JOB PORTAL

A PROJECT REPORT

Submitted By

YASHI JAIN
2000290140142
PRAKSHA TANDON
2000290140089
VYOM GUPTA
2000290140137

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Dr. Arun Kr. Tripathi
Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(JAN 2022)

TABLE OF CONTENTS

LIST OF FIGURES

CERTIFICATE

Certified that **Yashi Jain**(200029014005827**), Praksha Tandon**(200029014005774**),Vyom Gupta**(200029014005822) have carried out the project work having **"Job Portal"** for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:13/01/2022

> **Yashi Jain (2000290140142)**
> **Praksha Tandon(2000290140089)**
> **Vyom Gupta(2000290140137)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:13/01/2022

> **Dr. Arun Kr. Tripathi**
> **Professor**
> **Department of Computer Applications**
> **KIET Group of Institutions, Ghaziabad**

Signature of Internal Examiner   **Signature of Internal Examiner**

Dr. Ajay Shrivastava
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

Job portal is developed for creating an interactive job vacancy for candidates. This web application is to be conceived in its current form as a dynamic site-requiring constant updates both from the seekers as well as the companies. On the whole the objective of the project is to enable jobseekers to place their resumes and companies to publish their vacancies. It enables jobseekers to search for jobs, view personal job listings.

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 organization's objective, their work culture and current job openings. In addition, finding the right candidate with desired qualifications to fill their current job openings is an important task for the recruiters of any organization. Online Job Search Portals have certainly made job seeking convenient on both sides. Job Portal is the solution where recruiter as well as the job seeker meet aiming at fulfilling their individual requirement. They are the cheapest as well as the fastest source of communication reaching wide range of audience on just a single click irrespective of their geographical distance.

ACKNOWLEDGEMENT

1. Introduction

This project is aimed at developing an online search Portal for the Placement Details for job seekers. The system is an online application that can be accessed throughout the organization and outside as well with proper login provided. This system can be used as an Online Job Portal for job seekers. Job Search Portal is a web application, which serves jobseekers to find available job vacancies and Employers to identify eligible job seekers with the prospect of selecting the most qualified candidates. The only way to select best-qualified candidate is to have a pool of eligible applicants, which is possible by drawing the interest of individuals in the market. Job search portals best serve this purpose.

E-recruitment has become the standard means for employers and job seekers to meet their respective objectives. The traditional methods for recruitment includes Job fairs, University career employment services, Employee referrals, advertising in the newspapers, televisions etc. With the advancement in technology and growth of internet usage, the e-recruitment has revolutionized the way organizations hire and candidates search for jobs. With the Online Job search portals, the recruitment process is speeded up at every stage from job postings, to receiving applications from candidates, interviewing process. The cost of searching/posting jobs will be much less compared to the traditional way of advertising. Job search portal stands as an effective means for Employers to outline the job vacancies, responsibilities and qualifications to attract jobseekers.

1.1 Project Description

1. In many websites job seeker and job provider cannot contact with each other. (faces difficulty in communication)
2. Job Seekers faces problems in searching best jobs that fits its profile.
3. Previous track of applied jobs cannot be seen.
4. In some existing system user can not add job to wish list for future reference.
5. Complex Job search.
6. Duplication of jobs - Sometimes job seekers could not reach the employers who originally posted the job.
7. Lack of Freshers Job, Walk-in Jobs, internship.
8. No proper feedback through portal, after employer viewed the profile of job seeker who applied.
9. Cannot Upload and Download the latest updates.
10. No proper coordination between different Applications and Users.

1.2 Project Scope

There is ample scope of enhancement and adding functionalities to this application. This application can be extended to send automated interview scheduling through acceptance/rejection of Resume. Companies can delete jobs once the job availably period is over automatically. The application can have a job recommendation system based on the

frequent search results of different users. The portal can also send email notifications to candidates about certain job availabilities. There can be a feedback or review section for the application. The application can be more scalable by extending the search functionality based on country, city or area. While this application meets the basic requirements of a job portal eliminating few of the traditional challenges faced like time, money and effort, it can be extended to make the application more dynamic and robust. The User Interface can be made more attractive and user friendly. We can dig through more AngularJS magic capabilities to add additional features to the UI.

1.3 Hardware and Software Requirements

1.3.1 Hardware Specification

• HDD 20 GB Hard Disk Space and Above

• 4 GB RAM

• 2.8 GHz Processor and Above

1.3.2 Software Specification

• Operating System: Windows, Linux

• IDE: Visual Studio Code

• Backend - Django, Python

• Database - MYSQL

• Frontend - HTML, CSS , JavaScript, Ajax

2. Feasibility Study

2.1 Technical Feasibility

This system will be developed using Django framework, MySQL. All these technologies are easy to learn and can develop system very rapidly. After developing and deploying the system, any user can view this site on the Internet.

2.2 Economical Feasibility

Proposed System requires development tools and software such as visual studio code and python package which are free of cost and available on internet. For developing proposed system, we need various resources such as computers systems, internet connection for e-help, recommended disk space, and memory speed as mention in technical requirement. By looking at all these expenses and comparing with proposed system, we have many benefits from proposed system such are: As existing system is manual, where data may not accurate, up to date, and available on time. But proposed system will be computerized, so we can overcome all limitations of existing system. Also with this new system insertion, deletion, and modification of various data will be easier to handle. This system will reduce the paperwork. And quality of data will be improved. So keeping all above mentioned benefits and comparing with various expenditures of resources, we conclude that proposed system is economical feasible.

2.3 Operational Feasibility

No doubt the proposed system is fully GUI based that is very user friendly and all inputs to be taken all self-explanatory even to a layman. To put an orders user should have only basic knowledge of computer and Internet which is not a big issue.

2.4 Behavioural Feasibility

It evaluates and estimates the user attitude or behavior towards the development of new system. It helps in determining if the system requires special effort to educate, retrain, transfer, and changes in employee's job status on new ways of conducting business.

3. Database Design

A properly designed database provides you with access to up-to-date, accurate information. Because a correct design is essential to achieving your goals in working with a database, investing the time required to learn the principles of good design makes sense. In the end, you are much more likely to end up with a database that meets your needs and can easily accommodate change.

This article provides guidelines for planning a desktop database. You will learn how to decide what information you need, how to divide that information into the appropriate tables and columns, and how those tables relate to each other. You should read this article before you create your first desktop database.

Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

The main objectives behind database designing are to produce physical and logical design models of the proposed database system. To elaborate this, the logical model is primarily concentrated on the requirements of data and the considerations must be made in terms of monolithic considerations and hence the stored physical data must be stored independent of the physical conditions. On the other hand, the physical database design model includes a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as Database Management System (DBMS).

3.1 Flowchart Diagram

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams. Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence. They can range from simple, hand-drawn charts to comprehensive computer-drawn diagrams depicting multiple steps and routes. If we consider all the various forms of flowcharts, they are one of the most common diagrams on the planet, used by both technical and non-technical people in

numerous fields. Flowcharts are sometimes called by more specialized names such as ,
Process Map, Functional Flowchart, Business Process Mapping, Business Process Modeling
and Notation (BPMN), or Process Flow Diagram (PFD). They are related to other popular
diagrams, such as Data Flow Diagrams (DFDs) and Unified Modeling Language (UML)
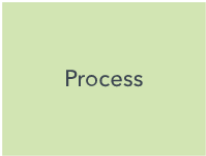Activity Diagrams.

Flowchart Annotations

Here are some of the common flowchart symbols.

Terminal/Terminator

Process

Decision

Document

Document
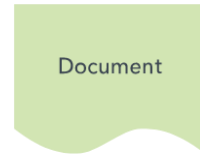
Data, or Input/Output

Data

Stored Data

Database

Flow Arrow

Comment or Annotation

Note { } Note

Predefined process

Predefined
Process
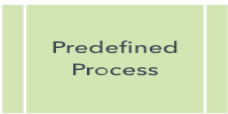
On-page connector/reference

A

Off-page connector/reference

Off-Page
Link

Flow Chart - Job Portal System

```
                              Start

         ┌───────────────────┼───────────────────┐
         ▼                    ▼                    ▼
      Admin               Student              Company

         ▼                    ▼                    ▼
      Login ──No──┐      Registration          Registration
                  │
       Yes                    ▼                    ▼
                         Login ──No──┐       Login ──No──┐
                                     │                   │
   Manage student        Yes                 Yes

   Manage Company      Search Job            Search Student

   Account             Apply Job             Post Job

                       Manage Job            Manage Job

                       Profile Match         Apply Job Report

                       Message               Message

                       Manage Account        Profile Match

                       Like Job              Manage Account

                       Change Password       Change Password

         └─────────────────Logout───────────────────┘

                              End
```
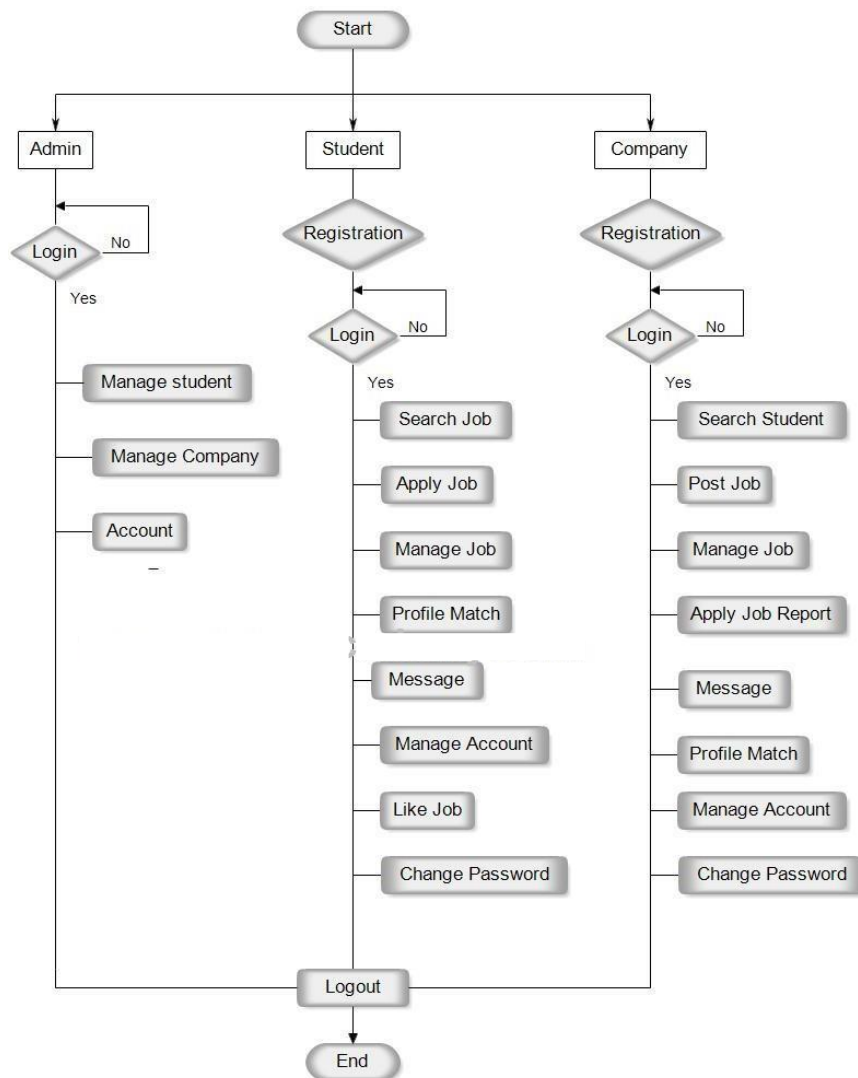
Fig 1: Flowchart Diagram

3.2 Usecase Diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State chart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

When the initial task is complete, use case diagrams are modelled to present the outside view.

In brief, the purposes of use case diagrams can be said to be as follows −

i.   Used to gather the requirements of a system.
ii.   Used to get an outside view of a system.
iii.   Identify the external and internal factors influencing the system.
iv.   Show the interaction among the requirements are actors.

How to draw a Use Case diagram?

It is essential to analyse the whole system before starting with drawing a use case diagram, and then the system's functionalities are found. And once every single functionality is identified, they are then transformed into the use cases to be used in the use case diagram. After that, we will enlist the actors that will interact with the system. The actors are the person or a thing that invokes the functionality of a system. It may be a system or a private entity, such that it requires an entity to be pertinent to the functionalities of the system to which it is going to interact.

Once both the actors and use cases are enlisted, the relation between the actor and use case/ system is inspected. It identifies the no of times an actor communicates with the system. Basically, an actor can interact multiple times with a use case or system at a particular instance of time.

Following are some rules that must be followed while drawing a use case diagram:

i.   A pertinent and meaningful name should be assigned to the actor or a use case of a system.

ii.   The communication of an actor with a use case must be defined in an understandable way.

iii.   Specified notations to be used as and when required.

iv.   The most significant interactions should be represented among the multiple no of interactions between the use case and actors.

Fig 2: Usecase Diagram

3.3 Sequence Diagram

A sequence diagram or system sequence diagram (SSD) shows object interactions arranged in time sequence in the field of software engineering. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

For a particular scenario of a use case, the diagrams show the events that external actors generate, their order, and possible inter-system events. All systems are treated as a black box; the diagram places emphasis on events that cross the system boundary from actors to systems. A system sequence diagram should be done for the main success scenario of the use case, and frequent or complex alternative scenarios.

i.   To model high-level interaction among active objects within a system.
ii.   To model interaction among objects inside a collaboration realizing a use case.
iii.   It either models generic interactions or some certain instances of interaction.

Sequence Diagram Notations –

i.   Actors – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram
ii.   Lifelines – A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram.
iii.   Messages – Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.
iv.   Guards – To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

Uses of sequence diagrams –

i.   Used to model and visualise the logic behind a sophisticated function, operation or procedure.
ii.   They are also used to show details of UML use case diagrams.

iii.   Used to understand the detailed functionality of current or future systems.

iv.   Visualise how messages and tasks move between objects or components in a system.
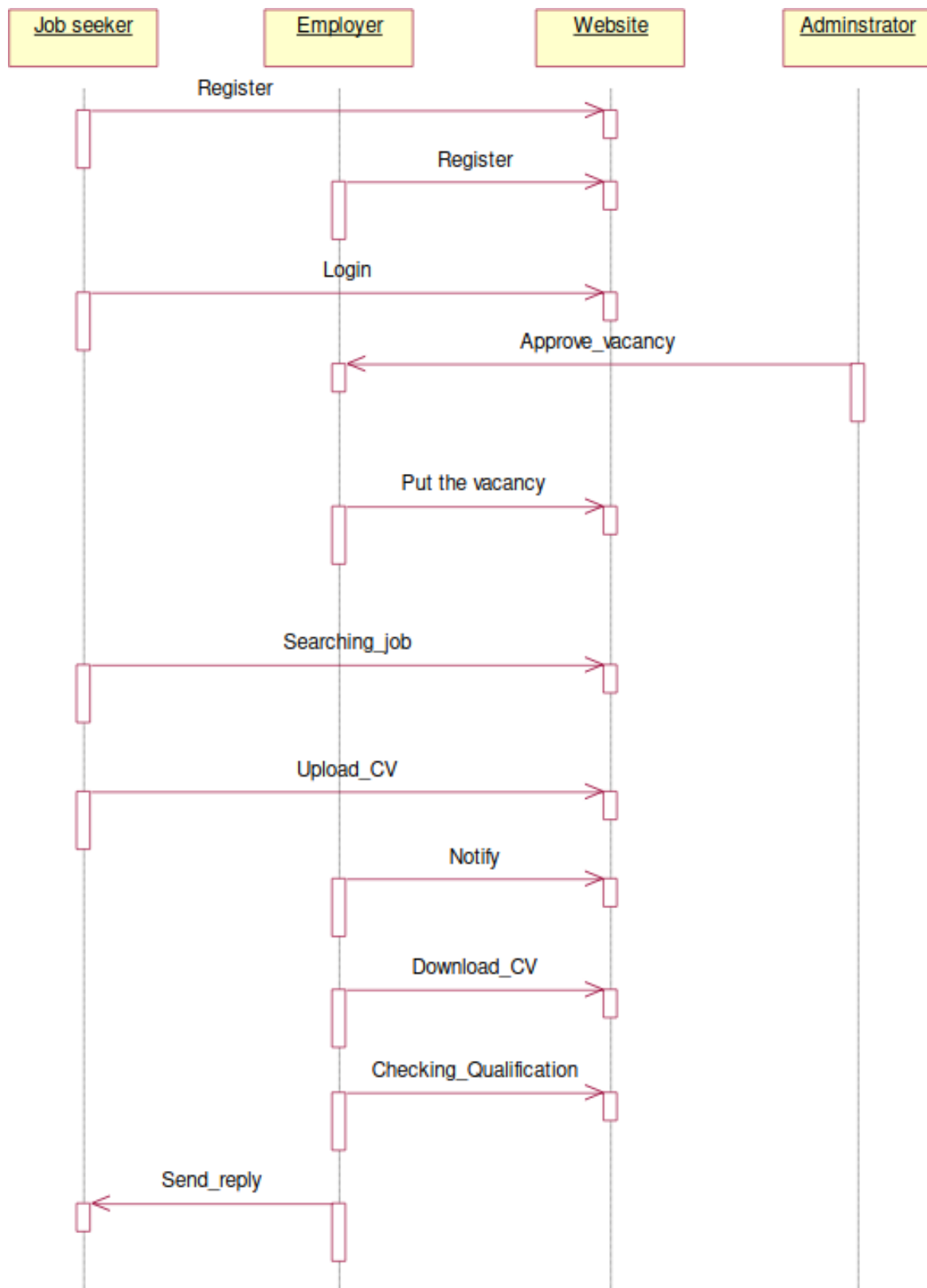
Fig 3: Sequence Diagram

3.4 Activity Diagram

The activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities. The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.

It is also termed as an object-oriented flowchart. It encompasses activities composed of a set of actions operations that are applied to model the behavioral diagram

Why use Activity Diagram?

An event is created as an activity diagram encompassing a group of nodes associated with edges. To model the behavior of activities, they can be attached to any modeling element. It can model use cases, classes, interfaces, components, and collaborations.

It mainly models processes and workflows. It envisions the dynamic behavior of the system as well as constructs a runnable system that incorporates forward and reverse engineering. It does not include the message part, which means message flow is not represented in an activity diagram.

It is the same as that of a flowchart but not exactly a flowchart itself. It is used to depict the flow between several activities.

How to draw an Activity Diagram?

An activity diagram is a flowchart of activities, as it represents the workflow among various activities. They are identical to the flowcharts, but they themselves are not exactly the flowchart. In other words, it can be said that an activity diagram is an enhancement of the flowchart, which encompasses several unique skills.
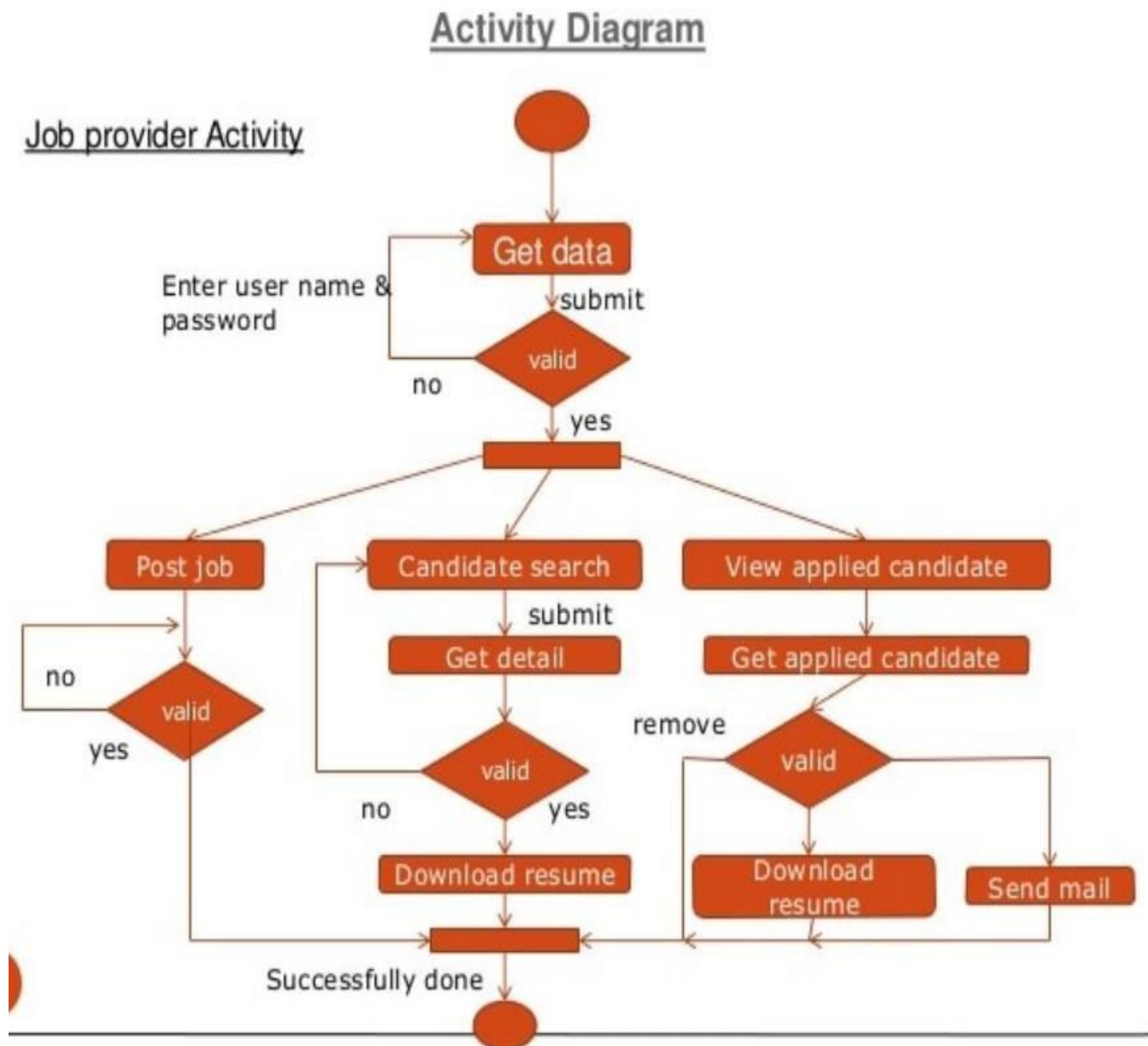
Since it incorporates swim lanes, branching, parallel flows, join nodes, control nodes, and forks, it supports exception handling. A system must be explored as a whole before drawing an activity diagram to provide a clearer view of the user. All of the activities are explored after they are properly analyzed for finding out the constraints applied to the activities. Each and every activity, condition, and association must be recognized.

After gathering all the essential information, an abstract or a prototype is built, which is then transformed into the actual diagram.

Following are the rules that are to be followed for drawing an activity diagram:

i.   A meaningful name should be given to each and every activity.
ii.   Identify all of the constraints.

iii.   Acknowledge the activity associations

Activity Diagram

Job provider Activity

Get data

Enter user name & password

submit

valid

no

yes

Post job

Candidate search

submit

Get detail

View applied candidate

Get applied candidate

no

valid

yes

valid

no

yes

remove

valid

Download resume

Download resume

Send mail

Successfully done

Fig 4 : Activity Diagram

3.5 Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

Purpose of Class Diagrams

The main purpose of class diagrams is to build a static view of an application. It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages. It is one of the most popular UML diagrams. Following are the purpose of class diagrams given below:

i.   It analyses and designs a static view of an application.

ii.   It describes the major responsibilities of a system.

iii.   It is a base for component and deployment diagrams.

iv.   It incorporates forward and reverse engineering

How to draw a Class Diagram?

The class diagram is used most widely to construct software applications. It not only represents a static view of the system but also all the major aspects of an application. A collection of class diagrams as a whole represents a system.
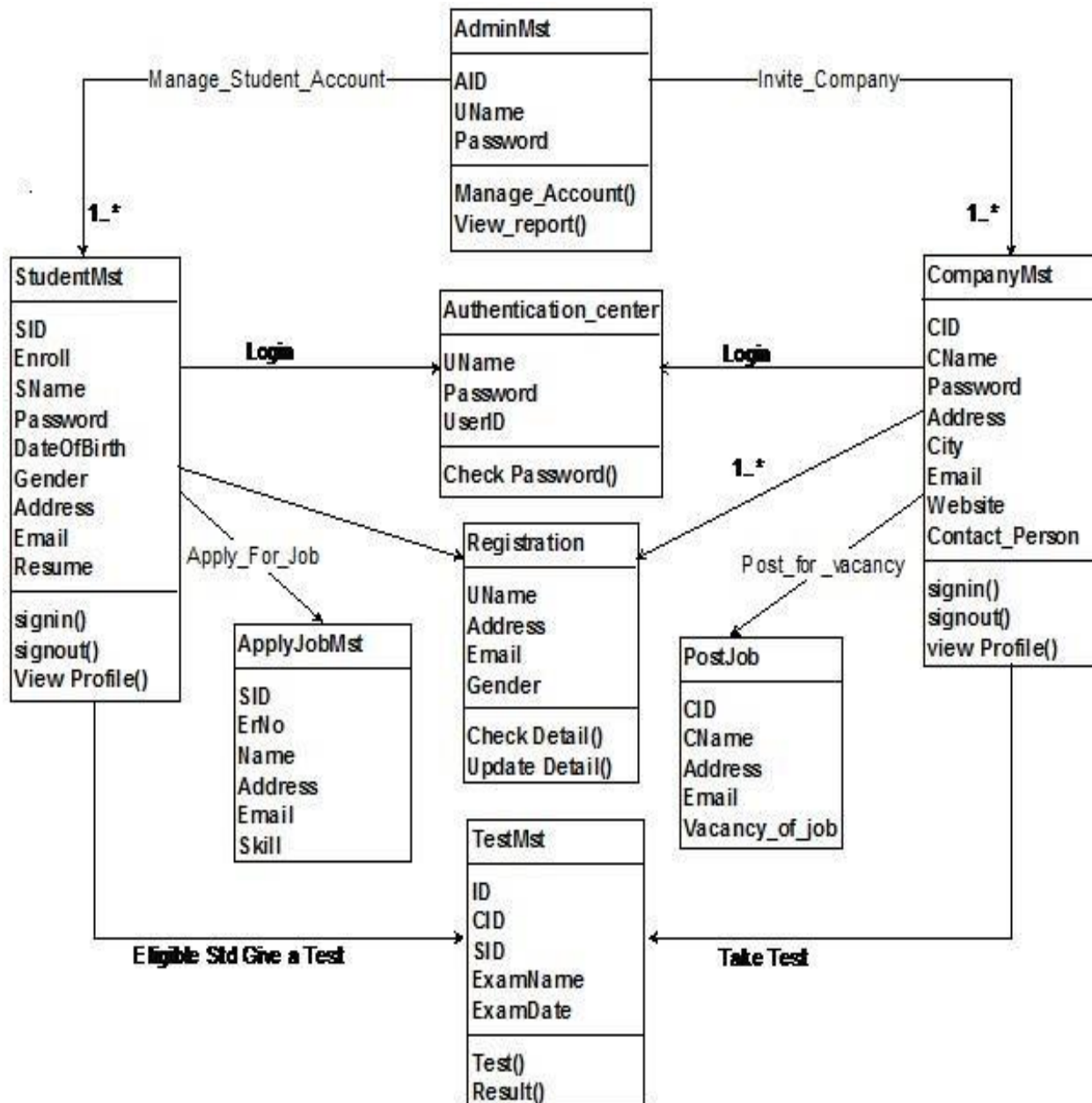
Some key points that are needed to keep in mind while drawing a class diagram are given below:

1.   To describe a complete aspect of the system, it is suggested to give a meaningful name to the class diagram.

2.   The objects and their relationships should be acknowledged in advance.

3.   The attributes and methods (responsibilities) of each class must be known.

4.   A minimum number of desired properties should be specified as more number of the unwanted property will lead to a complex diagram.

5.   Notes can be used as and when required by the developer to describe the aspects of a diagram.
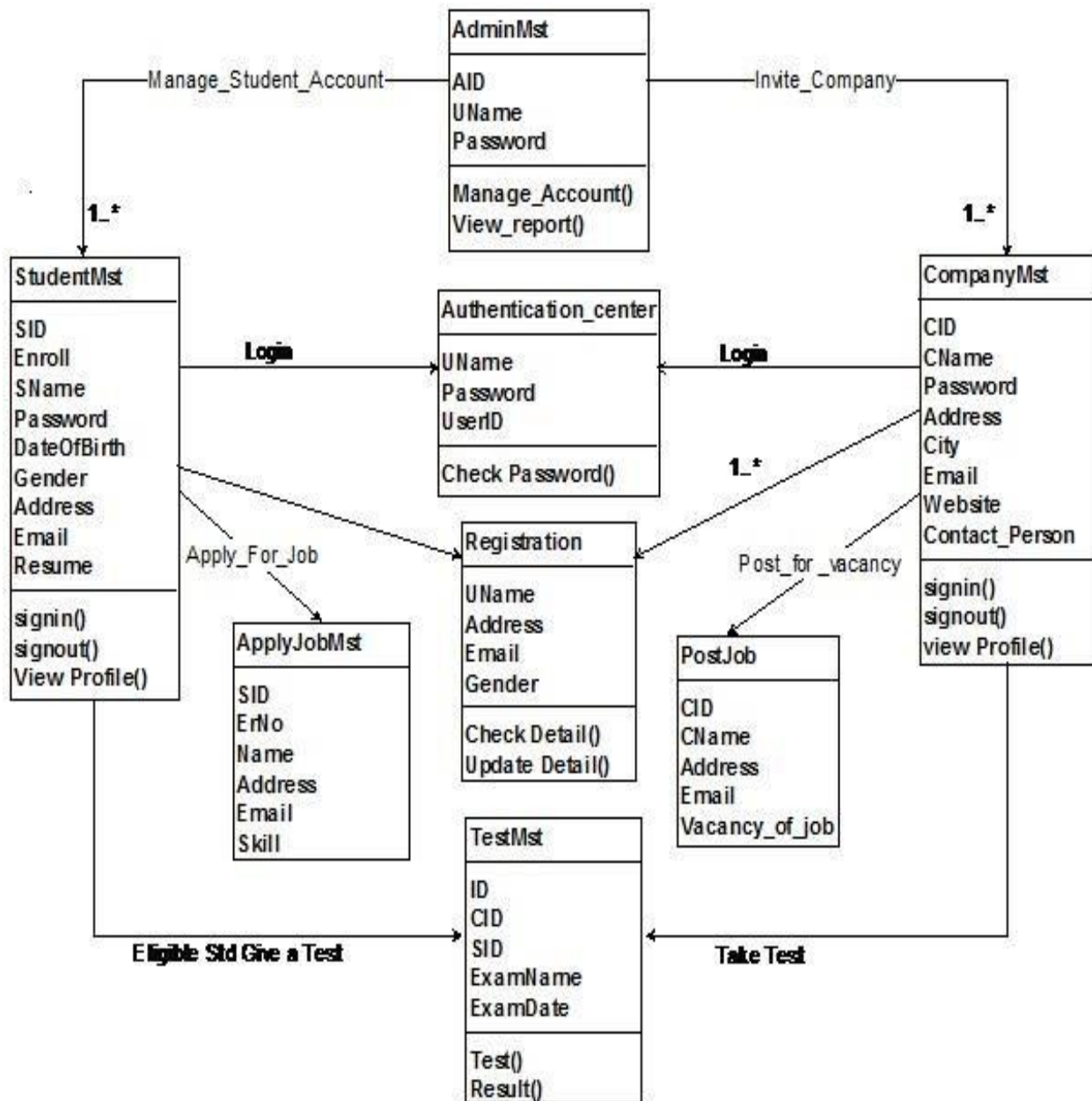
6. The diagrams should be redrawn and reworked as many times to make it correct before producing its final version.

Fig 3.5.1 Conceptual Class Diagram

AdminMst
AID
UName
Password

Manage_Account()
View_report()

—Manage_Student_Account—

—Invite_Company—

1_*

1_*

StudentMst
SID
Enroll
SName
Password
DateOfBirth
Gender
Address
Email
Resume

signin()
signout()
View Profile()

Login

Authentication_center
UName
Password
UserID

Check Password()

Login

CompanyMst
CID
CName
Password
Address
City
Email
Website
Contact_Person

signin()
signout()
view Profile()

Apply_For_Job

Registration
UName
Address
Email
Gender

Check Detail()
Update Detail()

1_*

Post_for_vacancy

ApplyJobMst
SID
ErNo
Name
Address
Email
Skill

PostJob
CID
CName
Address
Email
Vacancy_of_job

TestMst
ID
CID
SID
ExamName
ExamDate

Test()
Result()

Eligible Std Give a Test

Take Test

3.5.2 Detailed Class Diagram

AdminMst

AID
UName
Password

Manage_Account()
View_report()

—Manage_Student_Account—

—Invite_Company—

1_*

1_*

StudentMst

SID
Enroll
SName
Password
DateOfBirth
Gender
Address
Email
Resume

signin()
signout()
View Profile()

Login

Authentication_center

UName
Password
UserID

Check Password()

Login

CompanyMst

CID
CName
Password
Address
City
Email
Website
Contact_Person

signin()
signout()
view Profile()

Apply_For_Job

Registration

UName
Address
Email
Gender

Check Detail()
Update Detail()

1_*

Post_for _vacancy

ApplyJobMst

SID
ErNo
Name
Address
Email
Skill

PostJob

CID
CName
Address
Email
Vacancy_of_job

TestMst

ID
CID
SID
ExamName
ExamDate

Test()
Result()

Eligible Std Give a Test

Take Test

3.5 ER Diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs. ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

Components of ER Diagram

Entity
A definable thing—such as a person, object, concept or event—that can have data stored about it. Think of entities as nouns.
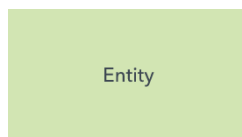
Entity

Entity categories

Entities are categorized as strong, weak or associative. A **strong entity** can be defined solely by its own attributes, while a **weak entity** cannot. An associative entity associates entities (or elements) within an entity set.

Weak Entity

Associative Entity

Relationship

How entities act upon each other or are associated with each other. Think of relationships as verbs

Relationship

Attribute

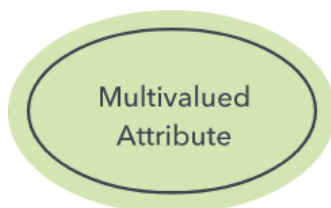A property or characteristic of an entity. Often shown as an oval or circle.

Attribute

Derived

Attributed is calculated or otherwise derived from another attribute, such as age from a birthdate.

Derived Attribute

Multi-value

More than one attribute value is denoted, such as multiple phone numbers for a person.

Multivalued Attribute

Cardinality

Defines the numerical attributes of the relationship between two entities or entity sets. The three main cardinal relationships are one-to-one, one-to-many, and many-many. A **one-to-one example** would be one student associated with one mailing address. A **one-to-many example (or many-to-one, depending on the relationship direction):** One student registers for multiple courses, but all those courses have a single line back to that one student. **Many-to-many example:** Students as a group are associated with multiple faculty members, and faculty members in turn are associated with multiple students.

Zero or one

Many

One (and only one)

One

Zero or many

One or many
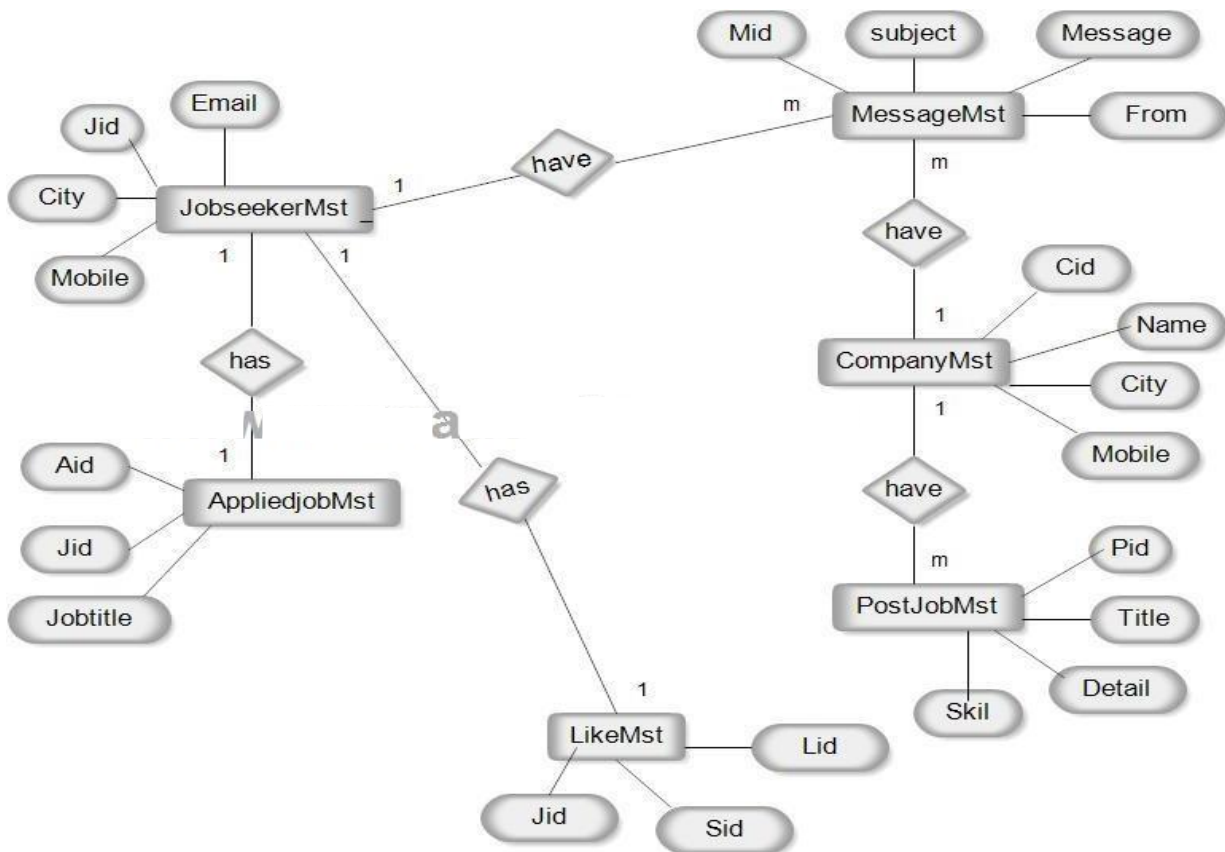
Fig 6: ER Diagram

3.6 State Diagram

A **state diagram** is used to represent the condition of the system or part of the system at finite instances of time. It's a **behavioral** diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as **State machines** and **State-chart Diagrams**. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. We can say that each and every class has a state, but we don't model every class using State diagrams. We prefer to model the states with three or more states.

Uses of state chart diagram –

- We use it to state the events responsible for change in state (we do not show what processes cause those events).
- We use it to model the dynamic behavior of the system .
- To understand the reaction of objects/classes to internal or external stimuli.

Basic components of a state chart diagram –

1. **Initial state**

   We use a black filled circle represent the initial state of a System or a class.

   Figure – initial state notation

2.Transition

We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.

   **Figure –** transition

3.State

We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.

Figure – state notation

4.Fork

We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent state and outgoing arrows towards the newly created states. We use the fork notation to represent a state splitting into two or more concurrent states.
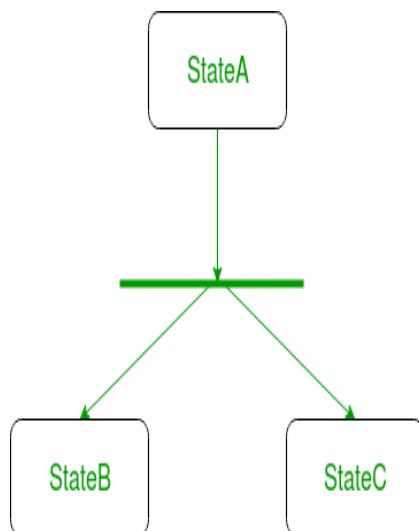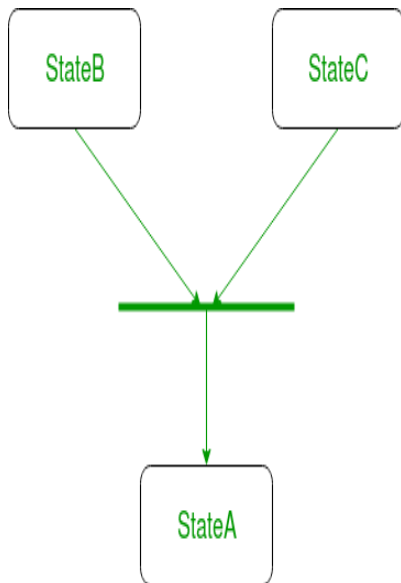

Figure – a diagram using the fork notation

5.Join

We use a rounded solid rectangular bar to represent a Join notation with incoming arrows from the joining states and outgoing arrow towards the common goal state. We use the join notation when two or more states concurrently converge into one on the occurrence of an event or events.

Figure – a diagram using join notation

6.Self transition

We use a solid arrow pointing back to the state itself to represent a self transition. There might be scenarios when the state of the object does not change upon the occurrence of an event. We use self transitions to represent such cases.
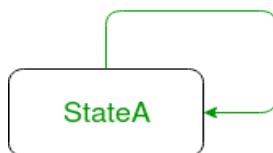
Figure – self transition notation

7. Composite state

We use a rounded rectangle to represent a composite state also .We represent a state with internal activities using a composite state.
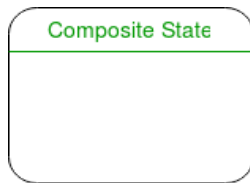
Figure – a state with internal activities

8.Final state

We use a filled circle within a circle notation to represent the final state in a state machine diagram.
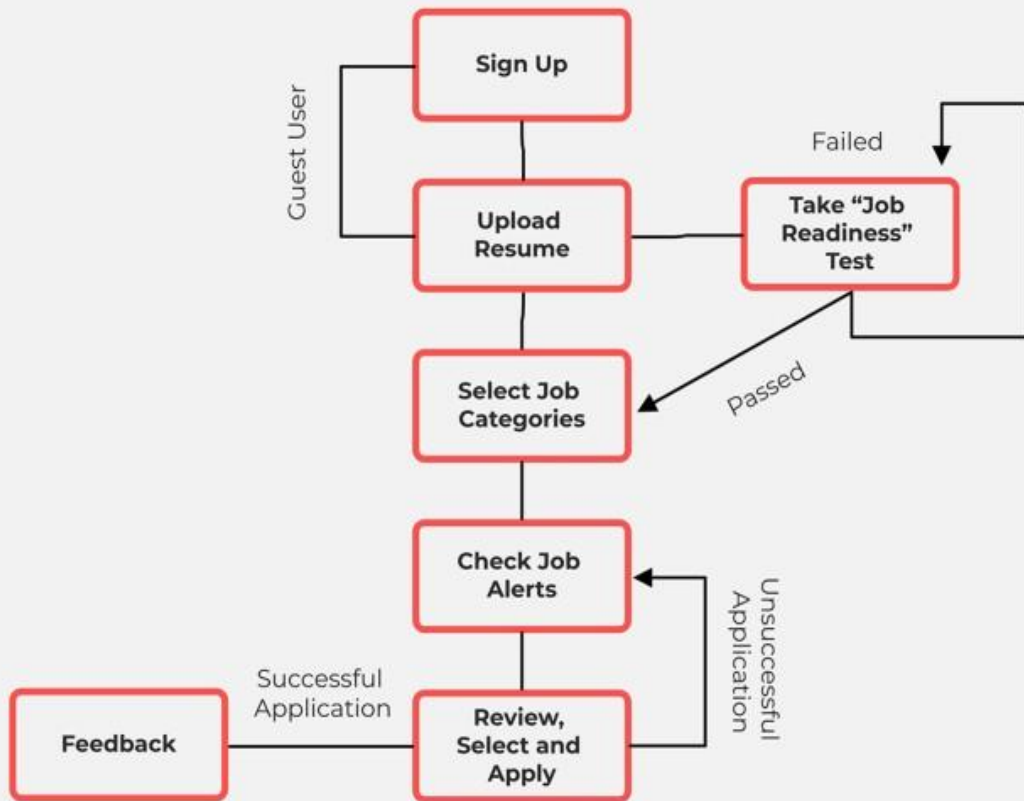

Figure – final state notation

Fig 7: State Chart Diagram

4   System Features

4.1   Functional Requirements

The proposed system has the following functionalities:

- ➢ Job Seeker Sign In
- ➢ Job Seeker Sign Up
- ➢ Job Seeker can view all available jobs
- ➢ Job seeker can search based on advanced search
- ➢ Advanced search can be done on location, technology, etc.
- ➢ Job seeker can apply to more than one jobs.
- ➢ Job seeker can view all applied jobs.
- ➢ Job seeker can update their profile information.
- ➢ Job seeker can retrieve their password if forgot.
- ➢ Companies can Register to the portal.
- ➢ Admin approves the company registration,
- ➢ Companies can post jobs.
- ➢ Companies can see applications.
- ➢ Companies can see all approved jobs.
- ➢ Admin can view all jobs.
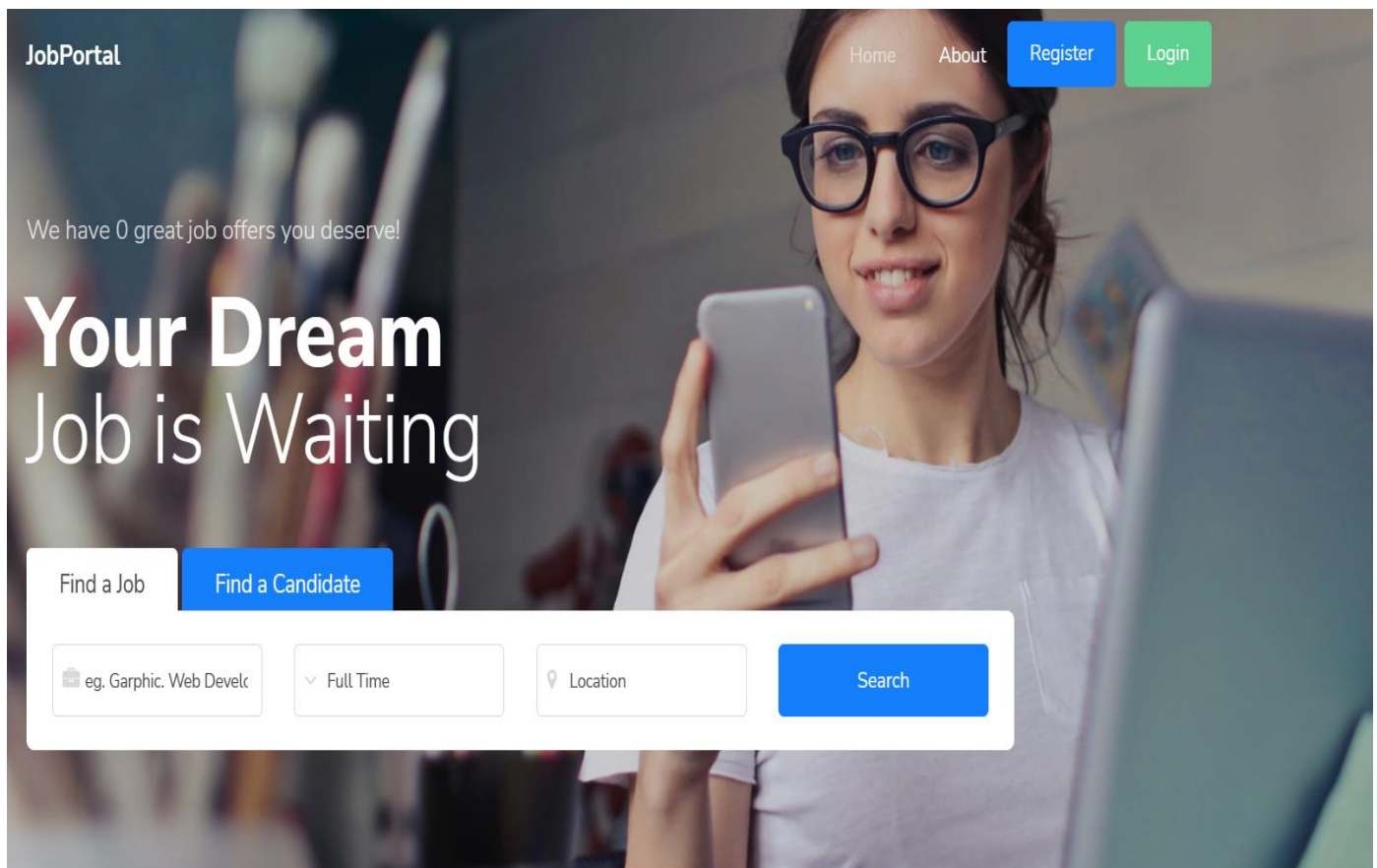- ➢ Admin approves the jobs posted by the company.

4.2 Non-Functional Requirements

- **Reliability requirements** - The system must perform accurately towards the administrator request. For example, when the administrator saves the edited details of the user, after he reviews the details later, they must be changed according to the latest details that was updated. Moreover, the client is not allowed to view the details that the administrator has. Besides that, the login form will have validity check to ensure that only the authorized users gain access to the system.

- **Usability requirements** - This system should be user-friendly and easy to use so that users can perform their tasks nicely.

- **Implementation requirements** - In implementing the system, it uses python as the main tool. This forms the front-end. At the back-end, the MYSQL will be synchronized and be used to maintain the information in the database. This is formed by the databases and other data stores.

- **Security requirements** - User credentials should be encrypted so as to ensure confidentiality, integrity and availability and the project ideas should be protected so as to avoid being stolen by other parties.

5   Form Design

Form design is **the process of creating a web form** — where your site visitors can input and submit their information — while keeping the form's layout, format, UX, appearance, and other factors in mind. Great form design enhances UX and helps boost conversions.

4.1   Home Page

4.2

4.2.1 Login

4.2.2

4.3

4.3.1 Register

4.3.2

4.4 When an employee registered on our portal
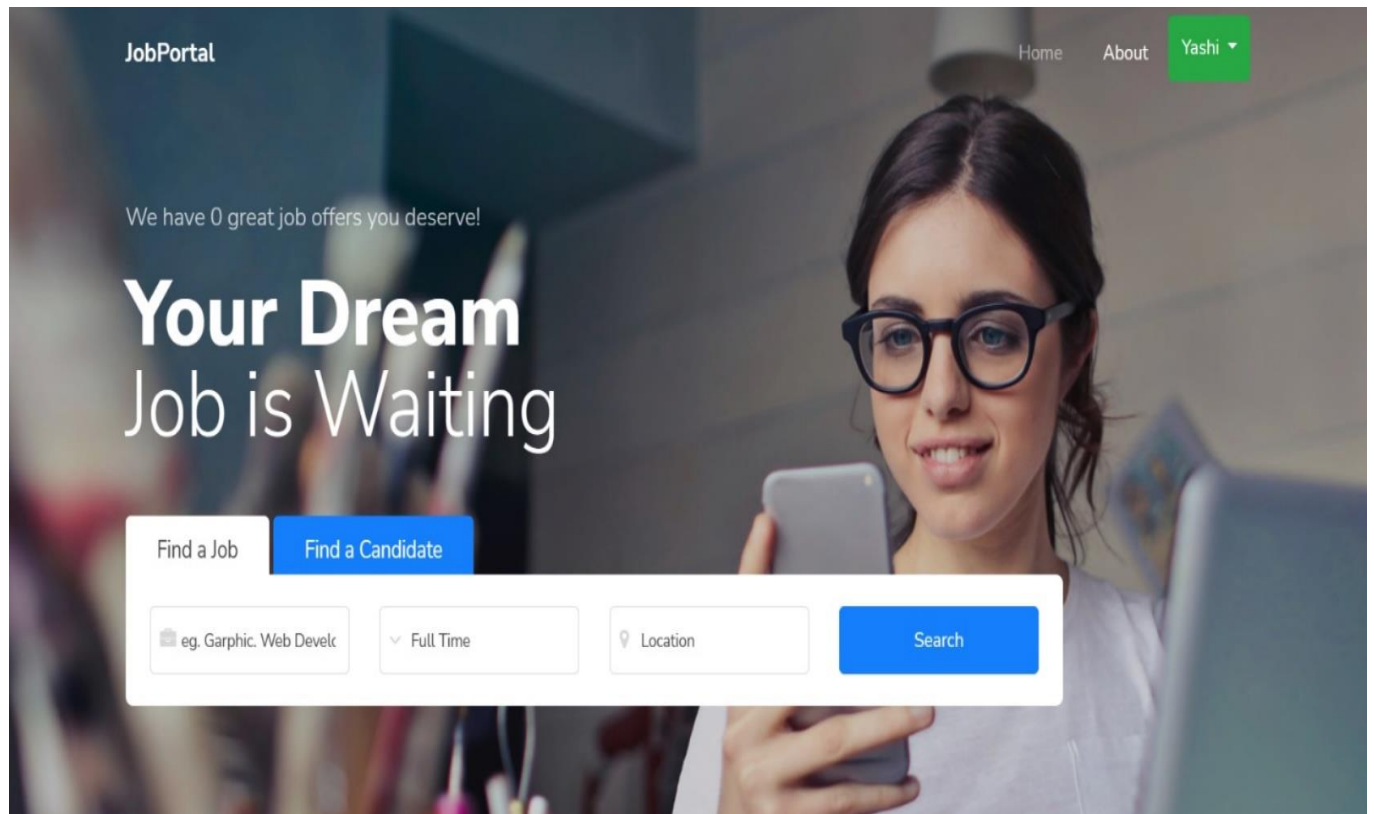
5.Test Case

5.1

TestCaseId	Component	Priority	Test cases	Description/Test Summary	Pre-requisites	Test Steps	Expected Result	Actual Result	Status	Test E
TC01	Registration	p10	TC01.1	To verify that when a user enters an unregistered email address and presses enter, an OTP must be sent to that email and a page should open that will ask the user to enter the OTP(One Time Password).	ELP website must be open.	1. Click on the register option. 2. Once the registration page is launched, enter a valid and unregistered email address. 3. Press enter	An OTP should be sent to the entered email address and next page should be launched where the user to enter the	An OTP sent to the entered email address and next page launched where the user needs to enter the OTP.	Pass	
			TC01.2	To verify that when a user enters an already registered email address then an error message should pop up saying "Account already exists, login instead".	A registered email address and ELP website must be open.	1. Click on the register option. 2. Once the registration page is launched, enter an already registered email address. 3. Press enter	An error message should pop up saying "Account already exists, login instead".	An error message pops up saying "Account already exists, login instead".	Pass	
			TC01.3	To verify that when a user enters an invalid email address then an error message should pop up saying "Enter a valid email address".	ELP website must be open.	1. Click on the register option. 2. Once the registration page is launched, enter an invalid email address. 3. Press enter	An error message should pop up saying "Enter a valid email address".	An error message pops up saying "Enter a valid email address".	Pass	
			TC02.1	To verify that when a user enters a registered email address and presses enter, next page should be launched where the user	ELP website must be open.	1. Click on the Login option. 2. Once the Login page is launched, enter a registered email address. 3. Press enter	Next page should be lunched where the user needs to enter the	Next page launched to enter the password.	Pass	

5.2

TestCaseId	Component	Priority	Test cases	Description/Test Summary	Pre-requisites	Test Steps	Expected Result	Actual Result	Status	Test E
02	Login	p10	TC02.1	To verify that when a user enters a registered email address and presses enter, next page should be launched where the user needs to enter the password.	ELP website must be open.	1. Click on the Login option. 2. Once the Login page is launched, enter a registered email address. 3. Press enter	Next page should be lunched where the user needs to enter the password.	Next page launched to enter the password.	Pass	
			TC02.2	To verify that when a user enters an unregistered email address and presses enter, an error message must pop up saying "No such account found, SignUp instead".	ELP website must be open.	1. Click on the Login option. 2. Once the Login page is launched, enter an unregistered email. 3. Press enter	An error message should pop up saying "No such account found, Login instead."	An error message pops up saying "No such account found, Login instead."	Pass	
			TC02.3	To verify that when a user enters an invalid email address and presses enter, an error message must pop up saying "Enter a valid email address".	ELP website must be open.	1. Click on the Login option. 2. Once the Login page is launched, enter an invalid email. 3. Press enter	An error message should pop up saying "Enter a valid email address."	An error message pops up saying "Enter a valid email address."	Pass	
03	Search_Bar_Module	P5	TC03.1	To verify that when a user writes a search term and presses enter, search results should be displayed.	ELP website must be open.	1. Write the write the name of the course or subject in the search bar. 2. Press enter	Search results related to the searched term should be displayed	Search results with 'searched term' keywords are displayed.	Pass	

6   Testing

6.1. Unit Testing

Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance.

A unit is a single testable part of a software system and tested during the development phase of the application software.

The purpose of unit testing is to test the correctness of isolated code. A unit component is an individual function or code of the application. White box testing approach used for unit testing and usually done by the developers.

Whenever the application is ready and given to the Test engineer, he/she will start checking every component of the module or module of the application independently or one by one, and this process is known as Unit testing or components testing.

Objective of Unit Testing:

The objective of Unit Testing is:

1. To isolate a section of code.

2. To verify the correctness of code.

3. To test every function and procedure.

4. To fix bug early in development cycle and to save costs.

5. To help the developers to understand the code base and enable them to make changes quickly.

6. To help for code reuse.

6.2 Integration Testing

Integration testing is done to test the modules/components when integrated to verify that they work as expected i.e., to test the modules which are working fine individually does not have issues when integrated.

When talking in terms of testing large application using black box testing technique, involves the combination of many modules which are tightly coupled with each other. We can apply the Integration testing technique concepts for testing these types of scenarios

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

Unit testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.

Integration Strategies:

1. Big-Bang Integration

2. Top-Down Integration

3. Bottom-Up Integration

4. Hybrid Integration

6.3 White Box Testing

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box is testing, structural testing, clear box testing, open box testing and transparent box testing. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

The term 'white box' is used because of the internal perspective of the system. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings.

Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer.

The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application

The white box testing contains various tests, which are as follows:

1. Path testing

2. Loop testing

3. Condition testing

4. Testing based on the memory perspective

5. Test performance of the program

6.4 Black Box Testing

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.

Generic steps of black box testing

1. The black box test is based on the specification of requirements, so it is examined in the beginning.

2. In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.

3. In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.

4. The fourth phase includes the execution of all test cases.

5. In the fifth step, the tester compares the expected output against the actual output.

6. In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

6.5   System Testing

System Testing includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software (any software is only a single element of a computer system). The software is developed in units and then interfaced with other software and hardware to create a complete computer system. In other words, a computer system consists of a group of software to perform the various tasks, but only software cannot perform the task; for that software must be interfaced with compatible hardware. System testing is a series of different type of tests with the purpose to exercise and examine the full working of an integrated software computer system against requirements

System testing falls under Black box testing as it includes testing of the external working of the software. Testing follows user's perspective to identify minor defects.

System Testing includes the following steps.

1. Verification of input functions of the application to test whether it is producing the expected output or not.

2. Testing of integrated software by including external peripherals to check the interaction of various components with each other.

3. Testing of the whole system for End-to-End testing.

4. Behaviour testing of the application via a user's experience

7. Bibliography

7.1 References

- https://in.indeed.com/
- https://www.naukri.com/
- https://www.linkedin.com/
- https://www.monsterindia.com/
- https://internshala.com/
- https://www.shine.com/
- https://www.firstnaukri.com/
- https://www.freshersworld.com/
- https://jobs.google.com/
- https://angel.co/