Learn Deep MILWAUKEE

Upcycle Startup Essentials

Created by UpCycle SDL team of 18-19

Section 1.0 - Setting Up the VM

Note: If you haven't already done so download and install VirtualBox

Link: https://www.virtualbox.org/

VM Installation

If you're proficient with VM's, just make an Ubuntu 18.04 LTS VM.

1. Go to the Ubuntu website and download the Ubuntu desktop (18.04 LTS was confirmed to work): https://www.ubuntu.com/download/desktop
2. After downloading open Virtualbox Manager and select the new icon.
3. Give the VM an appropriate name.
4. Keep type as linux and version as Ubuntu (64-bit), then select next.
5. Allocate 4+Gb for the memory size.
6. When you reach a page titled Hard disk select the option *create a virtual hard disk now*, then select create.
7. For page titled Hard disk file type, select VDI (VirtualBox Disk Image), and select next.
8. For "Storage on physical hard disk" select Dynamically allocated then select next.
9. For "File location and size", allocated 20Gb. You don't have to worry about the large size. Since it is dynamically allocated it only takes the amount that it needs. Allocating a smaller size such as 10Gb will cause the VM to run out of space.

10. Now the setup wizard should close and you should see the VM created. Select it and click start.
11. When asked for the operating system to use you will need to browse to where you saved the Ubuntu 16.04 LTS ISO file earlier and select start.
12. Select the install Ubuntu option .
13. On “Preparing to Install Ubuntu”, select *Download updates while installing Ubuntu*, then select continue
14. For installation type select “Erase disk and install Ubuntu” then select install now.

Application installation

15. Install pip with ***sudo apt install python-pip***
16. Now install something such as *Webstom*, or *Atom* so as to be able to edit the project.
17. Clone project repos:
    a. ***git clone https://github.com/LearnDeepMilwaukee/rea-app.git***
    b. ***git clone https://github.com/LearnDeepMilwaukee/valuenetwork.git***

Section 2.0 - Running the Project

After cloning Rea-app & value-network

- Open two terminals where the rea-app & value network have been cloned to.
- For the terminal within the rea-app folder
    - Run command **npm install** to install dependencies
    - Run **npm run** in the terminal to start the server
    - Cross fingers
- For the terminal within the value network folder
    - Install dependencies with ***pip install -r requirements.txt***
    - Then run the command **sudo ./manage.py runserver**
    - If it says some package isn’t available but was needed, install that package as well

Section 3.0 - Tech Stack

- *React*: React is a JavaScript library for building user interfaces. This is why we have components, which are exported from their home location to other locations.
- *React Apollo*: Essentially what allows us to fetch data using graphql from valuenetwork
- *React Redux*: Essentially what allows us to save state (variables, data, etc.) in project.
- *GraphQl*: Replaces REST, allows you to ask for just certain information instead of getting everything back and then having to sift to the item you want.
- *React Router*: What is used to route the pages on the website for user.

Note: What .TSX was used for : Essentially think of it as Javascript with strong types. It is essentially Javascript for large application development.

Note: This link has more info regarding TypeScript: https://msoese.atlassian.net/wiki/x/5YCAKQ

Very Important: You will not be editing value-network, only run it in the background. Bob & Lynn are the ones who make edits to the value-network project.

Section 4.0 - Helpful Resources / Hint

Useful Learning Guides:

- *Useful react Links*:
  - https://reactjs.org/
  - https://facebook.github.io/react-native/docs/getting-started
  - Longer Intro : https://hackernoon.com/react-js-a-better-introduction-to-the-most-powerful-ui-library-ever-created-ecd96e8f4621
  - short intro: https://medium.freecodecamp.org/learn-react-js-in-5-minutes-526472d292f4
  - Cheat Sheet: https://devhints.io/react
- Useful react apollo links:
  - https://www.telerik.com/blogs/managing-state-in-graphql-using-apollo-link-state
  - Simple example: https://www.robinwieruch.de/react-apollo-client-example/
- Useful react redux links:

        - https://github.com/markerikson/react-redux-links/blob/master/README.md
        - https://redux.js.org/basics/basic-tutorial
        - Cheat Sheet: https://devhints.io/redux
    - *Useful graphql links*:
        - https://graphql.org/learn/
        - Cheat Sheet: https://devhints.io/graphql
    - *Semantic UI*:
        - https://react.semantic-ui.com/
        - https://medium.com/@a.carreras.c/using-semantic-ui-react-your-react-js-app-523ddc9abeb3
        - Code Box (simple way test semantic code): https://codesandbox.io/s/jn0w4mxm5
- React Router:
        - Example: https://codeburst.io/getting-started-with-react-router-5c978f70df91
- Previous Project Atlassian: https://msoese.atlassian.net/wiki/spaces/UP19/overview
- Working Mutation GraphQLI Example (must be opened in VM): http://localhost:8000/api/graphiql?query=mutation(%0A%20%20%24username%3AString!%2C%0A%20%20%24password%3A%20String!%0A)%7B%0A%20%20createToken(%0A%20%20%20%20username%3A%20%24username%2C%0A%20%20%20%20password%3A%20%24password%0A%20%20)%7B%0A%20%20%20%20token%0A%20%20%7D%0A%7D%0A&variables=%7B%22username%22%3A%20%22connor%22%2C%20%22password%22%3A%20%22iuc4ur42%22%7D
- Working Query GraphQLI Example (must be opened in VM):
  http://localhost:8000/api/graphqli?variables=%7B%22token%22%3A%20%22eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImNvbm5vciIsImlhdCI6MTU0Njg4MDM4NiwicGFzc3dvcmQiOiI3YzA4ODliOWU5ZmNjYzAxZDIzMDcwNzljNDk5OTcyNDFlNTZlNzU0IiwiaWQiOjZ9.eG0F71jF-PHp3tlPVRUg9ijIncA-DMOJYSlWV1Pp3To%22%2C%20%22EconomicResourceId%22%3A%2022%7D&query=query(%24token%3A%20String)%20%7B%0A%20%20viewer(token%3A%20%24token)%20%7B%0A%20%20%20%20allAgentRelationships%7B%0A%20%20%20%20%20%20id%2C%0A%20%20%20%20%20%20note%2C%0A%20%20%20%20%20%20relationship%20%7B%0A%20%20%20%20%20%20%20%20id%0A%20%20%20%20%20%20%7D%2C%0A%20%20%20%20%20%20subject%20%7B%0A%20%20%20%20%20%20%20%20id%0A%20%20%20%20%20%20%7D%2C%0A%20%20%20%20%20%20object%20%7B%0A%20%20%20%20%20%20%20%20id%0A%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%0A%20%20%20%20%7D%0A%20%20%7D%0A%7D

Hint: Be careful with dependencies, the original version we tried working had a lot of intricate dependencies. Having many dependencies makes development difficult, since installing one packages brakes dependencies in another place in the project.

Section 5.0 - Mutations, Queries and Components etc.:

How to Test Queries & Mutations

To test out your mutations have value-network running open a web browser browser and enter http://localhost:8000/api/graphiql.

By Selecting the Docs button (refer to 4 in the picture below) you can look up the params needed for calling queries and mutations.

- Section (1) this is where you type your query or mutation.
- Section (2) this is where you supply the params for the query or mutation as an object.
- Section (3) is where the output will appear. Either, like in the example output pertaining to your query, or an error message detailing what you did wrong will appear.

Example Query, the process for a mutation is the same.

```
mutation(
  $note:String,
  $subjectId: Int!,
  $relationshipId: Int!,
  $token: String!,
  $objectId: Int!,
){
  createAgentRelationship(
    note: $note,
    subjectId: $subjectId,
    relationshipId: $relationshipId,
    token: $token
    objectId: $objectId
  ){
    agentRelationship{
        id
    }
  }
}
```

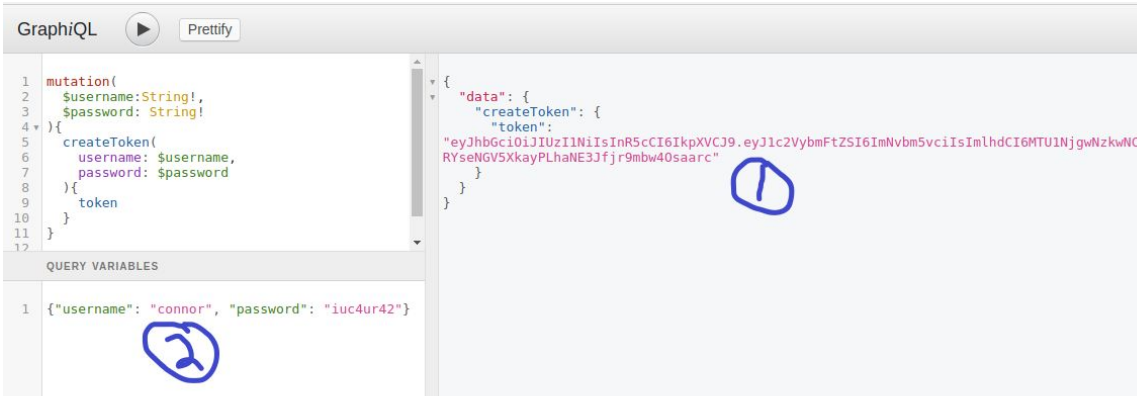
Example mutation, it is what would go into section (1) of the annotated picture.

How to Create Token Query

The token query is important. Used in every query, and mutation as a form of authentication.

Section (1) - If the token(This is a JWT) is created you can then copy and paste this into the token param of your query variables for your mutations and queries.

Section (2) - Used username and password, this will consist of the username and password of any account in valuenet-work.



Creating Graphql Query In Project:

```
import ...

const query = gql`
query($token: String) {
  viewer(token: $token) {
    myAgent{
      ...agentInterface
    }
  }
}
${agentInterface}
`;

export default compose(
  // bind input data from the store
  connect((state) => ({
    variables: {
      token: state.getUserInfo.currentUserToken,
    },
  })),

  graphql(query,  operationOptions: {
    // read query vars into query from input data above
    options: (props) => ({ variables: {
      ...props.variables,
    } }),
    // transform output data
    props: ({ ownProps, data: { viewer, loading, error, refetch } }) => (
      {
        loading,
        error,
        refetchAgent: refetch,  // :NOTE: call this in the component to force reload the data
        agent: viewer ? viewer.myAgent : null,
      }),
  })
);
```

- Create const query variable and have it equal to the working query placed between two single quotation marks.
    - Agent relationship is the type of object myagent query returns
- In export default compose it looks a lot more complicated than it really is. The comments do a good job at explaining it.

Calling Graphql Query In Project:

```
import getMyAgent from "../../queries/Agent/getMyAgent";
//...
var myAgentId = 0; //Global variable that holds value returned from GetMyAgent
//Global query that returns id of current user logged in else returns -1
export const GetMyAgent = getMyAgent(({ agent, loading, error}) => {
    if (loading) {
        myAgentId = -1;
    } else if (error) {
        myAgentId = -1;
    } else {
        myAgentId = agent.id;
    }
    return (<span></span>);
});
```

- Section (1) Importing query so it can be used in this page.

- Section (2) Export const GetMyAgent is what will cause the query to be run when its called. It will be called when it is rendered.
- Section (3) You need three sections: loading, error, and an else section.
  - The example was used in a special case. It does not follow the traditional way.
  - Inside of loading you would usually have:

```
return(
  <strong>Loading...</strong>
);
```

  - Inside of error you would usually have:

```
return(
  <p style={{color: "#F00"}}>API error</p>
);
```

  - Inside of the else block you would return the parameter (called agent in this example). Since you would be returning a single object the example below shows how you would return multiple objects instead. As you would want to return this more than a single object. However, to return a single object you would just remove the concat array.

```
return(
  <div>
    {concatArray(planList)}
  </div>
);
```

Creating Graphql Mutation In Project:

```
import { connect } from "react-redux";
import { graphql, compose } from "react-apollo";
import gql from "graphql-tag";
import { organizationInterface } from "./organizationInterface";

export const mutation = gql`
  mutation(
    $token: String!,
    $type: String!,
    $name: String!,
    $image: String,
    $note: String
  ) {
    createOrganization (
      token: $token,
      type: $type,
      name: $name,
      image: $image,
      note: $note
    ) {
      organization {
        ...organizationInterface
      }
    }
  }
${organizationInterface}
`;

export default compose(
  connect(state => ({
    token: state.getUserInfo.currentUserToken
  })),
  graphql(mutation, operationOptions: {name:"createOrgMutation"})
);
```

- Create export const mutation variable and have it equal to the working query placed between two single quotation marks.
- You are importing the organization Interface and using it in the mutation, since that is what the object you want back.
    - organizationInterface is the type of object the mutation returns.

Calling Graphql Mutation In Project:

```
import createOrganization from "../../../ui-bindings/Organization/CreateOrganization.tsx";
```

- You need to first import mutation into the page you want to use it in from the page you created it in.

```
export default createOrganization(Registration);
```

- Then when you export your page at the bottom, wrap your class name with the mutation name as seen in the picture above.

```
let variables = {
  name: this.state.name,
  type: this.state.type,
  image: this.state.logo,
  note: this.state.description
  // primaryLocationId: TODO
  // TODO add banner field
};
variables.token = this.props.token; // add the token in afterwards

// perform the mutation
this.props.mutate({variables}).then((response) => {
  let newOrganization = response.data.createOrganization.organization.id;
  if (newOrganization) {
    console.log(newOrganization);
    this.setState({newOrganizationID: newOrganization,});
    this.openModal();
  }
}).catch((error) => {
  console.log(error);
});
```

Then you can call your mutation.

- First assign the variables to it in an object then you pass it into the props (look at variable named variables in the picture above).
- Then pass this variable into this.props.mutate and do something with the response if successful. If unsuccessful handle it in the catch block. In the above example the error was just consoled out.

Creating React Component:

Multiple classes can be located in a page

- Below is a simple component the class is what it is called Greeting. (What does Render do/explain it simply). This method returns a header with a message in it. *Note*: Though you can have multiple classes, you can only export one.

```
import React, { Component } from 'react';

class App extends Component {
  render() {
    return (
      <div>
        <Greeting />
      </div>
    );
  }
}

class Greeting extends Component {
  render() {
    const greeting = 'Welcome to React';

    return <h1>{greeting}</h1>;
  }
}

export default App;
```

Using React Component:

If the component is located in the same page as were you will use the component you then don't need to use an import statement. However, if they are different then you need to use an import statement. The cool thing about components are that you can use components as if they were html tags.

```
import React, {Component} from 'react';
import Test from "../../testComponent.Tsx"
class App extends Component {
  render(){
    <div>
      <Test />
    </div>
  }
}
export default App;
```

Figure shows imported component name test being called as if it was html tag in render function of App.

```
import React, {Component} from 'react';
class Test extends Component {
  render(){
    <div>
      <Test />
    </div>
  }
}
export default Test;
```

The test class that you imported into the App

Using React Props:

Think props as a way of passing data as params from one React component to another component.

React vs State:

Props are read only and cannot be altered, however React states can be changed

```
import React, { Component } from 'react';

class App extends Component {
  constructor(props) {
    super(props);

    this.state = {
      isShow: true,
    };
  }

  toggleShow = () => {
    this.setState(state => ({ isShow: !state.isShow }));
  };

  render() {
    return (
      <div>
        {this.state.isShow ? <Greeting greeting={greeting} /> : null}

        <button onClick={this.toggleShow} type="button">
          Toggle Show
        </button>
      </div>
    );
  }
}

const Greeting = ({ greeting }) => <h1>{greeting}</h1>;

export default App;
```

You can pass props to other components

```
import React, { Component } from 'react';

class App extends Component {
  render() {
    const greeting = 'Welcome to React';

    return (
      <div>
        <Greeting greeting={greeting} />
      </div>
    );
  }
}

class Greeting extends Component {
  render() {
    return <h1>{this.props.greeting}</h1>;
  }
}

export default App;
```

Figure Below shows how "Welcome to React" is being passed to the Greeting component which is what they will render.

In a functionless state component, the props are received in the function signature as arguments (*image below*).

```
import React, { Component } from 'react';

class App extends Component {
  render() {
    const greeting = 'Welcome to React';

    return (
      <div>
        <Greeting greeting={greeting} />
      </div>
    );
  }
}

const Greeting = props => <h1>{props.greeting}</h1>;

export default App;
```

Figure shows example of functionless state component.

Styling with Semantic UI:

Semantic UI is a popular UI framework. It made development a lot quicker as it gave us premade, modern, components that we could use in our project.
To be able to use a component you first have to import it from semantic-ui-react, afterwards you can use it as if it was a html element.

Example 1:

```
import React from 'react'
import { Grid } from 'semantic-ui-react'

const ButtonExample = () => (
  <Grid>
    <Grid.Column>
      <p>Content lives here</p>
      <p>Content lives here</p>
      <p>Content lives here</p>
      <p>Content lives here</p>
    </Grid.Column>
  </Grid>
);
```

Figure example of importing Grid component from semantic-ui-react.

Example 2:

```
import React from 'react';
import { Button, Form } from 'semantic-ui-react';

const LoginForm = () => (
  <Form>
    <Form.Field>
      <label>Email Address</label>
      <input placeholder="Email Address" />
    </Form.Field>
    <Form.Field>
      <label>Password</label>
      <input placeholder="Password" />
    </Form.Field>
    <Button type="submit">Submit</Button>
  </Form>
);

export default LoginForm;
```

Figure example of importing 2 elements: Button & Form from semantic-ui-react.

apiKeys.json

This file is in the src directory of the project and will need to be created on each machine rea-app is run on. It is needed for when a user creates an account. The format of the file is as follows:

{“adminToken”: “TOKEN FROM createToken QUERY HERE”};

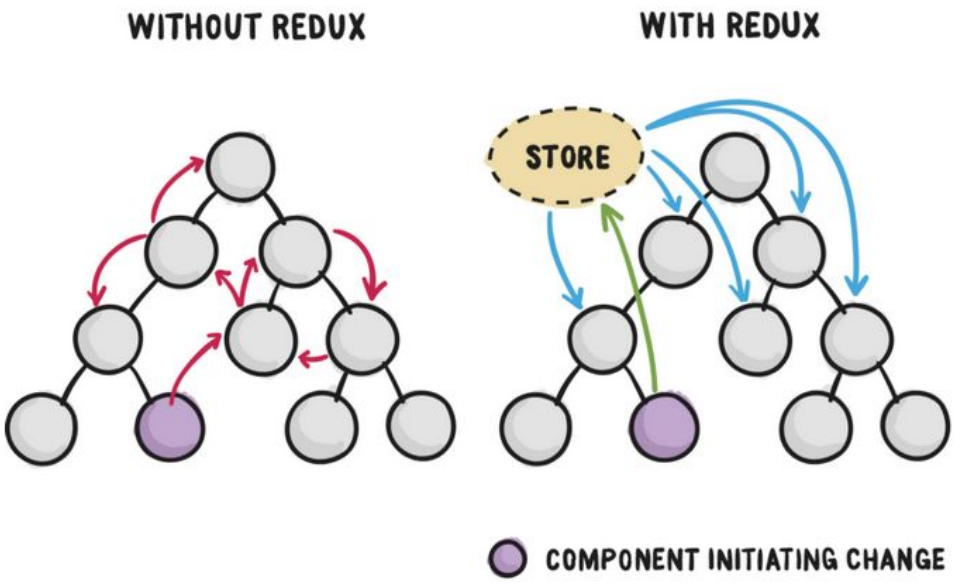
React Redux:

What is it?

It is a predictable state container for JavaScript apps; and helps you write applications that behave consistently, easy to test, and run in different environments. With complex react applications it is hard to manage state object when react app grows to multiple components

Why Redux?

React is javascript library that helps us to divide up our app into multiple components. However, it does not specify how to keep track of the data, and dealing with events. Redux is the only way to do this. A library for React that provides an easy way keep track of data and events. Redux isolates state object from components, allowing the creation of a react app by moving the state / actions to redux.



Redux can be described with fundamental principles: Actions, Reducers, Store, & Data Flow

Actions - Packets of information that send data from your application to your store. They are the only source of information for the store.
Reducers - Specify how the application's state changes in response to actions sent to the store (Actions describe only what happened, but not how application's state changes).
Store - In above sections, we defined the actions that represent the facts about “what happened,” and the reducers that update the state according to those actions. The store

couples the object and has the following responsibilities: holds application state, allows access to state (*getState()*), allows state to be updated via dispatch, registers listeners via subscribe, and handles unregistering of listeners via the function returned by subscribe.
Data Flow - All data in an app follows the same life cycle, making the app logic more predictable & more understandable. It encourages data normalization, to prevent creating multiple, independent copies of the same data that are unaware of one another.

Our redux store currently holds 2 pieces of information, the organization Id of the organization the user is currently representing(shown in the header), and the current user's token. This is used for running queries/mutations (state.getUserInfo.currentUserToken) and for running the getMyAgent query.

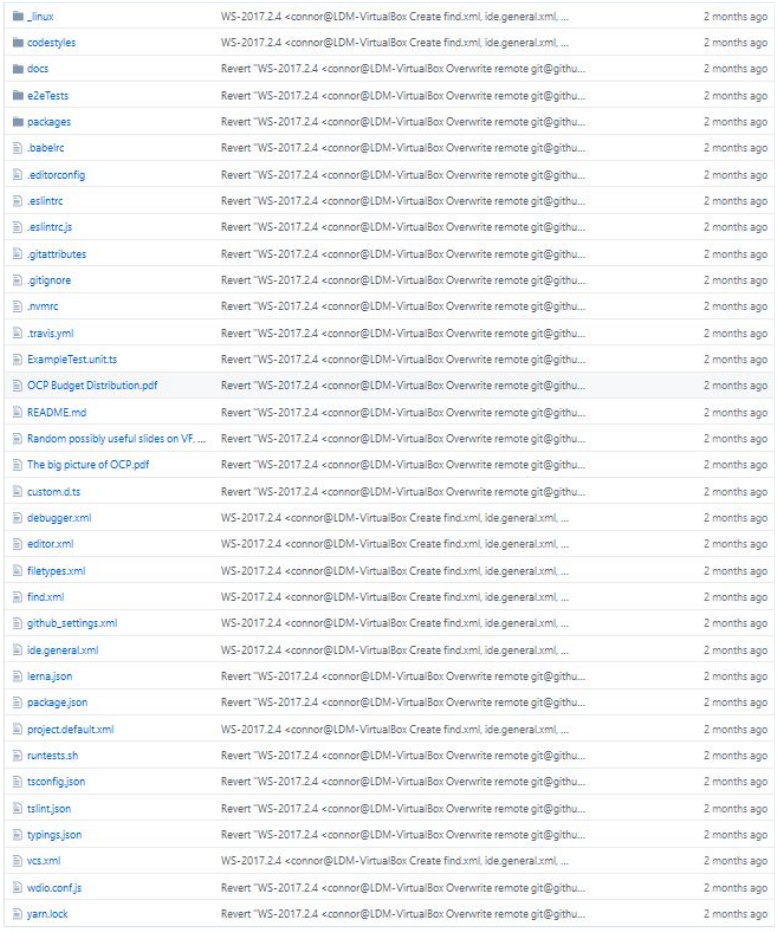
Known Issues With the Site:

Issue 1: Graphql is not fully used to the best of its ability in the project. It is better to not look too in depth into how graphql should work. Instead it's better to learn how it was used for the project.
Issue 2: The back button on the browser doesn't always work. Most likely an issue with React Router
Issue 3: Users can create items for organizations they are not apart of due to no verification on the createInventoryItem page.
Issue 4: Some pages(such as the organization registration page) don't match the rest of the site due to them not using semantic ui like the rest of the site.

The Repo:

Note: This only details the most important aspects of the repo. Many of the files are self explanatory

- packages:
    - App: Has all of the pages for the project
    - Services: Package.json and yarn.lock files
    - Ui-bindings - Queries and mutations for project that have been implemented
- Readme.md: is how to run the project, build it etc.
- Docs: has additional documentation to help you with the project



_linux	WS-2017.2.4 <connor@LDM-VirtualBox Create find.xml, ide.general.xml, ...	2 months ago
codestyles	WS-2017.2.4 <connor@LDM-VirtualBox Create find.xml, ide.general.xml, ...	2 months ago
docs	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
e2eTests	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
packages	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
.babelrc	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
.editorconfig	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
.eslintrc	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
.eslintrc.js	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
.gitattributes	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
.gitignore	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
.nvmrc	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
.travis.yml	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
ExampleTest.unit.ts	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
OCP Budget Distribution.pdf	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
README.md	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
Random possibly useful slides on VF, ...	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
The big picture of OCP.pdf	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
custom.d.ts	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
debugger.xml	WS-2017.2.4 <connor@LDM-VirtualBox Create find.xml, ide.general.xml, ...	2 months ago
editor.xml	WS-2017.2.4 <connor@LDM-VirtualBox Create find.xml, ide.general.xml, ...	2 months ago
filetypes.xml	WS-2017.2.4 <connor@LDM-VirtualBox Create find.xml, ide.general.xml, ...	2 months ago
find.xml	WS-2017.2.4 <connor@LDM-VirtualBox Create find.xml, ide.general.xml, ...	2 months ago
github_settings.xml	WS-2017.2.4 <connor@LDM-VirtualBox Create find.xml, ide.general.xml, ...	2 months ago
ide.general.xml	WS-2017.2.4 <connor@LDM-VirtualBox Create find.xml, ide.general.xml, ...	2 months ago
lerna.json	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
package.json	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
project.default.xml	WS-2017.2.4 <connor@LDM-VirtualBox Create find.xml, ide.general.xml, ...	2 months ago
runtests.sh	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
tsconfig.json	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
tslint.json	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
typings.json	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
vcs.xml	WS-2017.2.4 <connor@LDM-VirtualBox Create find.xml, ide.general.xml, ...	2 months ago
wdio.conf.js	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago
yarn.lock	Revert "WS-2017.2.4 <connor@LDM-VirtualBox Overwrite remote git@githu...	2 months ago

How to Setup Valuenetwork to be internet accessible

This is a step by step guide on how to run valuenetwork on a remote Ubuntu 18.04 GCP instance. You should be able to follow these steps relatively blindly, just running the bold commands (granted you ensure you're in the right directory) and be good to go.

Install apache2

sudo apt install apache2

enable apache2 proxy mod

sudo a2enmod proxy

sudo a2enmod proxy_http

add the apache config from the docs folder in valuenetwork repo to the apache config */etc/apache2/apache2.conf* (append not replace)

clone valuenetwork

git clone https://github.com/LearnDeepMilwaukee/valuenetwork.git

If it says *git* isn't a valid command, install git

sudo apt install git

run valuenetwork

cd valuenetwork

pip install -r requirements.txt

If it says pip is not installed run

sudo apt update

sudo apt install python-pip

Add these lines to the file

/home/learndeepmkedev/valuenetwork/valuenetwork/settings.py

but replace the first IP address with the IP of the VM you're using

```
# Adding to deploy through apache on google cloud
ALLOWED_HOSTS = ['35.243.188.32', 'localhost', '127.0.0.1']
```

run

python manage.py runserver

It may ask you to run migrations. It should be safe to do so, this has never caused an issue
python manage.py migrate

Preparing for Rea-app

Update npm to at least version 10.15.13, process for updating can be found here:
https://www.hostingadvice.com/how-to/update-node-js-latest-version/
use option #1

Emails are not being sent out when users register

We used a gmail account for this and thus used gmail's SMTP server.
Add following to settings.py

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_HOST_USER = 'GMAIL USERNAME'
EMAIL_HOST_PASSWORD = 'GMAIL PASSWORD'
DEFAULT_FROM_EMAIL = 'EMAIL HERE'
SERVER_EMAIL = 'EMAIL HERE'
EMAIL_SUBJECT_PREFIX = 'ADD PREFIX IF WANTED'
SEND_BROKEN_LINK_EMAILS = True
EMAIL_PORT = 465
EMAIL_USE_SSL = True
ACCOUNT_REQUIRED_EMAIL = True
ACCOUNT_EMAIL_VERIFICATION = True
ACCOUNT_EMAIL_AUTHENTICATION = True
```

Log into your google email account and then go to this link: https://www.google.com/settings/security/lesssecureapps and set "Access for less secure apps" to ON.
Go to https://g.co/allowaccess
Go to Gmail, go to your settings and the Forwarding and POP/IMAP tab. Ensure IMAP is enabled
Try having the backend send an email again.

How to keep session running in background:

`tmux` can be used to seperate ssh session from the console. We use it to keep the servers running even after closing our ssh terminal.

To start a service with `tmux`, just type tmux in the console. It will then open a tmux session and you can start your service inside of it.

To leave the tmux session, hit Ctrl + b and then d

To open the tmux session again tmux attach (If you only have 1 session). Otherwise you will have to specify the session number via the following command

tmux attach-session -t NUMBER_HERE

Reference here:
https://askubuntu.com/questions/8653/how-to-keep-processes-running-after-ending-ssh-session

---

Downloading database from google drive:
https://unix.stackexchange.com/questions/136371/how-to-download-a-folder-from-google-drive-using-terminal

---

Auto Start of services on Compute engine start:
https://cloud.google.com/compute/docs/startupscript

Process tips:

A process the group found very useful was peer programming. The group prefered this as it allowed for code to be more clean and concise as two people are looking at it. You are able to get a better understanding of your own code from explaining it to another person. It was very helpful when the team navigated tricky development sections.

Group Members:

Backend Developers - (Available Via Slack; not MSOE students they are friends of Pete Reynolds)
Bob Haugen - Works with actual server backend.
Lynn Foster - Works more with API for backend.

Previous SDL Team - (2018-19 SDL team; Reachable via Slack & available next year for help)

- Michael Larson: larsonme@msoe.edu
- Nic Strike: Strikenp@msoe.edu
- Aaron Murphy: Murphyad@msoe.edu
- Donal Moloney: Moloneyda@msoe.edu

If all else fails look at the prayer below for strength:

Lord, my heart is broken but You are near. My spirit is crushed, but You are my rescuer. Your Word is my hope. It revives me and comforts me in especially now. My soul faints, but you are the breath of life within me. You are my help, the One who sustains me. I am weak but You are strong. You bless those who mourn, and I trust You to bless me and my family with all that we need. You will rescue me from this dark cloud of despair because You delight in me. In Jesus' Name, Amen.