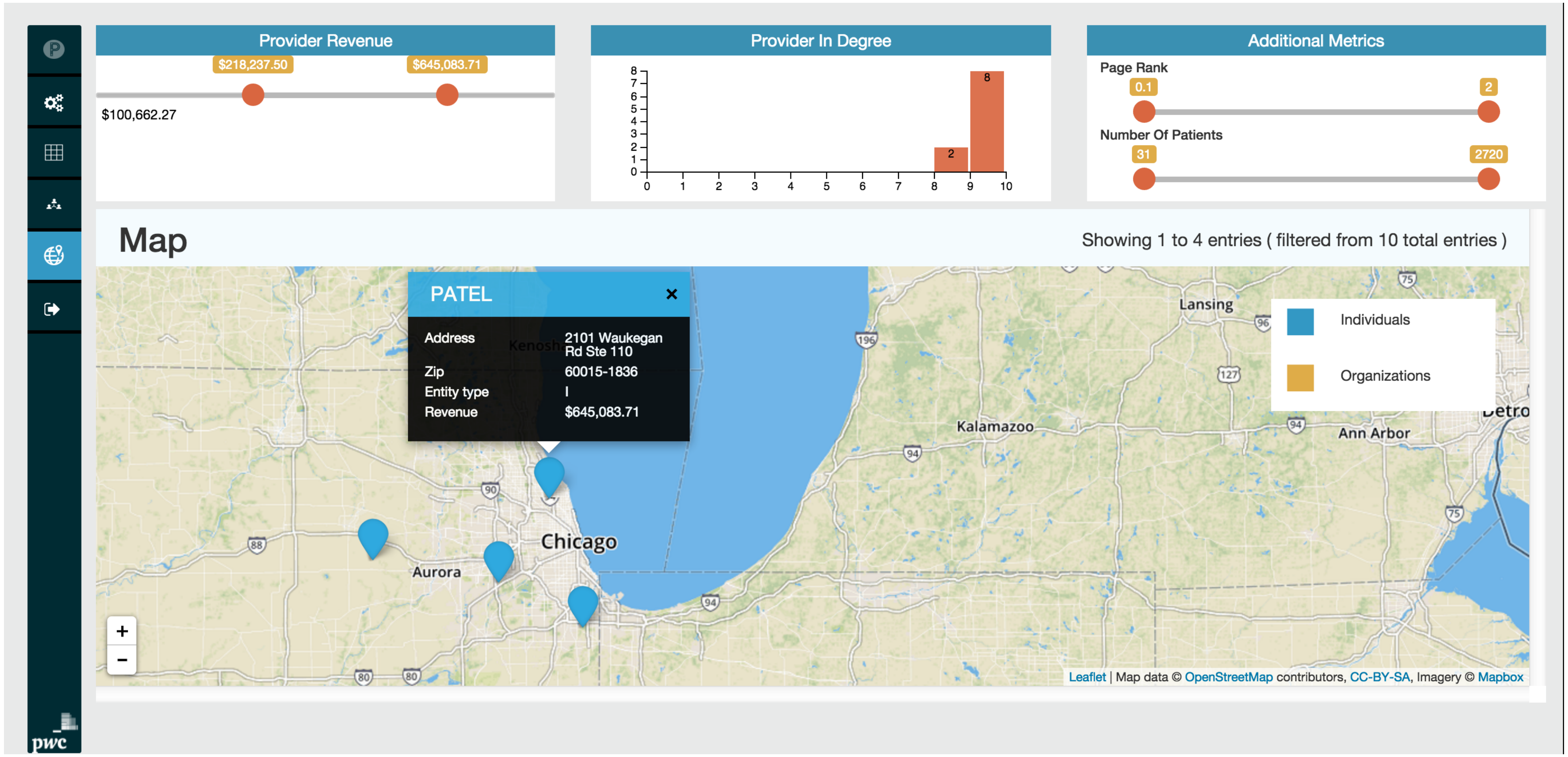


Map

Showing 1 to 4 entries (filtered from 10 total entries)

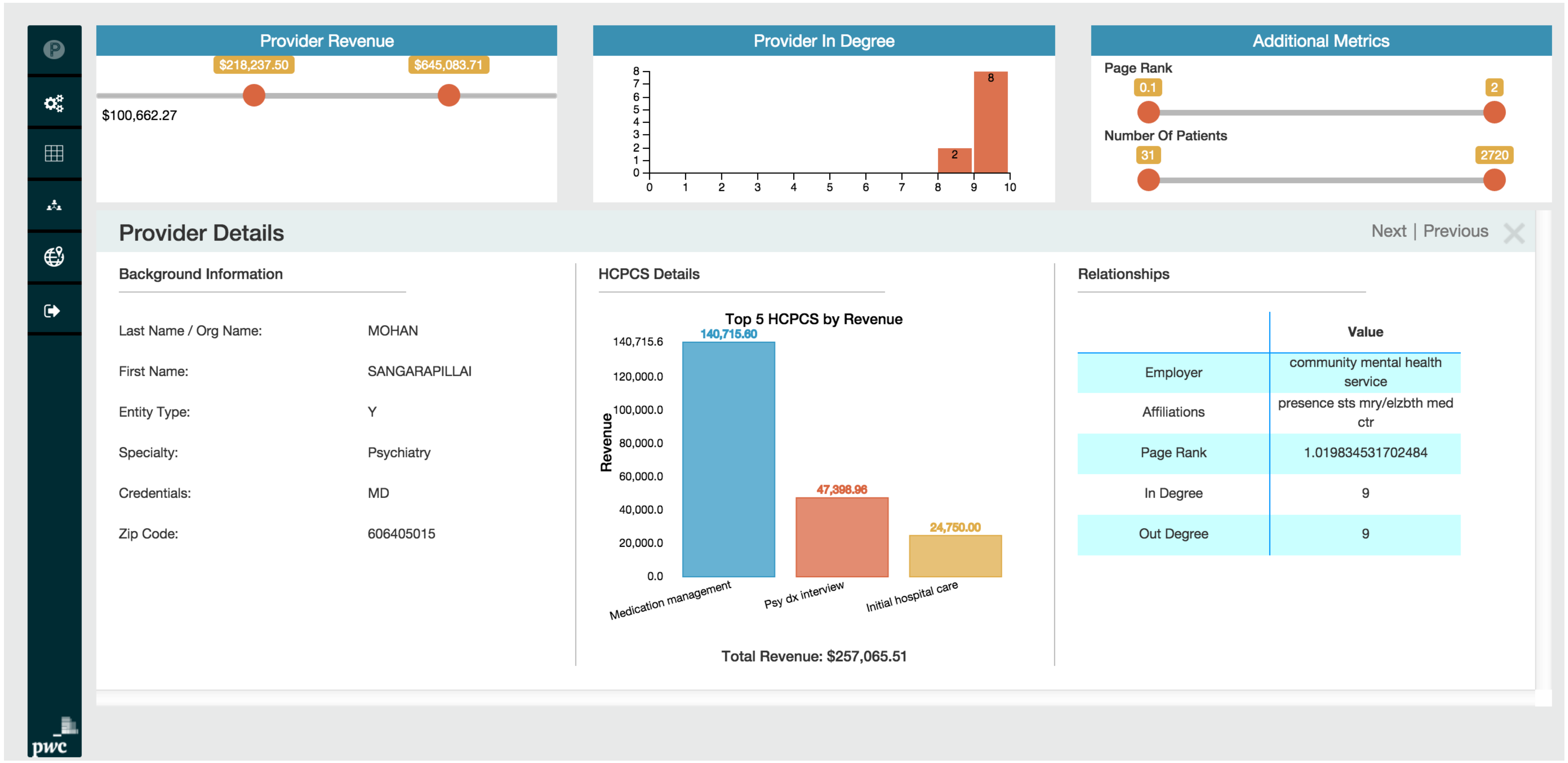
PATEL

Address	2101 Waukegan Rd Ste 110
Zip	60015-1836
Entity type	I
Revenue	$645,083.71

🟦 Individuals

🟨 Organizations

Lansing
Kalamazoo
Ann Arbor
Chicago
Aurora
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery © Mapbox
pwc



Provider Details

Background Information

Last Name / Org Name:	MOHAN
First Name:	SANGARAPILLAI
Entity Type:	Y
Specialty:	Psychiatry
Credentials:	MD
Zip Code:	606405015

HCPCS Details

Total Revenue: $257,065.51

Relationships

	Value
Employer	community mental health service
Affiliations	presence sts mry/elzbth med ctr
Page Rank	1.019834531702484
In Degree	9
Out Degree	9

Table of Contents

- Introduction
  - About this Document
  - Features
  - Used Technologys
  - How to Run

- Core Implementation
    - Architecture
    - Database
    - Server
        - Preparation
        - Application Structure
        - Datafeed
        - Database Query
    - Web: AngularJS Front-side（desktop version）
        - Application Structure
        - User Cases
        - Auth Login
            - Click OK button then goto signin page
            - Sign in
        - Query Builder
        - Slider
        - Draw the bar chart
            - Use dc.js
            - Use nv.d3.js
        - Responsive Design
        - Sigma.js
        - Map

Introduction

Doc Graph is a data analysis web app for showing relationships between doctors, hospitals, and insurance companies. App provides a powerful search matching function, can filter the entity according to the region and type of conditions, can provide a variety of views to show their distribution and relationship types, and can provide a great convenience for data mining.

About this Document

We create this document as some kind of conclusion so that we can easily pick it up when we need it in new project.

Features

This project is comprised of tools for improving your productivity and satisfaction when building a web project.

- We use Boorstrap to get extensive and beautiful documentation for common HTML elements, dozens of custom HTML and CSS components, and awesome jQuery plugins.
- We use Grunt to automate just about anything with a minimum of effort.
- We use npm to make it easy for JavaScript developers to share and reuse code, and make it easy to update the code that you're sharing.
- We use Bower to optimize the front-end, and reduce page load to a minimum.
- We use javascript charting library DCJS, D3 to support highly efficient exploration on large multi-dimensional dataset, and be utilized to perform data visualization and analysis in browser as well as on mobile device.
- We use DataTables which is a plug-in for the jQuery Javascript library to draw and optimize tables.
- We use Leaflet which is designed with simplicity, performance and usability in mind to draw maps, let maps work more efficiently across all major desktop and mobile platforms.
- We use Sigmajs to make it easy to customize how to draw and interact with networks. And we can add all the

interactivity we want, we also can use it to modify the data, move the camera, refresh the rendering, listen to events.
- We use Ion.RangeSlider to build any number of beautiful range sliders at one page without conflicts and big performance problems.

Used Technologys

- Foundation Framework - Bootstrap
- JavaScript Task Runner - Grunt
- Package Manager for Node.js - Npm
- Third-party Dependencies Manager - Bower
- Data Chart Generator - dc.js, D3
- Table Plug-in for jQuery - DataTables
- JavaScript library of Mobile-Friendly Interactive Maps - Leaflet
- JavaScript library dedicated to graph drawing - Sigmajs
- Cool, comfortable, responsive and easily customizable range slider - ion.rangeSlider

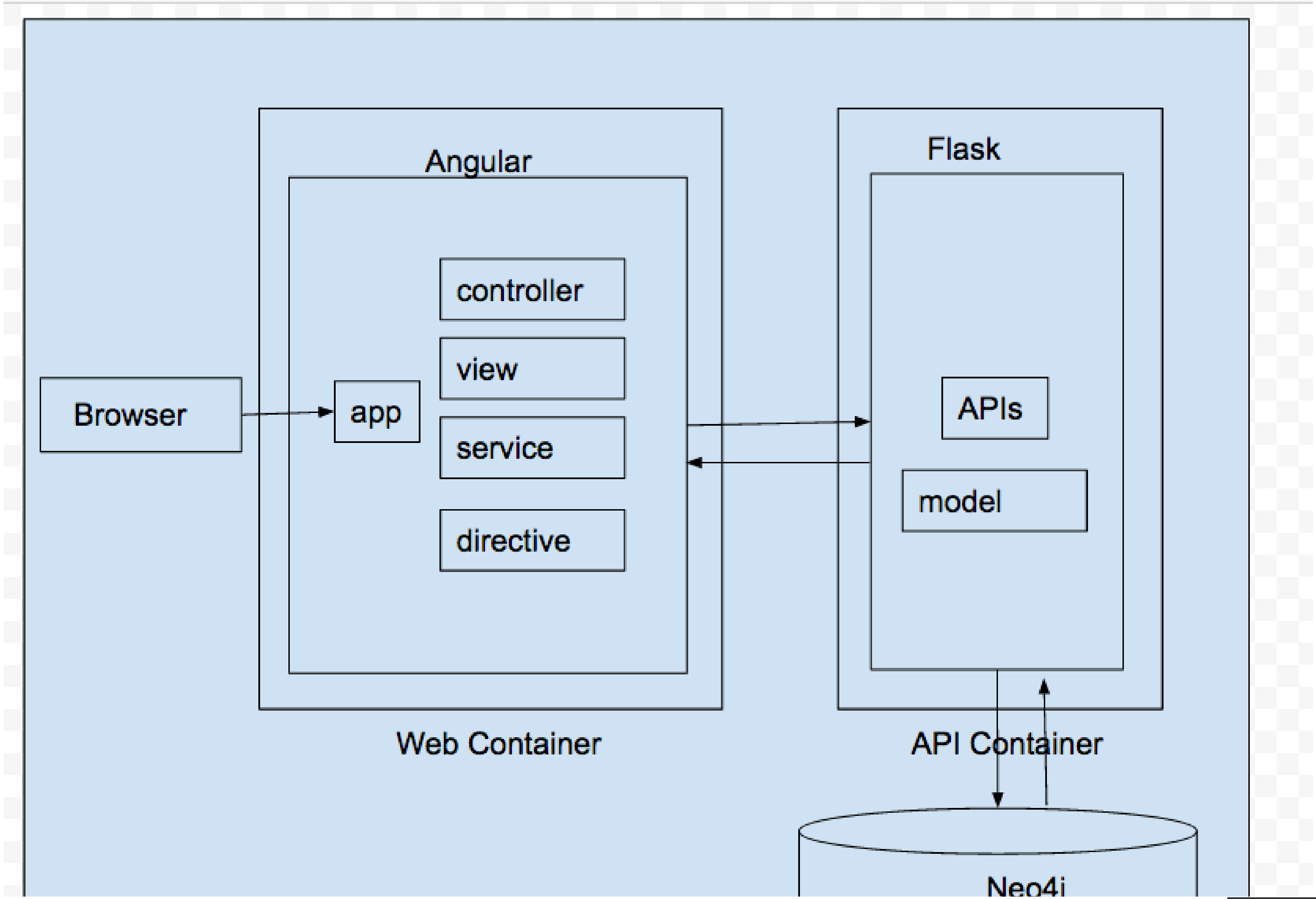
How to Run

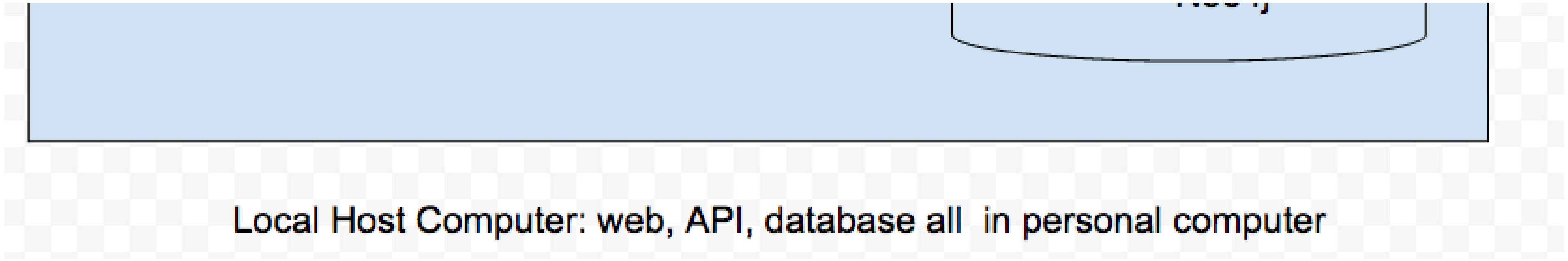
DocGraph's code was hosted on Github, you can config on your local environment according to the README. Or you can directly visit the deployed version on Heroku.

Core Implementation

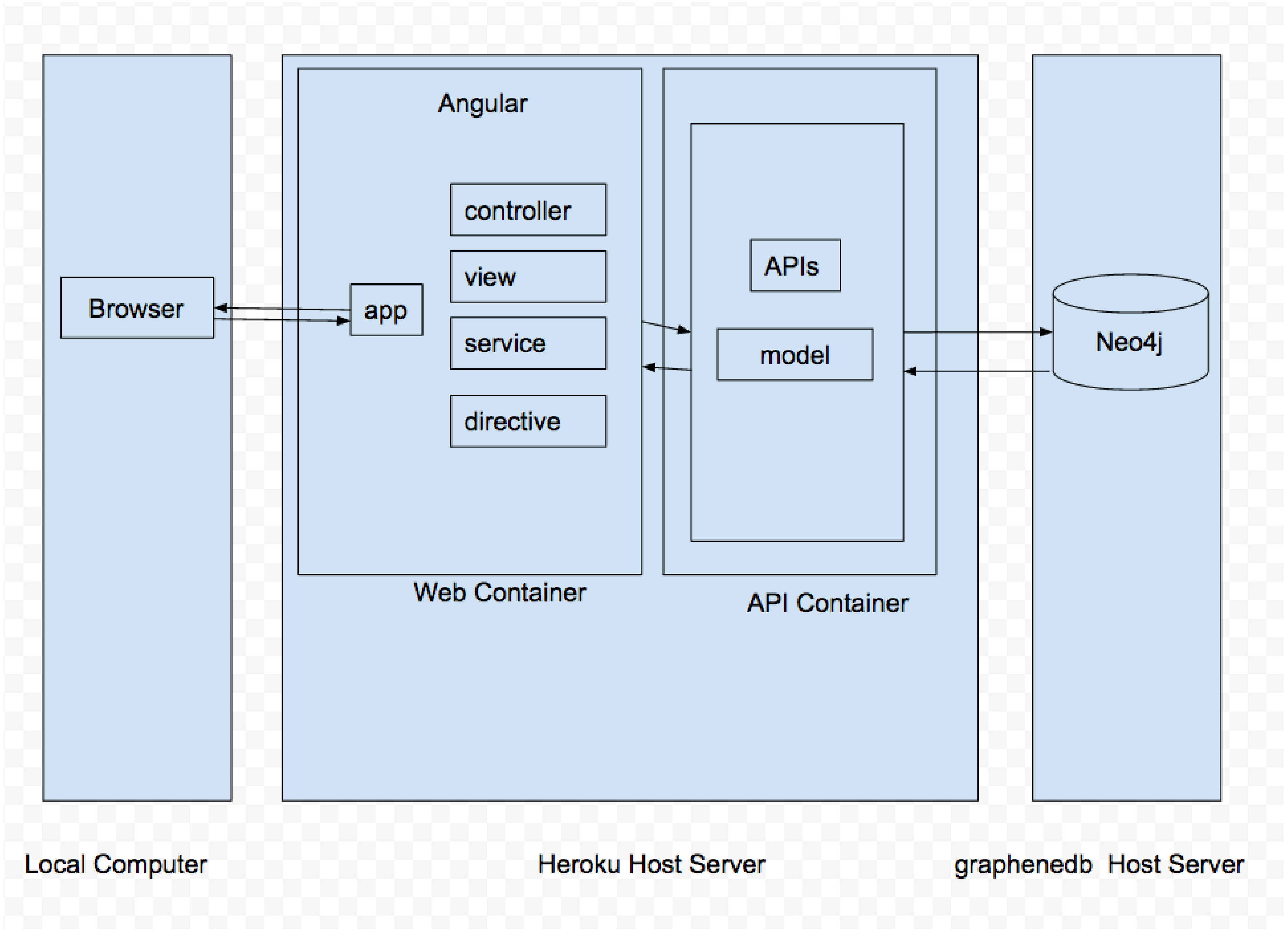
Architecture

- Dev Architecture: how does the application run in our local environment




Local Host Computer: web, API, database all in personal computer

- Deployed Architecture: the deployed application



Database: Neo4j

We use Neo4j as database because it's same-like with our bussiness needs, you can download it or just refer Github README's step: **"db environment"** .

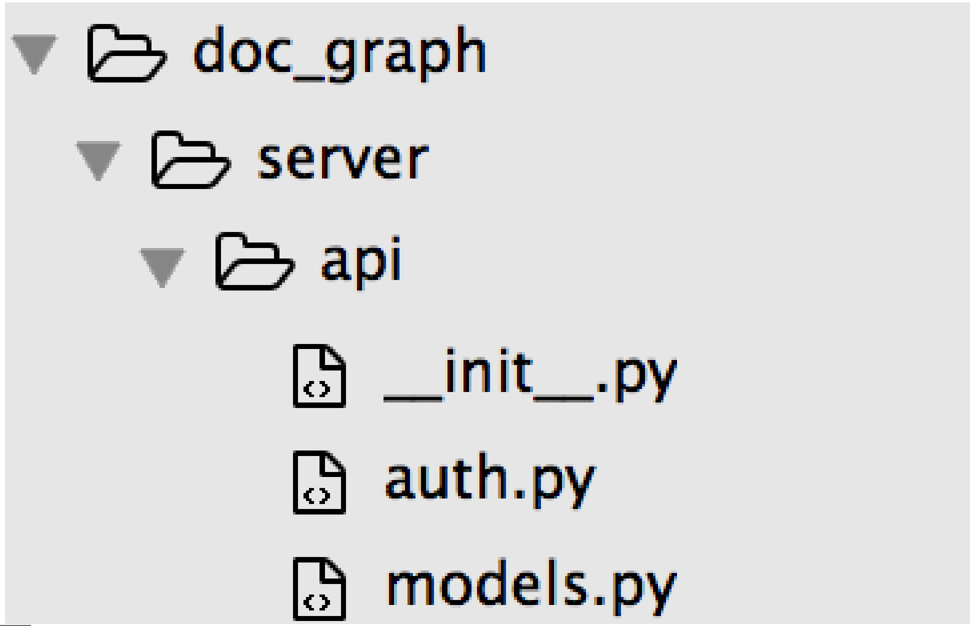
Server

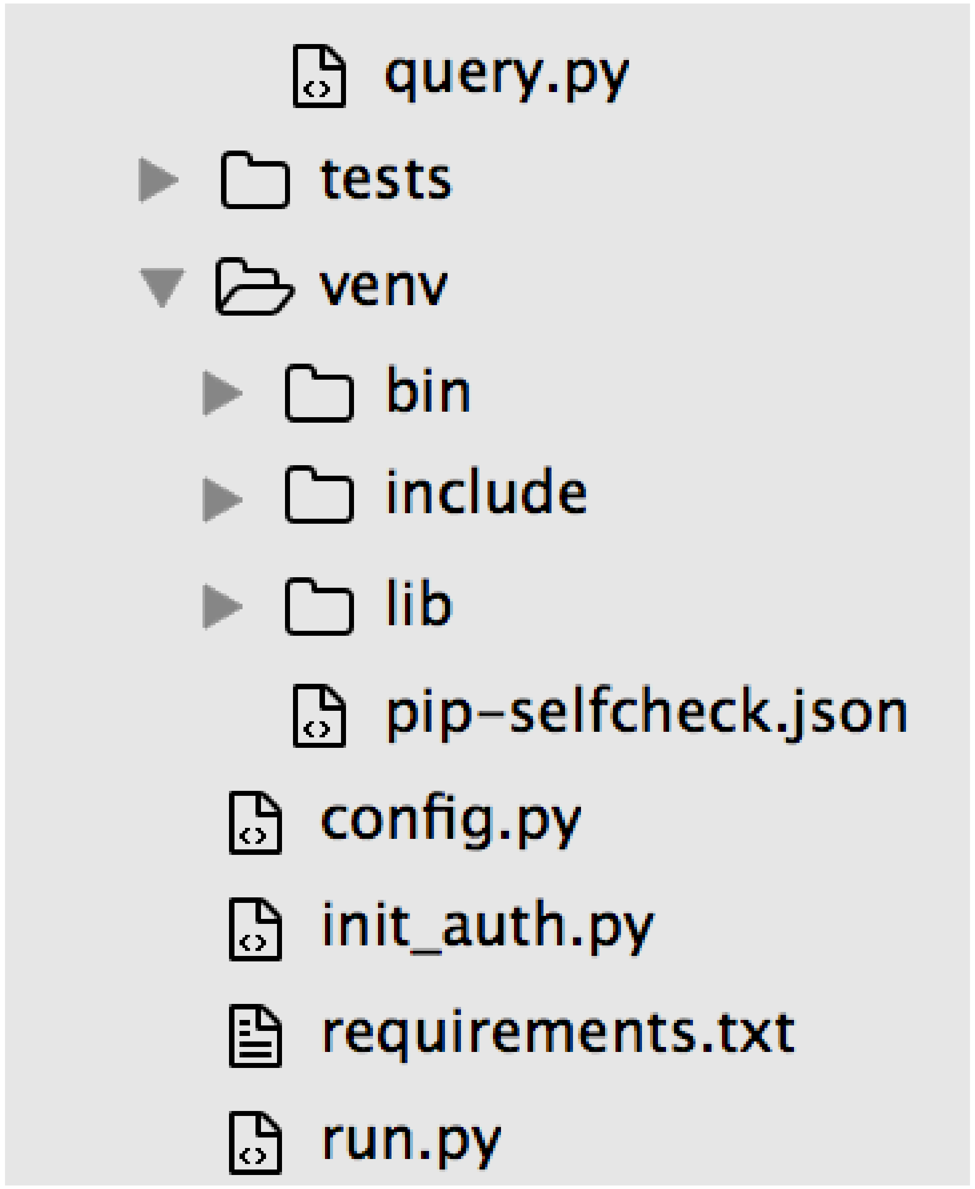
Preparation

The client provide the API Cohort-Builder, so we can call this server directly. Also we can create a server for doc graph app:

Make sure you have installed Python and Flask, if not please refer to the Github README's "**server environment**" part and make sure your environment right configured. After all that, let's get started to do some source code analysis!

Application Structure


```
doc_graph
  server
    api
      __init__.py
      auth.py
      models.py
```



```
      query.py
  ▶ tests
  ▼ venv
    ▶ bin
    ▶ include
    ▶ lib
      pip-selfcheck.json
    config.py
    init_auth.py
    requirements.txt
    run.py
```

- config.py: common used variables defined

- init_auth.py: create users for authentification

- run.py: execute instruction, application entrance, flask instance create here

- requirements.txt: contains a list of items to be installed, usually we can use *pip install*

- venv: provides support for creating lightweight "virtual environments" with our own site directories

- api/auth.py: query database for users, would been called by routes

- api/models.py: define user model, would been called in auth.py

- api/query.py: query database for data, would been called by routes

Datafeed

We feed the database with the DocPlot data zip file provided by customer via "Replace the *neo4j-community-2.2.0/data/graph.db* folder with the DocPlot *graph.db* folder". Then shut down any existing versions of neo4j you have running, start the docgraph version. First, import packages would been used：

/doc_graph/server/config.py

```
 # ...

'''
neo4jrestclient.client: to enable Python programmers already using Neo4j locally through python-embedded
'''

from neo4jrestclient.client import GraphDatabase
# ...
```

Then we can create the database instance:

/doc_graph/server/config.py

```
 #...
db = GraphDatabase('http://localhost:7474', username='...', password='...')
#...
```

Database Query

After create connection, we would need to initialize the database structure:

- Authentication: query and get user from database with two cases, by id and by name.

  /doc_graph/server/api/models.py

  ```
   #...
  def get_by_id(self, id):
      users_query = "match(u:User) where u.id='" + str(id) + "'  return u"
      users_sequence_object = db.query(users_query, params={}, returns=RAW)
      if len(users_sequence_object) == 1:
          data = users_sequence_object[0][0]['data']
          self.id = data['id']
          self.username = data['username']
          self.password = data['password']
      return self
  #...

  #...
  ```

```
def get_by_name(self, username):
    users_query = "match(u:User) where u.username='" + username + "'  return u"
    users_sequence_object = db.query(users_query, params={}, returns=RAW)
    user = None
    if len(users_sequence_object) == 1:
        data = users_sequence_object[0][0]['data']
        self.id = data['id']
        self.username = data['username']
        self.password = data['password']
    return self
#...
```

- Query: contains both query methods and data formated methods.

  Using provider_type as an example, to build provider type query.

  /doc_graph/server/api/query.py

```
 #...
provider_type = request.args.getlist('pt')

if len(provider_type) > 0:
    filters = True
    pt_tf = True
    path_list.append("(p:Provider)-->(s:Specialty)")
    provider_type = provider_type[0].split(',')
    pt_str = "['" + "','".join(str(pt) for pt in provider_type) + "']"
    pt_cy = "(s.specialty IN " + pt_str + ")"
    filter_list.append(pt_cy)
else:
    pt_tf = False
    pt_cy = ""
#...
```

Call the query, get the number of nodes it responded with. Initialize response dictionary, with the number of nodes and the query used to hit neo4j.

/doc_graph/server/api/query.py

```
 params = {}
query_sequence_object = db.query(q, params=params, returns=RAW)
resp_len = len(query_sequence_object)
json_response = dict(node_count=resp_len, cypher_query=q)
```

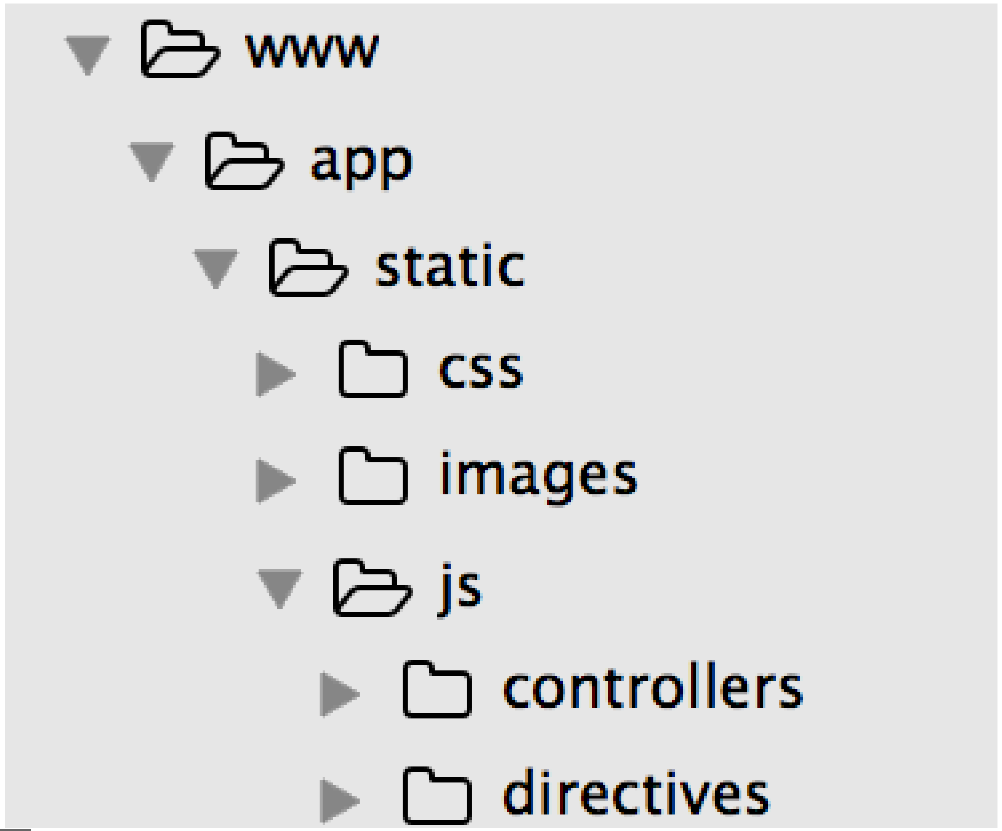
Build the whole query, loop each collection and all the relationships that each collection is connected to. We use the cypher query looks for all the relationships that the Provider has except for the 'Refers_To' relationship. The Neo4j cypher is a querying language which can let us execute the SQL directly. Also use provider_type as an example:

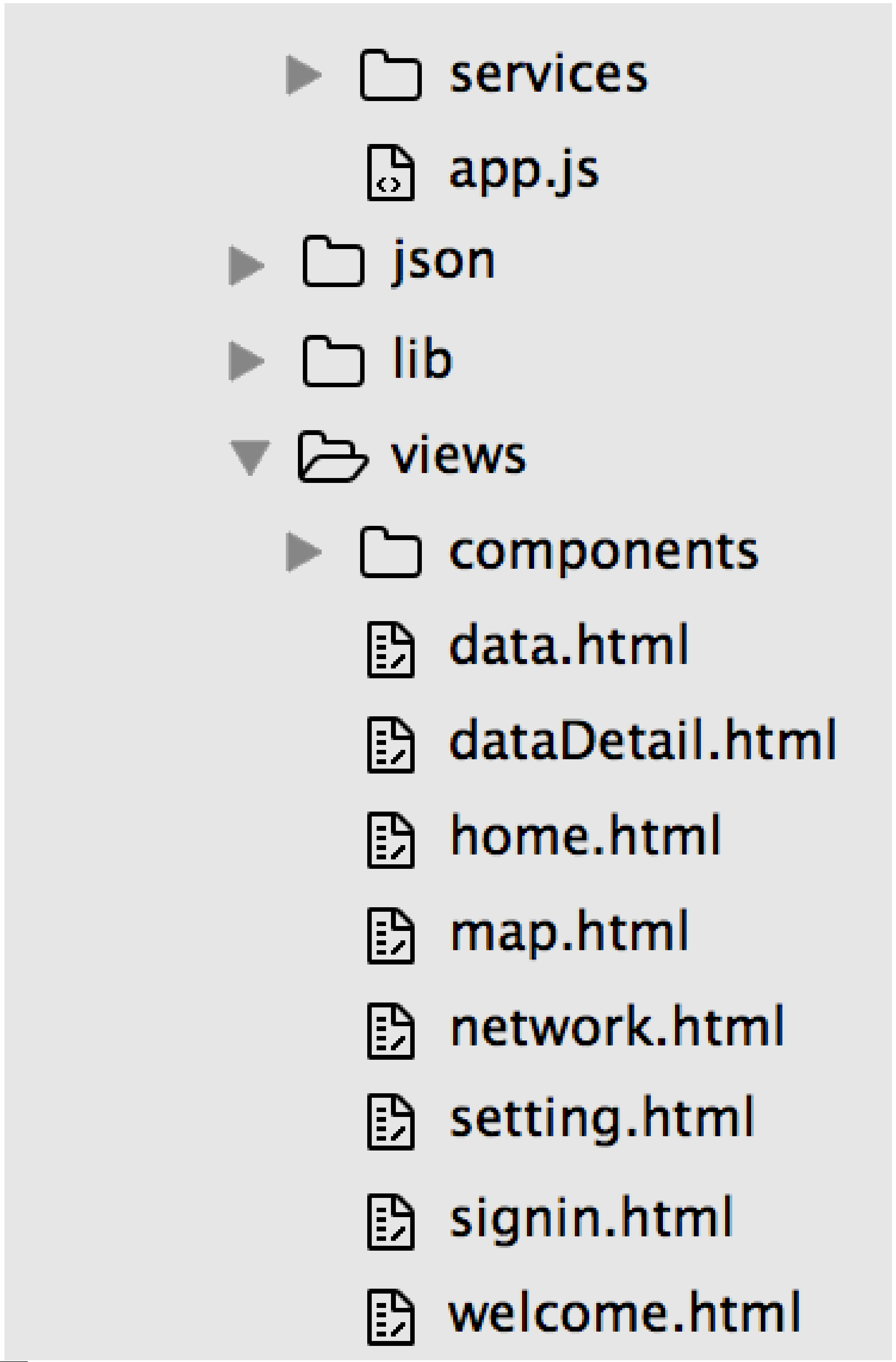
/doc_graph/server/api/query.py

```
 #...
def get_provider(node_json):
    #...
    for provider in provider_ids:
        q = "START p=node(" + str(provider) + ") MATCH (p)-[r]->(n) WHERE type(r)<>'Refers_To' RETUI
    params = {}
        query_sequence_object = db.query(q, params=params, returns=RAW)
        #...
        for rel in query_sequence_object:
        #...
#...
```

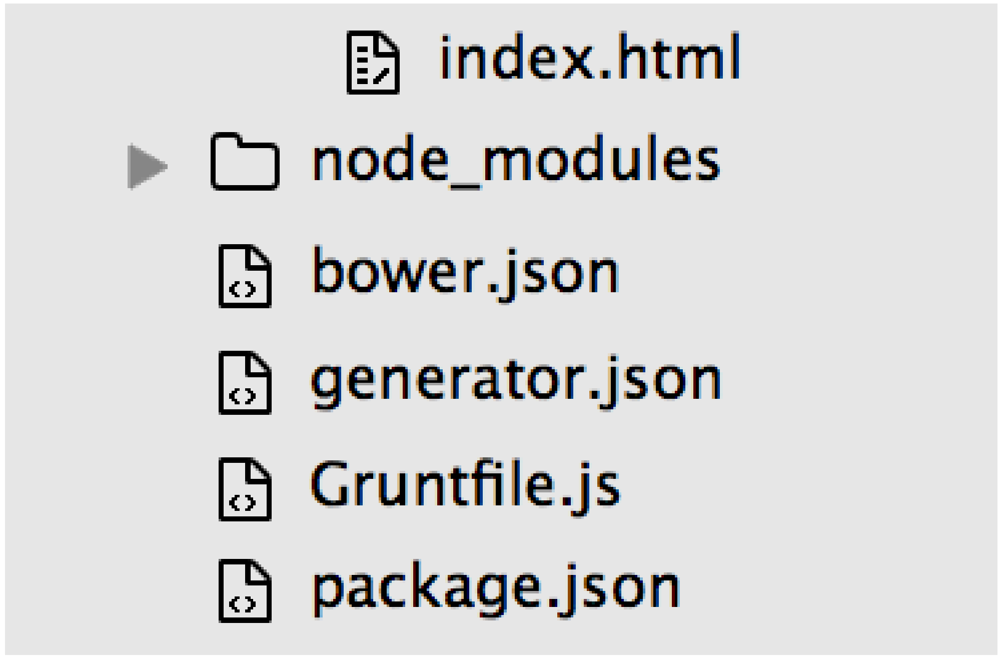
Web: AngularJS Front-Side

Application Structure





```
▶ services
  app.js
▶ json
▶ lib
▼ views
  ▶ components
    data.html
    dataDetail.html
    home.html
    map.html
    network.html
    setting.html
    signin.html
    welcome.html
```



- Gruntfile.js: used to configure or define tasks and load Grunt plugins

- package.json: used by npm to store metadata for projects published as npm modules. You will list grunt and the Grunt plugins your project needs as devDependencies in this file

- requirements.txt: contains packages would been used by grunt tasks

- index.html: home page

- node_modules: packages installed location

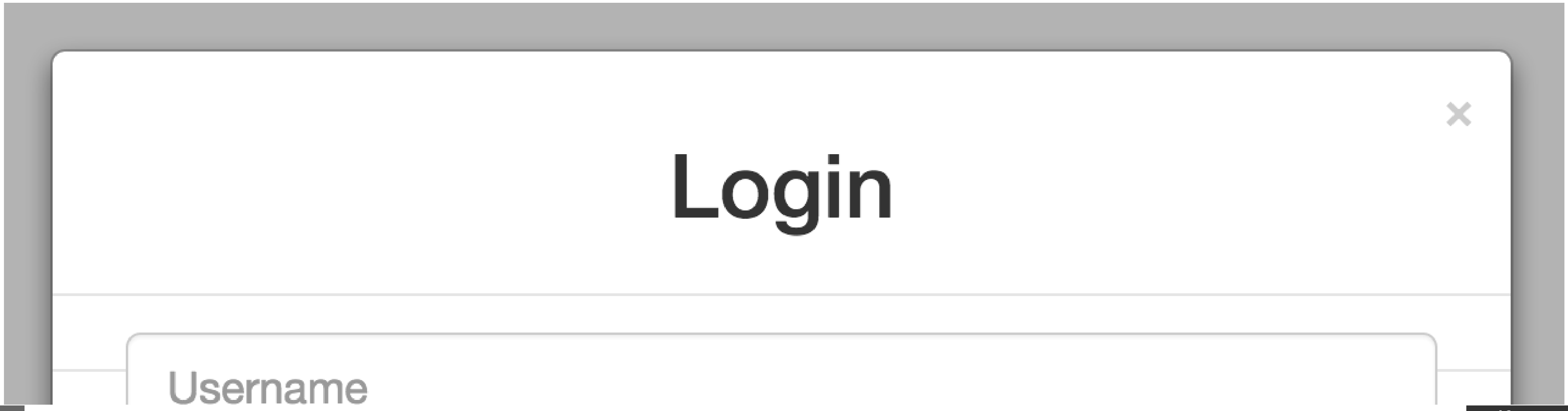
- css/js/images/libs/json: as it described

User Cases

- Auth Login
  - Click OK button then goto signin page
  - Sign in(use Bootstrap-Modal library)
- Query Builder
- Slider
- Draw the bar chart
  - Use dc.js
  - Use nv.d3.js
- Responsive Design
- Sigma.js
- Map

Auth Login

Click "OK" button in welcome page, there will pop up a login window, it need username and password to sign in. If you don't have account, you should register first(now not achieved).
If you have any question, you can click "Need help?"(now not achieved).



Password

Sign In

Need help? Register

docPLOT

Let's get started.

OK

Click OK button then goto signin page

- Front-end

```
 //to blind toHome() function to the button
<button type="button" class="btn btn-default" ng-click="toHome()">OK</button>
```

```
 <script>
    $scope.toHome = function () {
        (apis.offline || $cookies.get('user') ) ? $state.go('home') : $scope.toggleVisibility(true);
    }
</script>
```

Sign in

- Front-end
    - The signin page refer to Bootstrap-Modal.

      Modals are basically a dialog box that is used to provide important information to the user or prompt user to take necessary actions before moving on. Modal windows are widely used to warn users for situations like session time out or to receive their final confirmation before going to perform any critical actions such as saving or deleting important data.

You can easily create very smart and flexible dialog boxes with the Bootstrap modal plugin. The following example will show you how to create a simple modal with a header, message body and the footer containing action buttons for the user.

```
 <div class="modal fade" tabindex="-1" role="dialog">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label="Close"><spa
                <h4 class="modal-title">Modal title</h4>
            </div>
            <div class="modal-body">
                <p>One fine body&hellip;</p>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</butto
                <button type="button" class="btn btn-primary">Save changes</button>
            </div>
        </div><!-- /.modal-content -->
    </div><!-- /.modal-dialog -->
</div><!-- /.modal -->
```

- Back-end(database: neo4j; back-end language: python)
- connect to database
- Load json file to neo4j database

```
 user_json = json.loads(request.data)
username, password = user_json['username'], user_json['password']
```

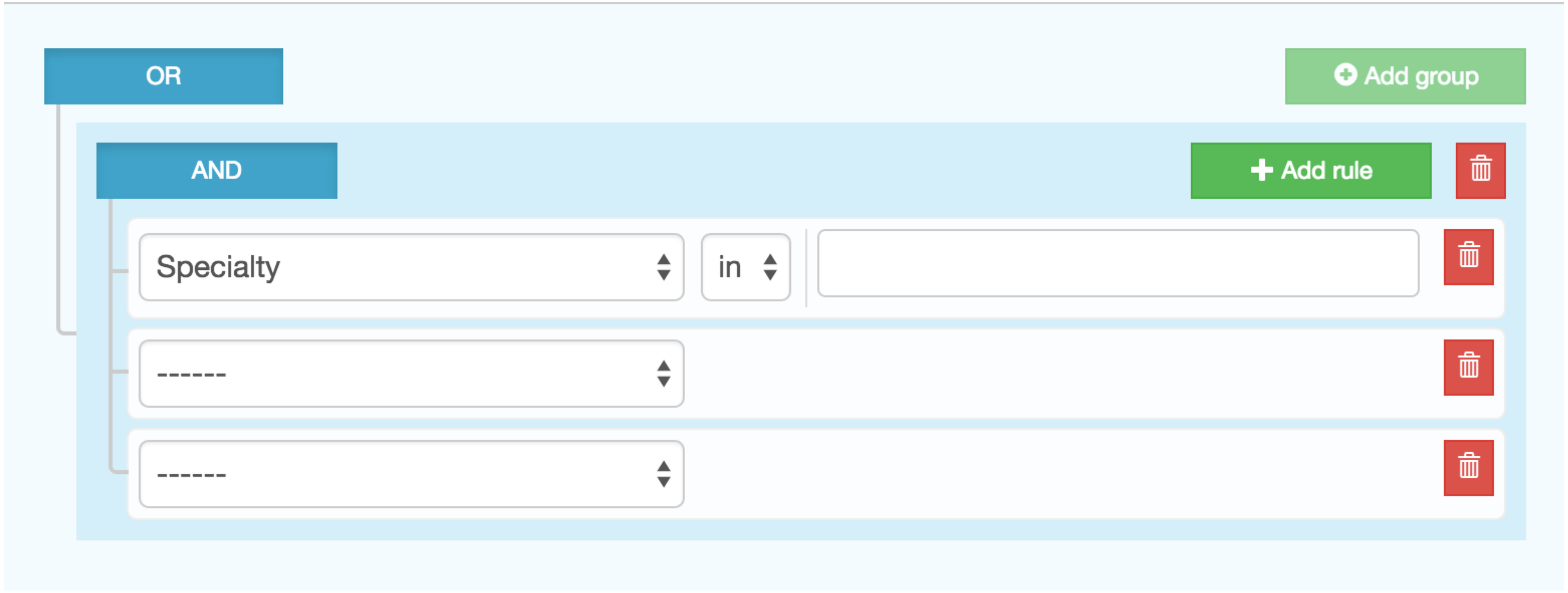
- Query users

```
def verify_user(username, password):

    if (username != None and password != None) or (username != '' and password != ''):
        users_query = "match(u:User) where u.username='" + username + "'  return u"
        users_sequence_object = db.query(users_query, params={}, returns=RAW)
        if len(users_sequence_object) == 1:
            data = users_sequence_object[0][0]['data']
            db_password = data['password']
            return check_password_hash(db_password,  password)
    return False
```

Query Builder

Filter the entity according to the region and type of conditions.



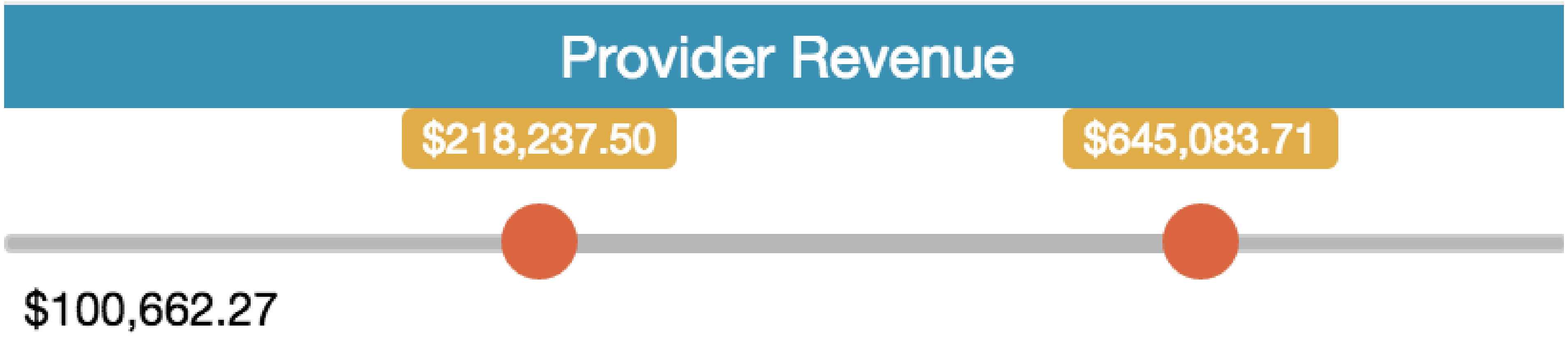
- We use the third-party library: Query Builder. QueryBuilder is an UI component to create queries and filters.It can be used on advanced search engine pages, administration backends, etc. It outputs a structured JSON of rules which can be easily parsed to create SQL/NoSQL/whatever queries.
- First download the Query Builder plugin and complete installation. Then include query-builder.css and query-builder.js in the index.html, add QueryBuilder to any <div> you want.

```
 //add QueryBuilder to <div> in the home.html
<div id="cohortContainer"></div>
```

```
 //below code add into cohort-service.js
<script>
    $('#cohortContainer').queryBuilder({
        filters: [ ... ]//add formatted data
    });
</script>
```

Slider

Set provider revenue value as a set of scope with ion.rangeSlider plugin to build a beautiful range slider. This slider shows the scope's min and max value and prettifies the prefixes with "$". This set of scope intergrates with map, data and network list. When change the value range on the slider, the map, data and network scope will change their each list with the corresponding data.



- We use the third-party library ion.rangeSlider to create the range slider. Ion.RangeSlider plugin is an easy, flexible and responsive range slider with tons of options. The plugin supports callbacks(onStart, onChange, onFinish, onUpdate), supports external methods (update, reset and remove) to control it after creation.
- First download the ion.rangeSlider plugin and add the library to project. Then include ion.rangeSlider.css and ion.rangeSlider.min.js in the index.html, add rangeSlider to any <div> you want.

```
 <div id="revenueSlider"></div>
```

```
 <script>
    $('#revenueSlider').ionRangeSlider({
        type: "double", //Set type to double
        grid: true, //Show grid
        min: 0,
        max: $scope.revenues.length - 1, //Set specify range
        prefix: "$", //Add prefix "$"
        from: $scope.revenues[0],
        to: $scope.revenues[$scope.revenues.length - 1], //set default own start and end value
        step: 1,
        prettify_enabled: true,
        prettify: function (num) {
            return $filter('currency')(parseFloat($scope.revenues[num]), '');
                },
        onFinish: function (data) {

        }
    });
</script>
```

- Intergrate with map, data and network scope.

There are three children states: data, map, network, and two different cases:

Case1: data state

Once the datalist change, or searchvalue change(which would cause datalist change), then the table will change. So it's not necessary to override this method. Because we could add angular two-way bind.
We use angular-datatables library to create dataList.

Case2: map/network states

Once the maplist/networklist change, we need to update the map or sigma graph separately, because the library draw map and graph was not angluar concerned, so override renderContent would be necessary.

- Define a method which should be override by children state controller when slider max/min changed or search input changed, this would be called.

  ```
   <script>
      $scope.renderContent = function() {console.log('home-controller.js method renderContent cal
  </script>
  ```

- Add below code to onFinish function which included in rangeSlider creation code.

  ```
   <script>
      //set slider intergrate with mapList, dataList, networkList
      $scope.$apply(function() {
          $scope.revenueFrom = $scope.revenues[data.from];
          $scope.revenueTo = $scope.revenues[data.to];

          //data filtered by revenue slider
          $scope.filteredProviders = $scope.filterProvidersWithSlider();
          $scope.filteredNetworks = $scope.filterNetworksWithSlider();
  ```

```
        //prepare for child page, there are based on filteredProviders, filteredNetworks
        $scope.mapList = $scope.prepareMapList();//description parepare map list for data page.
        $scope.dataList = $scope.prepareDataList();//description parepare data list for data pag
        $scope.networkList = $scope.prepareNetworkList();//description parepare network list for
        $scope.markerSearchValues = _.pluck($scope.mapList, 'node title');

        $scope.renderContent();
    });
</script>
```

- Add below code in mapController.js and networkController.js separately to override renderContent().

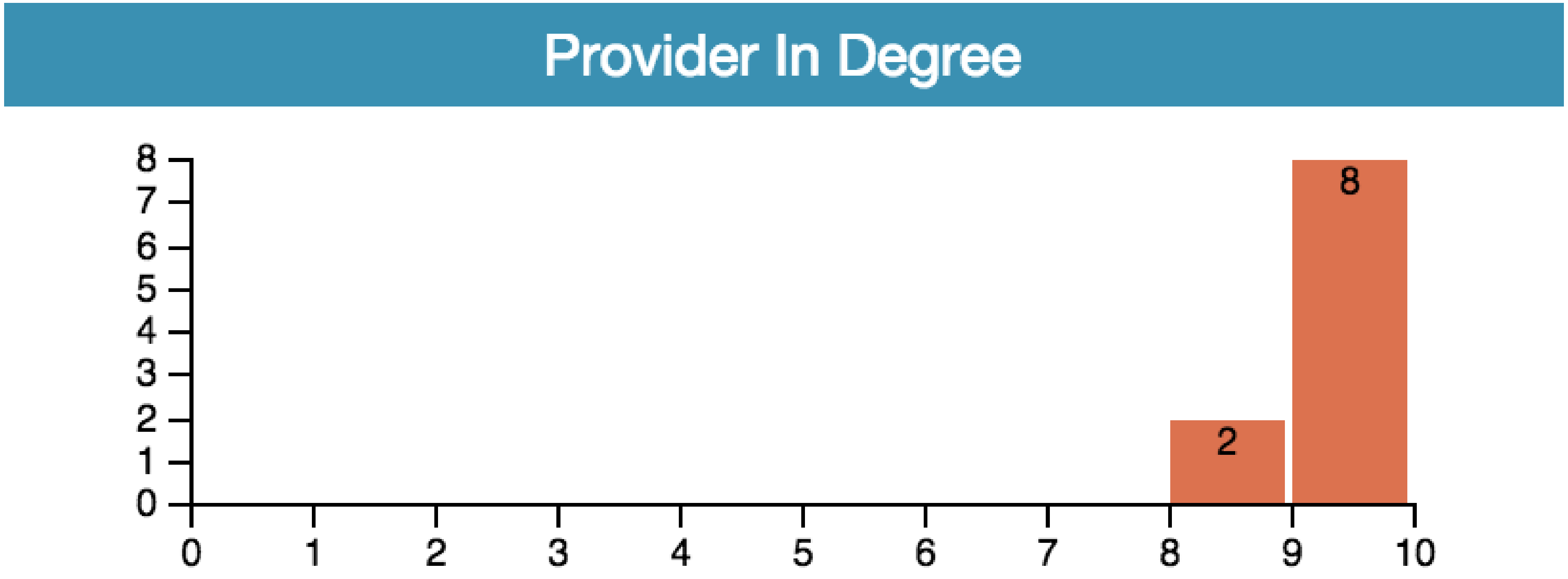
```
 <script>
    //override renderContent() in mapController.js to update mapList
    $scope.$parent.renderContent = function() {
        $scope.addMaker();
    }
</script>
```

```
 <script>
    //override renderContent() in networkController.js to update networkList
    $scope.$parent.renderContent = function(){
        $scope.drawGraph(true);
    }
</script>
```

Draw the bar chart

Use dc.js

The graph is a histogram representing the distribution of in degree values from the sample of data, and shows the number of each rawProvider.metrics.out_degree data.



- We use the third-party library dc.js to create the bar chart. Dc.js is a javascript charting library with native crossfilter support and allowing highly efficient exploration on large multi-dimensional dataset (inspired by crossfilter's demo). It leverages d3 engine to render charts in css friendly svg format. Charts rendered using dc.js are naturally data driven and reactive therefore providing instant feedback on user's interaction. The main objective of this project is to provide an easy yet powerful javascript library which can be utilized to perform data visualization and analysis in browser as well as on mobile device.
- First download the dc.js plugin and add the library to project. Then include dc.css and d3.js, dc.js and

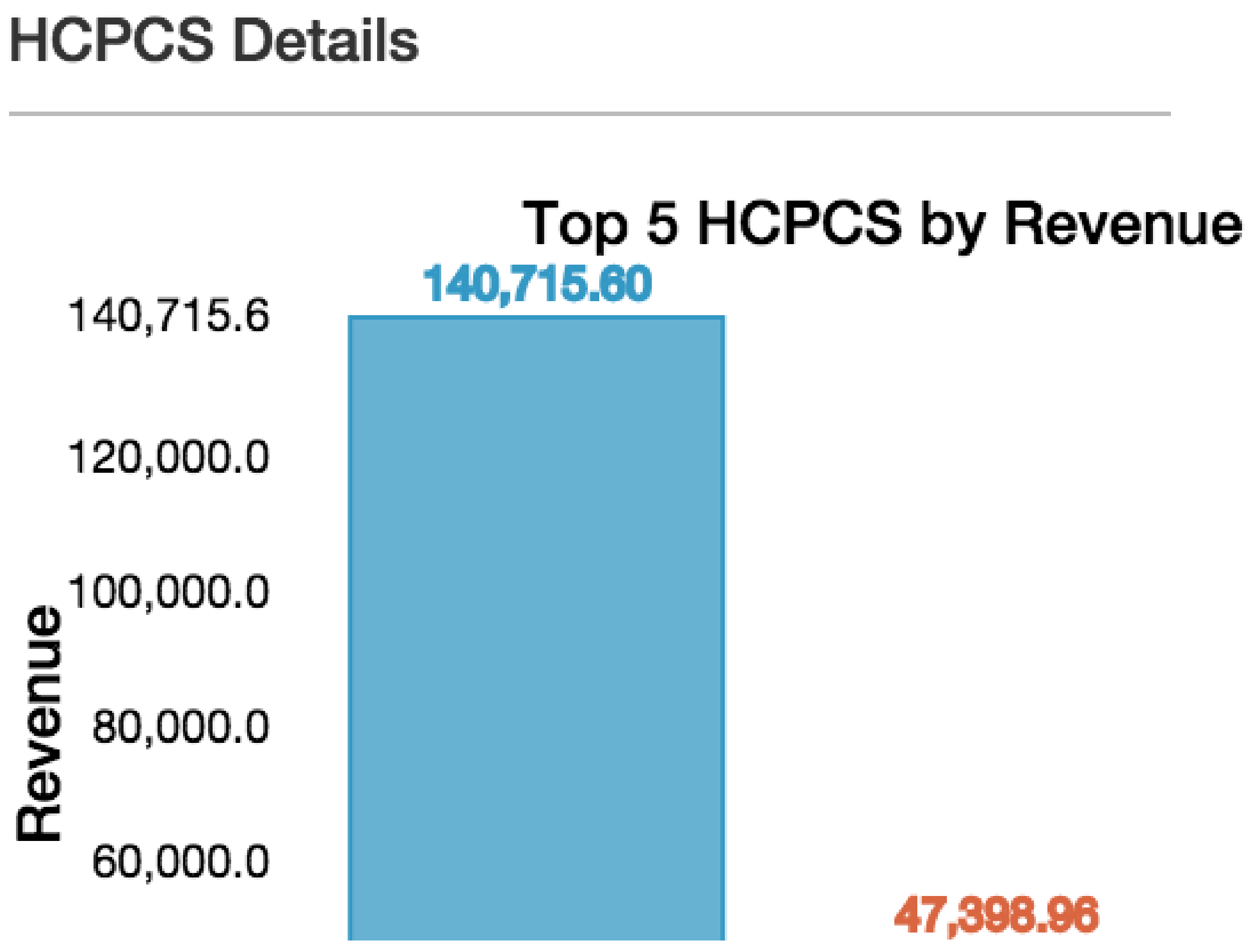
crossfilter.js(*must follow this sequence*, dc.js is based on d3.js) in the index.html, add barChart to any <div> you want.

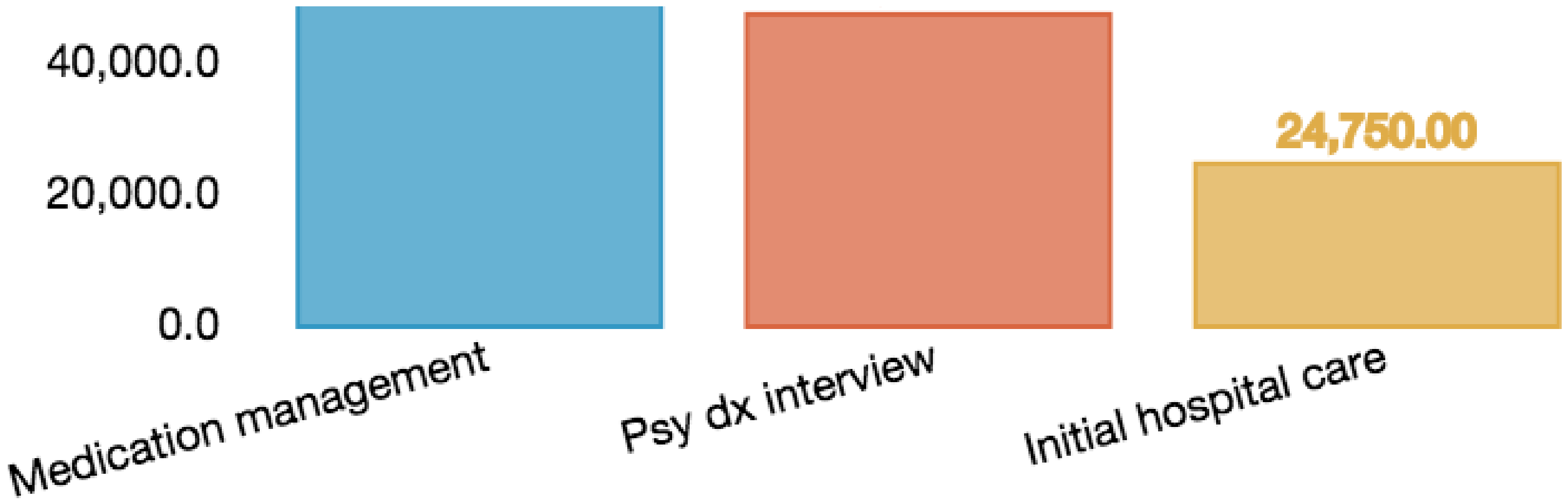
Note: css of bar chart(histogram) must be defined in the js file, otherwise invaild.

Provider In Degree

Display values on the top of bars, to add below code into renderlet() function.

Use nv.d3.js





Total Revenue: $257,065.51

- We use the third-party library: nvd3, we can view the reference document: D3, Generate a Histogram with D3, nvd3 documents
- Include nv.d3.css in the page head and d3.js, nv.d3.js(*must follow this sequence*, nv.d3.js is based on d3.js) after the dependencies.

The generate function should contain code that creates the NVD3 model, sets options on it, adds data to an SVG element, and invokes the chart model. The generate function should return the chart model.

```
<div id="chart_hcpcs">
   <span>HCPCS Details</span>
   <svg height="280" width="377"></svg>
   <span>Total Revenue:  {{totalRevenue}}</span>
```

```
</div>
```

```
 <script>
  nv.addGraph(function() {
    var chart = nv.models.discreteBarChart()//define a discreteBarChart model
        .x(function(d) { return d.label })//Proxy function to return the X value so adjustments can
        .y(function(d) { return d.value })//Proxy function to return the Y value so adjustments can
        .color($scope.$parent.$parent.colors)//Colors to use for the data.
        .staggerLabels(true)//Makes the X labels stagger at different distances from the axis so the
        .tooltips(false)//tooltips are unable
        .showValues(true)//Prints the Y values on the top of the bars.
        .transitionDuration(350);//Define take 350ms duration time when updating chart.

    d3.select('#chart_hcpcs svg')//blinding an SVG
    .datum($scope.dataHcpcs)//add data to an SVG
        .call(chart);//call chart

    return chart;
  });
</script>
```

- Extension: list examples about how to set x axis text css, add title in the top/left of SVG, and window resize.

Note: below code add to nv.addGraph()

```
 <script>
    //set x axis text css
    d3.select('#chart_hcpcs svg .nv-x.nv-axis>g>g')
    .selectAll('g')
    .selectAll('text')//select x axis text node
    .style('transform', function() {//rotate x axis text
        return 'rotate(-15deg)';
    })
    .style('text-anchor', function() {//align text to the end
        return 'end';
    });

    var x_title = $('#chart_hcpcs svg').width() / 2;
    var x_titleY = -$('#chart_hcpcs svg').height() / 2;
    var y = 11;
    var text_titleY = 'Revenue';

    //add title in the top of SVG
    d3.select('#chart_hcpcs svg')
          .append('text')
          .attr('class', 'title')
          .attr('x', x_title)
          .attr('y', y)
          .attr('text-anchor', 'middle')
          .attr('font-size', '1.3em')
          .attr('font-weight', 500)
          .text($scope.dataHcpcs[0].key);

    //add title in the left of SVG
    d3.select('#chart_hcpcs svg')
          .append('text')
          .attr('class', 'titleY')
          .attr('x', x_titleY)
          .attr('y', y)
          .attr('text-anchor', 'start')
```

```
            .attr('font-size', '1.3em')
            .attr('font-weight', 500)
            .style('transform', function() {
              return 'rotate(-90deg)';
            })
            .text(text_titleY);

      //The window is resized when execute each time. Prevent the barchart can't spread.
      nv.utils.windowResize(function() {
            chart.update();
            d3.select('#chart_hcpcs svg .nv-x.nv-axis>g>g')
                .selectAll('g')
                .selectAll('text')
                .style('text-anchor', function() {
                      return 'end';
                  });
            d3.select('#chart_hcpcs svg>text .title')
                  .attr('x', x_title)
                  .attr('y', y);
            d3.select('#chart_hcpcs svg>text .titleY')
                  .attr('x', x_titleY)
                  .attr('y', y);
      });
</script>
```

Responsive Design

//Todo

Sigma.js

Map

End