

CS5200 Database Management System

Project Report (Summer 2014)

CareerMedley

Group members:

Chi Zhang

ID: 001957029 Email: czhang79@ccs.neu.edu

Onyeka Igabari

ID: 001150568 Email: Onyeka.igabari@gmail.com

Introduction

Problem to Solve

Our goal in this project is to build database management system to well-organize our job applications and facilitate our job search. In current job market, we normally apply to multiple positions online in order to get an ideal intern/coop/job. The number of applications could range from tens to several hundreds. A typical naïve way to organize so many applications is: create a folder for each company and create a sub folder for each position in the company, then you put your resume/cv, login id/password, email address to that specific subfolder. It's easy to get lost in many folders. What's worse, after several weeks or months, you'll have no idea what you did before. And it's really embarrassing that sometimes you receive an interview invitation email, but you don't remember when and where you apply it.

The important difference between our job application system and existing job application services provided by indeed/monster is: this is for user-oriented. And the emphasis is put on the job that the user is interested, the effort the user puts on the position, and related info aggregation. E.g. if the user plans to apply the data scientist positions, the system can list the core skills that most data scientist positions require. In terms of info aggregation, for instance, the applicant may search for a job on Monster.com, read reviews about that job on indeed.com and find out more about the company and salaries ranges on glassdoor.com. The applicant may also get more details about the company's stocks and news from Bloomberg or yahoo finance, etc. Won't it be nice to have a one-stop site to have all these info?

Information Required

In order to build this system, we should collect information from several aspects:

1. Job positions info
 - Info about the position itself:
Title, location, job description, dept. or division, required skill sets, sponsorship, clearance, benefit (401k etc.), deadline, application link on the employer's website. (To get the latest job postings, there are two options: extract job positions from indeed/monster and populate our database with them with provided APIs; manually input the job position)
2. The employer info
Category (internet/mobile/software/hardware), main product, size, stock price, ratings, related news (such as hiring or layoff news), reviews and possible interview questions. (To get the info about the employer: retrieve info from Glassdoor, Bloomberg, CrunchBase, etc.)
3. The applicant info
Resumes, CVs, status (interested/applied/rejected/withdrawn), id and password pair for that position, referrer provided by user.

What we want to deliver (ultimately)

- Architecture:

A web-based client/server database management system that be hosted on a public clouding computing platform, such as Amazon AWS, Red Hat OpenShift, Salesforce Heroku.

- Services:

The user can view the history of his/her applications, such as which material has been submitted and which are not (E.g. resume submitted/cover letter missing etc.), what phrase now (E.g. pass phone screen/second round interview/onsite etc.), the incoming deadlines ranked by date.

For a specific position, the user can view its related info, such as: title, location, deadline, and short description of the job, etc.

For a specific employer, the user can view its related info: number of reviews, ratings by its employees, CEO, CEO approval rate, etc. The user can deduce the employer's reputation from the info.

The overall data analysis: what kind of skill set is required in those jobs that the user saves? Which language is most wanted among those jobs? (E.g. the swift programming language recently released by Apple?) How many open positions of those jobs now?

Solving Strategies

Project implementation can be broken down into several stages:

1. **Framework Investigation:**

In this stage, we start by analyzing different web technologies such as Ruby on Rails, Spring+Hibernate (Java), Django (Python), Flask (Python) etc. to figure out which would be advanced and fun to use nowadays.

2. **API Investigation:**

In this stage, we would investigate the API options we have from different job search engine sites and the possibility of obtaining data from these sites using their web services.

3. **Prototype Development:**

- In this stage, we would design the database that is needed for our application and write small applications to test the API functionality.
- Try to figure out how much data we should store in our database and how much data we should get and display dynamically.
- Develop a little test application for proof of concept. We would be using Northeastern Husky career service's website as a form of template for our development.

4. **Detail Implementation and Test Cycles:**

In this phase we would be continuing the design process towards a full product and fixing issues as we go. We will be using agile process of development to help keep us on track.

Database Design

E-R Diagram

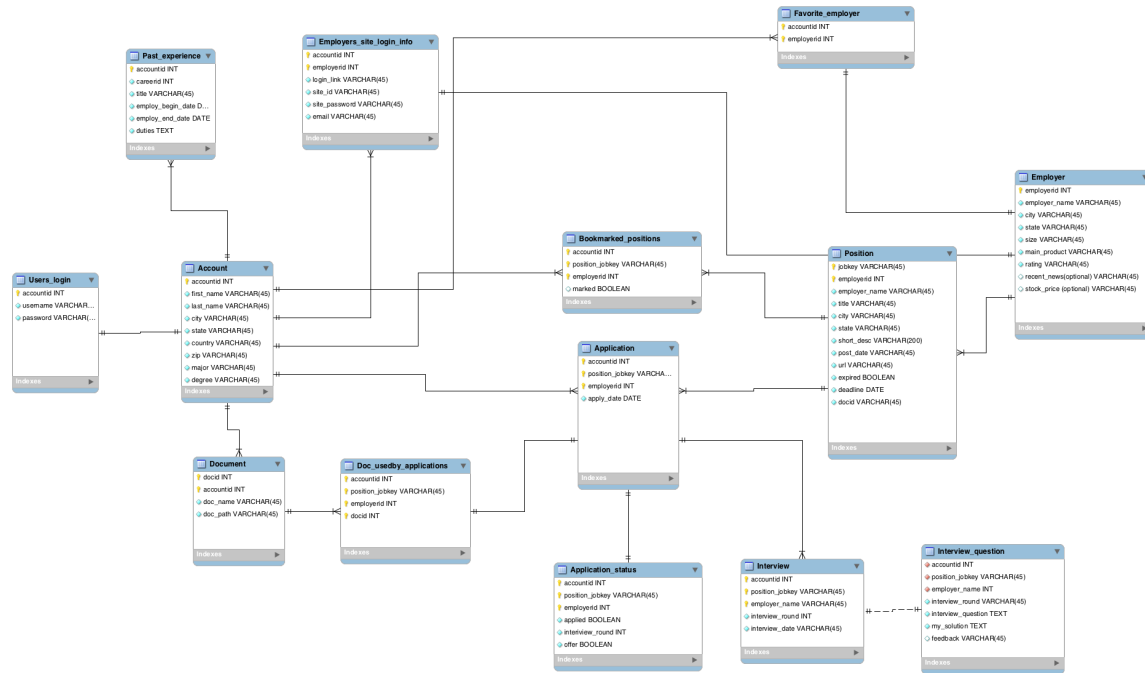


Fig. E-R Diagram of our project

The figure above shows the diagram of our project. Two figures below show the zoom-in left and right side of the same diagram.

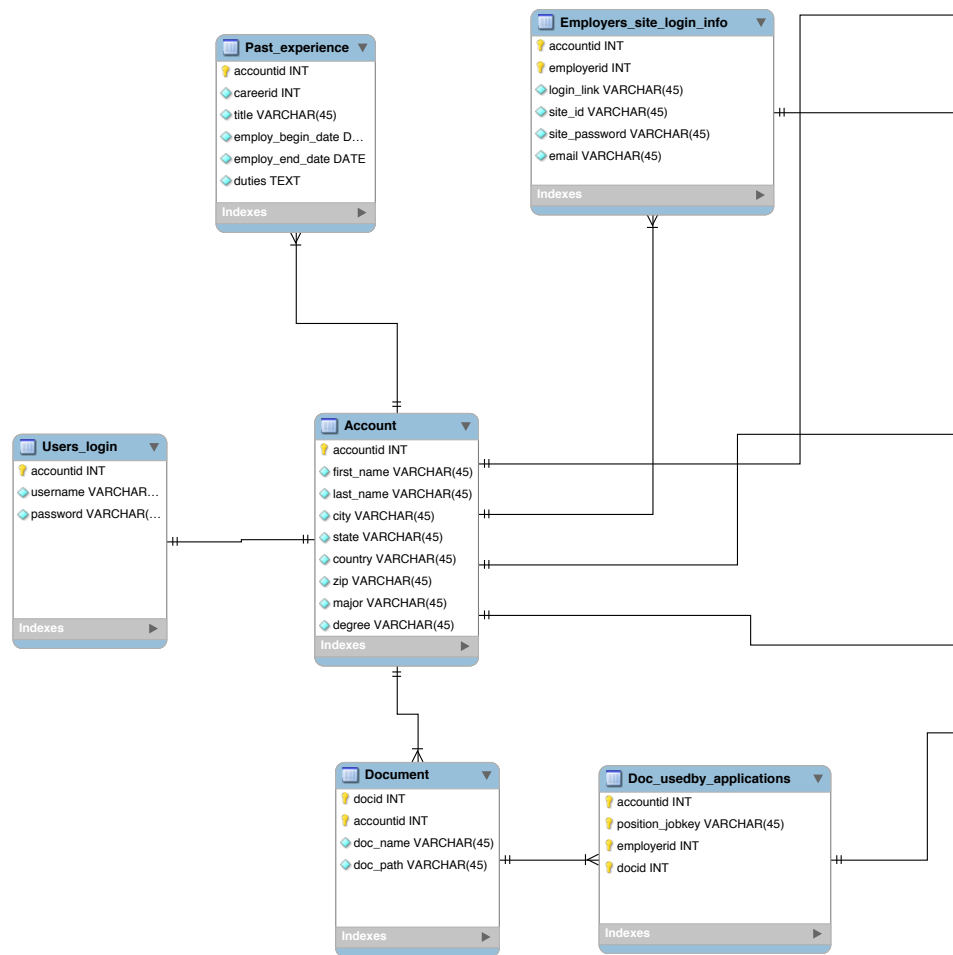


Fig. Left side of E-R Diagram

Tables on figure above:

- Users_login: store the user's login info.
- Account: store the user's account related info
- Past_experience: store the user's past work experience, like when to when, work for whom
- Employers_site_login_info: store the user's login info on the employers' sites
- Document: store the file names and paths of user's document
- Doc_usedby_applications: store which application uses which documents.

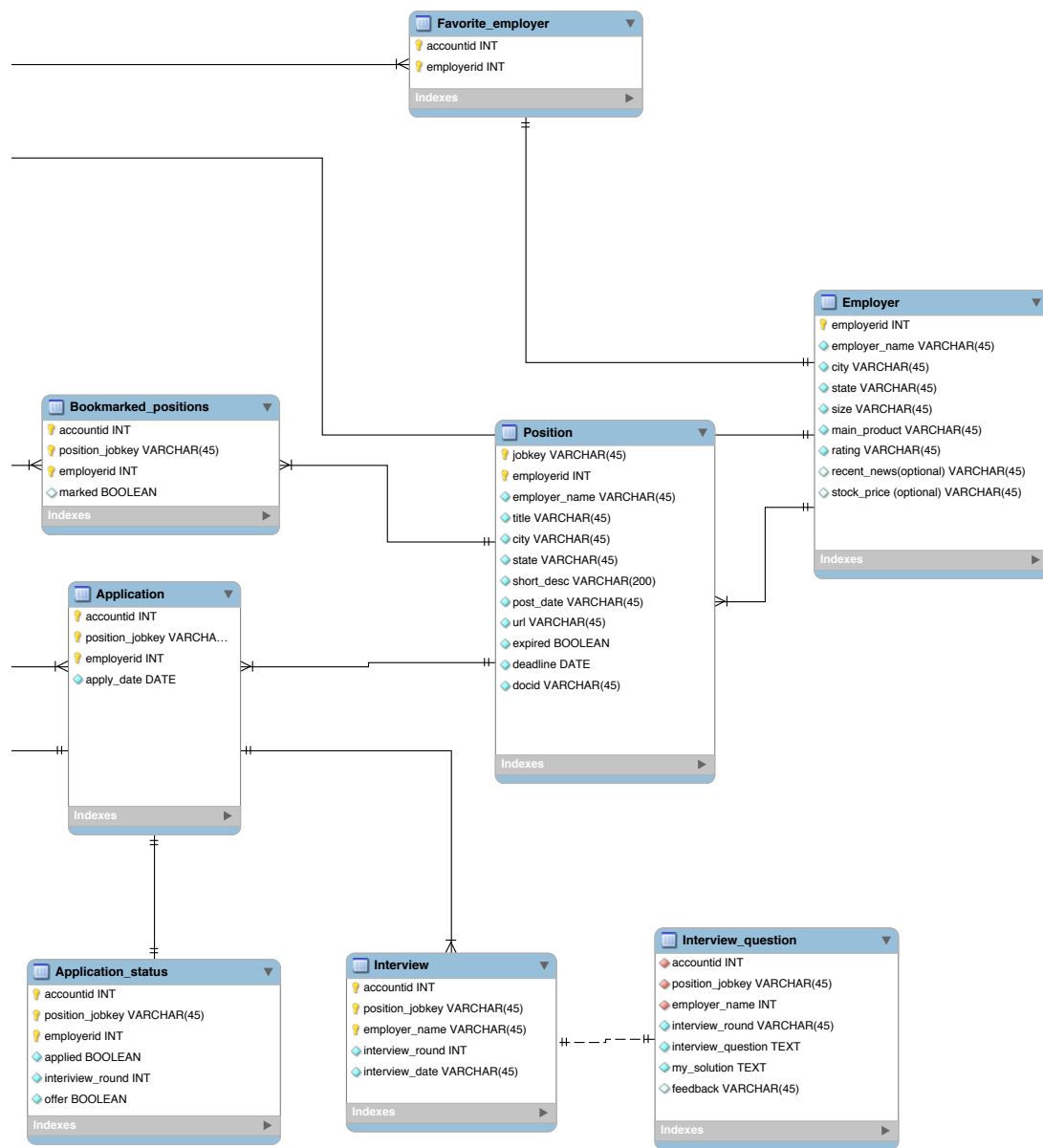


Fig. Right side of E-R Diagram

Tables on figure above:

- Position: store the related information about a position
- Bookmarked_position: indicate which position is bookmarked by user
- Application: store's application information
- Application_status: track the status of each application
- Interview: store interviews information for an application

- Interview_question: store the specific questions and solutions of an interview
- Employer: store the employer's information

Implementation

In our implement, we picked Flask and SQLAlchemy (ORM). It's a typical MVC application.

Architecture

```
~/careermedley
|-- run.py                # bring the application up
|-- config.py            # configuration
|-- db_create.py         # database creation
|-- /env                 # virtual environment for python
|-- /application         # application module
    |-- __init__.py      # general application setup
    |-- /static          # store statics files: images, js and css
        |-- /css
        |-- /img
        |-- /js
    |-- /templates       # holds jinja2 templates
        |-- base.html
        |-- login.html
        |-- user.html
        |-- main_page.html
        |-- ...
    |-- forms.py         # input data mappers and validators
    |-- models.py        # classes that be mapped to tables
    |-- view.py          # router and controller
```


Package dependencies:

Flask 0.10.1

Flask-Bootstrap3 3.1.1.3

Flask-Login 0.2.11

Flask-Mail 0.9.0

Flask-OpenID 1.2.1

Flask-WTF 0.10.0

Flask-SQLAlchemy 1.0

Flask-WhooshAlchemy 0.56 // database index building

Jinja2 2.7.3 // template rendering

SQLAlchemy 0.9.7 // database ORM

WTForms 2.0.1

Werkzeug 0.9.6 // routing

Technology stack

Web development framework- Flask (Python)

[Flask](#) is an open source micro web development framework for Python based on Werkzeug, Jinja2 and other components. Flask has built-in development server and debugger, integrated unit testing support, it's 100% WSGI1.0 compliant (Web Server Gateway Interface) and Unicode based. The **micro** means Flask aims to keep the core simple but extensible. Flask won't make many decisions for you, such as what database to use. Those decisions that it does make, such as what template engine to use, are easy to change. Everything else is up to you. By default, Flask doesn't include a database abstraction layer, form validation or anything else where different libraries already exist that can handle that. Instead, Flask supports extensions to add such functionality to your application as if it was implemented in Flask itself. Numerous extensions provide database integration, form validation, uploading handling, various open authentication technologies, and more. Flask may be "micro", but it's ready for production use on a variety of needs.

In terms of Flask's major components, Werkzeug is a Python WSGI Utility Library. Jinja2 is a modern and designer friendly template language for Python, modeled after Django (another popular Python web development framework)'s template. It's fast, widely used and secure with optional sandboxed template execution environment.

Further info about Flask can be found on it's official website:

flask.pocoo.org

Data access layer: SQLAlchemy (Python database toolkit)

In our project, [SQLAlchemy](#) is chosen as the database abstraction layer, which is the bridge that Flask uses to talk to backend database.

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. It provides a full suite of well-known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

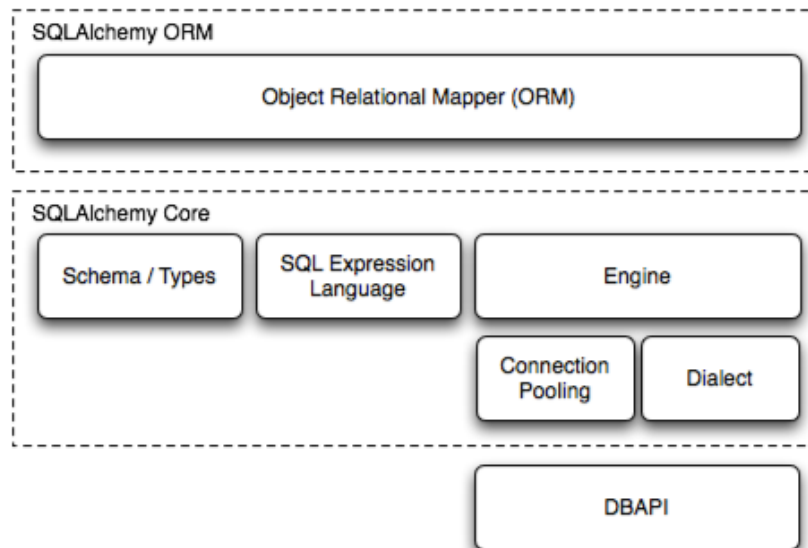


Fig 1. SQLAlchemy internal structure

One tip in installation: after SQLAlchemy is installed, Flask wrapper

extension Flask-SQLAlchemy can be installed to provide a friendlier programming API for developers.

Use cases (which reflects CRUD operations)

CareerMedley use case analysis

- List all open positions before / after login
- Update open positions
- List all employers
- Update employers
- List/Upload/Delete resumes or Cover Letter
- Register on employer's official website
- List all my job applications
- Create views for job applications
- View a specific job application view
- Apply a specific position
- Update application status

Details of each use case:

- Use Case: applicant lists all open positions before

Actors: applicant, IDE

Description: when an applicant logs in, he/she can see a list of open positions. If he/she logs in, he/she can see the SAVE/APPLY buttons attached to positions.

Steps:

1. Applicant types query and location and press SEARCH button,
2. Our program invokes indeed API, and returns a list of open positions (software engineer, system engineer, etc.): the title of the position, location, employer, etc.

- Use Case: applicant lists all favorite employers

Actors: applicant, IDE

Description: when an applicant logs in, he/she can see a list of all his/her favorite employers.

Precondition: the applicant has logged in

Steps:

1. IDE queries tables: Account, Employer and Favorite_employer;
2. IDE shows a list of all applicant favorite employers (Google, Facebook, etc.), their score, CEO name, number of ratings, etc.

- Use Case: applicant updates employers

Actors: applicant, IDE

Description: applicant can manually input other employers to table: employers

Steps:

1. input in related info, title, location, employer name, ratings manually.

- Use case: List/Upload/Delete resumes

Actors: applicant, IDE

Description: the applicant can list/upload/delete resumes

Precondition: the applicant has logged in

Steps:

1. IDE queries tables: accounts and resumes tables to get available resumes version for current applicant

For listing resumes: IDE returns the applicant's all resume versions

For uploading resume: upload a new resume to database, insert new entry in table: resumes to point to the new resume

For deleting resume: IDE displays list of resumes, applicant picks up one to

delete. IDE asks for delete confirmation, the applicant acknowledges the deletion, the IDE displays new list of resumes

- Use case: List/Upload/Delete cover letter

Similar case as above

- Use case: applicant registers on employer's official website

Actors: applicant, IDE

Description: the applicant registers on employer's official website then populates the table of our app: `employer_site_login_infos`

Precondition: the applicant has logged in

Steps:

1. The applicant visits the official website of the employer
2. The applicant chooses a combination of email, id, password to register on employer's website
3. The applicant also input the combination of email, id, password to the table: `employer_site_login_infos` (Since different websites require different id/password naming rules, if we don't store those combinations carefully, sometimes we forget the combination, and then we couldn't login the employer's official site again, always need to resort to "Forgot your username/password")

- Use case: applicant lists all his/her applications

Actors: applicant, IDE

Description: the applicant can view all his/her application related info and status

Precondition: the applicant has logged in

Steps:

1. IDE queries tables: Account and Position to get job applications for an applicant

2. IDE displays basic info of those job applications: the position, location, employer, etc.

- Use case: create view for job applications

Actors: applicant, IDE

Description: applicant creates view for job applications

Precondition: applicant has logged in

Steps:

1. applicant requests to create view for job applications
2. IDE queries the tables: accounts, positions and employers to create the view for job application with related info

- Use case: view a specific job application detail

Actors: applicant, IDE

Description: The applicant views job application detail

Precondition: applicant has logged in

Steps:

1. IDE displays view for job application with related info.

- Use case: the applicant applies a specific position

Actors: applicant, IDE

Description: applicant applies an open position

Precondition: applicant has logged in

Steps:

1. The applicant finds an interesting position in the list of positions and click the APPLY button. The click redirects it to the application page for that position.
2. The applicant fills in the fields, upload resumes and CVs, as required on the official job application link

2. The applicant fills in related info, such as date, which version of resume, which version of CV to local DB tables: applications

- Use case: update application status

Actors: applicant, IDE

Description: applicant can change the application status

Precondition: applicant has logged in

Steps:

1. IDE queries tables: Accounts and Application to find a specific application for a specific applicant
2. The applicant marks the job as applied, interviewed, offer etc.
3. If the applicant has been interviewed, he/she input the interview questions into table: Interview_question. In this way, the applicant can learn from the past interviews and improve him/herself.

- Use case: view deadlines

Actors: applicant, IDE

Description: applicant can check whether the incoming deadlines of positions

Precondition: applicant has logged in

Steps:

1. IDE queries the table: deadlines to display the incoming deadlines of positions (a month after now)

(Note: This is not a complete list of use cases or final use cases. Use cases will be added or deleted depending on further function discussion and the progress of development.)

Demo

Main page

Home Sign In

CareerMedley
helping you build your career...





Search

Login page (with OpenID technology)

Home Sign In

CareerMedley
helping you build your career...

Please enter your OpenID, or select one of the providers below:


   

☐ Remember me


Login

Profile page

Home
Profile
Documents
Applications
Saved Positions
Employers
Hi, onyeka! Sign Out



List of saved positions (you can delete or apply the positions)



Home
Profile
Documents
Applications
Saved Positions
Employers
Hi, onyeka.igabari! Sign Out

Software Developer
Jserver - Watertown, MA

Software Developer - Watertown, MA Are you looking for an opportunity to use your software development skills to make a difference? athenahealth is transforming healthcare by creating easy to use, ultra powerful, cloud based tools which allow medical professionals to focus on what they do best - treat patients. Our goal is to bring order to the chaos of the U.S. healthcare system by harnessing the...

Delete
Apply

Physician - Urgent Care
American Family Care- Doctors Express Urgent Care - Marlborough, MA

Full Time & Part Time Urgent Care physicians needed in Marlborough, MA We are currently looking for a Full & Part-Time Physicians in our Marlborough, MA Location.. Come Grow with Us!! Expenses are up and reimbursements are down!! These are two trends forecasted to continue!! In our practice we handle those concerns and you practice medicine!! How can Doctors Express change your life...

Delete
Apply


Physician - Emergency Medicine
American Family Care- Doctors Express Urgent Care - Burlington, MA

Full-Time Urgent Care Physician in Waltham & Burlington, MA We are currently looking for a Full-Time Physician in our Burlington & Waltham, MA Locations.. Come Grow with Us!! Expenses are up and reimbursements are down!! These are two trends forecasted to continue!! In our practice we handle those concerns and you practice medicine!! How can Doctors Express change your life? •Concentration on...

Delete
Apply

Apply a specific position

(position detail shown above, the form shown below)



helping you build your career...

[Home](#)
[Profile](#)
[Documents](#)
[Applications](#)
[Saved Positions](#)
[Employers](#)

Hi, onyeka.igabari! [Sign Out](#)

[Software Developer](#)
 Jserver - Watertown, MA

Software Developer - Watertown, MA Are you looking for an opportunity to use your software development skills to make a difference? athenahealth is transforming healthcare by creating easy to use, ultra powerful, cloud based tools which allow medical professionals to focus on what they do best - treat patients. Our goal is to bring order to the chaos of the U.S. healthcare system by harnessing the...

Apply Date


Resume

CV

Username

Password

Apply a specific position (popup date plug-in)



helping you build your career...

[Home](#)
[Applications](#)
[Saved Positions](#)
[Employers](#)

Hi, onyeka.igabari! [Sign Out](#)

[Software Developer](#)
 Jserver - Watertown, MA

Software Developer - Watertown, MA Are you looking for an opportunity to use your software development skills to make a difference? athenahealth is transforming healthcare by creating easy to use, ultra powerful, cloud based tools which allow medical professionals to focus on what they do best - treat patients. Our goal is to bring order to the chaos of the U.S. healthcare system by harnessing the...

27 28 29 30 31 1 2
 3 4 5 6 7 8 9
 10 11 12 13 14 15 16
 17 18 19 20 21 22 23
 24 25 26 27 28 29 30
 31 1 2 3 4 5 6


Resume

CV

Username

Password

Apply a specific position (dropdowns for resume and CV, so that applicant can pick up resume and CV directly instead of inputting themselves.)



[Home](#)
[Profile](#)
[Documents](#)
[Applications](#)
[Saved Positions](#)
[Employers](#)

Hi, onyeka.igabari! [Sign Out](#)

Software Developer
Jserver - Watertown, MA

Software Developer - Watertown, MA Are you looking for an opportunity to use your software development skills to make a difference? athenahealth is transforming healthcare by creating easy to use, ultra powerful, cloud based tools which allow medical professionals to focus on what they do best - treat patients. Our goal is to bring order to the chaos of the U.S. healthcare system by harnessing the...

Apply Date

Date

Resume

Documents

CV

Documents

Username

1sam.doc


Username

Password

Password

Submit

Favorite employers (index is created for employers, and we can use the index to search jobs with a keyword. Pagination is applied to the list of employers as well.)



[Home](#)
[Profile](#)
[Documents](#)
[Applications](#)
[Saved Positions](#)
[Employers](#)

Hi, Christopher! [Sign Out](#)

Rank by Score
Reorder

Employer	Num_ratings	Score	CEO	CEO_reviews	CEO_approval
Google	2439	4.4	Larry Page	1281	96
Facebook	725	4.3	Mark Zuckerberg	628	95
amazon.com	3514	None	Jeff Bezos	2445	80
Pivotal	37	None	Paul Maritz	28	93
Intuit	1603	None	Brad Smith	683	93
salesforce.com	1132	None	Marc Benioff	899	95
Qualcomm	1578	None	Steve Mollenkopf	130	96
EMC	1893	None	Joe Tucci	1255	91
Microsoft	7870	None	Satya Nadella	748	87

[<< Prev employers](#)
[Next employers >>](#)

(note: these snapshots don't reflect all the pages and functions in our application)

Statistics

Number of

Tables: 14

fields: 81

foreign keys: 27

mapping tables: 14

classes: 19+

CRUD (create/read/update/delete) operations: 35

interfaces: 10+

pages: 12+

- /main.html
- /login.html
- /logout.html
- /profile.html
- /user.html
- /document.html
- /applications.html
- /user/userid/jobkey/apply
- /user/userid/jobkey/applydetail
- /user/saved_positions
- /user/employer_search.html
- /user/favorite_employers.html

Web services

Indeed

<https://ads.indeed.com/jobroll/xmlfeed>

- Case 1: dynamically retrieve job openings by submitting a query to indeed;
- Case 2: dynamically retrieve the detail of a specific job by submitting a jobkey.

Glassdoor

<http://www.glassdoor.com/api/index.htm>

- Case 1: dynamically retrieve the detail of a specific employer by submitting the employer name.

USAToday

<http://developer.usatoday.com/docs/read/articles>

- Case 1: dynamically retrieve latest tech news by sending requests to USAToday Articles API.

Architecture

Flask web development framework. It's a MVC web framework, which is similar to Django or Ruby on Rails, but it's more lightweight. Developer can have more freedom to add or remove components.

ORM

SQLAlchemy, which is a Python database toolkit and can be chained with Flask.

Summary

Key lessons - anything that surprised, delighted or concerned you.

- There are a lot of available web development frameworks out there. I never imagined there are so many frameworks there before I took our database course. LAMP, MEAN, Hibernate/Spring, .NET, Django, Ruby on Rails, Flask. I was dazzled and confused by so many terminologies initially. Then the side effect is: when we began our project, we had to investigate different ones and picked up the one that's most appropriate in terms of functionalities and learning curve, which causes a significant time overhead.

But then you found that most of them share the same philosophies, like MVC, but in a different way.

- There are so many pieces of small knowledge you need to know to really develop some web based database application. Html, CSS, ajax, REST, HTTP requests, cloud hosting, SSH, security, front-end scripting JavaScript, back-end scripting PHP... you can name it. It's OK that you are not familiar some of them, but you really need to learn about the big picture of the technology stack, their application background and where you can apply them: displaying dynamic content? Retrieving records from database, etc. Most of the time, you only need to know how to invoke the APIs. But if you want to be highly capable web / database developer, you really need to attack the above items one by one and then you can become a so-called full-stack developer.
-
- For Team projects:
 - Your specific contributions to the project
 - Comments and Feedback for your team members.

References:

indeed: <http://www.indeed.com/>

monster: <http://www.monster.com/>

Simplyhired: <http://www.simplyhired.com/>

Glassdoor: <http://www.glassdoor.com/index.htm>

Ruby on Rails: <http://rubyonrails.org/>

Python Flask web framework: <http://flask.pocoo.org/>