GETJOB

SKILL/JOB RECOMMENDER WEB APPLICATION

Developed By : Kantamsetti Mohan Durga Sathish

Walk Through video : getjobWalkThrough.

Source code @ : Github SmartInternz02

Hosted @ : getjob.apps.pcfdev.in

INTRODUCTION

1.1 Overview

Having lots of skills but wondering which job will best suit you? Don’t need to worry! we have come up with a skill recommender solution through which a fresher or a skilled person can login and find jobs by using the search option or they can directly interact with the chatbot and get their dream job.

1.2 Organisation

This web application is developed as part of a Build-a-thon conducted by smart Internz with VMware Tanzu. This web application is built as a project.

1.3 Target Market

The target market for this web application are job seekers, undergraduates and working professional who are looking for a job.

1.4 Purpose

To develop an end to end web application capable of displaying the current job openings based on the skillset of the users. The user’s credentials and their information are stored in the Database. Whenever there is an job opening based on the user's skillset, it is recommended to the user. The user will interact with the chatbot and can get recommendations based on their skills. Users can use the job search API of the web application to get the current job openings in the market which will fetch the data directly from the webpage.

2. System Analysis

2.1 Product Perspective

GetJob webapp is a skill job recommender app which

- Takes details of the user that include skills and other personal data and recommends jobs to the users.
- The application has three interfaces to interact with, that include a graphical web application, a chat bot and through a REST api.
- Apart from job recommendations the webapp can also search for job openings in the market by various categories such as skill, role, location and organization.
- The application also provides a job gallery with job roles and job organizations.
- The application provides learning resources to the user, to improve their skills.

2.2 Product Functions

The main function of the web application is to provide job recommendations to the user, from all the three interfaces i.e., through REST api, chatbot and the graphical web app based on user skills.

2.3 Phases

Since the web application has multiple interfaces it is proposed that it would be developed in phases. The web application has three interfaces for the user or the job seeker, even though it was not thought of initially an employer interface was also added. Where in an employer can login and check job postings by their organization and also can accept or reject a job application. The employer can also change the status of a job posting.

- Phase 1 : Graphical User Interface
    - User Registration
    - Login and profile pages
    - Dashboard, Landing page, Search pages.

- Phase 2 : Chatbot

    - Chatbot’s responses to job recommendations
    - Chatbot’s responses to help user navigate and find

- Phase 3 : REST api
    - To search jobs using the api based on various parameters
    - To check user’s status of job applications

- Phase 4 : Employer portal
    - Employer’s login
    - Employer’s Dashboard
    - Employer ability to accept, reject or close job openings.

- Phase 5 : Documentation
    - API Documentation for the user to understand and use the api
    - Website documentation for the user
    - Website documentation for the employer

2.4 User Characteristics

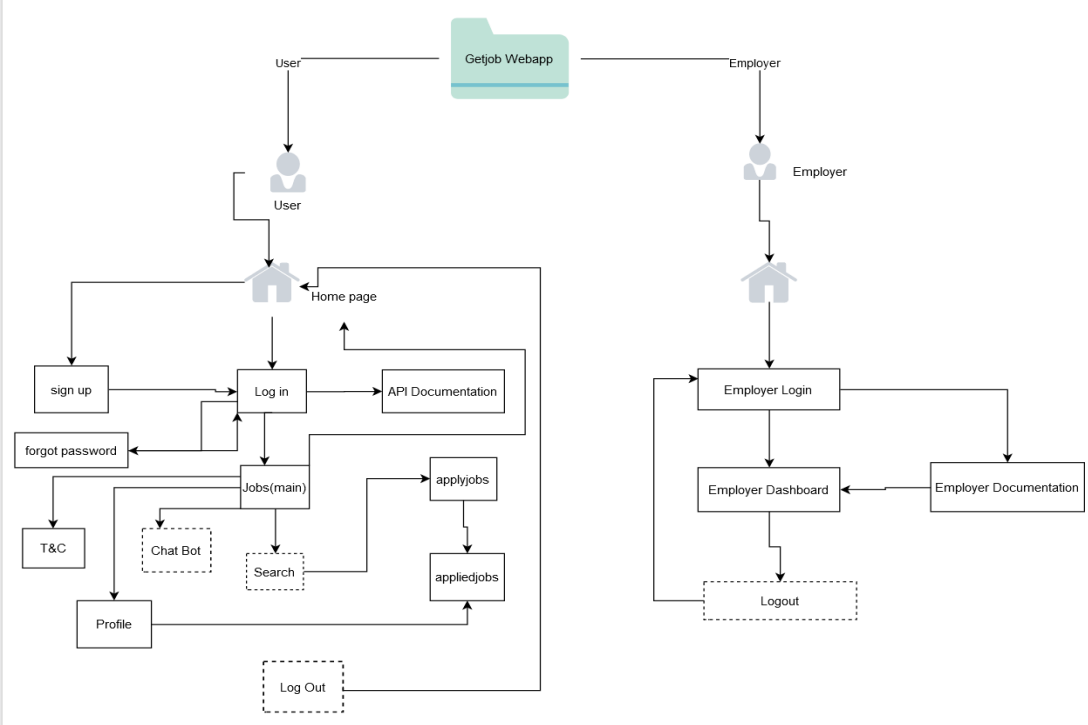
The users of the web application are expected to be able to use a browser to be able to use the web application. The user can refer the documentation provided for the api to use the api, user might not require any special training to be able to use the chatbot.

2.5 Operating Environment

The web application is a python flask based application, that uses Jinja web template engine. The web application is bundled and containerized using docker and then is hosted on vmware tanzu service.
The web application uses remotemysql database to store and retrieve data of the user and that of the job openings and applications. The database is remote and is hosted at remotemysql.com
.
Technologies : HTML, CSS, Bootstrap, MySQL, Python-Flask, Cloud Foundry, Tanzu Application Service, REST API's.

2.6 Design and Implementation Constraints

Site map



Implementation

This webapp uses [IBM Watson Assistant]for the chatbot of the website.

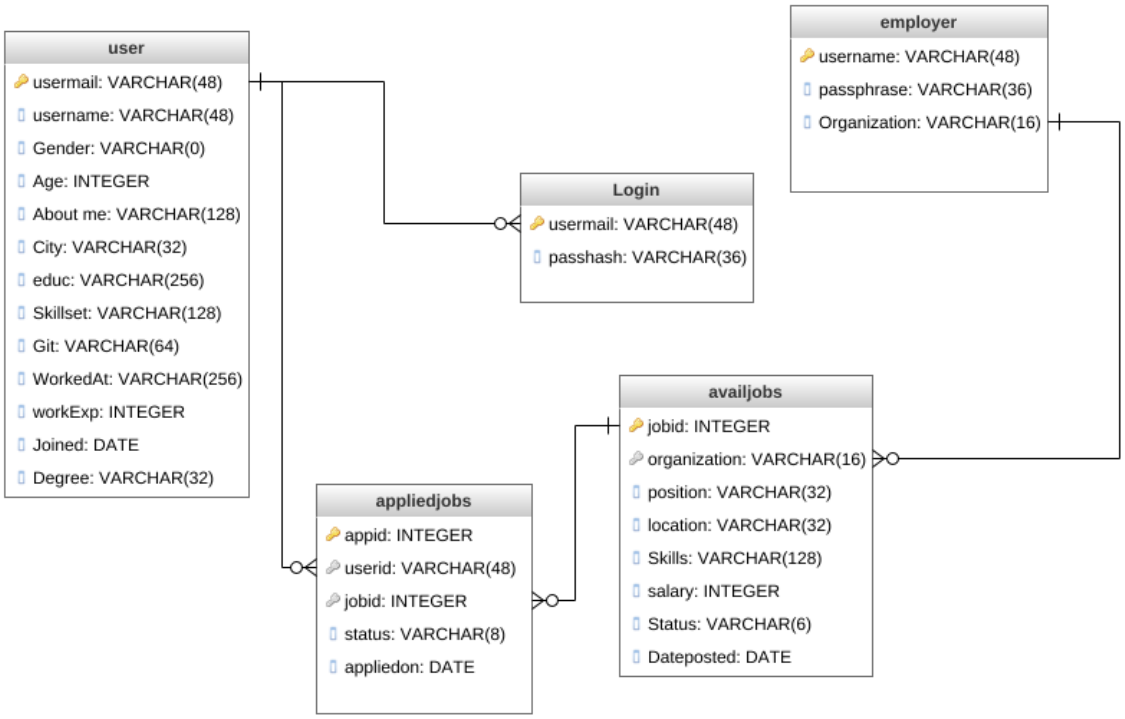
Exception handling is put in place in the backend of the web application, which prevents the web app from displaying error messages to the user and crashing. Whenever an exception is generated, code is written to safely handle the exception and to inform the administrator through logs and the user through alert messages.

Database Constraints

- **Entity Integrity** :
  All rows in the tables of the database have unique identifiers called Primary Keys.
- **Domain Integrity** :
  All data stored in a column must follow the same format and definition. The domains of the data are well defined and verified using js before inserting into the db.

- **Referential Integrity** :
  Foreign key constraints are put in place to keep table relationships consistent. Therefore, a user cannot delete a record which is related to another one.
- **User-defined Integrity** :
  All required data and fields from the user are well defined and handled. Various Ids are used across the website.

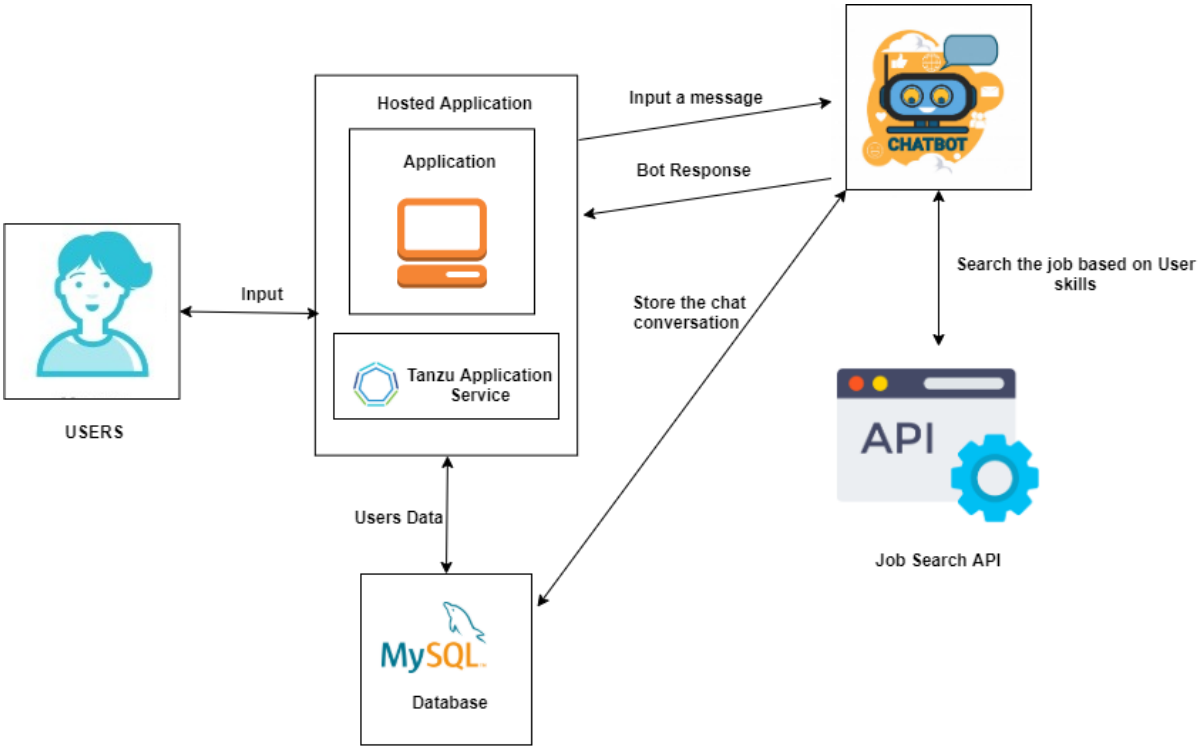
Database Schema



Input data Constraints (implemented with js)

- All input data is validated through the use of javascript functions and to minimize the load of server performing the validations.
- Html tag attributes of max length, min value and max value are also put in place to prevent data imputation.

2.6 Proposed Architecture



The web application uses mysql database to store and retrieve user data, and data of job postings. The user interacts with the web application directly using the interface or using the chatbot or by using the api. The chatbot also makes use of the api to interact with the website and answer user's questions.

3. Software and Hardware Requirements

- Operating system : Windows / linux with Docker installed.
- Database : RemoteMySQL database
- Hosting: VMware Tanzu Application Service
- A browser through which the web application could be accessed.
- Interactive Support : Python Flask , jinja2 and JavaScript are used for interactive support.
- Cloud Foundry

This web application is developed as a modern web application which uses microservices, containers and Kubernetes help to free apps from infrastructure.This web application can work independently and run anywhere. With VMware Tanzu, the delivery of containerized workloads is automated, the code of the application is pushed to github. Whenever a commit is made to master of git repository. A trigger is generated and a new image of the web application is built on the infrastructure of docker hub. Restarting the hosted web application on vmware tanzu will build the latest image of the webapp and hosts it.

To be installed on a local machine, the local machine should have docker installed.

4. Functional Requirements

The user should be able to

- sign up by providing the necessary Details.
- login to webapp with their credentials.
- Get recommendations for their skills.
- The user should be notified with email alerts.
- Should be able to search for jobs based on skill, role or location.
- Should be able to chat with the chatbot and get
    - Job recommendations
    - Navigation Help
- Should be able to use the api
    - To search for jobs
    - To get status of applied jobs
    - To apply for a job.

5. Non Functional Requirements

5.1 PERFORMANCE REQUIREMENTS

- **RESPONSE TIME**:
    - This web application loads within 1-2 seconds, when first loaded and loads within a second when refreshed.
    - Job recommendations are loaded as soon as the user logs in.
    - The results of the search querry are fast due to optimized queries.
    - The response of the api is fast.
    - The chatbot doesn’t freeze or stop responding.

- **OPTIMIZATION** :
    - To decrease the loading time of the web application all the images were compressed.
    - Optimization of JavaScripts was done very keenly.

5.2 SECURITY REQUIREMENTS

- **Authentication** :

  The user can only access the features of this web application after successful authentication through logging in.

  The user is not allowed to create their own password, instead they are given a secure 10 character alphanumeric(also symbols) password, which is communicated to the user by an email.

  The password of the user is not stored as plain text in the database, instead md5 hash of the password is calculated and stored in the database. Whenever user enters their password it’s md5 hash is calculated and compared with the db.
- **Encryption** :

  Apart from the encryption provided by mysql db, the access to the database is protected by a secure password, and the data is encrypted.

5.3 SOFTWARE QUALITY ATTRIBUTES

- **Ease of Use :**
    - The interface of the website is fairly easy to understand and documentation is also provided.
    - The user will need some sort of understanding of the api to be able to use the api.
    - Interacting with the chatbot is fairly simple and doesn't differ from talking to a humans through messaging.

- **Flexibility:**
    - Containerizing a web application makes it very flexible as one has to no longer worry about the prerequisites to run this application on their machine.
    - All that is required is having Docker installed to run the containerized applications.
- **Correctness :**
    - Even though the data is used in the website is for representational purpose. The web app accurately displays data from the db without fail or error.
- **Responsive design**
    - This web application is made to work on a wide range of devices and screen sizes. Bootstrap classes are used in every element to make it fit for all screen sizes. Some sections also use, different elements for smaller and larger screen sizes.
- **Accessibility**
    - To support screen readers and people with impairities color coding, hue of the website is adjusted, and descriptions of the elements items, images is given which is read by screen readers and other assistive devices.

6. Future Scope

This web application has a lot of scope to be improved.

- The API interface can be enriched with parameters to perform multi parameter search and display more personalized information of the user.
- The chatbot can have more intents and dialogues and can be made more human like.
- The graphical interface can have more learning resources, notifications within the webapp, chat features, and even a feed post feature where job seekers can socialize.
- The employer interface can be improvised with features such as ranking the applicants, graphical job posting, Customised dashboards with stats, direct communication with the user.

7. Bibliography and References

- Set Up flask environment
- Cloud Foundry Installation
- Python flask
- REST API
- flask-mysql
- Docker-Containerize
- StackOverflow-flask
- Python-general