

Da Meng
Jin ho Kim
Kevin Valle
Chanhee Park
Seung Youp Baek

Phase I report

Introduction

Problem

We find that when a person is searching for a job, normally he could only filter the job information with a few criteria, mostly location and company. This will lead to lots of inconvenience. For example, a person has to do some research about which companies are out there and are currently hiring. He also has to know the location that provides a lot of jobs so that he could have enough choices. The most annoying part is that the person has to look through the description of a job posting to find out if he is qualified or interested in the job. Our website will save a lot of work for the client. Our clients can find their satisfied jobs with just a few clicks.

The use cases

Our website is a navigable database for job postings. Our job posting data is categorized by locations, companies, mainly-used languages and skill sets. Each individual job posting belongs to a company of a specific location. Each job posting requires proficiency of one main stream programming language and general skills of one skill set. In the listing or the page of each job, there are clickable links that connect to the company, location, language and skillset that is associate with it.

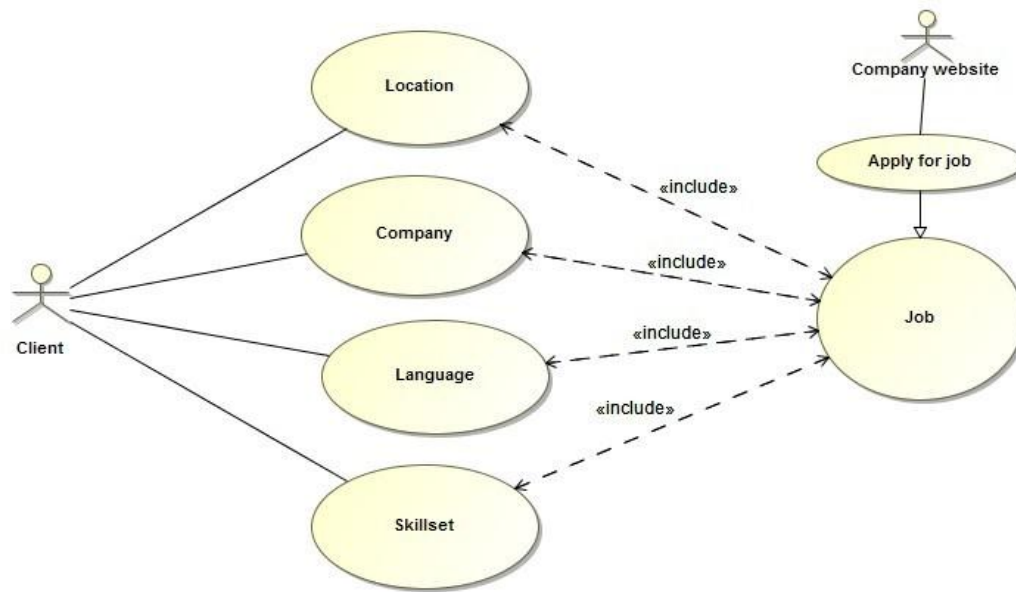


Figure 1 Use Case Diagram

From our splash page, the client can navigate among location, company, language and skillset. In each of these options, there will be a number of specific choices for client to click, such as “Austin, TX” in location option, “Google” in company option, “Java” in language option and “Mobile computing” in skillset option. In each choice that a client decides to look into, there is a list of available jobs. The client can navigate to another section related to a job he likes using the links on the job listing. And finally he will find the most suited one.

Model Design

RESTful API

The RESTful API is accessible from our server, because we have it as a public API. The main idea for it is for clients to be able to extract information from our database. The client should be able to get list all jobs, locations, programming languages, and companies, as well as retrieve them by searching IDs. Following is the documentation for listing and retrieving. Currently, the data is being

accessed from JSON files in our repo. The next step would be add model logic to the methods responsible for our GET endpoints

Jobs

List all jobs [GET]

Jobs [/job]

Response 200 (application/json)

```
[{
  "id": 1,
  "title": "Web Developer",
  "locationID": 1,
  "companyID": 1,
  "languageID": 1,
  "skillsetID": 1,
  "description": "something",
  "link": "some link"
}]
```

Retrieve a job [GET]

Job [/job/{id}]

Response 200 (application/json)

```
Body
{
  "id": 1,
  "title": "Web Developer",
  "locationID": 1,
  "companyID": 1,
  "languageID": 1,
  "skillsetID": 1,
  "description": "something",
  "link": "some link"
}
```

Response 404

Header

Content-type: application/json

Body

```
{ "error": "Item does not exist" }
```

Locations

List all Location [GET]

Locations [/location]

Response 200 (application/json)

```
[{
  "Location_ID" : 1
  "Location": "Austin, TX",
  "Location_description" : "A great city to live in",
  "Location_image" : "http://www.seesawaustin.com/wp-content/uploads/2014/07/Austin-
jobs.jpg"
},{
  "Location_ID" : 2
  "Location": "Seattle, WA",
  "Location_description" : "A big city",
  "Location_image" :
"http://upload.wikimedia.org/wikipedia/commons/2/2f/Space_Needle002.jpg"
}]
```

###Retrieve a location [GET]

Location [/location/{id}]

Response 200 (application/json)

```
Body
{
  "Location_ID" : 1
  "Location": "Austin, TX",
  "Location_description" : "A great city to live in",
  "Location_image" : "http://www.seesawaustin.com/wp-content/uploads/2014/07/Austin-
jobs.jpg"
}
```

Response 404

Header

Content-type: application/json

Body

```
{"error" : "Item does not exist."}
```

Languages

List all Languages [GET]

Languages [/language]

Response 200 (application/json)

```
[{
  "Language_ID" : 1,
  "Language_Name" : "java",
  "Language_description" : "multi platform",
  "Language_image" : "link"
},{
  "Language_ID" : 2,
  "Language_Name" : "python",
  "Language_description" : "snake",
  "Language_image" : "link"
}]
```

Retrieve a language [GET]

Language [/language/{id}]

Response 200 (application/json)

```
Body
{
  "Language_ID" : 1,
  "Language_Name" : "java",
  "Language_description" : "muti platform",
  "Language_image" : "link"
}
```

Response 404

Header

Content-type: application/json

Body

```
{"error" : "Item does not exist."}
```

Company

List all Companies [GET]

Companies [/company]

Response 200 (application/json)

```
[{
  "Company_ID": 1,
  "Company_Name": "Google",
  "Company_description" : "Search engine",
  "Company_image" :
"http://www.google.com/doodle4google/images/splashes/featured.png"
},{
  "Company_ID": 2,
  "Company_Name": "Oracle",
  "Company_description" : "Database"
  "Company_image" : "http://192.95.57.65/wp-content/uploads/2014/07/logo-oracle.jpg"
}]
```

###Retrieve a company [GET]

Company [/company/{id}]

Response 200 (application/json)

Body

```
{
  "Company_ID": 1,
  "Company_Name": "Google",
  "Company_description" : "Search engine",
  "Company_image" :
"http://www.google.com/doodle4google/images/splashes/featured.png"
}
```

Response 404

Header

Content-type: application/json

Body

```
{"error" : "Item does not exist."}
```

Skillset

List all skills [GET]
<p>Skillsets [/skillset]</p> <p>Response 200 (application/json)</p> <pre>[{ "Skillset_ID" : 1, "Skillset": "Spring, J2EE, Hibernate", "Skillset_description" : "Spring, J2EE, Hibernate", }, { "Skillset_ID" : 2, "Skillset": "Mobile computing", "Skillset_description" : "Mobile computing, IOS, Andriod", }]</pre>
Retrieve a skillset [GET]
<p>Skillset [/skillset/{id}]</p> <p>Response 200 (application/json)</p> <p>Body</p> <pre>{ "Skillset_ID" : 1, "Skillset": "Spring, J2EE, Hibernate", "Skillset_description" : "Spring, J2EE, Hibernate", }</pre> <p>Response 404</p> <p>Header</p> <p>Content-type: application/json</p> <p>Body</p> <pre>{"error" : "Item does not exist."}</pre>

Setting up Rackspace Server

Setting up the Rackspace server was not that difficult because we followed along with the PowerPoint presentation given during class at <http://rack.to/cs373>.

Flask

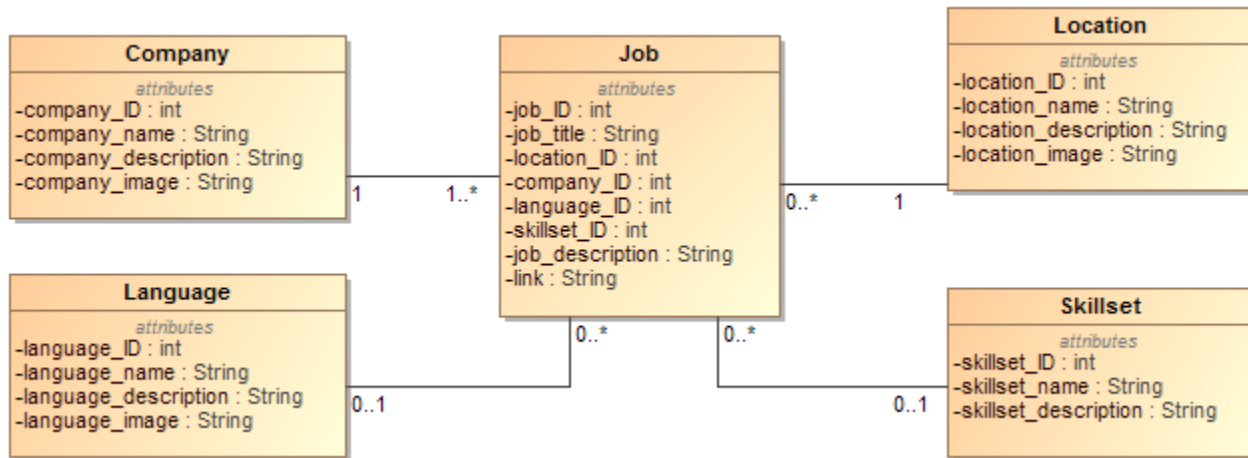


Figure 2 UML Diagram

The UML diagram above illustrates our model. We have five major tables: Job, Company, Location, Language and Skillset. The relationships we create are many-to-one relations. Each job will be mapped to one company, one location, one language and one skillset. The attributes location_ID, company_ID, Language_ID, and skillset_ID are foreign key that connects Job table to other tables. In our model, we majorily focus on Jobs.json's entry, so it was more efficient to have Job class to be the main port for other classes to be linked to.

Model Example

```
class Job(db.Model):
    __tablename__ = 'job'
    __table_args__ = {'useexisting': True}
    id = db.Column(db.Integer, primary_key=True, nullable=False)
    title = db.Column(db.Text, nullable=False)
    description = db.Column(db.Text)
    link = db.Column(db.Text)
    locationid = db.Column(db.Integer, ForeignKey('location.id'))
    companyid = db.Column(db.Integer, ForeignKey('company.id'))
    languageid = db.Column(db.Integer, ForeignKey('language.id'))
    skillsetid = db.Column(db.Integer, ForeignKey('skillset.id'))
```


As seen in the example above, the models on top of a database. Currently there is only a PostgreSQL database initialized on the cloud server. Defining the class for a model should make it easier to query since SQL will not have to be explicitly written. Next steps include replacing the logic in the API methods with model logic and binding models to templates in order to switch from a static page to a dynamic page.

Unit tests

```
def testCreatingLocation(self):

    location = Location('Austin, TX', 'Live music capital of the world', 'austin.jpg')

    assert location.Location == 'Austin, TX'

    assert location.Location_description == 'Live music capital of the world'

    assert location.Location_image == 'austin.jpg'


def testModifyingLocation(self):

    location = Location('Austin', 'Live music capital of the world', 'austin.jpg')

    location.Location = 'New York, New York'

    location.Location_description = 'City that never sleeps'

    location.Location_image = 'newyork.jpg'

    assert location.Location == 'New York, New York'

    assert location.Location_description == 'City that never sleeps'

    assert location.Location_image == 'newyork.jpg'
```

The tests above are an example of unit tests written to test our models. They test the creation of model objects and the modification of those objects after creation.

Data

We gain fine-grained data from Indeed.com using its built-in advance search. We looked for qualified jobs of each company that we are looking into. Then we filter and categorized them by location. Finally, we check which skillset does a particular job belong to and which language that the job mainly required.

Front-end

In order for us to create the static pages, we incorporated HTML, CSS, JavaScript, and components from Twitter BootStrap 3. We decided to use three different categories (Languages, Companies, and Locations) in our splash page as

well as on our navigation bar for the client to look up jobs. Each category contained more specific listing for the client to explore into. For example, by going into Companies, the client would now be able to go into a specific company that is listed on that page. Within a specific listing, the client can now explore posted jobs that are related to that specific listing. For example, once the client has selected a specific company, all the jobs that that

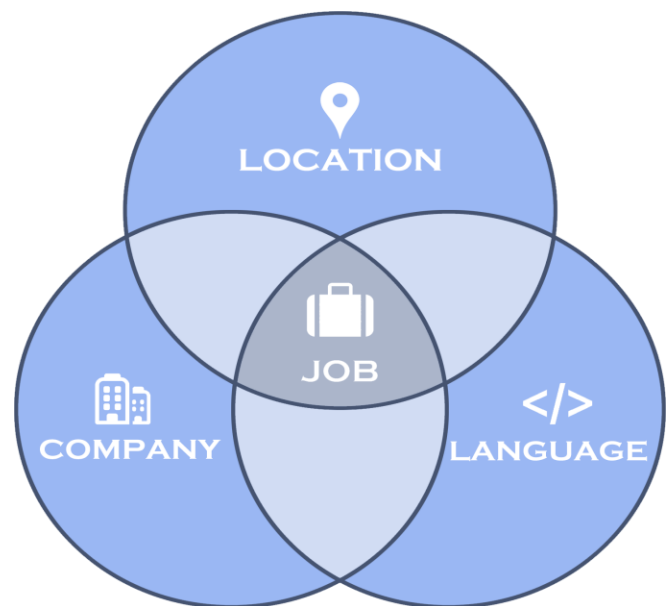


Figure 3 Three categories Venn diagram

specific company has posted will be listed there. The jobs are listed in an accordion panel view and once expanded, it should show the job title, company, location, and job description. And in order for the client to view more related jobs within a single page, they can click on the “Show More” button



Figure 4 Job Listing



Figure 5 Navigation Bar

Another way to navigate through our website is by using the navigation bar at the top. Due to the small number of languages, companies, and locations, we decided to put all our specific listings under each category. This may change into a search bar as we expand on our project. The navigation bar gives the client the option to choose those specific listings for each category without having to go back to splash page. Essentially, it becomes a shortcut for clients. Through navigation bar, it does not give the client an option to go to the specific categories page but it does give them the ability to go to the specific listing within that category.

that takes them to the individual job page that show all the jobs for similar language, similar skillset, similar company, and similar location.

Programmer Jobs
Languages
Companies
Location
About

Application Developer, Corporate Asset Management Engineering

Google // New York City, NY

Description

- Build custom front-ends (web and mobile), back-end services, and third-party integrations to automate business processes.
- Propose, design, and implement business workflows to streamline asset management lifecycles.
- Maintain strong development practices including technical design and writing clean, modular, well tested and self-sustaining code.
- Collaborate with business analysts, program managers, end users and other internal teams to translate business requirements into technical solutions that work at Google scale.
- Integrate third-party products with Google's internal systems as well as support and upgrade implemented systems.

Apply here!

Web Front-end

Software Developer 2

Python

Google

Systems Engineer


New York City, NY

© 2015 Team Programmer Jobs

Figure 6 Job Page

Within our static pages, there are few links that are not referenced anywhere because we do not have the specific location/language. Among these are “Los Angeles, CA”, “New York City, NY”, and “Python”. Once we have our database model working, we may expand it to include all the locations in our Locations.json file but for now, we decided to leave it out.

Programmer Jobs
Languages
Companies
Location
About




Google
www.google.com

Description

Search engine

Picture



Job Listings

Application Developer, Corporate Asset Management Engineering

Google New York City, NY

Job Description

Build custom front-ends (web and mobile), back-end services, and third-party integrations to automate business processes. Propose, design, and implement business workflows to streamline asset management lifecycles. Maintain strong development practices including technical design and writing clean, modular, well tested and self-sustaining code.

Collaborate with business analysts, program managers, end users and other internal teams to translate business requirements into technical solutions that work at Google scale. Integrate third-party products with Google's internal systems as well as support and upgrade implemented systems.

Required Skill(s)

Web Front-end

Required Language(s)

Python

Show More

Systems Engineer, Site Reliability Engineering

© 2015 Team Programmer Jobs

Figure 7 Company Page

For each job listings within a specific language, company, or location, we used a collapsible panel, or “accordion”, to display all the jobs related to that category. Once a specific job is selected, it would expand that specific job listing while collapsing every other job. This would allow the client to look at the descriptions for individual job without crowding up the entire page.

Design

For our design, we used Twitter Bootstrap components and made our custom CSS script. With the Bootstrap, we used responsive panels, tables, images and more to organize and align the data and assets to make our static pages multi-platform friendly, including mobile.

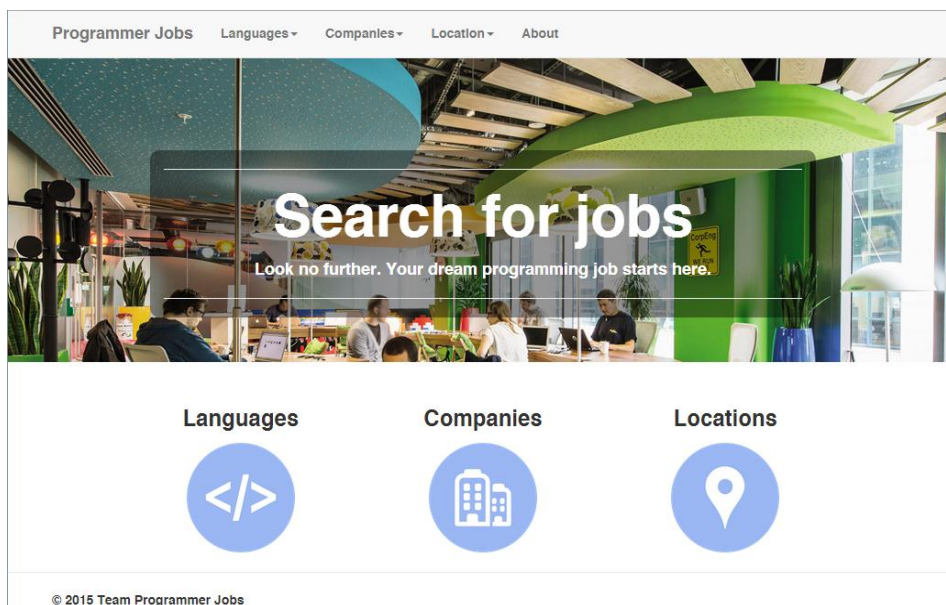


Figure 8 Wide View

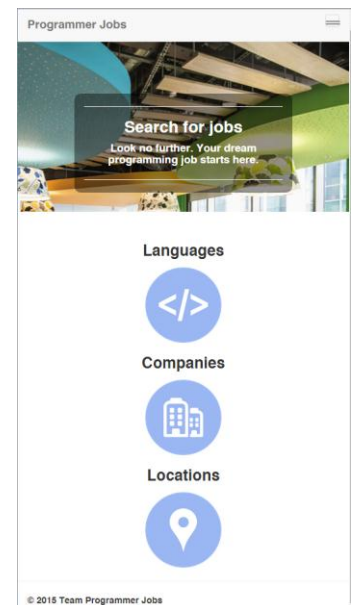


Figure 9 Narrow View

Since two people developed the static pages, we had to match up our styles and designs to avoid incompatibility between pages. Also we sought a way to make each categories have their own style while keeping the design uniformed. To achieve this, we followed each other's CSS style and Bootstrap design while developing. For example, on each specific pages, we placed our images on left

top, used collapsible panels to display jobs and clickable buttons for terms. Furthermore, we applied column feature provided by Bootstrap in our html codes to make our page more clean, well-organized, and responsive.