Job Search System - Application of Intelligent Agents

V.Jayavardhan[1], S.Praneetha[2], P.Rishitha[3], P.Pavitra[4], N.Sunny[5]

[1,2,3,4]Dept. of Computer Science and Engineering, VR Siddhartha Engineering College, Vijayawada
[5]Assistant Professor, Dept. of Computer Science and Engineering, VR Siddhartha Engineering College, Vijayawada

Abstract - *Finding jobs that best suits the interests and skill set is quite a challenging task for the job seekers. The difficulties arise from not having proper knowledge on the organisation's objective, their work culture and current job openings. Summer jobs are becoming year round side work.Even I'm rolling up my sleeves on the path of researching the best apps that will put to work on tasks, jobs and chores in one's extra time. We set the hours and the amount of time you want to carve out for this side work. An app for finding small paid work in your local area. It is for users who are in need of some quick cash and willing to do small works like repairing a computer, babysitting, mowing a lawn and other similar tasks.*

Key Words: **Native Android Framework, Firebase cloud messaging, Google Maps SDK, MVC Architecture.**

1. INTRODUCTION

In today's global economy, the challenges associated with finding a suitable job is amplified by the technicalities associated with the Job search process which is seen by experience. Normally when we want to apply for a job, we search the newspapers; listen to radio and television broadcasts that may advertise vacancies and also job seekers register themselves with job site portals such as Academickeys.com, Monster.com, and Careerbuilder.com and so on. In general, employers do not register themselves with these mediums to provide full details of the job specifications but instead post important details on their own website only. Also with the growing number of online job search engines, making it almost impossible for job seekers to get an overview of all relevant positions [0]. Therefore we do not always get to know all the vacancies, the nature and status of the employer to decide if this is the sort of job that is being sought for. Also at times we get flattered by the job providers profile but don't get information about the rating of the company by the existing or past employee in terms of salary and so. Taking all these into consideration we propose to develop an intelligent agent (instead of a human agent) to perform the same search operations by interacting with the employer and job search coordinator agents. We propose to use an agent based utility concept to provide suitability profiling based on configurable factors such as distance from work, days and shift requirements, work environment, safety and hazard considerations, remuneration, skill-set, etc.

The purpose of developing an Online Job Search System comes from my idea to make the job search efficient and handy. It helps the job seekers to search for current vacancies at a single point. Therefore, we can say that Online Job Search Portal act as a bridge of communication between job providers and job seekers. With the evolution of technology and internet being the main source of information for the Job Seekers, these job systems and have become an excellent method to reach wide range of audience. Initially, when I am unaware of these job portals or systems, I used to do research about companies and their technology stack through their respective websites and apply if the job responsibilities matches my interests. This requires lots of effort and time. However, later when I realised the importance of job search systems, I am able to access jobs in preferred place, locations that I might not otherwise have learned.

Information: Nagavarappadu is a village in Krishna district of the Indian state of Andhra Pradesh. It is located in Unguturu mandal of Nuzvidu revenue division. It is a part of Andhra Pradesh Capital Region. It is the area where it is full of fields lakes etc. The village Nagavarappadu is populated with total of 1007 people of which some will be performing agriculture works for daily living and others who are job less.

Purpose of visit: On 14th May 2019, we team members along with our guide have visited the village named Nagavarappadu. This village is located around the outskirts of Vijayawada. This visit was undertaken with he aim of investigating the technical problems, unemployment and lack of facilities prevailing in rural areas. In this regard we have met children, school going children, farmers, school principal, people having some jobs, house hold women etc.

Problem statement: Our main aim is to provide an easy going application for people who are in search for small paid work in our local area. It is for users who are in need of some quick cash and willing to do small works.

Objective: To understand the problems and struggle faced by the rural people in their daily life and try to relate the solution to their problems by applying the basic understanding of our engineering knowledge.

2. LITERATURE SURVEY

In this paper they introduced a prototype job portal which uses semantically annotated job offers and applicants. In their opinion, using Semantic Web technologies substantially increase market transparency, lower transaction costs and speed up the procurement process. How-ever adding semantics is not a panacea for everything. We identify some outstanding problems in job search using the system and outline how the technique of query approximation can be the basis for a solution. Through an Industry-Research co-operation we are extending the prototype with these semantic techniques to demonstrate a more accurate job search. [0]

The advent of software agents gave rise too much discussion of just what such an agent is, and of how they differ from programs in general. Here we propose a formal definition of an autonomous agent which clearly distinguishes a software agent from just any program. We also offer the beginnings of a natural kinds taxonomy of autonomous agents, and discuss possibilities for further classification. Finally, we discuss subagents and multi-agent systems. [1]

Intelligent agents are a new paradigm for developing software applications. More than this, agent-based computing has been hailed as 'the next significant breakthrough in software development '(Sargent, 1992), and 'the new revolution in software '(Ovum, 1994). Currently, agents are the focus of intense interest on the part of many subfields of computer science and artificial intelligence. Agents are being used in an increasingly wide variety of applications, ranging from comparatively small systems such as email filters to large, open, complex, mission critical systems such as air traffic control. At first sight, it may appear that such extremely different types of system can have little in common. And yet this is not the case: in both, the key abstraction used is that of an agent our aim in this article is to help the reader to understand why agent technology is seen as a fundamentally important new tool for building such a wide array of systems. More precisely, our aims are five-fold:

- to introduce the reader to the concept of an agent and agent-based systems,
- to help the reader to recognize the domain characteristics that indicate the appropriateness of an agent-based solution,
- to introduce the main application areas in which agent technology has been successfully deployed to date,
- to identify the main obstacles that lie in the way of the agent system developer, and
- to provide a guide to the remainder of this book.[2]

Agents and agent systems are becoming more and more important in the development of a variety of fields such as ubiquitous computing, ambient intelligence, autonomous computing, intelligent systems and intelligent robotics. The need for improvement of our basic knowledge on agents is very essential. We take a systematic approach and present extended classification of artificial agents which can be useful for understanding of what artificial agents are and what they can be in the future. The aim of this classification is to give us insights in what kind of agents can be created and what type of problems demand a specific kind of agents for their solution. [3]

Functional and non functional requirements: Based on the Job Search System application's requirements, I have made the choice of frameworks to be used. This application primarily consists of an android application that displays the jobs that are available and applied by the applicants. This android application is developed by using the Native Android Framework: Used for building an android application, Google Maps SDK: Used for integrating the application with Google maps, Firebase: Used for the mobile clients without any extensive backend programming.

Google Maps SDK: With the Maps SDK for Android, you can add maps based on Google Maps data to your application. The API automatically handles access to Google Maps servers, data downloading, map display, and response to map gestures. You can also use API calls to add markers, polygons, and overlays to a basic map, and to change the user's view of a particular map area. These objects provide additional information for map locations, and allow user interaction with the map. The API allows you to add these graphics to a map:

- Icons anchored to specific positions on the map (Markers).
- Sets of line segments (Polylines).
- Enclosed segments (Polygons).
- Bitmap graphics anchored to specific positions on the map (Ground Overlays).
- Sets of images which are displayed on top of the base map tiles (Tile Overlays).

Firebase SDK: Firebase is an app development platform with tools to develop high-quality apps, grow your user base.

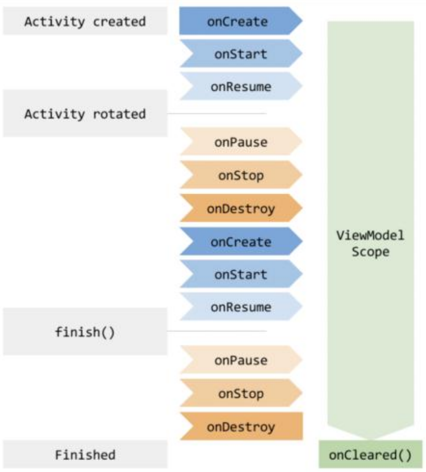


Fig -1: Firebase SDK

3. PROPOSED METHODOLOGY

The existing system for job recruitment includes traditional methods like Employment agencies, advertising through newspapers, televisions and radios, college fairs etc., which are too slow and stressful. With the advancement of internet, jobseekers rely on the online job portals, which makes the job search efficient. Again, most of these are limited to the web/desktop applications, which requires jobseekers to have a laptop or desktop connected to internet and is not handy. And disadvantages include: Time Consuming, Stressful, Challenging.

Proposed System: Job Search System is a Java-based android application that provides functionalities of e-recruitment on portable devices like Android based smart phones/tablets. The applications do not require internet to perform the desired functionalities. Advantages: Cost and Time efficient, Portable.

Purpose of the System: Job Search System is developed to provide an effective means for the employers to post job openings with required qualification to have a better penetration into the job market and jobseekers to find out the information regarding the current openings in the organization or in the market. In addition, job seekers can view the reviews provided by the applicants to make necessary improvements in their system if needed. Job search System is an android application providing flexibility for the users.

Process flow diagram:

System Design: Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements [6]. System designs are modelled using

UML which is a standard object-oriented analysis and design language. The UML is a collection of diagrams and standard set of notations for specifying and visualising various aspects such as requirements and design of software systems.

Use Case Diagram: A diagram is a visualisation of set of elements and the relationships between them. Use case is a set of scenarios, which defines functionalities of the system from a user's perspective. The main components of a use case diagram include actors, use cases and their relationships. They depict the interaction between actors and system to achieve certain goal. This, a use case diagram is important in modelling the behaviour of a system.

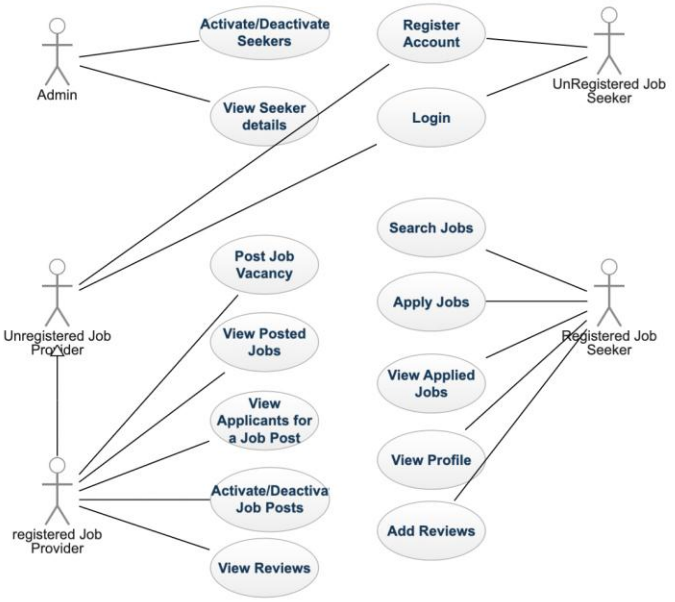


Fig -2: Use Case Diagram

Actors: The actors of the system are Unregistered Employer, Register Employer, Unregistered Jobseeker and Registered Jobseeker.

Use cases:

I have identified a set of use cases based on the functionalities and goals of the application.

- Register Account- This use case denotes a set of actions required for Employer and Job seeker to register with the application.
- Login- This use case denotes a set of actions required for Employer and Job seeker to login into the application.
- Activate/De-activate Account- This use case denotes a set of actions required for admin to activate or deactivate the Employers.

- View Employers- This use case denotes a set of actions required for admin to view the Employers that are registered with the application.

- Add Job Vacancy- This use case denotes a set of actions required for Employer to post a job vacancy.

- Activate/De-activate Job Post- This use case denotes a set of actions required for Employer to change the status of the Job Post.

- View Job seekers for a Job Post- This use case denotes a set of actions required for Employer to view the list of applicants for a particular job post.

- View Reviews- This use case denotes a set of actions required for Employer to view Reviews provided by the applicants.

- ViewJobPosts- This use-case denotes as to factions required for Employer to view all the jobs posted by the Employer.
  Actors of the Admin,

- Search Job Posts - This use case denotes a set of actions required for Job Seeker to search available and active jobs.

- Apply for Job - This use-case denotes as to actions required for JobSeeker to apply for an available job vacancy.

- Add Reviews - This use case denotes a set of actions required for Job Seeker to add Reviews for an organization that can be viewed by the Employer.

Class Diagram: Class diagram is a graphical representation of the static view of the system. It describes the design and structure of the system by displaying the system's classes, their attributes, methods and relationships among objects.

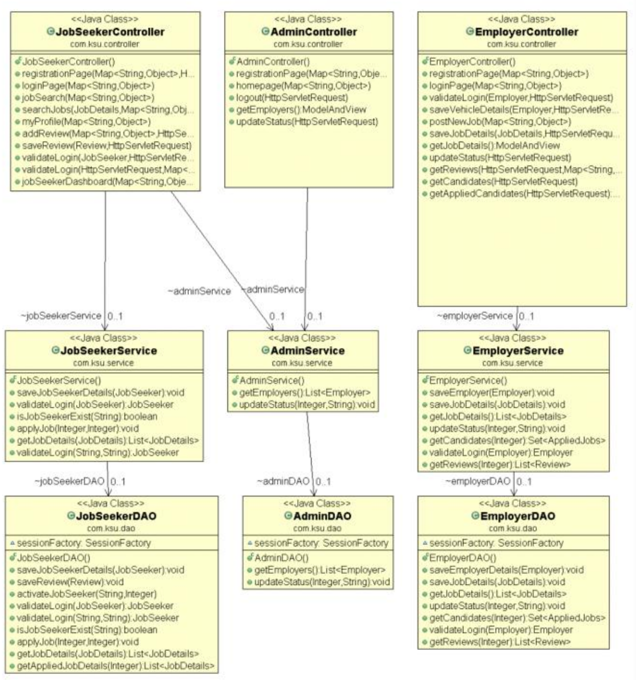

<<Java Class>>
JobSeekerController
com.ksu.controller
JobSeekerController()
registrationPage(Map<String,Object>,H...
loginPage(Map<String,Object>)
jobSearch(Map<String,Object>)
searchJobs(JobDetails,Map<String,Obj...
myProfile(Map<String,Object>)
addReview(Map<String,Object>,HttpSe...
saveReview(Review,HttpServletRequest)
validateLogin(JobSeeker,HttpServletRe...
validateLogin(HttpServletRequest,Map<...
jobSeekerDashboard(Map<String,Obje...
<<Java Class>>
AdminController
com.ksu.controller
AdminController()
registrationPage(Map<String,Obje...
homepage(Map<String,Object>)
logout(HttpServletRequest)
getEmployers():ModelAndView
updateStatus(HttpServletRequest)
<<Java Class>>
EmployerController
com.ksu.controller
EmployerController()
registrationPage(Map<String,Object>)
loginPage(Map<String,Object>)
validateLogin(Employer,HttpServletRequest)
saveVehicleDetails(Employer,HttpServletRe...
postNewJob(Map<String,Object>)
saveJobDetails(JobDetails,HttpServletRequ...
getJobDetails():ModelAndView
updateStatus(HttpServletRequest)
getReviews(HttpServletRequest,Map<String,...
getCandidates(HttpServletRequest)
getAppliedCandidates(HttpServletRequest):...
~adminService
~adminService
~jobSeekerService 0..1
0..1
0..1
~employerService 0..1
<<Java Class>>
JobSeekerService
com.ksu.service
JobSeekerService()
saveJobSeekerDetails(JobSeeker):void
validateLogin(JobSeeker):JobSeeker
isJobSeekerExist(String):boolean
applyJob(Integer,Integer):void
getJobDetails(JobDetails):List<JobDetails>
validateLogin(String,String):JobSeeker
<<Java Class>>
AdminService
com.ksu.service
AdminService()
getEmployers():List<Employer>
updateStatus(Integer,String):void
<<Java Class>>
EmployerService
com.ksu.service
EmployerService()
saveEmployer(Employer):void
saveJobDetails(JobDetails):void
getJobDetails():List<JobDetails>
updateStatus(Integer,String):void
getCandidates(Integer):Set<AppliedJobs>
validateLogin(Employer):Employer
getReviews(Integer):List<Review>
~jobSeekerDAO 0..1
~adminDAO 0..1
~employerDAO 0..1
<<Java Class>>
JobSeekerDAO
com.ksu.dao
sessionFactory: SessionFactory
JobSeekerDAO()
saveJobSeekerDetails(JobSeeker):void
saveReview(Review):void
activateJobSeeker(String,Integer)
validateLogin(JobSeeker):JobSeeker
validateLogin(String,String):JobSeeker
isJobSeekerExist(String):boolean
applyJob(Integer,Integer):void
getJobDetails(JobDetails):List<JobDetails>
getAppliedJobDetails(Integer):List<JobDetails>
<<Java Class>>
AdminDAO
com.ksu.dao
sessionFactory: SessionFactory
AdminDAO()
getEmployers():List<Employer>
updateStatus(Integer,String):void
<<Java Class>>
EmployerDAO
com.ksu.dao
sessionFactory: SessionFactory
EmployerDAO()
saveEmployerDetails(Employer):void
saveJobDetails(JobDetails):void
getJobDetails():List<JobDetails>
updateStatus(Integer,String):void
getCandidates(Integer):Set<AppliedJobs>
validateLogin(Employer):Employer
getReviews(Integer):List<Review>

Fig -3: Class Diagram

Controller classes: The Project contains three controller classes i.e. AdminController, JobSeekerController and EmployerController.

These classes are responsible for handling HTTP requests and returns HTTP response. Service classes: The Project contains three Service classes i.e. AdminService, JobSeekerService and Employer Service. The Controller classes pulls data from the request and passes it to the appropriate service class. The Service classes are responsible for called or more DAO class.

DAO classes: The Project contains three Service classes i.e. AdminDAO, JobSeekerDAO and EmployerDAO. The DAO classes contains the query code and directly interacts with the model classes. The DAO classes send back model classes to the Controller class in order to be sent to the view layer.

Methodology: The project is finished in 3 basic steps: 1. login/signup 2. If role is Job Provider:- choose location and post the job 3. If role is Job Seeker:- choose the job needed

Advantages: Accuracy, Classification

Disadvantages: Processing time

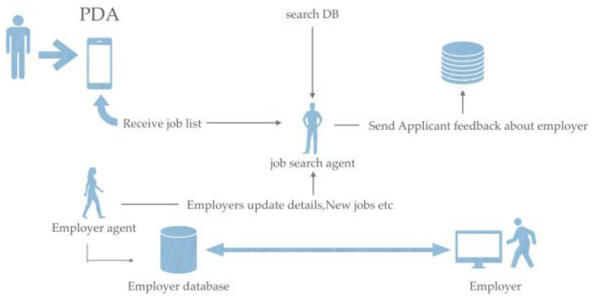


Fig -4: Architecture

Job Search Algorithm: Job-search theory attempts to propose strategies for making optimal employment decisions by considering factors that determine individual's demands and their prospect for finding an acceptable job offers.

The variables to be considered are: Industry, Occupation, Education, Job Type (Full-time, part-time, contract, etc.), Career Level (amount of experience obtained versus what is required for the job),Salary and Allowances (salary and all additional benefits).

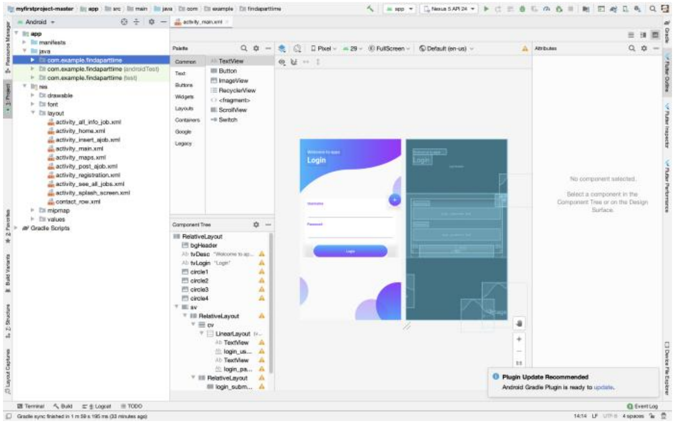
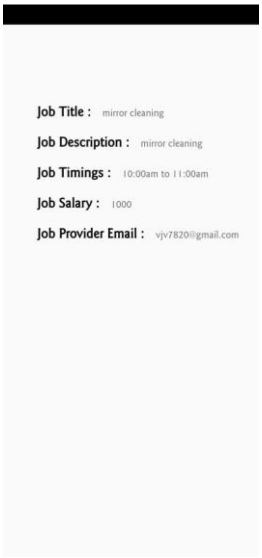


Fig -5: Implementation

PROPOSED APPROACH:

Once the user selects the job(which are present in the job database), the user will get the details of all the job from the job database. In order to get the location of the employer, the user will have a button to get the location. Here we are using Google Maps API to view the location of the user. The job database is filled by the Employer database. When any new user will add job details to the Employer Database, the job will be automatically updated in the Job database by using Normalisation techniques in Databases.



Job Title : mirror cleaning

Job Description : mirror cleaning

Job Timings : 10:00am to 11:00am

Job Salary : 1000

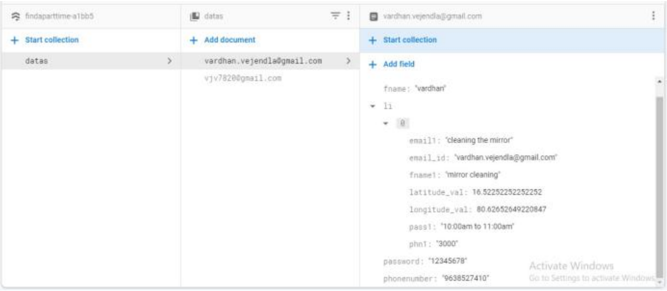
Job Provider Email : vjv7820@gmail.com

Fig -6: New Job Entry

Database design:

The database that is used to design the application is Firebase. Firebase Realtime database is a cloud hosted database that supports multiple platforms Android, iOS and Web. All the data is stored in JSON format and any changes in data, reflects immediately by performing a sync across all the platforms & devices. This allows us to build more flexible realtime apps easily with minimal effort. Here, we have identified five tables to achieve desired functionality.

- Employer table: holds details of Employer
- Jobseeker table: holds details of applicant
- Applied_Jobs table: holds details of jobs applied by the job seeker
- Posted_Jobs table: holds details of jobs posted by the Employer
- Reviews table: holds the reviews for interview, salary, work life provided by the jobseeker.



datas

vardhan.vejendla@gmail.com

vjv7820@gmail.com

```
fname: "vardhan"
▾ li
  ▾ 0
      email1: "cleaning the mirror"
      email_id: "vardhan.vejendla@gmail.com"
      fname1: "mirror cleaning"
      latitude_val: 16.52252252252252
      longitude_val: 80.62652649220847
      pass1: "10:00am to 11:00am"
      phn1: "3000"
password: "12345678"
phonenumber: "9638527410"
```

Fig -7: Database Design

When employer registers with the application, the application inserts the details of the employer into the Employer table. Similarly, when a jobseeker creates an account, his/her details will be inserted into the Jobseeker table. When jobseeker searches for the available job vacancies, the application queries the database to retrieve the job vacancies that are posted by the employer from the Posted_Jobs table. Similarly, when an employer wishes to view the applicants for a particular job posting, the application queries the database to retrieve the details of the job and job seeker from the Applied_Jobs table. In addition, the employer can activate or deactivate the job status thus updating the database. The jobseeker can provide reviews about an organization and will be saved in the Reviews table.

4. RESULTS AND ANALYSIS

Testing is a process of executing a program with the intent of finding bugs that makes the application fail to meet the expected behaviour. Regardless of the development methodology, the ultimate goal of testing is to make sure that what is created does what it is supposed to do. Testing plays a critical role for assuring quality and reliability of the software. I have included testing as a part of development process. The test cases should be designed with maximum possibilities of finding the errors or bugs. Various level of testing are as follows.

Testing Levels:

Unit testing: Unit testing tests the functionality of individual units of source code. It is the smallest component of a testable software that works in isolation with other parts of the code. I have done unit testing for various individual components of the source code to uncover errors within the boundary of the application.

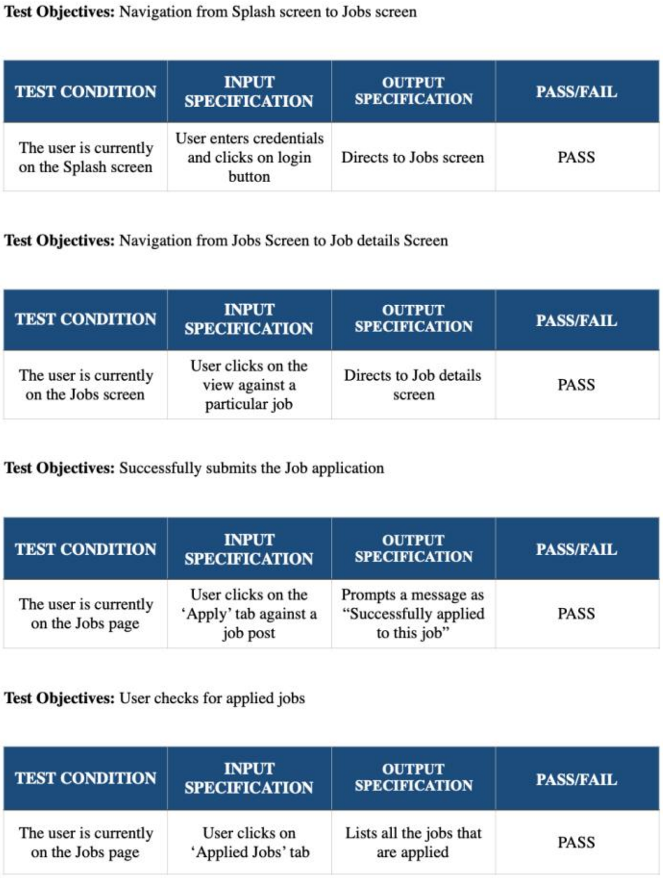
Integration testing: Integration testing focuses on the design and construction of the software. Here, the individual components that are tested using unit tests are combined and tested as a group. Its primary purpose is to expose the defects associated with the interfacing of modules. It checks if the modules perform the desired functionality when integrated together.

System testing: System testing is performed on a completely integrated system to see if it meets the requirements.

Regression testing: Regression testing aims at verifying the functionality of the software that is previously tested and to which changes are made. It is to ensure the old software still works with new changes.

Acceptance testing: Acceptance testing is conducted to verify if the system compliance the business requirements. Adhering to the levels of testing, Unit testing is performed on individual components of the system ensuring the expected behaviour. Later, I have integrated various components together and performed Integration testing. Once the integration testing is done, I have performed System 30 testing and ensured the application works as per the requirements. Finally, acceptance testing is performed to check if the client accepts the system. Test Case: A test case is a set of rules or conditions to check if the system or one of its feature works in accordance to the requirement. It is a document with a set of details which includes, set of test data, expected results, actual results, environment information and soon. I have designed and executed a few test cases to check if the application meets the functional requirements.

Below are the test cases for the Job Search Portal System application:



Test Objectives: Navigation from Splash screen to Jobs screen

TEST CONDITION	INPUT SPECIFICATION	OUTPUT SPECIFICATION	PASS/FAIL
The user is currently on the Splash screen	User enters credentials and clicks on login button	Directs to Jobs screen	PASS

Test Objectives: Navigation from Jobs Screen to Job details Screen

TEST CONDITION	INPUT SPECIFICATION	OUTPUT SPECIFICATION	PASS/FAIL
The user is currently on the Jobs screen	User clicks on the view against a particular job	Directs to Job details screen	PASS

Test Objectives: Successfully submits the Job application

TEST CONDITION	INPUT SPECIFICATION	OUTPUT SPECIFICATION	PASS/FAIL
The user is currently on the Jobs page	User clicks on the 'Apply' tab against a job post	Prompts a message as "Successfully applied to this job"	PASS

Test Objectives: User checks for applied jobs

TEST CONDITION	INPUT SPECIFICATION	OUTPUT SPECIFICATION	PASS/FAIL
The user is currently on the Jobs page	User clicks on 'Applied Jobs' tab	Lists all the jobs that are applied	PASS

Fig -8: represents test cases passed for the proposed application.

5. CONCLUSION AND FUTURE SCOPE

Job Search is a very involved process that could require hours of interaction with different search sites, applications, human agents, etc. The developed system intelligently anticipates the needs of the user and makes intelligent decisions based on fuzzy preference rules and dynamically make location, salary markup and markdown, and allowances choices that are perceived as beneficial to the user. This is evident in the results presented in the form of scenarios and supporting screenshots. The system could be extended to include a secure application process where the applicant's experience and education is verified possibly by including biometric data along with the job application details which has been published elsewhere. In addition the job search process could enhance the calculation of utility by including risk factors of success in choosing one job over another. This could enhance the probability of applying for the job that would be most suitable for an applicant on many levels.

This project fulfils the primary requirements of the job seekers and employers. It can be extended in several ways – We can provide recommendations and email updates for new job postings based on the job seeker's search history. Since, the job seekers might be interested in building a strong Resume, we can provide tips and information for the same. We can also provide templates for building the Resumes which might interest most applicants. The mobile application is developed fulfilling the functionalities of job seeker, it can be extended to support functionalities of Employer as well.

REFERENCES

[0]Mochol, Malgorzata, Holger Wache, and Lyndon Nixon. "Improving the accuracy of job search with semantic techniques." Berlin, Germany, 2007

[1]Franklin, Stan, and Art Graesser. " Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents." Third International Workshop on Agent Theories Architectures and Languages. Springer-Verlag, 1996.

[2]Jennings, N. R., and M. Wooldridge. Applications of Intelligent Agents. London: University of London, 1998.

[3]Hayes-Roth, B. "An Architecture for Adaptive Intelligent Systems." Artificial Intelligence: Special Issue on Agents and Interactivity, 1995: 72, 329-365.

[4] h#ps://www.genmymodel.com/use-case-diagram-online