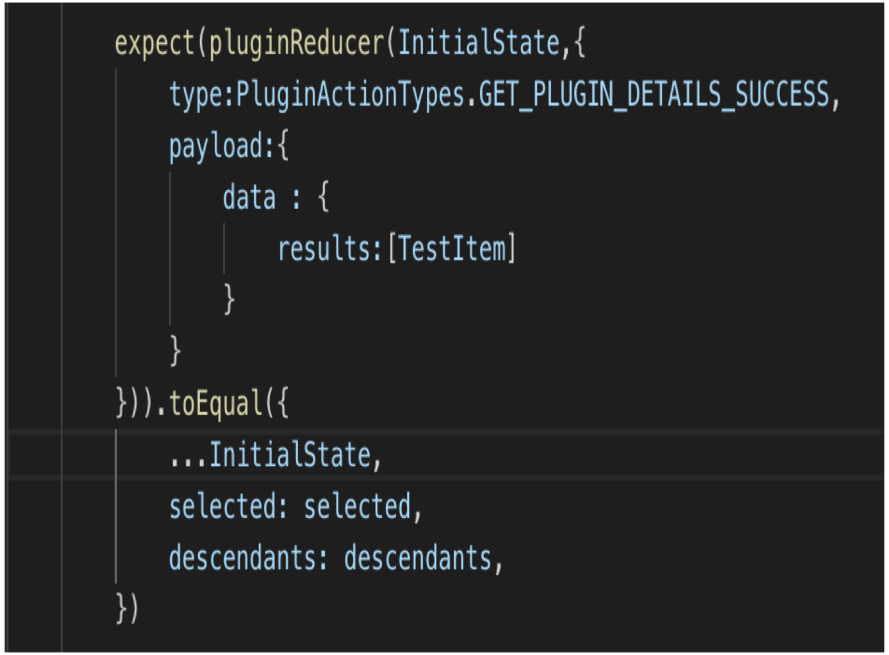
Demo

- You can now access our ChRIS Store using Safari or Edge at http://chris-store-demo-bu528-ui-for-cloud-mpc.k-apps.osh.massopen.cloud

Testing problem from last sprint

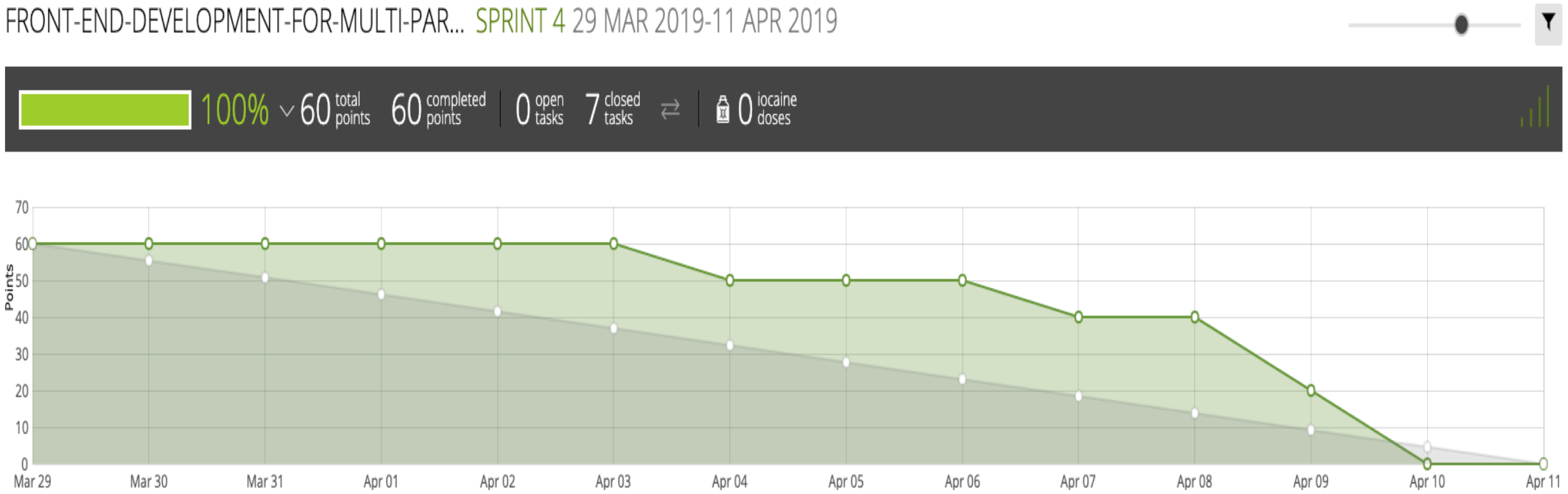


```
FAIL __tests__/store/plugin/reducer.test.ts
● Reducer of plugin › getPluginParametersSuccess should return

  TypeError: Cannot read property 'results' of undefined

          }
          case PluginActionTypes.GET_PLUGIN_PARAMETERS_SUCCESS: {
  >         return { ...state, parameters: action.payload.data.results };
                                                               ^
          }
          case PluginActionTypes.GET_PLUGIN_DETAILS_SUCCESS: {
            const descendants = action.payload.data.results;
```



```
expect(pluginReducer(InitialState,{
    type:PluginActionTypes.GET_PLUGIN_DETAILS_SUCCESS,
    payload:{
        data : {
            results:[TestItem]
        }
    }
})).toEqual({
    ...InitialState,
    selected: selected,
    descendants: descendants,
})
```

Burn Down Chart


FRONT-END-DEVELOPMENT-FOR-MULTI-PAR... SPRINT 4 29 MAR 2019-11 APR 2019

100% 60 total points 60 completed points 0 open tasks 7 closed tasks 0 iocaine doses

Sprint 5- Next to do(Stretch Goal)

- Deploy the Chris Store backend to MOC using OpenShift
- Monitor the website traffic

Questions?