Acknowledgements

I would like to extend my sincere gratitude to my advisor, **Kailash Chandra** for his motivation, support and constant guidance throughout the development of this project. I truly appreciate the trust he has placed in me.

I am deeply grateful to my project guider **Kailash Chandra** for his help and support provided at every step of the project.

I would like to thank my parents and family for their unconditional love and support .Special thanks to my beloved friends for being there for me during my odds. Thank you for your strong belief in me.

# Durgesh Jaiswal

**Course: MCA\_NEW**

**Enroll: 2250958510**

Declaration

I, **Durgesh Jaiswal**, hear by declare that this project entitled “Online Job Portal” and the work descried in it is my own work, and am carried out in accordance with the regulations of the **Indira Gandhi National Open University**.

I declared that the work is completed with full dedication. I have taken special references of the various texts and no part of this project has been submitted for any other degree.

**Table of Contents**

1. Introduction.……………………………………………………………………………...03

1.1 Objective…………………………………………………………………………......04

1.2 Scope…………………………………………………………………………............04

1. System Analysis………………………………………………………………………….05

2.1 Identification of Need….……………………………………………………………..05

2.2 Project Planning and Project Scheduling…………………………………………….25

2.3 Software requirement specifications….……………………………………………...44

2.4 Software Engineering Paradigm applied……………………………………………..44

2.5 Data models & Flow diagrams…………………………………………………….....68

1. System Design……………………………………………………………...…………….80

3.1 Modularization details………………………………………………………………..80

3.2 Data integrity and constraints………………………………………………………...81

3.3 Database design, Procedural Design/Object Oriented Design……………………….83

3.4 User Interface Design………………………………………………………………...86

3.5 Test Cases…………………………………………………………………………….99

1. Coding…………………………………………………………………….....................103

4.1 SQL commands…………………………………………………………..…………103

4.2 Complete Project Coding…………………………………………………………...103

4.3 Comments and Description of Coding segments…………………………………...103

1. Standardization of the coding…………………………………………………………...172

5.1 Code Efficiency…………………………………………………………………......172

5.2 Error handling ……………………………………………………………………....174

1. Testing ………………………………………………………………………………….175

6.1 Testing techniques and Testing strategies used …………………………….............176

6.2 Testing Plan used……………………………...........................................................179

6.3 Test reports for Unit Test Cases and System Test Cases……………………….......185

6.4 Debugging ……………………….............................................................................190

1. System Security measures……………………………………………………..………..191

7.1 Database/data security……………………………………………………................191

7.2 Creation of User profiles and access rights………………………………………....192

1. Cost Estimation of Project………………………………………………………………193
2. Future scope and further enhancement of the project…………………………..………199
3. Bibliography…………………………………………………………………………….200
4. Glossary……………………………………………………………………………..…..201
5. Introduction

Online Job 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 to find individuals in the market. Online Job Portal best serve this purpose .E-recruitment has become the standard means for employers and jobseekers 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 advance mention technology and growth of internet usage, thee-recruitment has revolutionized the way organizations hire and candidates search for jobs. With the Online Job Portal, the recruitment process is speed educate 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. Using the portal jobseekers can extensively search for jobs in companies, organizations and regions they may otherwise have not learnt. In addition, candidates/Employers can write are view about an organization, which might help them to change the way things are done.

1.1) Objective

The objective of the web as well as android application is to provide flexibility to the jobseekers by providing the functionalities of both job search and job application in a single application. In addition, this application provides an effective means for the employers to post job vacancies and view the job applications by the interested applicants in a single application. Employers can also view the reviews provided by the jobseekers.

1.2) Scope

This project fulfills the primary requirements of the jobseekers 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 jobseeker, it can be extended to support functionalities of Employer as well.

1. System Analysis

System Analysis is the study of sets of interacting entities, including computers analysis. This field is closely related to requirements analysis or operations research. It is also an explicit formal enquiry carried out to help someone to identify a better course of action and make a better decision than he might otherwise have made. System analysis can also be defined as a series of components that perform organic function together.

2.1) Identification of Need

According to ‘Crawford’, information needs depend on…

* Work activity
* Discipline/ Field / Area of interest
* Availability of facilities
* Hierarchical position of individuals
* Motivation factors for information needs
* Need to take a decision
* Need to seek new ideas
* Need to validate the correct ones
* Need to make professional contributions
* Need to establish priority for discovery etc.

Information needs are affected by a variety of factors such as:

* The range of information sources available;
* The uses to which the information will be put;
* The background, motivation, professional orientation and other individual characteristics of the user;
* The social, political, economic, legal and regulatory systems surrounding the user;
* The consequences of information use.

Information needs identification is a complex process. Some of the factors adding to the complexity are…

* Different users perceive same information differently as their information need. (Value system of users differs due to the nature of work and it affects factors such as the information products and services).
* Researchers need original documents whereas planners need digests of `point of view ' / opinions);
* Information is put to different uses (R & D personnel, Application developers and Technicians all put information to different uses);
* Need is satisfied by having access to the identified information in a particular package and form, and at a suitable time;
* The flow of information and channels of communication are complex and add to the complexity; and
* Individual preferences and behavioral aspects add a further dimension.

Apart from the complexities mentioned above, there are problems due to individual behavior too:

* A user may not like to reveal his/her need as s/he may like to show that s/he is above (any) requirements and that s/he knows his/her sources. To divert or mislead s/he may give or just because someone else is getting information s/he may give a pseudo need to receive the same information. A super ordinate, to avoid a subordinate getting informed, may give it as a pseudo need and try to withhold the information received.

Further, an interaction of all the above factors makes the process of identifying information needs akin to carrying out an investigation or detective work, and is one of the 'grey areas' of research in system and Information Science for the past few decades.

So, in our application  **– Online Job Portal** the needs are as follows…

This application is basically an online or web based system that means Employer and Jobseeker will interact to each other. Employer can post multiple job opportunities for the jobseeker as per organization requirements. Jobseeker can apply for the job as per designation and eligibility.

Preliminary Investigation

The purpose of the preliminary investigation is to collect information for developing broad solutions for the purpose of feasibility study. Material i.e. information and facts to be collected in the preliminary investigation not only act as a basis for forming the several board solutions of proposed system but it also provides the much needed feedback for selection of the final candidates system among the solutions suggested in the course of feasibility study. Actually it means to be finding out the way that how the proposed web application will be developed containing which facts and figures.

**2.2.a) Django**

With Django, you can take Web applications from concept to launch in a matter of hours. Django takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

### 2.2.b) Ridiculously fast

Django was designed to help developers take applications from concept to completion as quickly as possible.

### 2.2.c) Fully loaded

Django includes dozens of extras you can use to handle common Web development tasks. Django takes care of user authentication, content administration, sitemaps, RSS feeds, and many more tasks—right out of the box.

### 2.2.d) Reassuringly secure

Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-sites cripting,c rosés-site request for garland click jacking. Its user authentication system provides a secure way to manage user accounts and passwords.

### 2.2.e) Exceedingly scalable

Some of the busiest sites on the planet use Django’s ability to quickly and flexibly scale to meet the heaviest traffic demands.

### 2.2.f) Incredibly versatile

Companies, organizations and governments have used Django to build all sorts of things — from content management systems to social networks to scientific computing platforms.

# 2.2.g) Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation far betray expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

# 2.2.h) Django Template Language

This document explains the language syntax of the Django template system. If you’re looking for a more technical perspective on how it works and how to extend it, see The Django template language: for Python programmers.

Django’s template language is designed to strike a balance between power and ease. It’s designed to feel comfortable to those used to working with HTML. If you have any exposure too their text-based template languages, such as Smarty or Ninja2, you should feel right at home with Django’s templates.

* A template a text file. It can generate any text-based format (HTML,XML,CSV, etc.).
* A template contains variables, which get replaced with values when the template is evaluated, and tags, which control the logic of the template.
* Below is a minimal template that illustrates a few basics.

The data collected during preliminary investigations have been gathered through some primary methods. These are as follows:-

1. **Gathering Data**

Choose data-gathering technique based on the project's needs. Such techniques include in-person interviews, questionnaires sent via regular mail, phone interviews or use of the Internet. Keep in mind that different cultures and subcultures respond differently to various techniques, and that we should not limit ourselves to one technique if combining two or three methods provides optimal results.

* In-person interviews. Researchers who conduct in-person interviews can benefit by:
* Receiving high contact and response rates
* Conducting long, in-depth interviews
* Supplementing participants' responses with observations of them
* Allowing respondents a chance to interact with stimuli (e.g., pictures, products)
* At the same time, in-person interviews have their disadvantages. Such interviews:
* Can take weeks to complete
* Are expensive (Rs100 to Rs200-plus per interview)
* Fail to contact gated communities
* Can be skewed by possible interviewer bias through expressions, comments, etc.
* Mail. Like in-person interviews, questionnaires sent via regular mail provide participants the chance to submit thoughtful responses, since they have time to look up information and reflect before responding, plus there is a high contact rate. Mailing out questionnaires is effective for surveying homogeneous groups, and there is relatively low cost involved with surveying a wide geographic area.

This technique should not be used for general population studies, however. Some people cannot read or write well, so their responses can be unclear or skewed by misinterpretation of the question. Also, mailing questionnaires offers low return rates and can take weeks to complete. Hence, a company should entice potential participants by providing:

* An immediate prize (e.g., a small gift sticker, etc., included with questionnaire)
* A future prize (e.g., collectible gift, stamp, etc., sent after participants return questionnaires)
* A chance to win something (e.g., entering name in raffle drawing)
* Additionally, the company should explain the value of the study and send a letter before the study announcing it and a reminder after potential participants receive the questionnaire with a set deadline.

**2. Phone**

While mailing questionnaires works when targeting a homogeneous group, a phone interview is the most popular method for researching general populations. This method boasts a good response rate, is effective in reaching most people and is quick and inexpensive. However, phone interviews are typically limited to 8 to 10 minutes, which demands that participants give short, top-of-mind responses. Phone interviews also limit whom you contact, since most people have answering machines or caller ID, or they mistake a researcher for a telemarketer.

**3. Database Generators**

For a more accurate depiction of continuing behavior, you may want to merge existing information the company has about its customers' lifestyles with information obtained by tracking how often they use checks, credit cards, discount cards, etc. Likewise, employee databases provide a way to assess productivity by tracking their sales volume and expenses, for instance. Also, you can combine employment statistics (e.g., length of employment, job classification) with behavioral statistics (e.g., use of in-house communication, e-mail, Internet) to develop communication. Internet tracking. Internet tracking that records how often a web application is "hit" can be skewed by search engines that count a hit even if a person does not see the page. Also, a hit does not indicate that a person paid attention or found the information useful. To address this, install a cookie on the computer that keeps track of an audience's behavior while on the web application. Internet and e-mail. Using the Internet and e-mail to gather information is inexpensive and quick, and offers immediate analysis of perhaps thousands of responses. This technique, useful for surveying within an organization, is not suggested for general population studies, as it is limited to those with Internet access. Setting up such a complex database can prove costly, and privacy concerns arise. Researchers who use this method encounter similar problems as those who use regular mail. Additionally, we should consider the following when conducting an Internet survey:

* How to control who gets a questionnaire
* How to prevent the same person from responding twice
* Whether all computers format the questionnaire correctly

**4. Developing A Questionnaire**

* When designing a questionnaire, we always keep in mind:
* Questions and response categories embody our variables.
* Response categories quantify variables for analysis.
* Standardizing how data is collected reduces bias and allows the study to be replicated.

Keep in mind also that a general population study is usually limited to an eight-minute phone call or four-page mail.

**a. Types of questions.**

Researchers typically ask questions leading to responses that are fixed or open-ended. Fixed responses, used when you know the likely answers, are easy to code and analyze. Consider open-ended questions when you are uncertain of the response. Such questions should be used sparingly because they take time away from other short questions, and answers must be categorized. These questions should be written with "probes" (e.g., "Are there any other issues?") that allow respondents to elaborate.

**b. Rules of thumb.**

These rules will make questionnaires more efficient and less biased. Sometimes there is a good reason for breaking these rules, but for the most part, we should:

* Keep it short, simple and to the point
* Write clearly in the language of the target population
* Keep a logical order for asking questions
* Save sensitive items for the end
* Avoid leading/loaded questions or combining multiple questions into one

**c. Pre-test.**

Have people fill out the questionnaire or answer on the phone, and ask them at the end if they found anything difficult to answer. Make all necessary adjustments, and pretest the questionnaire again.

After the preliminary investigation we try to make an outline of the database of the proposed system. Entity- Relationship diagram gives a picture of the entities to be used, their relationship with each other and their attributes. Actually the study phase is considered as the requirement analysis, which is the most vital part of the software development life cycle (SDLC). Without proper analysis something will remain hidden from the developer may generate certain fatality in near future even in after the implementation, too. Analysis is a software engineering task that bridges the gap between system level requirements engineering and system design.

Feasibility Study

A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats present in the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained.

A well-designed feasibility study should provide a historical background of the business or project, a description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, feasibility studies precede technical development and project implementation.

A feasibility study evaluates the project’s potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions. It must, therefore, be conducted with an objective, unbiased approach to provide information upon which decisions can be based.

Investment proposals, involving huge capital outlay are invariably irreversible. Therefore, before starting a project/proposal, it is necessary and imperative to find out whether the same is feasible or not.

A feasibility study is a preliminary study undertaken to determine and document a project's viability. Also known as feasibility analysis. The term **feasibility study** is also used to refer to the resulting document. The results of this study are used to make a decision whether or not to proceed with the project. If it indeed leads to a project being approved, it will — before the real work of the proposed project starts — be used to ascertain the likelihood of the project's success. It is an analysis of possible alternative solutions to a problem and a recommendation on the best *alternative*. It, for example, can decide whether an order processing be carried out by a new system more efficiently than the previous one. A **feasibility study** is an important part of creating a business plan for a new enterprise, since it has been estimated that only one idea in fifty is commercially viable. If a project is seen to be feasible from the results of the study, the next logical step is to proceed with it. The research and information uncovered in the feasibility study will support the detailed planning and reduce the research time.

**Feasibility study** is one of the most important factors to undergo a process of software development, it is considered as one of the first step of SDLC (i.e. Software Development Life-Cycle). There remain various kinds of feasibility studies, but most important of them to be required actually for a software development is discussed below: -

This mainly occurred to make the system feasible. Mainly I am interested about the following feasibilities, such as :-

1. Technical Feasibility
2. Economic Feasibility
3. Operational Feasibility
4. Social Feasibility
5. Management Feasibility
6. Time Feasibility
7. Legal Feasibility

**Technical Feasibility:-**

This is concern with specifying equipment and software that will successfully satisfied the user requirement. The technical needs of the system may very considerably but might include some feature such as-

* The facility to produce output in a given time.
* Response time under certain conditions.
* Ability to process a certain volume of transudation at a particular speed.
* Facility to communicate data to distant location.

In our system the technical feasibility is mostly emphasis on the above points because in the rush hour it will provide the speed and accuracy to the user of the system. It gives the idea to smooth operation of the system and communicate between them.

**Economic Feasibility:-**

It is the most important now a days the economic performance of software that how much cost cutting provided by the software, how much it cost effective. From the point of view of my software it is give more than ninety percent of cost benefit because it provide stationary less office and account also partly maintain by my system itself.

Economic analysis is the most frequently used method for evaluating the effectiveness of a candidate system. More commonly known as analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system.

**Operational Feasibility:-**

It is mainly related to human organizational and political aspects. The point to considered are:

What changes will be brought with the system, what organizational structures are distributed, what new skills are required, do the existing staff members have these skills? These types of questions are executed in this feasibility. To solve this types of question my system is implemented such a way that no high skill personal are not required, only the man who know some English language and has data entry knowledge who can drive this system very effectively and existing staffs are enough for my system.

Operational feasibility is dependent upon the human resources available for the project. It also concerns I self with whether or not the system will operate and work once it is implemented. Do the current work practices and procedures support a new system? Also social factors i.e. how the organizational changes will affect the working lives of those affected by the system.

**Social Feasibility: -**

Social feasibility is a determination of whether a proposed project will be acceptable to the people or not. I think this project highly appreciable by the user because it helps them lots and also take some thanks from the tourist who are also want to gather the information about the spot they want to visit.

**Management Feasibility: -**

Management feasibility is a determination of whether a proposed project will be acceptable to management. This project get the support from the manager of some renowned travel company they appreciate me to doing this project and give moral support also.

**Time Feasibility: -**

Time feasibility is a determination of whether a proposed project can be implemented fully within a stipulated time frame. This project is a complete project within stipulated time but if I get more time I can make this project more advanced one.

**Legal Feasibility: -**

Legal feasibility is determined with legal aspects this project is done under supervision of my project guide no rule violation is occurred in this project.

Determines whether the proposed system conflicts with legal requirements, e.g. a Data Processing system must comply with the local Data Protection Acts. When an organization has either internal or external legal counsel, such reviews are typically standard. However, a project may face legal issues after completion if this factor is not considered at this stage.

For the system maintenance the all reasons are required. In this system all the above topics is considered then it is need worry about the systems feasibility. In this organization all the person are computer literate through corporate training, so there no questions to unused the system by improper handling. From the economic point of view there is no such investment on money such as computer, printer, operator etc. All of them are computer literate so any one can handle the machine so the recurring cost of operation minimized. The installation of machine does need not a factor because it is the computer related field. From the technical point of view, if it is necessary to change certain criteria or print the different format there should have one programmer to prepare the report format.

The system when required to update for reporting and other application it can be done without interrupting the existing software. The system is so handy that can be transform to all the personal computer. The data security is maintained by copy the system in floppy disk. The regular backup should be done to avoid the system crush.

**Cost-Benefit Analysis…**

**Cost-Benefit Analysis (CBA)** estimates and totals up the equivalent money value of the benefits and costs to the area of projects to establish whether they are sensible.

**Principles of Cost Benefit Analysis:-**

One of the problems of CBA is that the computation of many components of benefits and costs is intuitively obvious but that there are others for which intuition fails to suggest methods of measurement. Therefore some basic principles are needed as a guide.

**There Must Be a Common Unit of Measurement:-**

In order to reach a conclusion as to the desirability of a project all aspects of the project, positive and negative, must be expressed in terms of a common unit; i.e., there must be a "bottom line." The most convenient common unit is money. This means that all benefits and costs of a project should be measured in terms of their equivalent money value. A program may provide benefits which are not directly expressed in terms of Rupee but there is some amount of money the recipients of the benefits would consider just as good as the project's benefits. For example, a project may provide for the aged in an area a free monthly visit to a doctor. The value of that benefit to an elderly recipient is the minimum amount of money that that recipient would take instead of the medical care. This could be less than the market value of the medical care provided. It is assumed that more esoteric benefits such as from preserving open space or historic sites have a finite equivalent money value to the public.

Not only do the benefits and costs of a project have to be expressed in terms of equivalent money value, but they have to be expressed in terms of rupee of a particular time. This is not just due to the differences in the value of rupee at different times because of inflation. A rupee available five years from now is not as good as a rupee available now. This is because a rupee available now can be invested and earn interest for five years and would be worth more than a rupee in five years. If the interest rate is r then a rupee invested for t years will grow to be (1+r)t. Therefore the amount of money that would have to be deposited now so that it would grow to be one rupee t years in the future is (1+r)-t. This called the discounted value or present value of a rupee available t years in the future.

When the rupee value of benefits at some time in the future is multiplied by the discounted value of one rupee at that time in the future the result is discounted present value of that benefit of the project. The same thing applies to costs. The net benefit of the projects is just the sum of the present value of the benefits less the present value of the costs.

The choice of the appropriate interest rate to use for the discounting is a separate issue that will be treated later in this paper.

The procedure is to determine the benefits and saving that are expected from a candidate system and compare them with costs. If the benefits outweigh costs, then decision is made to design and implement the system. Considering the facts it is become evident that the system will be economically feasible both for developer as well as for client’s respect.

Technical feasibility centers on the existing computer system (hardware, software, etc.)And to what level it can support the proposed addition. If the budget is a serious constraint, then the project is judged not feasible. Here it is not a constraint altogether.

Now, let us discuss briefly, what are there actual jobs are? Different related sites are studied thoroughly with all their different facilities and systems. The discussions are done with the associates to get the proper requirements for conduction of such a system. The overall system is the most hectic one and watched properly how it is operated actually and then the process of site development is going on.

In this fast changing world, competition has taken a huge toll over millions. In this competitive world, everyone is seeking for a better lifestyle and a source of livelihood. As a result, communications between the nearer and dearer ones seems to have deteriorated to a very large extent. The same could have resulted due to two primary reasons:

* Monetary Consideration
* Lack of cheap communication awareness.

The very serious concern needs to be purged and as a result it has became necessary to implement an automated communication system which will efficiently handle each process of the interaction and will maximize the communication utilization and minimize the unavoidable human constraints thereby increasing the notion of “Stay Connected”.

So in above discussion we can say regarding this project Basic structure of CBA is given below: -

**1. Software Costs**

Relates to the buying of the software required to develop the project as well as the software required to run the application in the organization.

**2. Facility Costs**

Are expenses incurred in the preparation of the physical site in organization where the application or the computer will be in operation? This includes wiring, flooring, acoustics, lighting and air conditioning. These costs have been treated as one-time costs.

**3. Operating Costs**

Include all cost associated with the day-to-day operation of the system. The amount depends on the number of shifts, the nature of the applications and the caliber of the operating staff.

**4. Supply Costs**

Are variable costs that increase with increase use of papers, ribbons, disk, and the like. They have been estimated and included in the overall cost of the system.

The management & operators desire to be well acquainted with the requisite skill needed.

**N**ow the preliminary investigation is carried out as follows:

**i. Interviewing: -**

**T**he people who will use the system (also called **end-users**) are interviewed thoroughly to find out what they actually want from the system to serve the customers better.

**ii. Group Communication: -**

After the interview session, group discussion is conducted. Here, the overall functionality of the system is discussed within the group.

**iii. Record Review: -**

**T**he current records of the existing system, which are manually handled, are reviewed to take stake how these records would appear in the automated system.

**Market Feasibility:-**

Market Feasibility Study typically involves testing geographic locations for a real estate development project, and usually involves parcels of real estate land. Developers often conduct market studies to determine the best location within a jurisdiction, and to test alternative land uses for a given parcel. Jurisdictions often require developers to complete feasibility studies before they will approve a permit application for retail, commercial, industrial, manufacturing, housing, office or mixed-use project. Market Feasibility take into account the importance of the business in the selected area.

**Technology and System Feasibility:-**

This involves questions such as whether the technology needed for the system exists, how difficult it will be to build, and whether the firm has enough experience using that technology. The assessment is based on an outline design no system requirements in terms of Input, Processes, Output, Fields, Programs, and Procedures. This can be quantified in term so volume so data, trends, frequency of updating, etc in order to estimate if the new system will perform adequately or not.

**Resource Feasibility:-**

This involves questions such as how much time is available to build the new system, when it can be built, whether it interferes with normal business operations, type and amount of resources required, dependencies, etc.

**Cultural Feasibility:-**

In this stage, the project's alternatives are evaluated for their impact on the local and general culture. For example, environmental factors need to be considered and these factors are to be well known. Further an enterprise's own culture can clash with the results of the project.

**Schedule Feasibility:-**

A project will fail if it takes too long to be completed before it is useful. Typically, this means estimating how long the system will take to develop, and if it can be completed in a given time period using some methods like pay back period.

2.2) Project Planning and Scheduling

**Project Planning:-**

Planning phase, plan is made and strategies are set, taking into consideration the company policies, procedures and rules

Project Planning is an aspect of Project Management that focuses a lot on Project Integration. The project plan reflects the current status of all project activities and is used to monitor and control the project.

The Project Planning tasks ensure that various elements of the Project are coordinated and therefore guide the project execution.

Planning provides direction, unifying frame work, performance standards, and helps to reveal future opportunities and threats

In Planning, the following steps are followed.

* The objectives of the projects in definite words
* Goals and stages intermediate to attain the final target
* Forecast and means of achieving goals i.e., activities.
* Organization resources-financial, managerial and operational-to carry out activities and to determine what is feasible and what is not.
* Alternatives-individual courses of action that will allow accomplishing goals.
* For consistency with company’s policies.
* An alternative which is not only consistent with its goals and concept but also one that can be accomplished with the evaluated resources.
* Decision on a Plan

**Project Planning helps in …**

* Facilitating communication
* Monitoring/measuring the project progress, and
* Provides overall documentation of assumptions/planning decisions

**The Project Planning Phases can be broadly classified as follows:-**

* Development of the Project Plan
* Execution of the Project Plan
* Change Control and Corrective Actions

Project Planning is an ongoing effort throughout the Project Lifecycle.

Project planning is a discipline for stating how to complete a project within a certain timeframe, usually with defined stages, and with designated resources. One view of project planning divides the activity into:

* Setting objectives (these should be measurable)
* Identifying deliverables
* Planning the schedule
* Making supporting plans

Supporting plans may include those related to: human resources, communication methods, and [risk management](http://searchcio.techtarget.com/definition/enterprise-risk-management).

Computer hardware and software project planning within an [enterprise](http://searchwinit.techtarget.com/definition/enterprise) is often done using a project planning guide that describes the process that the enterprise feels has been successful in the past.

Tools popularly used for the scheduling part of a plan include the [Gantt chart](http://searchsoftwarequality.techtarget.com/definition/Gantt-chart) and the [PERT chart](http://searchsoftwarequality.techtarget.com/definition/PERT-chart).

**Steps in Planning:-**

Project Planning spans across the various aspects of the Project. Generally Project Planning is considered to be a process of estimating, scheduling and assigning the projects resources in order to deliver an end product of suitable quality. However it is much more as it can assume a very strategic role, which can determine the very success of the project. A Project Plan is one of the crucial steps in Project Planning in General!

Typically Project Planning can include the following types of project Planning:

**Step 1: Explain the project plan to key stakeholders and discuss its key components**

"Project plan" is one of the most misunderstood terms in project management. It is a set of living documents that can be expected to change over the life of the project. Like a road map, it provides the direction for the project.

And like the traveler, the project manager needs to set the course for the project. Just as a driver may encounter road works or new routes to the final destination, the project manager may need to correct the project course.

A common misconception is that the plan equates to the project timeline - that is only one of the components of the plan. The project plan is the major work product from the entire planning process, so it contains all the planning documents.

For example, a project plan for constructing a new office building needs to include not only the specifications for the building, the budget and the schedule, but also the risks, quality metrics, environmental impact, etc.

Components of the project plan include:

* **Baselines:** These are sometimes called performance measures because the performance of the entire project is measured against them. They are the project's three approved starting points for scope, schedule and cost. These are used to determine whether or not the project is on track during execution
* **Baseline management plans:** These include documentation about how variances will be handled throughout the project
* Other work products from the planning process, which include plans for risk management, quality, procurement, staffing and communications.

**Step 2: Define roles and responsibilities:-**

Identifying stakeholders - those who have a vested interest in either the project or its outcome - is challenging and especially difficult on large, risky, high-impact projects. There are likely to be conflicting agendas and requirements among stakeholders, as well as different slants on who needs to be included.

For example, the stakeholder list of the city council where a new office building is being constructed could differ from that of an engineering consulting firm. It would certainly include the developer who wants to build the complex, the engineering firm that will build it, citizens who would prefer a park, consultants to study the environmental impact, the city council itself, etc.

The engineering firm may have a more limited view. It is important for the project manager to get clarity and agreement on what work needs to be done by whom, as well as which decisions each stakeholder will make.

**Step 3: Develop a scope statement:-**

The scope statement is arguably the most important document in the project plan. It is used to get common agreement among the stakeholders about the project definition.

It is the basis for getting the buy-in and agreement from the sponsor and other stakeholders and decreases the chances of miscommunication.

This document will most likely grow and change with the life of the project. The scope statement should include:

* Business need and business problem
* Project objectives, stating what will occur within the project to solve the business problem
* Benefits of completing the project, as well as the project justification
* Project scope, stated as which deliverables will be included or excluded from the project
* Key milestones, the approach and other components as dictated by the size and nature of the project

It can be treated like a contract between the project manager and sponsor - one that can only be changed with sponsor approval.

**Step 4: Develop the project baselines**:-

**Scope baseline-** Once the deliverables are confirmed in the scope statement, they need to be developed into a work breakdown structure of all the deliverables in the project.

The scope baseline includes all the deliverables produced on the project, and therefore identifies all the work to be done. These deliverables should be inclusive.

Building an office building, for example, would include a variety of deliverables related to the building itself, as well as such things as impact studies, recommendations, landscaping plans, etc.

* Schedule and cost baselines
* Identify activities and tasks needed to produce each of the deliverables identified in the scope baseline. How detailed the task list needs to be depends on many factors, including the experience of the team, project risk and uncertainties, ambiguity of specifications, amount of buy-in expected, etc
* Identify resources for each task, if known
* Estimate how many hours it will take to complete each task
* Estimate cost of each task, using an average hourly rate for each resource
* Consider resource constraints, or how much time each resource can realistically devote to this one project
* Determine which tasks are dependent on other tasks, and develop critical path
* Develop schedule, which puts all tasks and estimates in a calendar. It shows by chosen time period (week, month, quarter or year) which resource is doing which tasks, how much time each task is expected to take, and when each task is scheduled to begin and end
* Develop the cost baseline, which is a time-phased budget, or cost-by-time period

This process is not a one-time effort. Throughout the project, you will most likely be adding to and repeating some or all of these steps.

**Step 5: Create baseline management plans:-**

Once the scope, schedule and cost baselines have been established, create the steps the team will take to manage variances to these plans.

All these management plans usually include a review and approval process for modifying the baselines. Different approval levels are usually needed for different types of changes.

Not all new requests will result in changes to the scope, schedule or budget, but a process is needed to study all new requests to determine their impact on the project.

**Step 6: Communicate:-**

One important aspect of the project plan is the communications plan. This document states such things as:

* Who wants which reports, how often, in what format and using what media
* How issues will be escalated and when
* Where project information will be stored and who can access it
* What new risks have surfaced and what the risk response will include
* What metrics will be used to ensure a quality product is built
* What reserves have been used for which uncertainties

Once the project plan is complete, it is important that its contents be delivered to key stakeholders. This communication should include such things as:

* Review and approval of the project plan
* Process for changing the contents of the plan
* Next steps - executing and controlling the project plan and key stakeholder roles/responsibilities

Developing a clear project plan takes time. The project manager will probably be tempted to skip the planning and jump straight into execution.

However, the traveler who plans the route before beginning a journey ultimately reaches the intended destination more quickly and more easily than the disorganized traveler, who often gets lost along the way.

Similarly, the project manager who takes time to create a clear project plan will follow a more direct route to project success.

**1) Project Scope Definition and Scope Planning:** -

In this step we document the project work that would help us achieve the project goal. We document the assumptions, constraints, user expectations, Business Requirements, Technical requirements, project deliverables, project objectives and everything that defines the final product requirements. This is the foundation for a successful project completion.

**2) Project Activity Definition and Activity Sequencing**: -

In this step we define all the specific activities that must be performed to deliver the product by producing the various product deliverables. The Project Activity sequencing identifies the interdependence of all the activities defined.

**3) Time, Effort and Resource Estimation: -**

Once the Scope, Activities and Activity interdependence is clearly defined and documented, the next crucial step is to determine the effort required to complete each of the activities. The Effort can be calculated using one of the many techniques available such as Function Points, Lines of Code, Complexity of Code, Benchmarks, etc. This step clearly estimates and documents the time, effort and resource required for each activity.

**4) Risk Factors Identification: -**

“Expecting the unexpected and facing it” .It is important to identify and document the risk factors associated with the project based on the assumptions, constraints, user expectations, specific circumstances, etc.

**5) Cost Estimation and Budgeting:-**

Based on the information collected in all the previous steps it is possible to estimate the cost involved in executing and implementing the project. A Cost Benefit Analysis can be arrived at for the project. Based on the Cost Estimates Budget allocation is done for the project.

**6) Organizational and Resource Planning:-**

Based on the activities identified, schedule and budget allocation resource types and resources are identified. One of the primary goals of Resource planning is to ensure that the project is run efficiently. This can only be achieved by keeping all the project resources fully utilized as possible. The success depends on the accuracy in predicting the resource demands that will be placed on the project. Resource planning is an iterative process and necessary to optimize the use of resources throughout the project life cycle thus making the project execution more efficient. There are various types of resources – Equipment, Personnel, Facilities, Money, etc.

**7) Schedule Development:-**

The time schedule for the project can be arrived at based on the activities, interdependence and effort required for each of them. The schedule may influence the cost estimates, the cost benefit analysis and so on.

Scheduling is the allocation of resources, Resources in conceptual sense are time & energy but in practical sense are the time, man power, equipment applied to material.

Scheduling is the process of formalizing the planned functions, assigning the starting and completion dates to each activity which proceeds in a logical sequence and in an orderly and systematic manner.

In Scheduling, the following steps are followed.

* Detailed control information is to be calculated.
* Timings to events & activities are assigned
* Consideration must be given to resources generally concerned with those resources whose availability is limited and which there by impose a constraint on the project. Important ones are skilled, technical and supervisory manpower and capital investment
* Resource Allocation

Project Scheduling is one of the most important task of Project Planning and also the most difficult tasks. In very large projects it is possible that several teams work on developing the project. They may work on it in parallel. However their work may be interdependent.

Again various factors may impact in successfully scheduling a project

* Teams not directly under our control
* Resources with not enough experience

**8) Quality Planning: -**

The relevant quality standards are determined for the project. This is an important aspect of Project Planning. Based on the inputs captured in the previous steps such as the Project Scope, Requirements, deliverables, etc. various factors influencing the quality of the final product are determined. The processes required to deliver the Product as promised and as per the standards are defined.

**9) Risk Management Planning: -**

Risk Management is a process of identifying, analyzing and responding to a risk. Based on the Risk factors Identified a Risk resolution Plan is created. The plan analyses each of the risk factors and their impact on the project. The possible responses for each of them can be planned. Throughout the lifetime of the project these risk factors are monitored and acted upon as necessary.

**10) Project Plan Development and Execution: -**

Project Plan Development uses the inputs gathered from all the other planning processes such as Scope definition, Activity identification, Activity sequencing, Quality Management Planning, etc. A detailed Work Break down structure comprising of all the activities identified is used. The tasks are scheduled based on the inputs captured in the steps previously described. The Project Plan documents all the assumptions, activities, schedule, and timelines and drives the project.

Each of the Project tasks and activities are periodically monitored. The team and the stakeholders are informed of the progress. This serves as an excellent communication mechanism. Any delays are analyzed and the project plan may be adjusted accordingly.

**11) Performance Reporting:-**

As described above the progress of each of the tasks/activities described in the Project plan is monitored. The progress is compared with the schedule and timelines documented in the Project Plan. Various techniques are used to measure and report the project performance such as EVM (Earned Value Management) A wide variety of tools can be used to report the performance of the project such as PERT Charts, GANTT charts, etc.

**12) Planning Change Management: -**

Analysis of project performance can necessitate that certain aspects of the project be changed. The Requests for Changes need to be analyzed carefully and its impact on the project should be studied. Considering all these aspects the Project Plan may be modified to accommodate this request for Change.

Change Management is also necessary to accommodate the implementation of the project currently under development in the production environment. When the new product is implemented in the production environment it should not negatively impact the environment or the performance of other applications sharing the same hosting environment.

**13) Project Rollout Planning:-**

In Enterprise environments, the success of the Project depends a great deal on the success of its rollout and implementations. Whenever a Project is rolled out it may affect the technical systems, business systems and sometimes even the way business is run. For an application to be successfully implemented not only the technical environment should be ready but the users should accept it and use it effectively. For this to happen the users may need to be trained on the new system. All this requires planning.

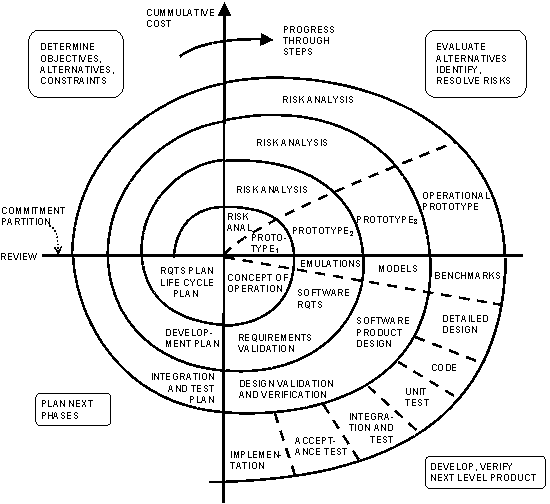
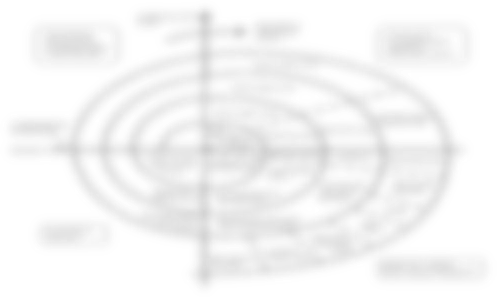
**Considering** all those things, it can be decided that to develop such type of system the **“Spiral Model”**-proposed by Boehm is the most suitable one. It consists the iterative nature of prototyping model and it is the rapid development of incremental version of software. Using spiral model, software is developed in a series of incremental releases. It is a realistic approach and suitable for development of large-scale system and can be used a risk reduction mechanism and enables the developer to apply prototyping approach at new stage in the evolution of the product.

A spiral model is divided into a number of framework activities, also called task region, typically there are 3-6 tasks regions.

* Tasks required establishing effective communication between developer and customer,
* Tasks required defining resources, timelines and other projects.
* Tasks required to build one or more representations of the applications.
* Tasks required constructing, testing, installing, and providing user support (e.g. documentation and training).
* Tasks required obtaining customer feedback based on evaluation of the software representations created during the engineering stage and implemented during the installation stage.

Each of the regions is populated by a set of work tasks, called a task set, that re adapted to the characteristics of the project to be undertaken. For small projects the number of work tasks and their formality is low. For larger, more critical projects each task region contains more work tasks that are defined to achieve a higher level of formality. This evolutionary process begins, the software engineering team moves around the spiral in a clockwise direction, beginning at the center. The first circuit around the spiral might results in the development of a product specification; subsequent passes around the spiral might be used to develop a prototype and then progressively more sophisticated versions of the software. Each pass thorough the planning region results in adjustment to the project plan. Cost and schedules are adjusted based on the feedback derived from customer evaluation. In addition the project manager adjusts the planned number of iterations required to complete the software.

### Fig: Spiral Life Cycle Model of Software Development



**Project Scheduling:-**

In project management, a **schedule** consists of a list of a project's terminal elements with intended start and finish dates. Terminal elements are the lowest element in a schedule, which is not further subdivided. Those items are often estimated in terms of resource requirements, budget and duration, linked by dependencies and scheduled.

By using following Table below describes the project’s activities and expected durations the system will be developed and installed by computer programmers, systems analysts, and personnel from the different function. The organization can hired sufficient programmers and has sufficient accountants on staff, but the systems analysts who mediate between the ‘tetchiest’ and the staffs are in short supply. Currently, only three systems analysts are available to the firm, and the number of analysts needed for each activity is listed in the table. The table also specifies immediate predecessors, the smallest possible list of tasks, which must be completed before starting each activity.

Scheduling is the allocation of resources, Resources in conceptual sense are time & energy but in practical sense are the time, man power, equipment applied to material.

Scheduling is the process of formalizing the planned functions, assigning the starting and completion dates to each activity which proceeds in a logical sequence and in an orderly and systematic manner.

In Scheduling, the following steps are followed.

* Detailed control information is to be calculated.
* Timings to events & activities are assigned
* Consideration must be given to resources generally concerned with those resources whose availability is limited and which there by impose a constraint on the project. Important ones are skilled, technical and supervisory manpower and capital investment
* Resource Allocation

Project Scheduling is one of the most important task of Project Planning and also the most difficult tasks. In very large projects it is possible that several teams work on developing the project. They may work on it in parallel. However their work may be interdependent.

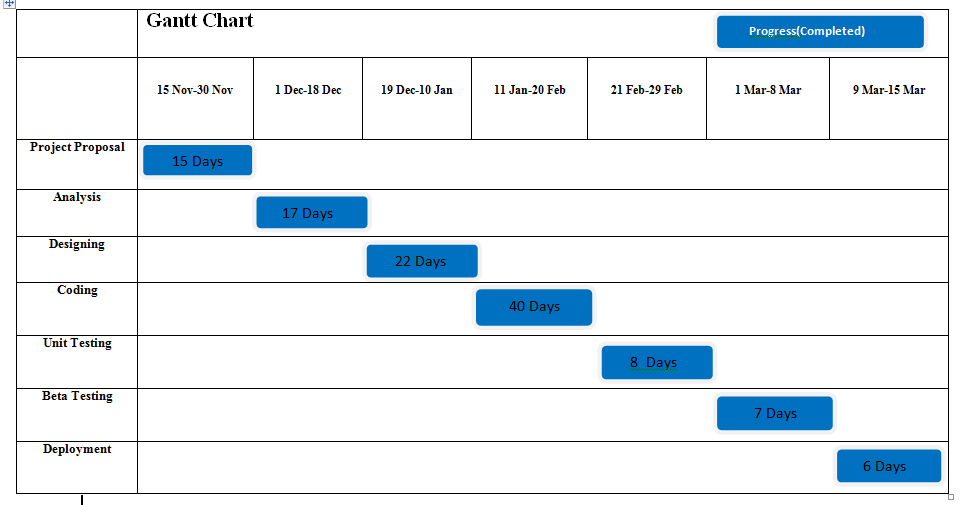
Again various factors may impact in successfully scheduling a project

* Teams not directly under our control
* Resources with not enough experience

**Gantt Chart :**

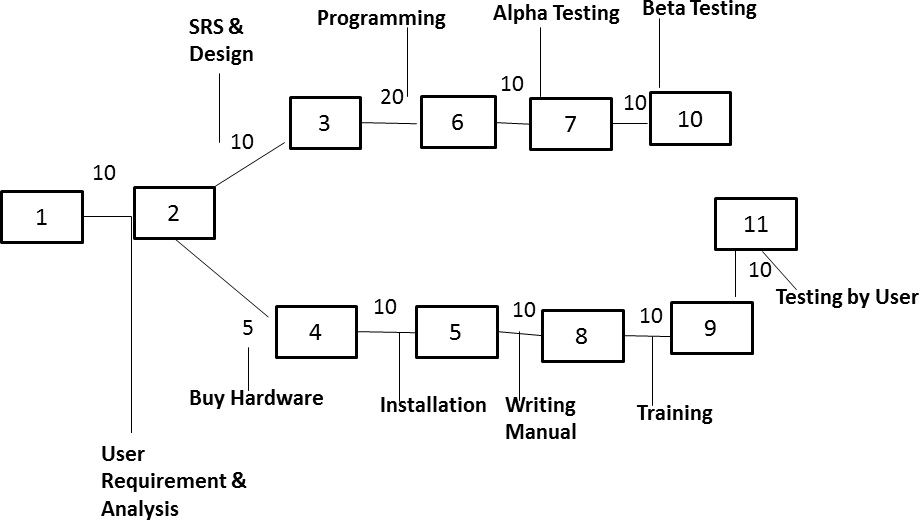
Gantt chart is also known as Time Line Charts. A Gantt chart can be developed for the entire project or a separate chart can be developed for each function. A Gantt chart can be developed for the entire project. All project tasks are listed in the left hand column. A tabular form is maintained where rows indicate the tasks with milestones and columns indicate duration (weeks/months). The horizontal bars indicate the duration of each task.

A Gantt chart is constructed with a horizontal axis representing the total time span of the project, broken down into increments (for example, days, weeks, or months) and a vertical axis representing the tasks that make up the project.



**Pert Chart :**

A PERT chart is a project management tool that provides a graphical representation of a project's timeline. The Program Evaluation Review Technique (PERT) breaks down the individual tasks of a project for analysis. PERT charts are considered preferable to Gantt charts because they identify task dependencies, but they're often more difficult to interpret.



2.3) Software Requirement Specification

**2.3.a) Software Requirements**

* Operating System: **Windows10**
* IDE: **Pycharm**
* Application Server:: **Django Web Server**
* Frameworks and APIs: **Django Web Framework**
* Database: **SQLiteDatabase**
* Front End: **HTML5,CSS3,JavaScript,jQuery**
* Browser: **Chrome or Firefox or Internet Explorer**

**2.3.b) Hardware Requirements**

* Processor: **Intelcorei7**
* Processor speed: **3.40 GHz**
* RAM: **8 GB**

2.4) Software Engineering Paradigm Applied

Software engineering is the application of principles used in the field of engineering, which usually deals with physical systems, to the design, development, testing, deployment and management of [software](https://searchapparchitecture.techtarget.com/definition/software) systems.

The field of software engineering applies the disciplined, structured approach to programming that is used in engineering to software development with the stated goal of improving the quality, time and budget efficiency, along with the assurance of structured testing and engineer [certification.](https://whatis.techtarget.com/definition/certification)

Software engineering is typically used for large and intricate software systems rather than single [applications](https://searchsoftwarequality.techtarget.com/definition/application) or [programs.](https://searchsoftwarequality.techtarget.com/definition/program) Development, however, is simply one phase of the process. While a software engineer is typically responsible for the design of systems, programmers are often responsible for [coding](https://whatis.techtarget.com/definition/code) its implementation.

Software engineering involves a number of fields that cover the process of engineering software and certification including: requirements gathering, software design, software construction, software maintenance, software configuration management, software engineering management, software development process management and creation, software engineering models and methods, software quality, software engineering professional practices as well as foundational computing and mathematical and engineering study.

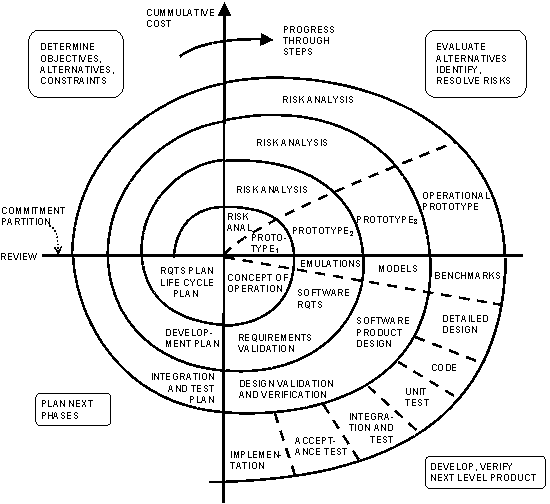
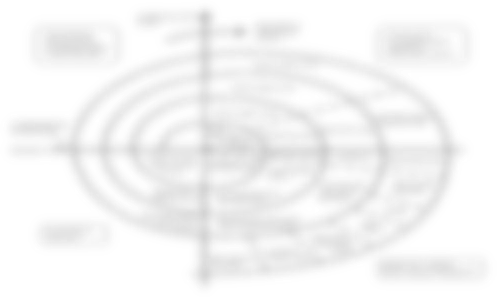
The software engineering paradigm which is also referred to as a software process model or Software Development Life Cycle(SDLC) model is the development strategy that encompasses the process, methods and tools. There are common software process tasks, phases and activities that are modeled by software models.

**Spiral model**

**Spiral model** is one of the most important Software Development Life Cycle models, which provide support for **Risk Handling**. In its diagrammatic representation, it looks like a spiralwith many loops. The exact number of loops of the spiral is unknown and can vary from project to project. **Each loop of the spiral is called a Phase of the software development process.** The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using spiral model.

The Radius of the spiral at any point represents the expenses (cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

Below diagram shows the different phases of the Spiral Model:



Each phase of Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

* **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
* **Identify and resolve Risks:** During the second quadrant all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution is identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is built for the best possible solution.
* **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
* **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the software developed version of the software. In the end, planning for the next phase is started.

## Advantages of Spiral Model:

* Risk Handling
* Flexibility in Requirements
* Customer Satisfaction

Every new software engineering paradigm that gets invented seems to be greeted with the battle cry “Revolution!” as if everywhere software engineers and programmers were freed suddenly of all the evils that the previous paradigm had foisted upon them. Each new generation of unspoiled programming acolytes is admonished to go out into the world, to spread the word, and to suffer not the sins of the previous generation. The truth is that each and every one of these so-called revolutions was built unequivocally on the features of the paradigm that was its predecessor. Software engineering paradigms did not revolt; they evolved! None of the lessons learned was ever forgotten. In many cases, the only things that were changed were those features that had been left completely neglected by an earlier paradigm.

In some respects, that explains why we have progressive shifts anyhow. At the end of any one paradigm phase, it may seem that everyone knows everything needed to write complete, consistent, and clear code. So, we are completely at the mercy of …. *researchers*. Those are the people who can watch what it is that master programmers do differently from other programmers and put a catchy name onto it. Of course, there are many researchers (more than there are master programmers), and some of the researchers are better than others. Their keen observation and insightful analysis is vital in order to tell us how to write the software in the future. It didn’t happen overnight. It took many lines of code, many master programmers, and scores of researchers for us to be able to look back at what we can see as the mainstay of computer programming paradigms.

Benefits of the Paradigm:-

**Re-usability:** anyone that needs a particular functionality can use an appropriate module, without having to code the algorithm from scratch.

***Specialization*:** one person can concentrate on writing a best possible module (function) for a particular task while others look after other areas.

**Upgradability:** if a programmer comes up with a better way to implement a module then he/she simply replace the code within the function. Provided the interface remains the same - in other words the module name and the order and type of each parameter are unchanged - then no changes should be necessary in the rest of the application.

However procedural modules have serious limitations:

* For a start, there is nothing to stop another programmer from meddling with the code within a module, perhaps to better adapt it to the needs of a particular application.
* There is also nothing to stop the code within the function making use of global variables, thus negating the benefits of a single interface providing a single point of entry.

Obviously, the paradigm is best suited for the *waterfall model* of software development.

## Design-

A particular software system is viewed in terms of its modules and data flowing between them starting with a high-level view.

In this case, software design methodology can be categorized as a Top-down modular design

(functional design viewpoint).

The basic design concepts include:

**Modularity**

Modules are used to describe a functional decomposition of the system

* A module is a unit containing:
* executable statements
* data structures
* other modules
* **A module:**
* has a name
* can be separately compiled
* can be used in a program or by other modules
* **System design generally determines what goes into a module**
* Cohesive
* Single clearly defined function
* Description of when and how used
* Loosely Coupled Modules (Modules implement functionality, but not parts of other modules)
* **Black Boxes (information hiding)**
* each module is a black box
* each module has a set of known inputs and a set of predictable outputs
* inner workings of module are unknown to user
* can be reusable
* **Preliminary and Detailed Design specify the modules to carry out the functions in the Data Flow Diagrams (DFD).**

Preliminary design deals mainly with Structure Charts

* Hierarchical tree structure
* Modules - rectangle boxes
* calling relationships are shown with arrows
* arrows are labeled with the data flowing between modules
* Module Design
* Title
* Module ID - from structure charts
* Purpose
* **Method - algorithm**
* Usage - who calls it
* External references - other modules called
* Calling sequence - parameter descriptions
* Input assertion
* Output assertion
* Local variables
* Author(s)
* Remarks
* **Preliminary Design Document**
* Cover Page
* Table of Contents
* Design Description
* Software Structure Charts
* Data Dictionary
* Module Designs
* Module Headers
* Major Data Structures Design
* Design Reviews (Examination of all or part of the software design to find design anomalies )
* **Overview of Detailed Design**
* select an algorithm for each module
* refine the data structures
* produce detailed design document
* **Implementation**
* Coding (for each Module)
* Source Code
* Documentation
* **Integration**
* Decide what order the modules will be assembled
* Assemble and test integration of modules
* After final assembly perform system test
* Note, coding and testing are often done in parallel
* **Testing**

Types of testing

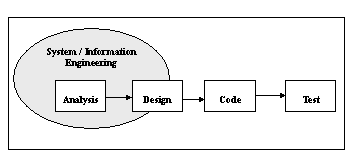
* Unit testing
* Integration testing
* Acceptance testing

As it was mentioned above, the paradigm is best suited for the *waterfall model* of software development. Implementing change requirements and especially rapid prototyping are weak points of the programming paradigm.

To develop the software **“Online Job Portal”** I maintain various records of the traveling company I do follow the spiral model which suggest a systematic sequential approach to software development that begin at the system level and process through analysis, design, coding, testing and support.

The Model encompasses the following phases:

* + Software Requirement Analysis
  + Design
  + Code Generation
  + Testing
  + Support



**Fig: The Phases of a Software Development**

* **Software Requirement Analysis:** The requirement gathering process is intensified and focused specifically on the software. Requirement for both the system and the software are documented and reviewed with the customer.
* **Design:** It is a multistep process that focus on four attributes of a Programme: data structure, software architecture, interface representation and procedural (algorithmic) details. The design process translates requirements into a representation of the software that can be accessed for quality before code begins. Like requirements into a representation of the software that can be accessed for quality before coding begins. Like requirements, the design is documented and becomes part of the software configuration.
* **Code Generation:** the design must be translate into a machine readable form. The code generation steps perform this task. If design is performed in a detail manner, code generation can be accomplished mechanistically.
* **Testing:** Once code has been generated, programmes testing begin. This processes focus on the logical internal of the software, ensuring that all statements have been tested, and on the functional external, that is conducting test to uncover errors and ensure that defined input will produce actual result that agree with required results.
* **Support:** Software will undoubtedly undergo change after it is delivering to the customer.

**Advantages of A Model:-**

* + It is easy to explain.
  + Stages and activities are well defined.
  + Verification at each stage ensures early detection of errors.
  + Helps to plain and schedule project.
  + Ensures that the system meets the user’s need.
  + Minimize the cost of rectifies error.
  + Feedback creates an environment that allows corrections.

Structured Analysis and Design (SASD):-

It is deadly needed for a software development to minimize the error from the very beginning stages. So, after detail investigations and analysis and fact-findings or information gathering, etc. it is revealed that the needs of the organization towards the application are as follows…

For the system, it is identified by the several problems may arise and after successful operation we complete the system properly. There are several problems may arise that are given below…

* **The Problem of Reliability:** A system suffers from the problem of reliability when procedures work some time not all of the time, or when use of the same procedure leads to different result. Analysts must work continually to improve the reliability of system.
* **The Problem of Validity:** Systems that produce invalid results are often most troublesome to users and system managers. There systems might be highly reliable. They may work all of the time, but they draw incorrect conclusions. A report might show that demand is increasing and that additional stock should be ordered for inventory.
* **The Problem of Accuracy**: The problem of accuracy is similar to the problems of reliability and validity. A system is inaccurate when processing is error-phone. If the posting procedure is complex and the number of transactions large, a fair number of errors may occur. Because of inaccuracy, the entire budget system might be viewed as unreliable and often invalid.
* **The Problem of Economy:** Besides improving processing accuracy, organizations seek to improve processing which also economic. A system suffers from the problem of economy when existing methods of transmitting, processing and storing information are very costly. An organization might discover that the cost of handling the paperwork associated with each purchase order is Rs.25. This cost is determined to be problem of economy.
* **The Problem of timeliness:** The problem of timeliness relates more to the transmission of information than to the processing or storing of it. A system suffers from the problem of timeliness if information is available but cannot be retrieved when and where it is needed. As people become more familiar with information systems and how they functions, they generally realize how much easier it is to process and store information than it is to retrieve it.
* **The Problem of Capacity**: The problem of capacity occurs when a system component is not large enough. Capacity problems are especially common in organizations that experience peak periods of business. During peak period, inadequate processing capacity, transmission capacity, storage capacity, staff capacity and the like may all exist. Capacity problems are also evident in rapidly growing organization.
* **The Problem of Throughput:** The problem of throughput may be viewed as the reverse of the problem of capacity. Throughput deals with the efficiency of a system. If system capacity is high and production low, a problem of throughput occurs.

By obeying the all terms and condition I have defined the problem. The tour and traveling company conducts the tour by giving the advertisements to the leading newspaper and other communication media like Local Cable Channels, Hand Bills and Banners, sometimes in road campaign etc. There is another approach to get the more sales the system invites the membership from the person who have very much interest in traveling at the different place frequently. Some discount is offered to the all members and charges will very nominal. The system is not up to the mark and not be so presentable due to the manual system. Not all the purpose is served by the system like enquiry and other monitory information with audiovisual effect because there are several booking counters and no such connections between them. So need computer and intercommunications.

**Problem Definition-**

The whole approach of analysis of problem should however be based around critical factors like the availability of information for making the decision, the time available for processing the data i.e. the realism. System Requirement Specification or SRS had been prepared after proper discussion with the persons attached with the mentioned **“EBS”**. Software project management begins with a set of activities collectively called project planning. Software project planning actually encompasses all of the activities. Planning involves estimation- to determine how much money, how much effort, how many resources, and how much time it will take to build a specific software-based system or product.

**Existing Problem:-**

In the traditional system files were used to maintain the database which was done manually. This existing system consumes a lot of time. This time consuming evaluation coupled by the huge maintenance problem and may also lead to erroneous results. The various operations performed on these files like sorting, adding, modifying and deletion of the records are very tedious. Moreover these manually maintained files have the possibility of getting worn out. Thus, less durability is achieved.

Thus the demerits of the existing system can be summarized as follows…

* There is no consistency as the data may not be updated on time.
* Feasibility is reduced
* Less reliability
* Security is not provided and any one can access
* Prioritization of records is difficult.
* More erroneous
* Difficult to maintain
* As everything is done manually its slow process
* No timely acknowledgement service

Taking the demerits into consideration, an alternative system which uses Oracle as both front end and back end was used. In front end, retrieval of the data from the database is done through SQL queries i.e. using D2K forms. This is not a web application and the data is not distributed as only a single system is used. As it is confined only to a particular system, scope is limited and there is a hindrance to the reliability if the system fails.

The demerits of this alternate system are…

* Only single system used.
* If the system crashes then the data is lost
* too overburdened
* not reliable
* slow processing
* less flexible
* not so user friendly

**Proposed System-**

The proposed system is developed based on the client server architecture, a request-response paradigm and is implemented with the help of advanced java using the tomcat web container. The employees can maintain and do the transactions online.

The application starts by asking for user name and password which provides authentication. This system provides high security where the unauthorized users cannot access the data.

Later we have different options for the employee like

* Consumer Management
* Revenue Collection
* Billing and Accounting
* Reports
* **Meter Details**
* **Ledger Process**
* **Journal Details**

The objectives of the proposed system are as follows:

* Easy to use, effective and efficient
* Accurate results.
* Easy maintenance.
* Fast access
* More feasibility
* More secure.
* Provides high consistency.
* More reliable

**Potential Users of the System: -**

There are two groups of users in our system, they are **“IMS”** members and system administrators. They have different authorities in our system which are shown as follows…

* **Members** – They are MCU members. They can view the detailed product information and their shopping history. Besides, they are able to create, view, modify and cancel their shopping orders.
* **Administrator**s – They are authorized MCU staffs to control the system. They are assigned with different level of authority to maintain the information of inventory, invoicing, members and administrators. Besides, they can make conditional refunds and print out transaction invoices. In addition, administrators can execute summary reports and be responsible to maintain database.

**Requirement Specification: -**

In the following, we describe the functional requirements and non-functional requirements of our system.

It is actually the inventory keeping and maintenance system of any store/shop, so the requirements are collected from both the shop owner’s side and along from the dealers and stockiest side who also deal with this store.

These requirements indicate that to keep their stock properly and maintain them by maintaining a proper accounting also. So, the shop/store owners need one type of user who login the system as a super user and having all the facilities to access where as the ordinary dealers will login as a normal user having limited access to handle the package and to get limited number of reports also. The shop deals with any kind of materials where stock valuation method is generally FIFO (First In First Out) or any other as specified by the shop owners and stockiest. The shop/stock holders first can purchase goods of different types such as either in liquid or in solid i.e. paper, mineral water from the dealers or distributors with whom they have daily connections. The purchasing is done through generating purchase challan, which only affects the existing stock, whereas the purchase bill dealing with payments of the debtors effects directly on the accounts. Though this software is not directly based on account keeping but the daily transactions are affecting related ledgers. Moreover, bill adjustments, pending bill receive; stock verifications, etc. are maintained successfully.

In case of selling of materials, same also done through generating order, challan, and bills. Order and challan effects on stock and billing will effect on accounts portion of the software. The sale also categorized into two types i.e. regular customer wise sales and direct sale, Billing also associated with different duties, taxes.

**Preliminary Product Description: -**

The Developer must think whether to take the charge of this project or not! So, they gone thought several feasibility study period i.e.

More commonly known as cost/benefit analysis. The procedure is to determine the benefits and saving that are expected from a candidate system and compare them with costs. If the benefits outweigh costs, then decision is made to design and implement the system.

Considering the facts it is become evident that the system will be economically feasible both for developer as well as for client’s respect.

Technical feasibility centers on the existing computer system (hardware, software, etc.)And to what level it can support the proposed addition. If the budget is a serious constraint, then the project is judged not feasible. Here it is not a constraint altogether.

People are inherently resistant to change, and computers have been known to facilitate change. An estimate should be made of how strong a reaction the user staff is likely to have toward the development of a computerized system. This is not a major problem in this area.

A determination of any infringement, violation or liability that could result from the development of the system. But the system to be developed will be 100% legal.

The management & operators desire to be well acquainted with the requisite skill needed.

* + 1. **Functional Requirements:-**

In [software engineering](http://en.wikipedia.org/wiki/Software_engineering), a functional requirement defines a function of a [software system](http://en.wikipedia.org/wiki/Software_system) or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, [data manipulation](http://en.wikipedia.org/wiki/Data_manipulation) and processing and other specific functionality that show how a [use case](http://en.wikipedia.org/wiki/Use_case) is to be fulfilled. They are supported by [non-functional requirements](http://en.wikipedia.org/wiki/Non-Functional_Requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability).As defined in [requirements engineering](http://en.wikipedia.org/wiki/Requirements_analysis), functional requirements specify particular behaviors of a system. This should be contrasted with [non-functional requirements](http://en.wikipedia.org/wiki/Non-Functional_Requirements) which specify overall characteristics such as cost and [reliability](http://en.wikipedia.org/wiki/Reliability).

**Other Functional Requirements:-**

1. **Input/output**
2. **Processing**
3. **Error Handling**

**Technical Requirements:-**

* Performance Requirements
* Safety Requirements
* Security Requirements
* Hardware Constraints
* Software Constraints
* Design Constraints
  + 1. **Nonfunctional Requirements:-**

Non-functional requirements define the operational requirements and project schedule that are requested by our stakeholder.

* All user manuals should be provided in the necessary format
* Application should support 5 simultaneous users.
* There will be backup procedure to maintain records.

**Operational Requirements: -**

Based on the Online Job Portal application’s requirements, I have made the choice of frameworks to be used. This application primarily consists of 2 main components;

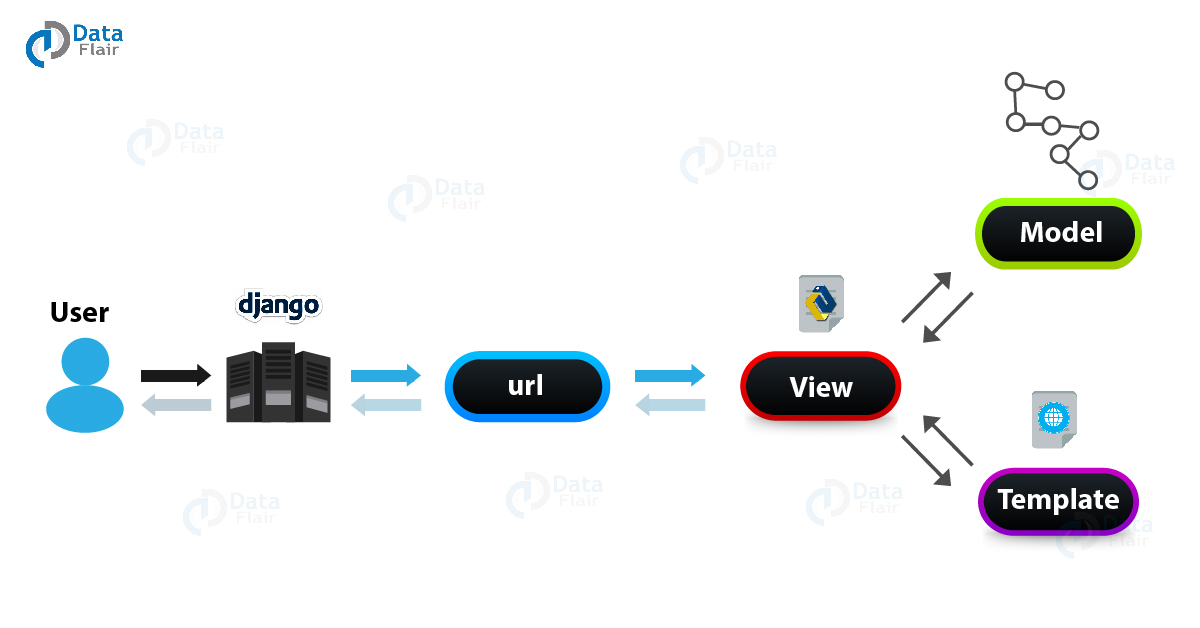
**A web application** that allows recruiters to outline job vacancies with required information for which jobseekers can view and apply according to their interests.

**The Python** based web application is developed using Django Framework to connect to the SQLite database. The front end is developed using HTML, CSS, JavaScript and JQuery.

**Django Framework**

Django is a web application framework written in Python programming language. It is based on MVT (Model View Template) design pattern. The Django is very demanding due to its rapid development feature. It takes less time to build application after collecting client requirement.

This framework uses a famous tag line:**The web framework for perfectionists with deadlines.**

By using Django, we can build web applications in very less time. Django is designed in such a manner that it handles much of configure things automatically, so we can focus on application development only.

The Django architecture diagram below shows the working cycle of Django MVC architecture.

#### Model

The Model is the part of the web-app which acts as a mediator between the website interface and the database. In technical terms, it is the object which implements the logic for the application’s data domain. There are times when the application may only take data in a particular dataset, and directly send it to the view (UI component) without needing any database then the dataset is considered as a model.

Although today if we want any kind of website we need to have some sort of database as we must be requiring some user input even if we are creating a simple blog site.

#### View

This component contains the UI logic in the Django architecture.

View is actually the User Interface of the web-application and contains the parts like HTML, CSS and other frontend technologies. Generally, this UI creates from the Models component, i.e., the content comes from the Models component.

**For example:**

When you click on any link or interact with the website components, the new webpages that website generates is actually the specific views that stores and generates when we are interacting with the specific components.

#### Controller

The controller as the name suggests is the main control component. What that means is, the controller handles the user interaction and selects a view according to the model.

The main task of the controller is to select a view component according to the user interaction and also applying the model component.

This architecture has lots of advantages and that’s why Django is also based on this architecture. It takes the same model to an advanced level.

**For example:**

When we combine the two previous examples, then we can very clearly see that the component which is actually selecting different views and transferring the data to the model’s component is the controller.

## MTV Pattern

Django is mainly an MTV (Model-Template-View) framework. It uses the terminology Templates for Views and Views for Controller.

Template relates to the View in the MVC pattern as it refers to the presentation layer that manages the presentation logic in the framework and essentially controls the content to display and how to display it for the user.

Thus our [**Python code**](https://data-flair.training/blogs/python-syntax-semantics/) will be in views and models and HTML code will be in templates.

## SQLite3

SQLite in general is a server-less database that you can use within almost all programming languages including Python. Server-less means there is no need to install a separate server to work with SQLite so you can connect directly with the database

## Django Web Server

Django, being a web framework, needs a web server in order to operate. And since most web servers don’t natively speak Python, we need an interface to make that communication happen.

Django currently supports two interfaces: WSGI and ASGI.

* [WSGI](https://wsgi.readthedocs.io/en/latest/) is the main Python standard for communicating between Web servers and applications, but it only supports synchronous code.
* [ASGI](https://asgi.readthedocs.io/en/latest/) is the new, asynchronous-friendly standard that will allow your Django site to use asynchronous Python features, and asynchronous Django features as they are developed.

**2.5) Data Model, Control Flow Diagrams, State Diagrams, ERD’s/ Class Diagrams/CRC Models/Collaboration Diagrams / Use-Case Diagrams /Activity Diagrams Depending On Project Requirements**

**Data Model: -**

Data modeling is a method used to define and analyze data requirements needed to support the business processes of an organization. The data requirements are recorded as a conceptual data model with associated data definitions. Actual implementation of the conceptual model is called a logical data model. To implement one conceptual data model may require multiple logical data models. Data modeling defines the relationships between data elements and structures, Data modeling techniques are used to model data in a standard, consistent, predictable manner in order to manage it as a resource. The use of this standard is strongly recommended for all projects requiring a standard means of defining and analyzing the data resources within an organization.

**Data Design: -**

**Data Flow Diagram (DFD):**

**Data flow diagrams** can be used to provide a clear representation of any software function and the technique starts with an overall picture of the software and continues by analysing each of the functional areas of interest. This analysis can be carried out to precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

**External Entity:-**

###### User

An external entity is a source or destination of a data flow, which is outside the area of study. Only those entities, which originate or receive data are represented on a business process diagram. The symbol used is an oval containing a meaningful and unique identifier.

**Process:-**

A process shows a transformation or manipulation of data flows within the system. Firstly an identification number appears in the upper corner. This is allocated arbitrarily at the top level and serves as a unique reference. Finally, a descriptive title is placed in the center of the box. This should be a simple imperative sentence with a specific verb, for example 'SignIn' or 'Signup'.

**Data Flow:-**

**Data\_name**

A data flow shows the flow of information from its source to its destination. A data flow is represented by a line, with arrowheads showing the direction of flow. Information always flows to or from a process and may be written, verbal. The processes or data stores at its head or tail may reference each data flow.

Data Store:-

## 1 userdetail PrivateRooms

A data store is a holding place for information within the system. It is represented by an open ended narrow rectangle. Data stores may be long-term Database such as user detail. Each data store should be given a reference followed by an arbitrary number.

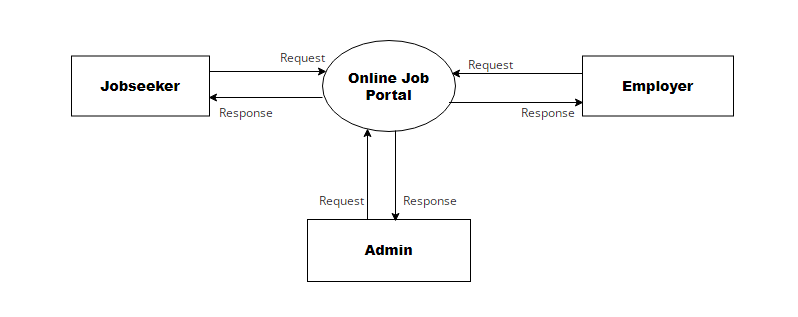
**Context Diagram:-**

* The context diagram represents the entire system under investigation. This diagram should be drawn first, and used to clarify and agree the scope of the investigation.
* The components of a context diagram are clearly shown on this screen. The system under investigation is represented as a single process, connected to external entities by data flow, resources flow.
* The context diagram clearly shows the interfaces between the system under investigation and the external entities with which it communicates. Therefore, whilst it is often conceptually trivial, a context diagram serves to focus attention on the system boundary and can help in clarifying the precise scope of the analysis.
* The context diagram shown below represents a book of chat web server. Chat web server receives details of users, and shows every detail of users who are currently on line.

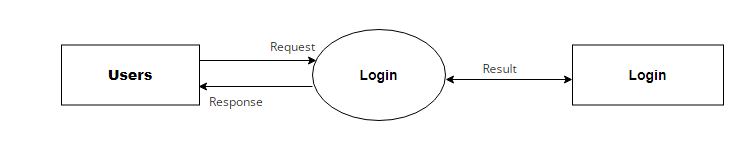
**Data Flow Diagram: -**

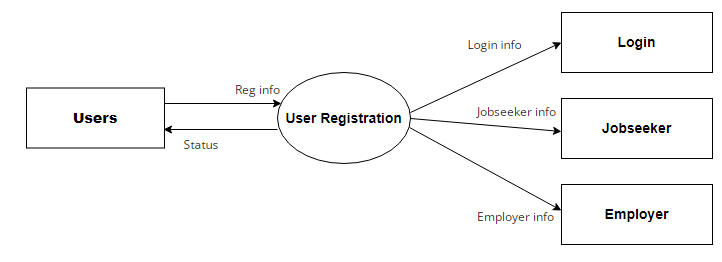
**Context Level Diagram:-**

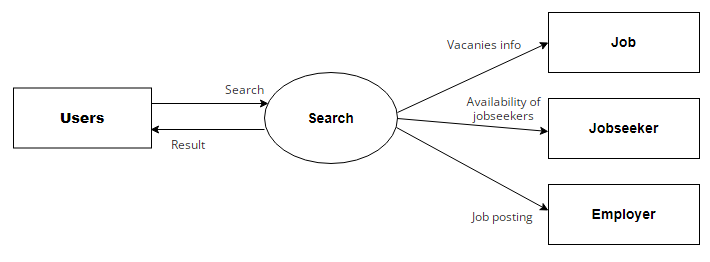
***0 level DFD***

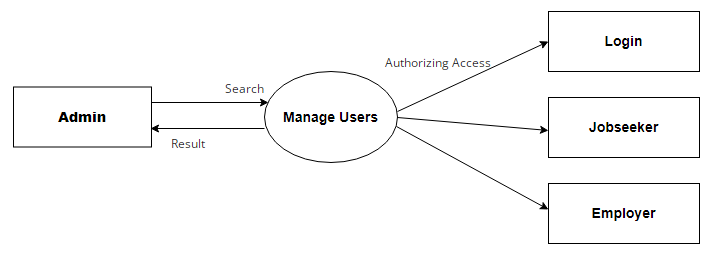


***1 level DFD***

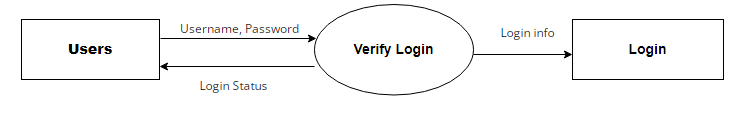
******

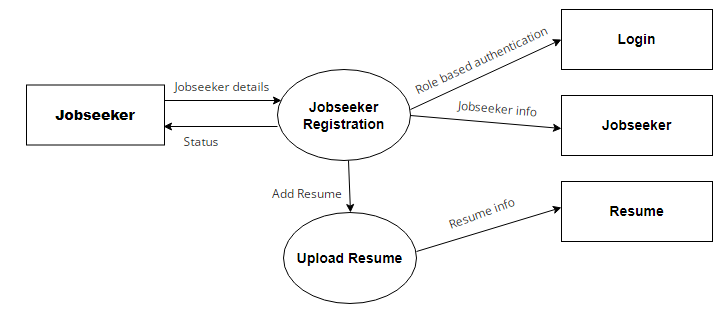
******

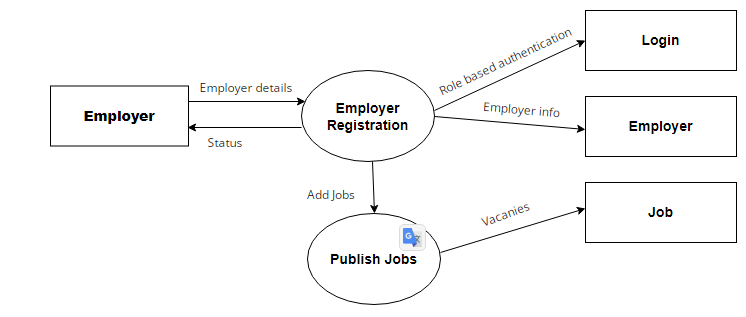
******

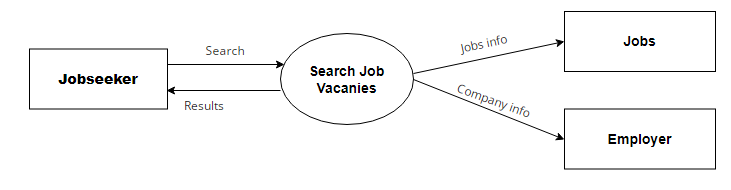
******

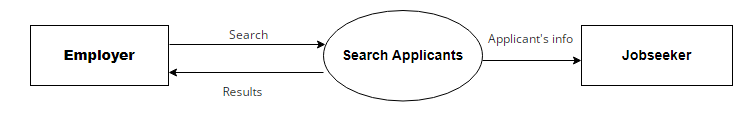
***2 level DFD***

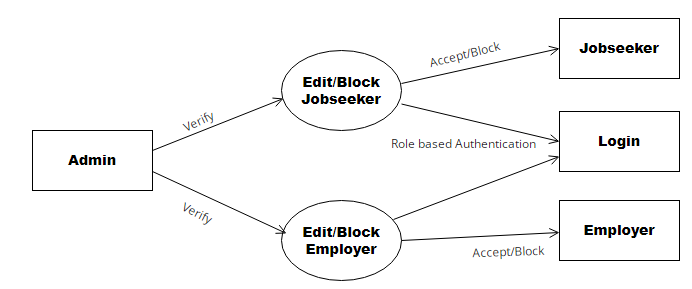






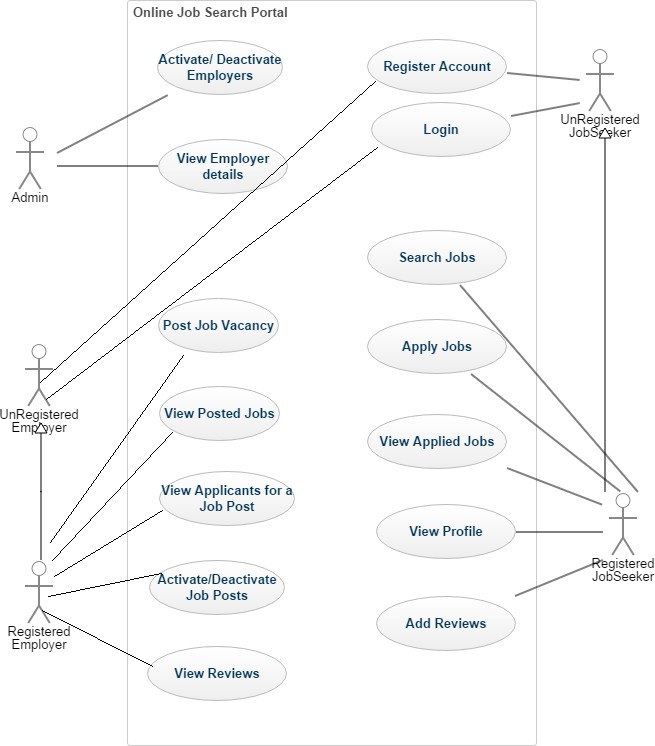






**Use Case Diagram**

A diagram is a visualization 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 modeling the behavior of a system.

**Use Case Diagram for Online Job Portal**

**Figure4:Use Case Diagram**

**Actors:**

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

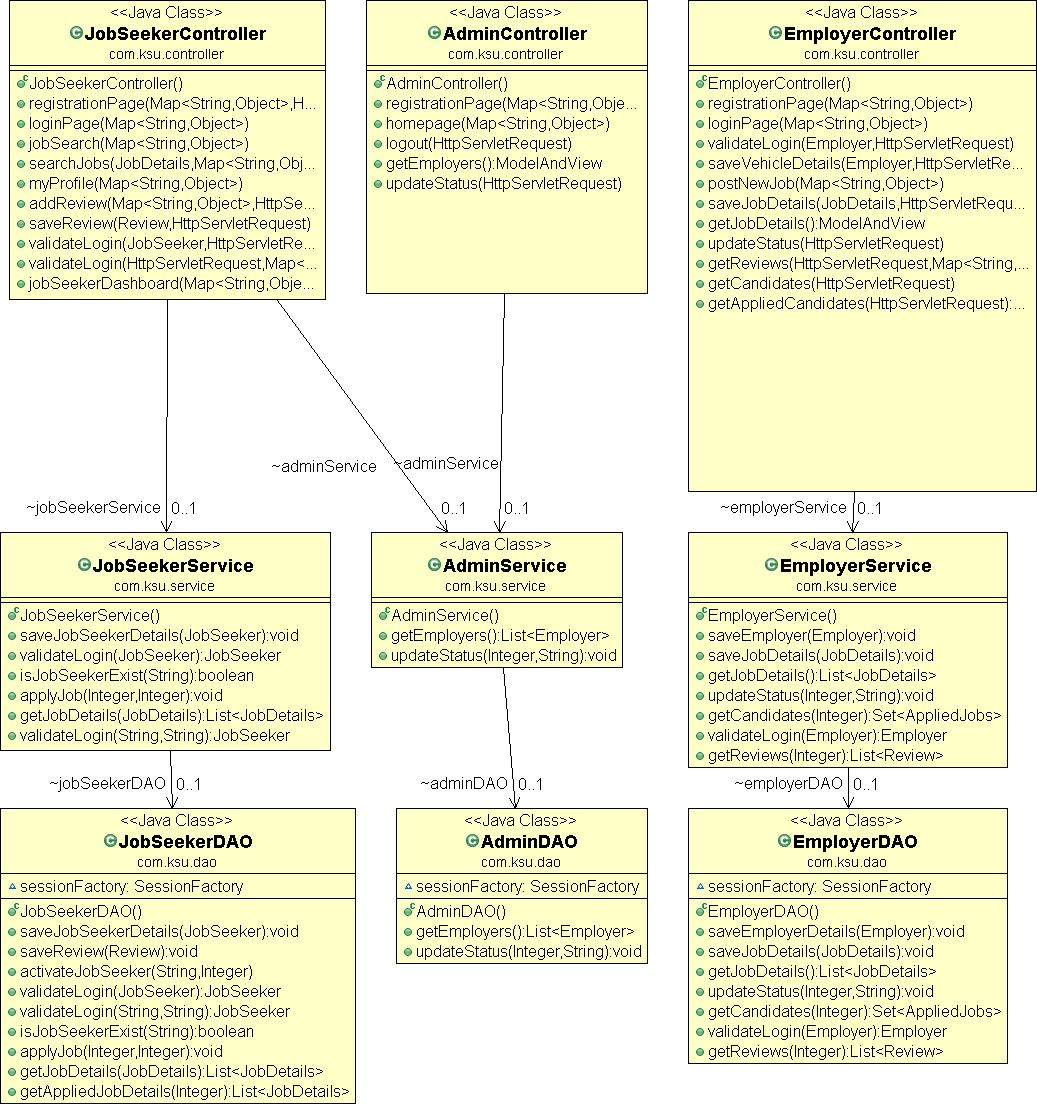
**Use cases:**

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 Jobseeker to register with the application.
* **Login-**This use case denotes a set of actions required for Employer and Jobseeker to login into the application.
* **Activate/De-activate Account-**This use case denotes a set of actions required for admin to activate order-activate 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 Applicants 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 or Employer to view Reviews provided by the applicants.
* **View Job Posts-**This use case denotes a set of actions required for Employer to view all the jobs posted by the Employer.
* **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 a set of actions required for Job Seeker to apply for an available job vacancy.

### 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.



**Controller classes**: The Project contains three controller classes i.e. Admin Controller, Job Seeker Controller and Employer Controller. These classes are responsible for handling HTTP requests and returns HTTP response.

**Service classes:** The Project contains three Service classes i.e. Admin Service, Job Seeker Service and Employer Service. The Controller classes pull data from the request and pass 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. Admin DAO, Job Seeker DAO and Employer DAO. 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.

# 

# 3. System Design

# 3.1) Modularization Details

The Online Job Search Portal is a web-based and android application, which revolutionizes the way companies hire the candidates and jobseekers search for job vacancies. The employers can view reviews given by the jobseekers and make improvements in their system accordingly. The application provides a flexible and easy to use environment on desktops as well as portable devices like smart phones/tablets for the users to achieve their respective objective.

The modules that I have implemented in the Online Job Portal are as listed.

* Admin
* Employer
* Jobseeker

## Admin

Spring Security provides the Admin login. The Admin module provides various functionalities. The Admin users are responsible for activating and deactivating the employer accounts. In addition, Admin users can view the list of employers registered with the application.

## Employer

Employer users will be able to perform functions such as registering with the application and creating an account by providing the details of Employer Name, Employer Code, Address, Company E-mail, Mobile Number, Login Name, and Password that are stored in the Employer table of SQLite database. Once the account is activated, this module allows employers to post jobs summarizing responsibilities and expected skills that will be saved in the Posted Jobs table of SQLite database. The employer will also be given privilege to activate or deactivate jobs. He/she can view the list of job postings that are active. He/she can also view the applicant details that have applied for a particular job posting. The employer will be able to view reviews provided by the jobseeker.

## Jobseeker

The Jobseeker users will be able to perform functions such as registering with the application and creating an account by providing the details of First Name, Last Name, E-mail, Password, Mobile Number, Primary Skill and Experience that are stored in the Jobseeker table of SQLite database.

# 3.2) Data Integrity and Constraints

Data integrity is the overall accuracy, completeness, and consistency of data. Data integrity also refers to the safety of data in regards to regulatory compliance — such as GDPR compliance — and security. It is maintained by a collection of processes, rules, and standards implemented during the design phase. When the integrity of data is secure, the information stored in a database will remain complete, accurate, and reliable no matter how long it’s stored or how often it’s accessed. Data integrity also ensures that your data is safe from any outside forces.

## a) Entity integrity

Entity integrity relies on the creation of primary keys, or unique values that identify pieces of data, to ensure that data isn't listed more than once and that no field in a table is null. It's a feature of relational systems which store data in tables that can be linked and used in a variety of ways.

## b) Referential integrity

Referential integrity refers to the series of processes that make sure data is stored and used uniformly. Rules embedded into the database’s structure about how foreign keys are used ensure that only appropriate changes, additions, or deletions of data occur. Rules may include constraints that eliminate the entry of duplicate data, guarantee that data is accurate, and/or disallow the entry of data that doesn’t apply.

The **constraints** are a rule which works into the table, it disallows the that data which are not following the rule of data constraints.

**For example:** Suppose, you have table there may have more than one column, among them there is one phone no column and you want, its would be unique then you will have to attach a Unique Key Constraint. disallows the duplicity, Oracle engine will check the data for uniqueness, which is going to be enter if it’s a unique among them which has already stored into the table, then it will be accepted otherwise it will be rejected.

### Types of Data Constraints:

There are two types of Data Constraints that can be applied to data being inserted into the table.

* **First one is the Input/output Constraint**. This data constraint determines the speed at which data can be inserted or extracted from the table.
  + - The Primary Key Constraint
    - The Foreign Key Constraint
* **Second one is the Business Rule Constraint**
* Check Constraint
* Unique Constraint
* NOT NULL Constraint

**3.3) Database design, Procedural Design/Object Oriented Design**

The database that is used to design the web application is SQLite. SQLite Studio is used to create tables and run queries. In this application development, we have used SQLite to store employer details, jobseeker details, applied jobs by the applicants, jobs posted by the employer. Hence, we have identified five tables to achieve desired functionality.

* + - * + **Employer table:** holds details of Employer
        + **Job seeker table:** holds details of applicant
        + **Applied Jobs table:** holds details of jobs applied by the jobseeker
        + **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.

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.

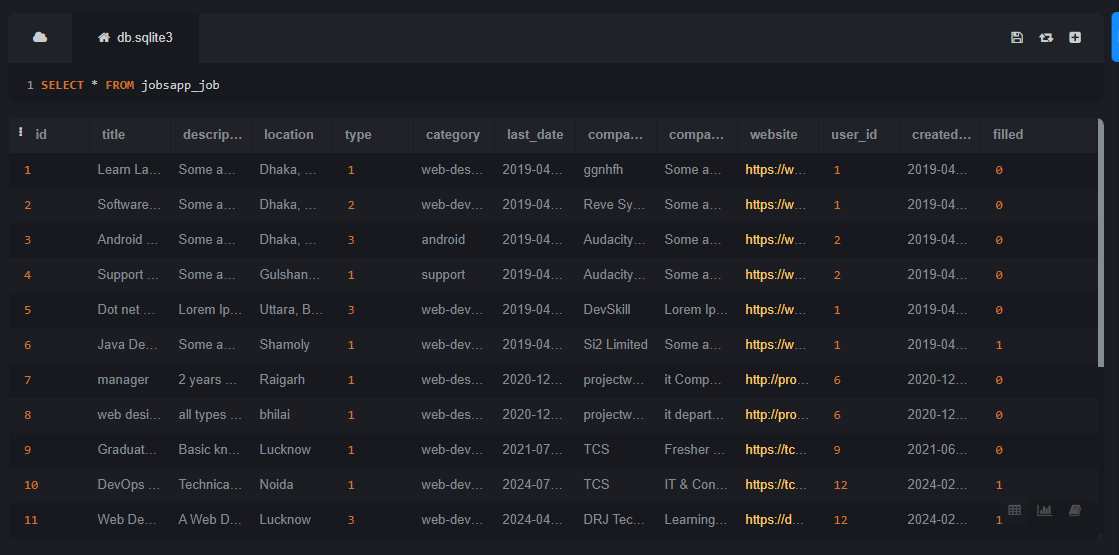
**Table : accounts\_user**

****

**Table : jobsapp\_applicant**

****

**Table : jobsapp\_job**

****

**3.4) User Interface Design**

User interface is the front-end application view to which user interacts in order to use the software. The software becomes more popular if its user interface is:

* Attractive
* Simple to use
* Responsive in short time
* Clear to understand
* Consistent on all interface screens

There are two types of User Interface:

1. **Command Line Interface:** Command Line Interface provides a command prompt, where the user types the command and feeds to the system. The user needs to remember the syntax of the command and its use.
2. **Graphical User Interface:** Graphical User Interface provides the simple interactive interface to interact with the system. GUI can be a combination of both hardware and software. Using GUI, user interprets the software.

The analysis and design process of a user interface is iterative and can be represented by a spiral model. The analysis and design process of user interface consists of four framework activities.

**A) User, task, environmental analysis, and modeling:**

Initially, the focus is based on the profile of users who will interact with the system, i.e. understanding, skill and knowledge, type of user, etc, based on the user’s profile users are made into categories. From each category requirements are gathered. Based on the requirements developer understand how to develop the interface. Once all the requirements are gathered a detailed analysis is conducted. In the analysis part, the tasks that the user performs to establish the goals of the system are identified, described and elaborated. The analysis of the user environment focuses on the physical work environment. Among the questions to be ask dare:

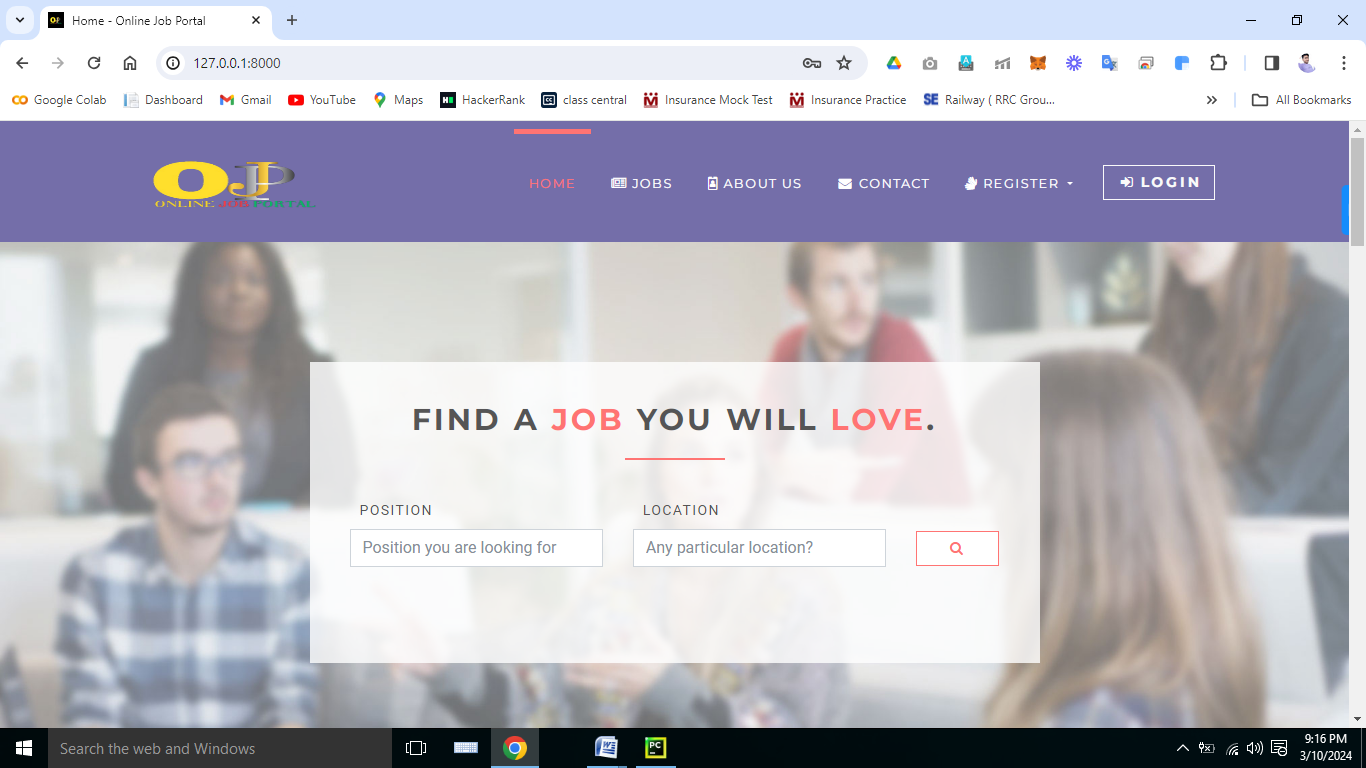
* Where will the interface be located physically?
* Will the user be sitting, standing, or performing other tasks unrelated to the interface?
* Does the interface hardware accommodate space, light, or noise constraints?
* Are there special human factors considerations driven by environmental factors?

**B) Interface Design:** The goal of this phase is to define the set of interface objects and actions i.e. Control mechanisms that enable the user to perform desired tasks. Indicate how these control mechanisms affect the system. Specify the action sequence of tasks and subtasks, also called a user scenario. Indicate the state of the system when the user performs a particular task. Always follow the three golden rules stated by The oMandel. Design issues such as response time, command and action structure, error handling, and help facilities are considered as the design model is refined. This phase serves as the foundation for the implementation phase.

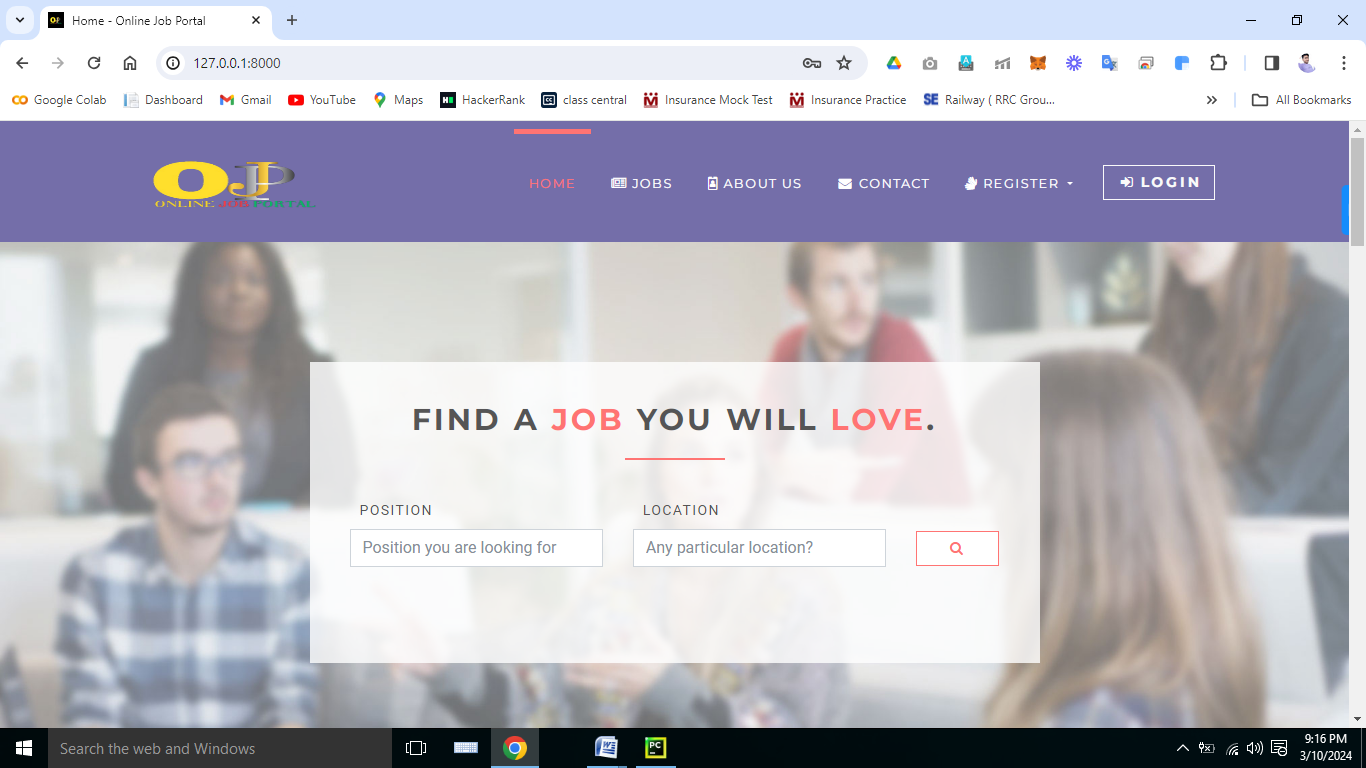
**a) Interface construction and implementation:** The implementation activity begins with the creation of prototype(model) that enables usage scenarios to be evaluated. As iterative design process continues a User Interface toolkit that allows the creation of windows, menus, device interaction, error messages, commands, and many other elements of an interactive environment can be used for completing the construction of an interface.

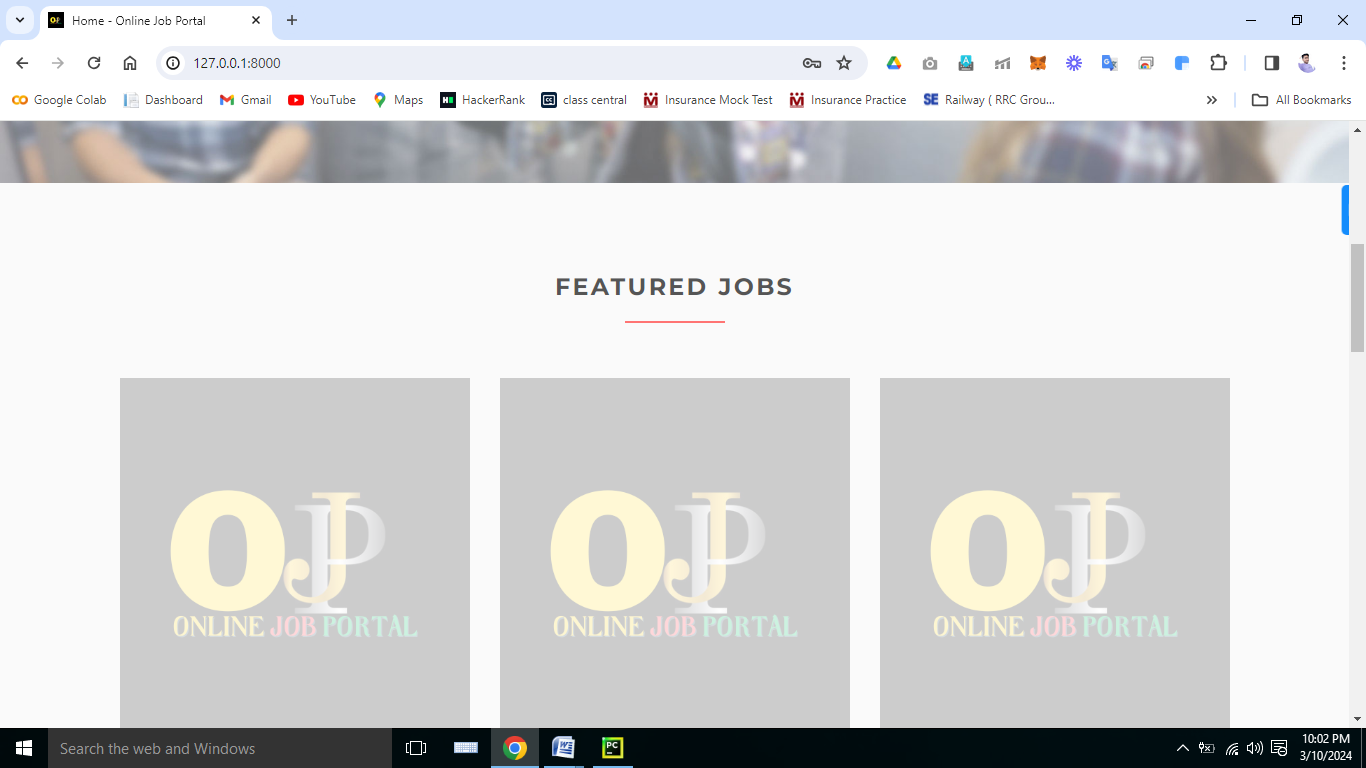
**b) Interface Validation:** This phase focuses on testing the interface. The interface should be in such a way that it should be able to perform tasks correctly and it should be able to handle a variety of tasks. It should achieve all the user’s requirements. It should be easy to use and easy to learn. Users should accept the interface as a useful one in their work.

**Interface: Dashboard**

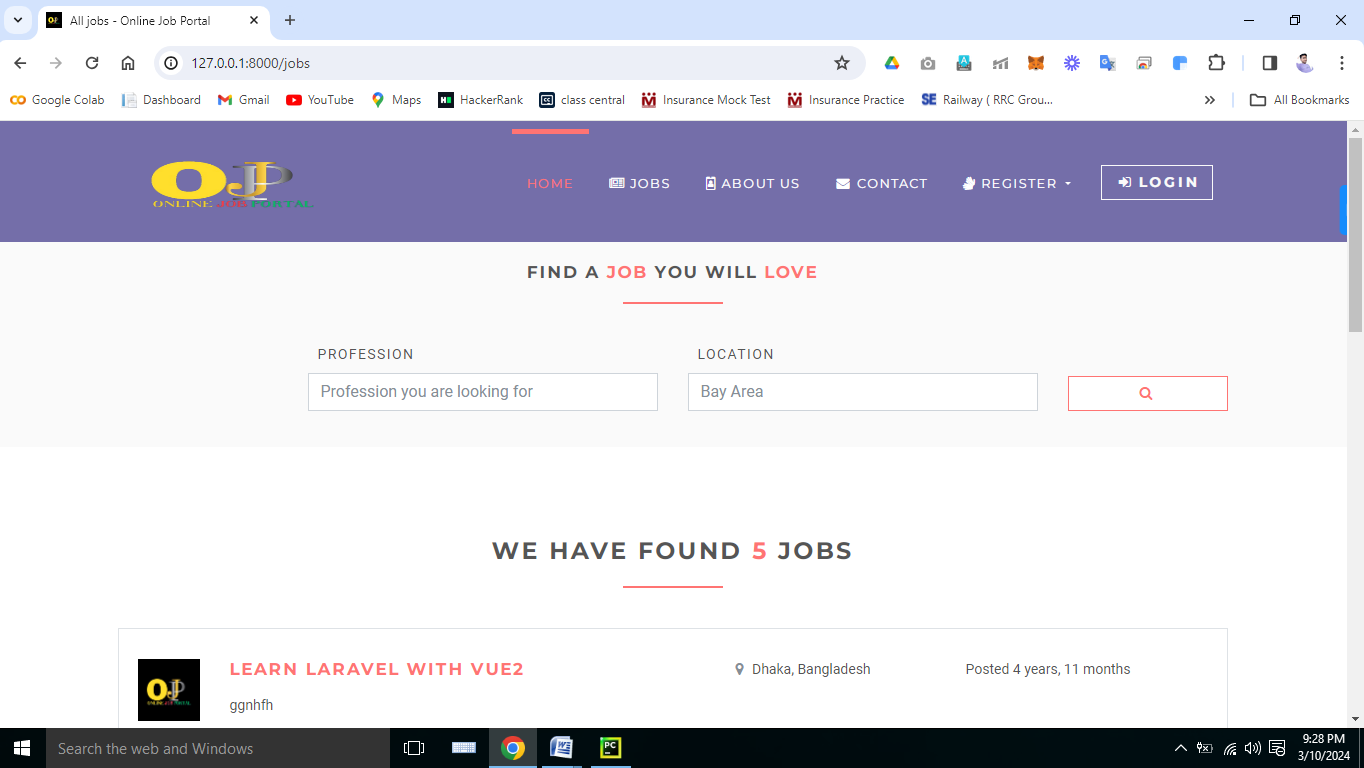
****

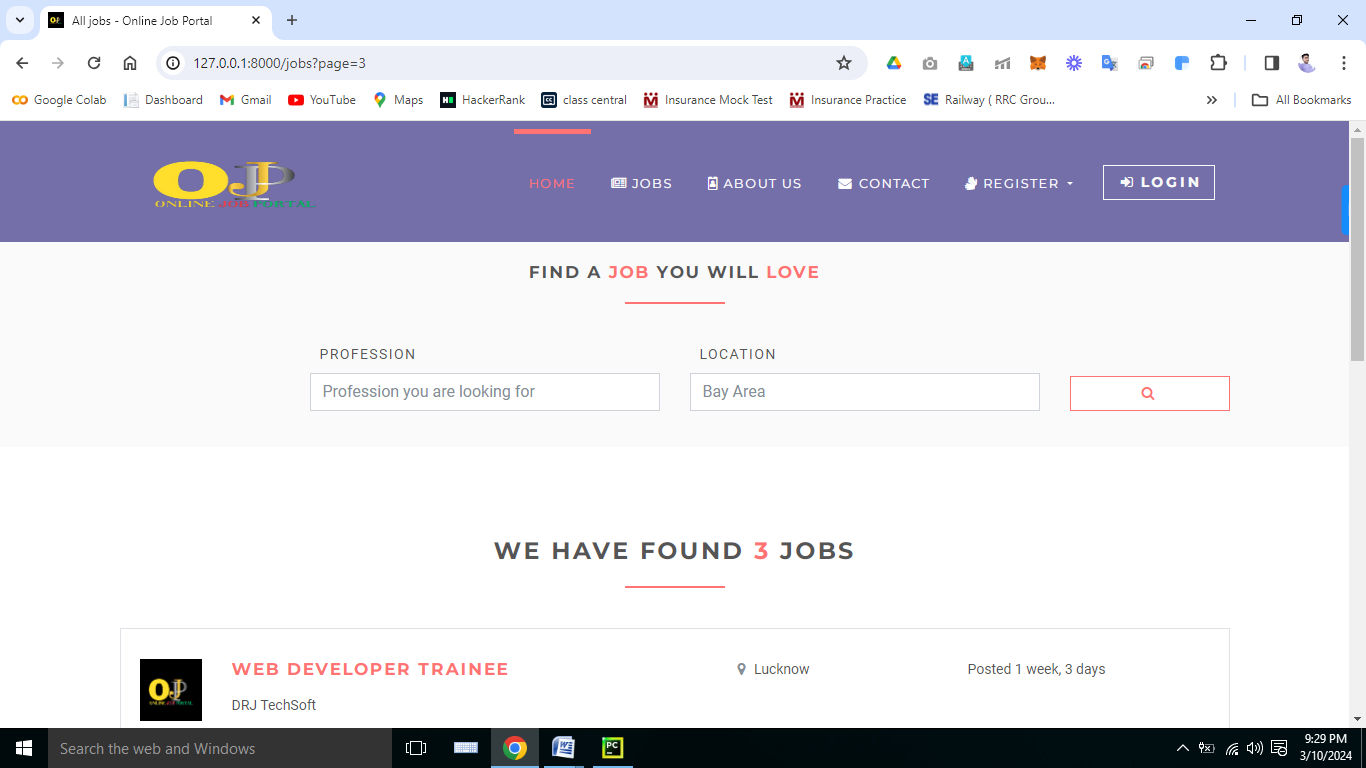
**Interface: Home**



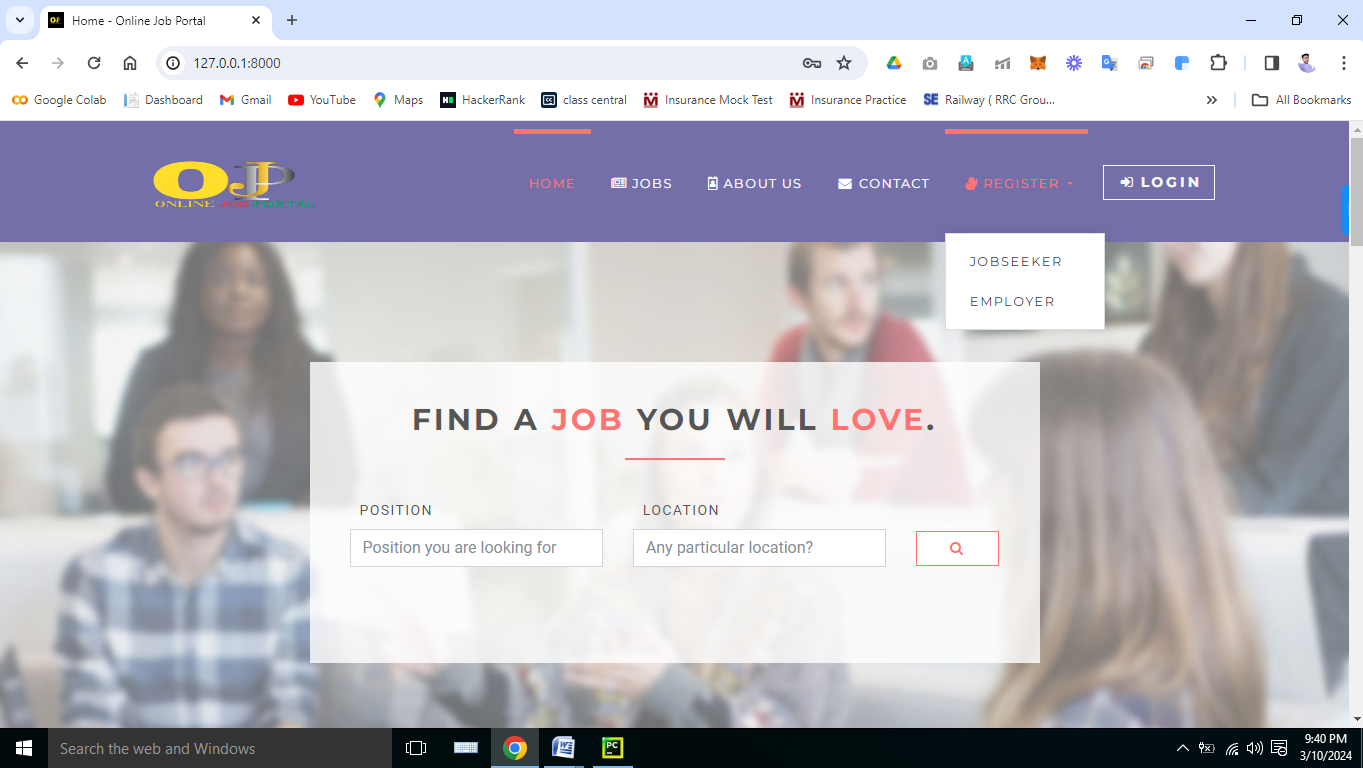


**Interface: Jobs**

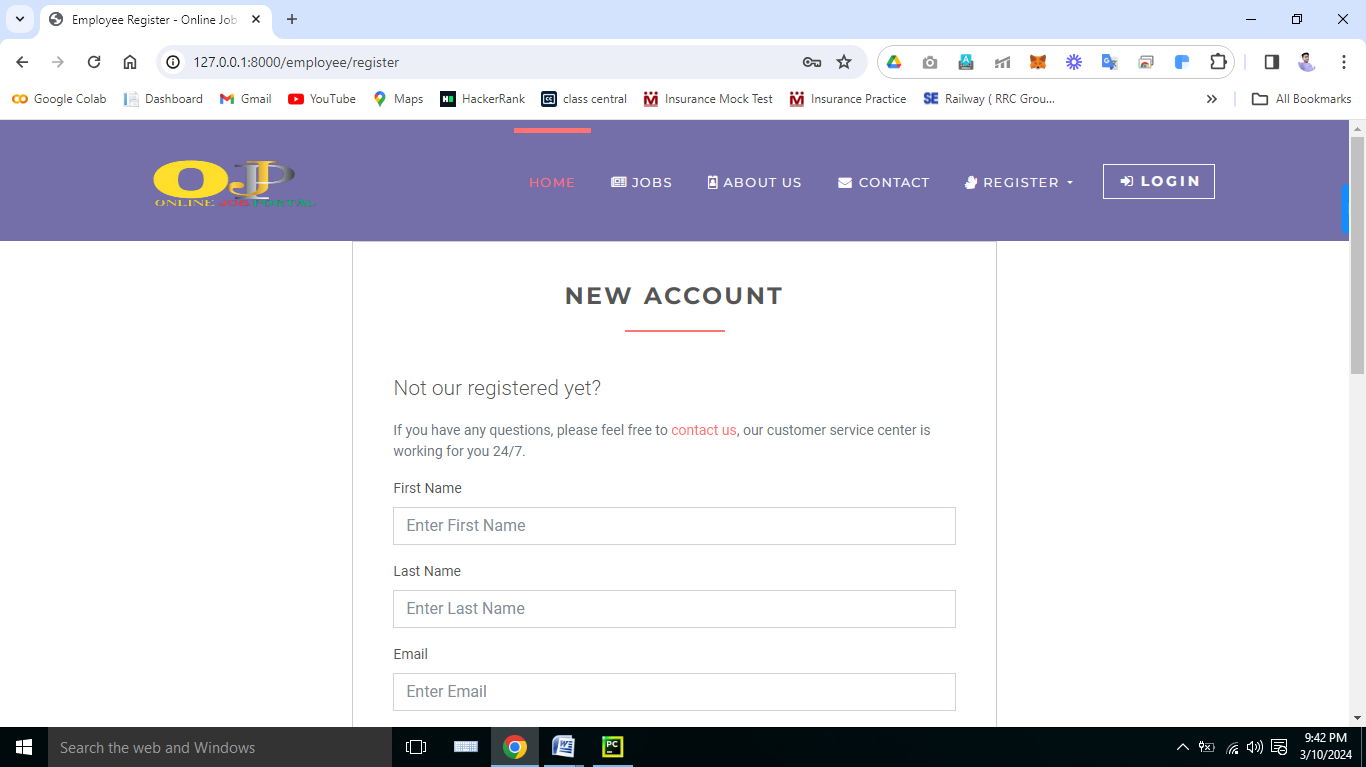
****

****

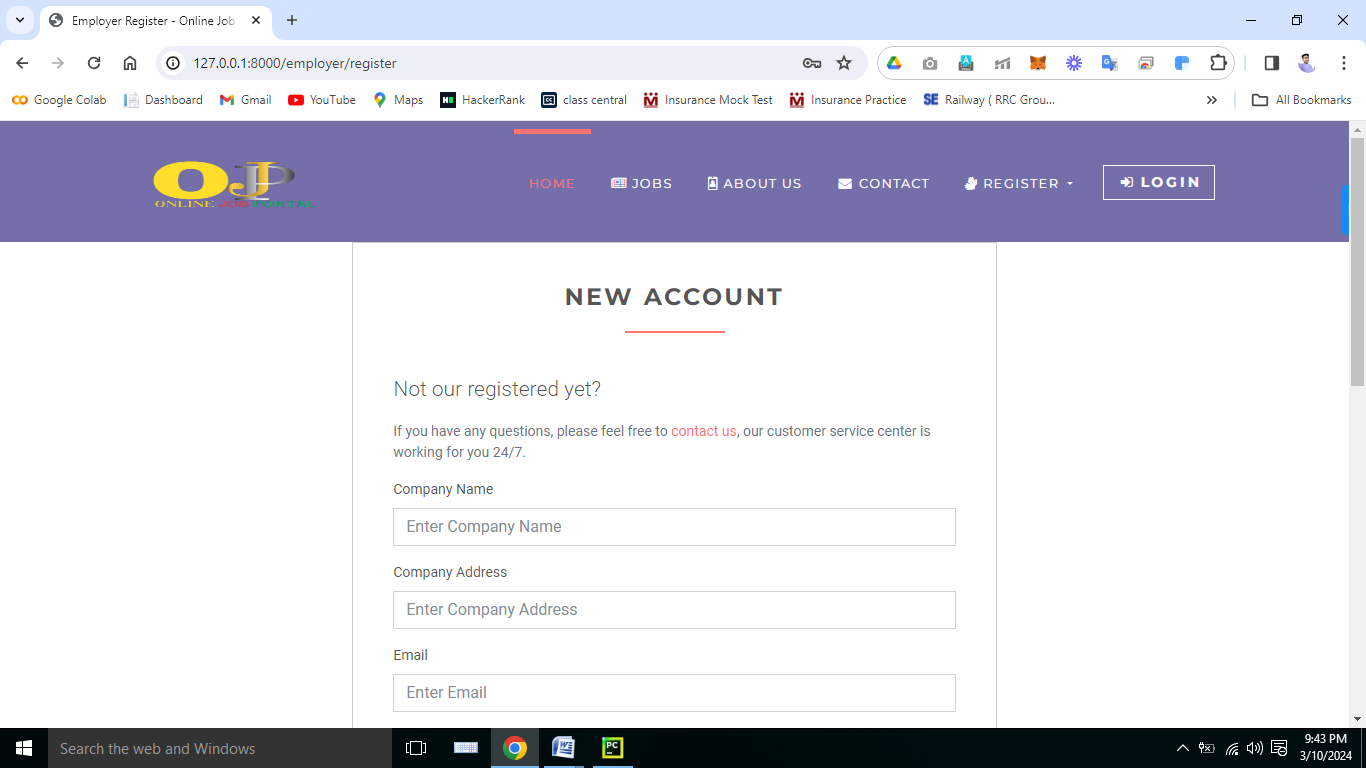
**Interface: Register**

****

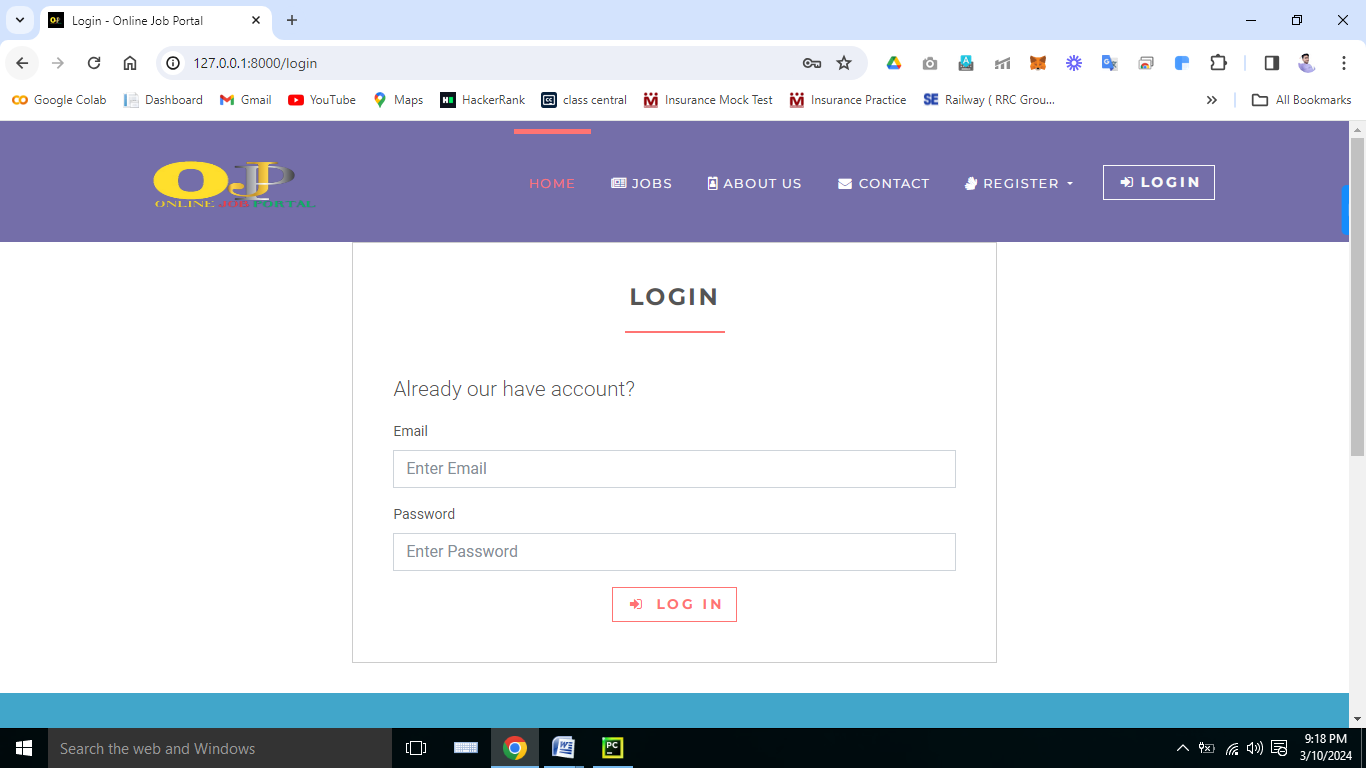
**Interface: Register (Jobseeker)**

****

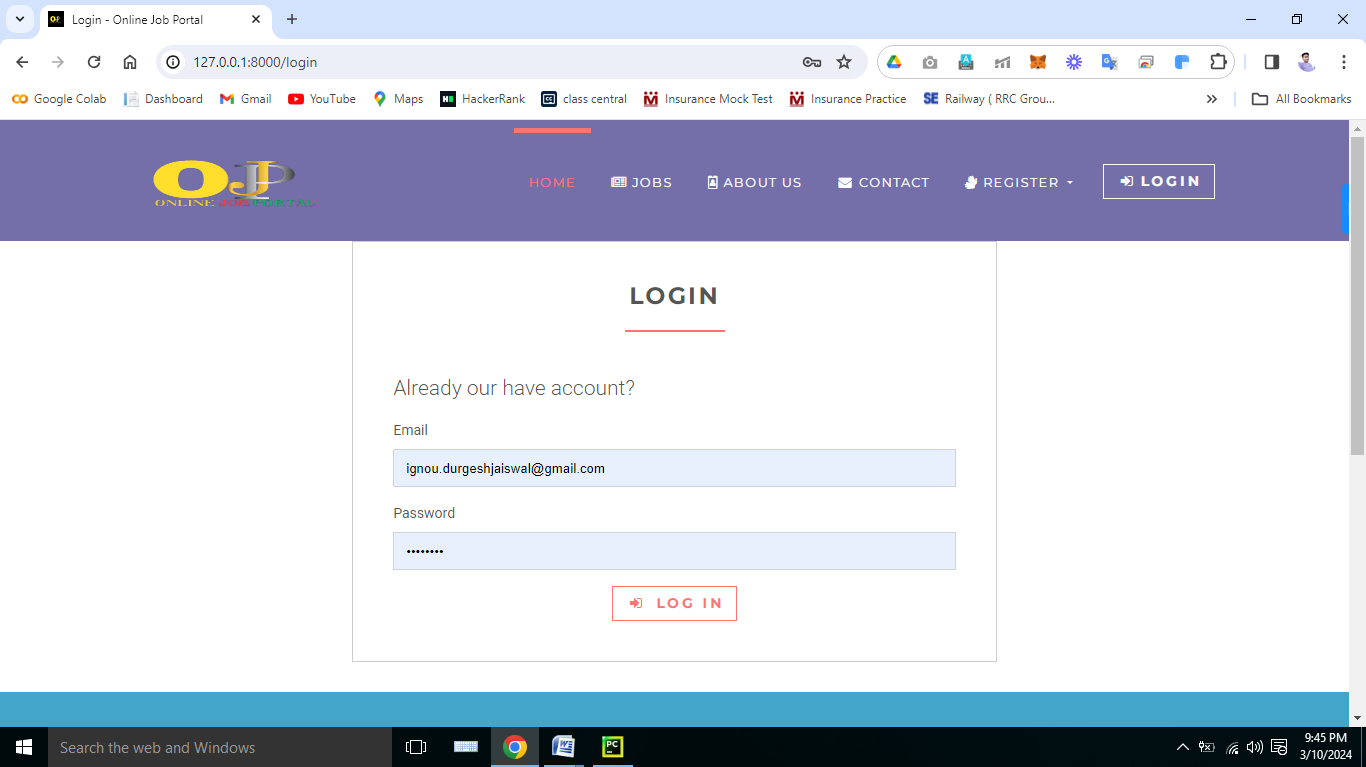
**Interface: Register (Employer)**

****

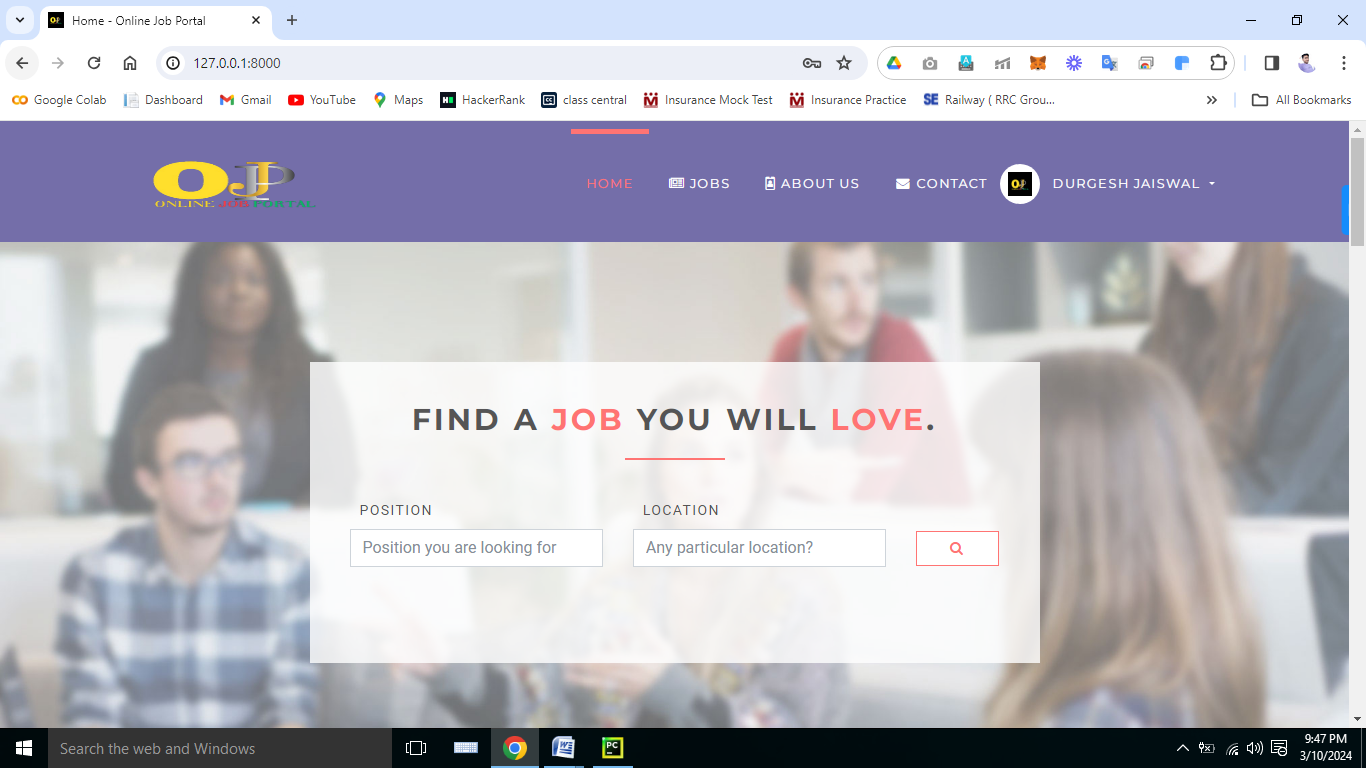
**Interface: Login Screen**

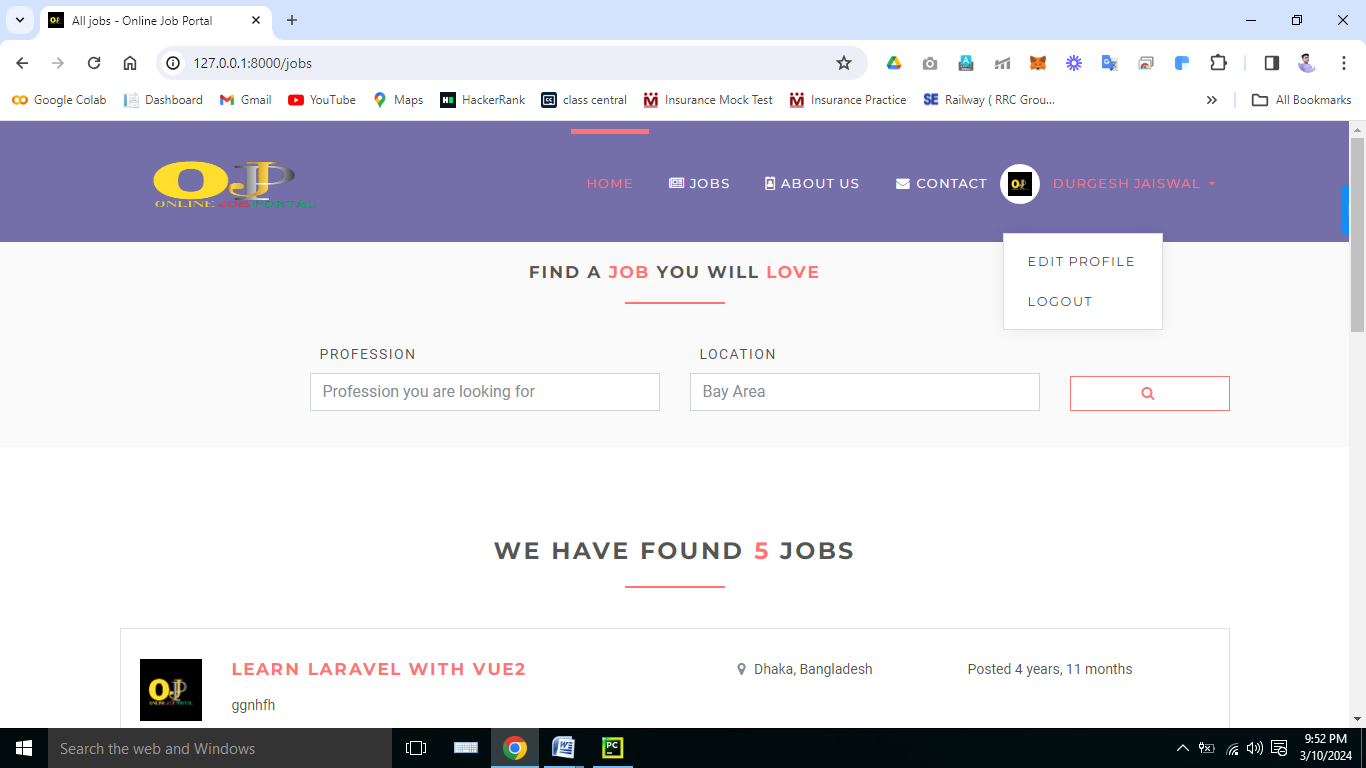
****

**Interface: Login (Jobseeker)**

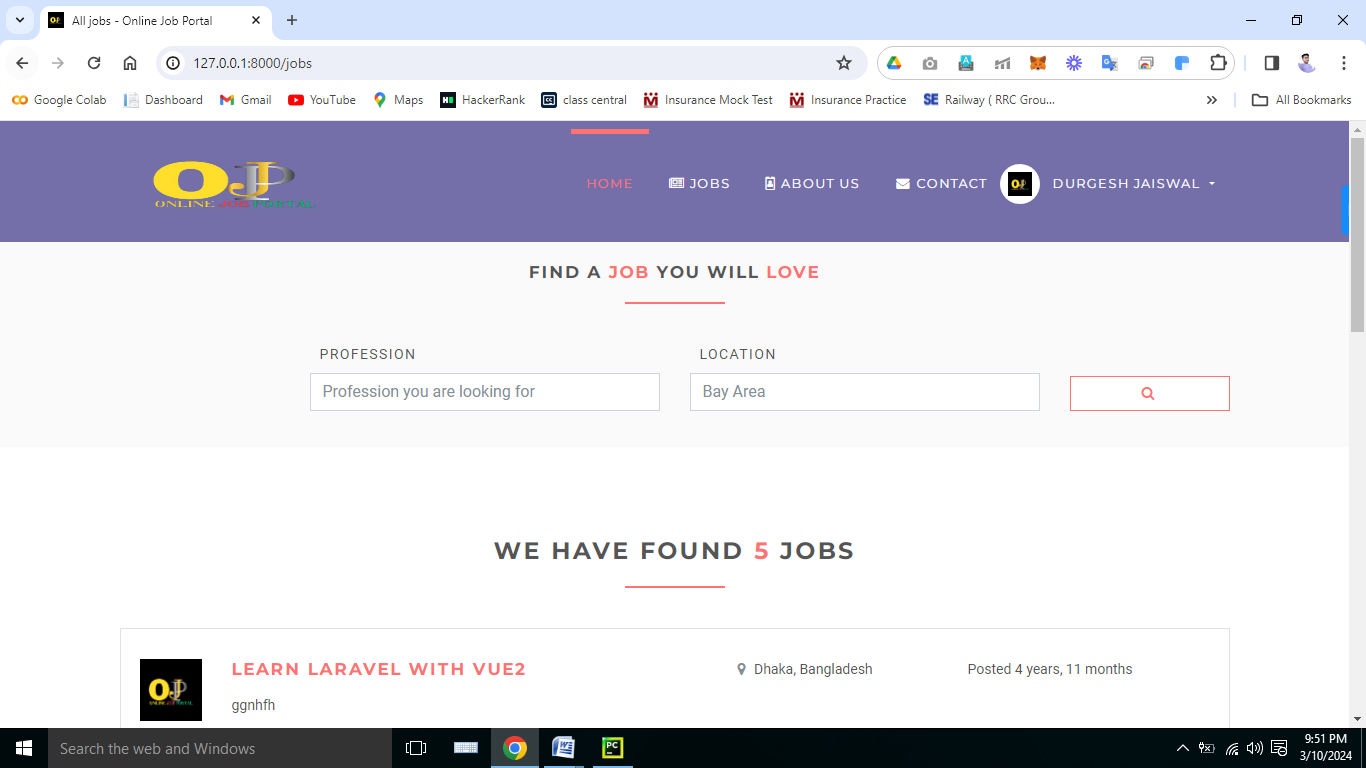
****

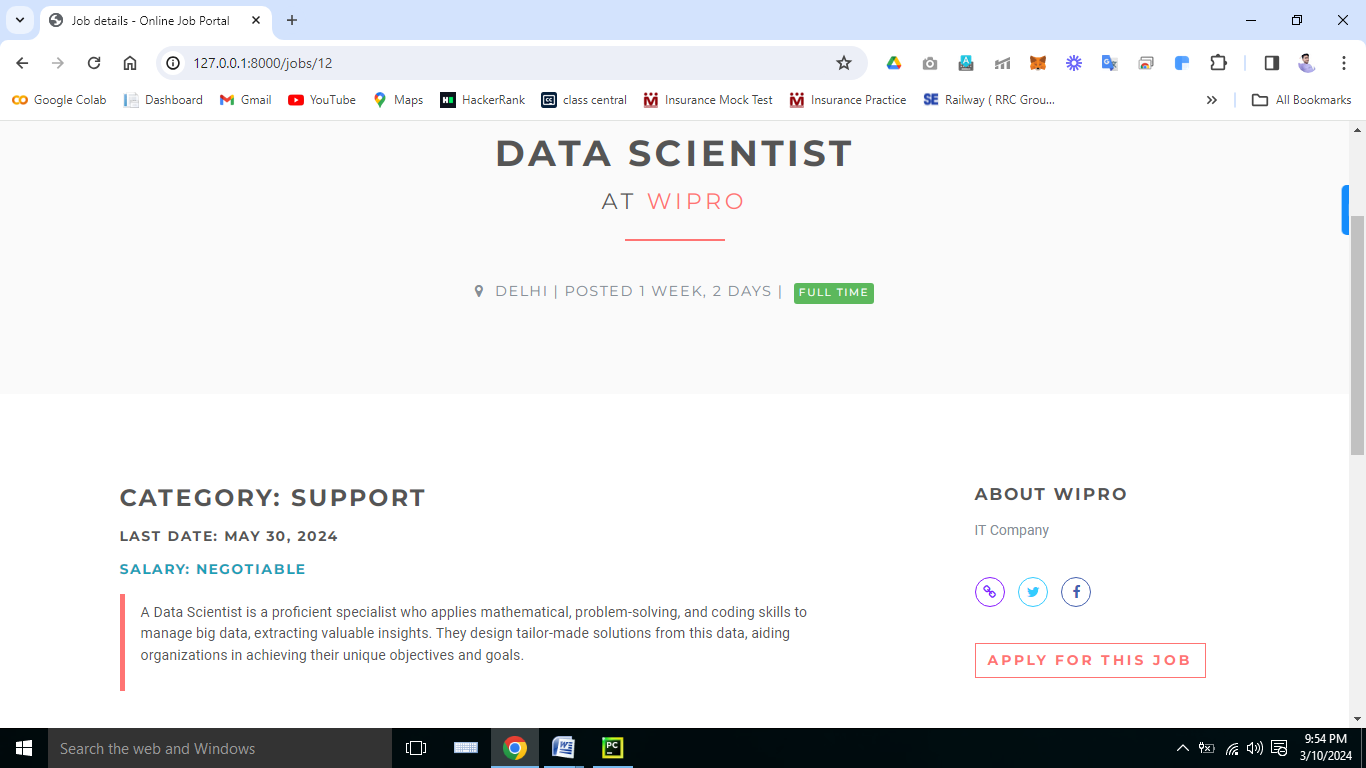
**Interface: Jobseeker Dashboard**

****

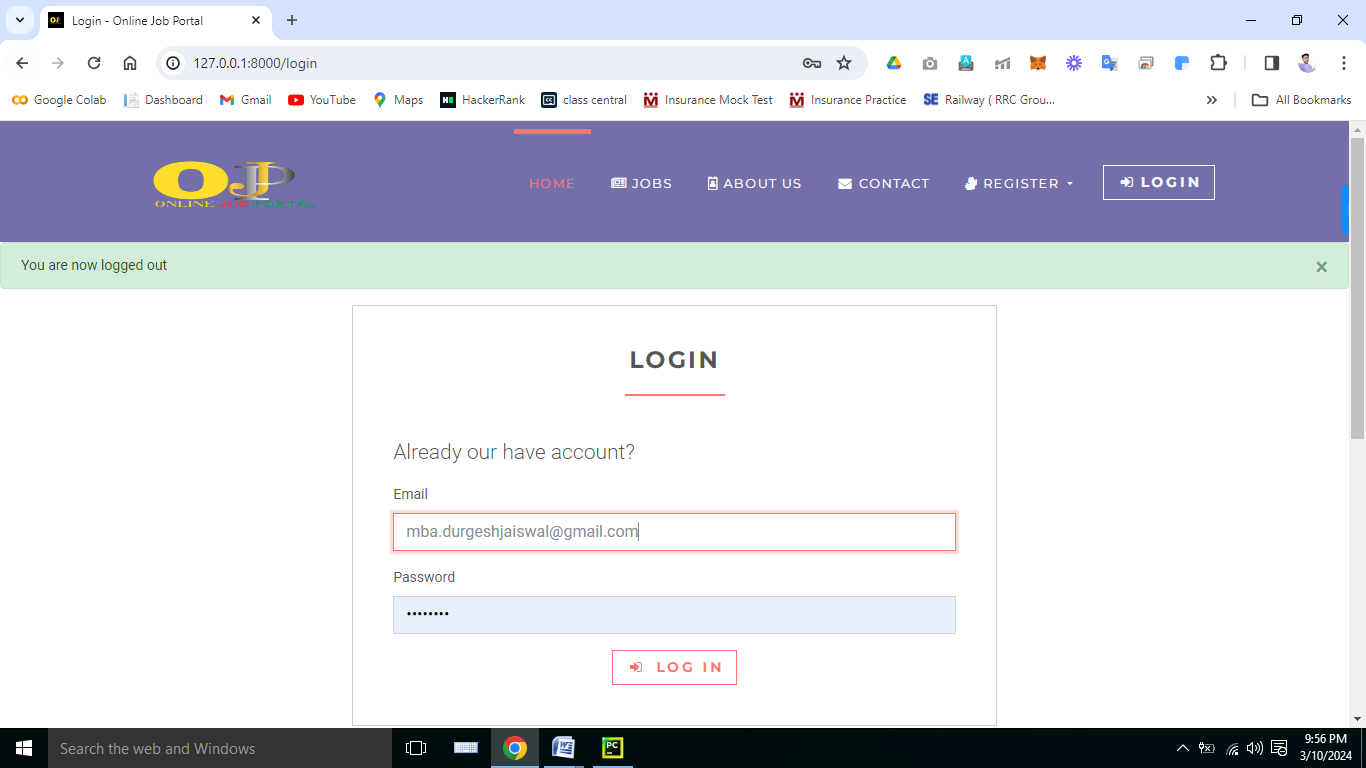
****

**Interface: Jobs (Jobseeker)**

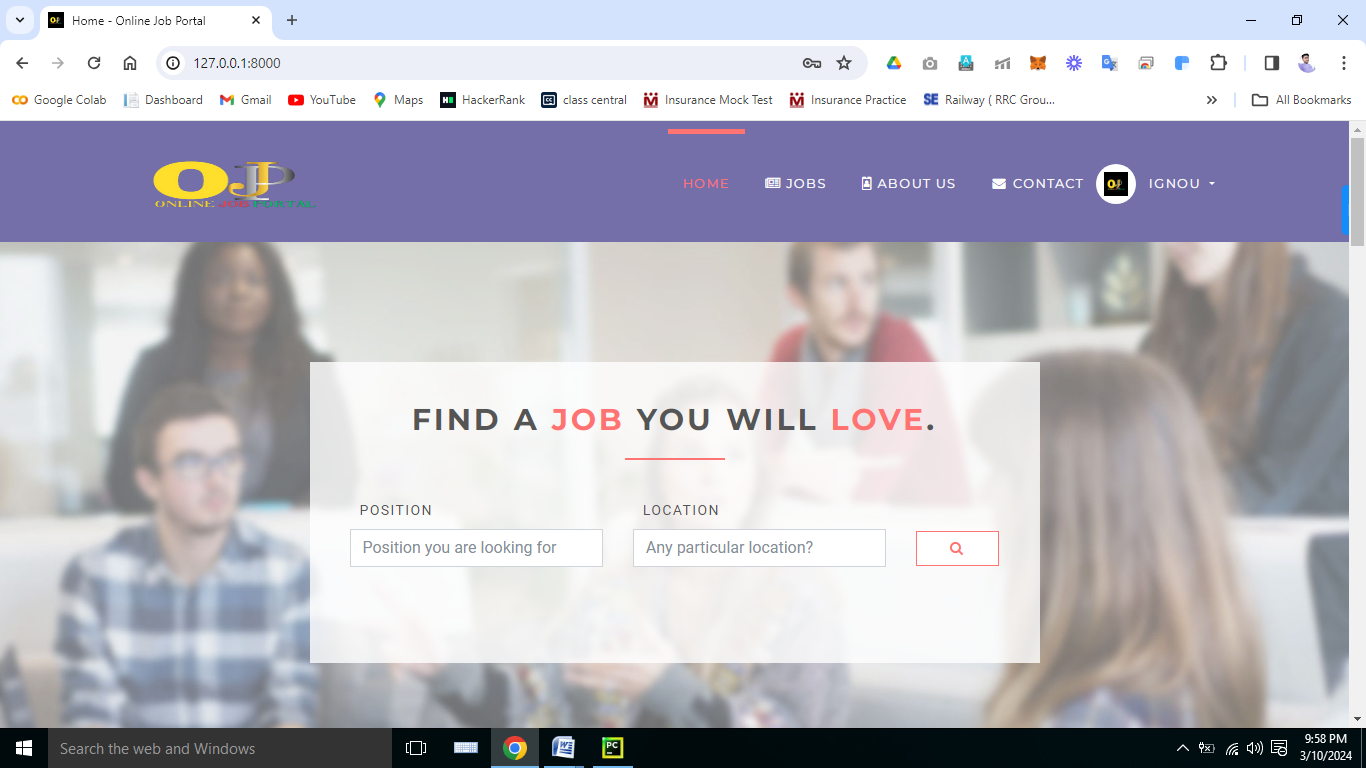
****

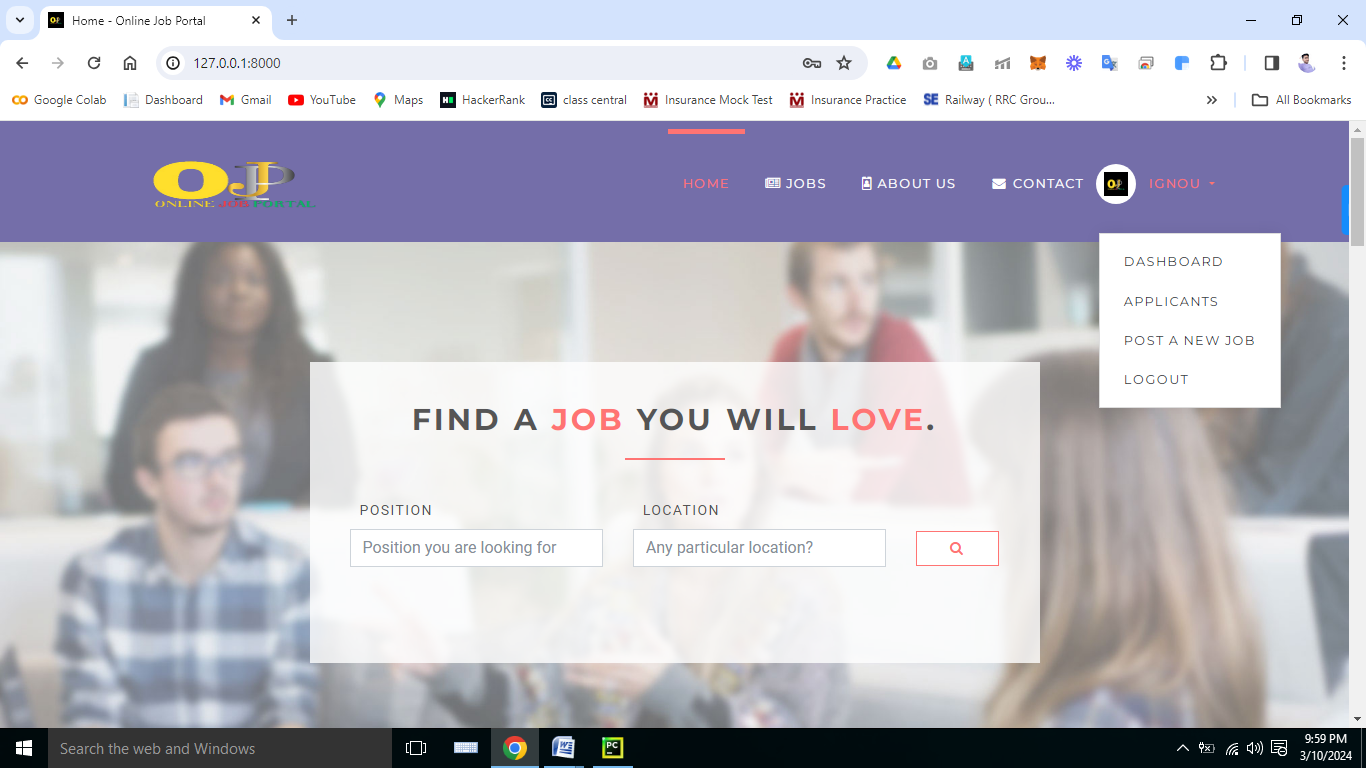
****

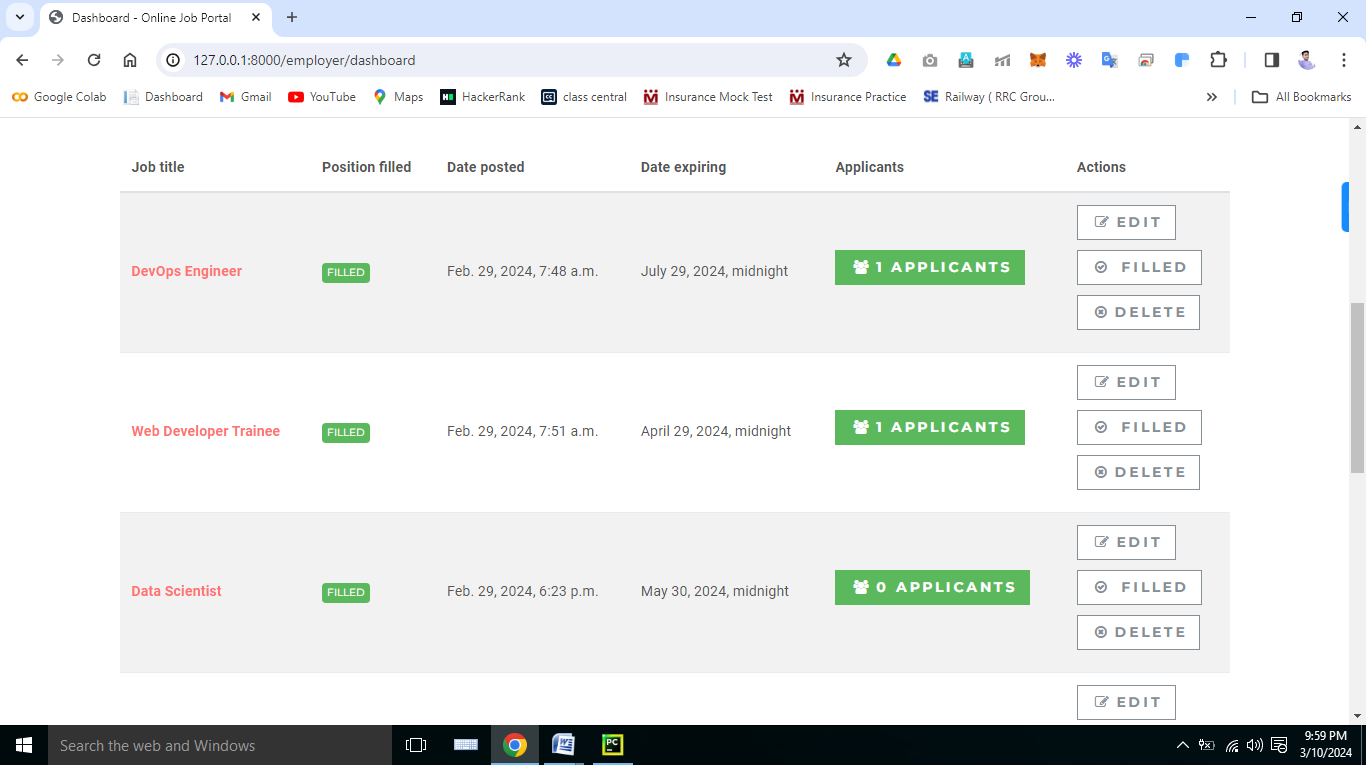
**Interface: Login (Employer)**

****

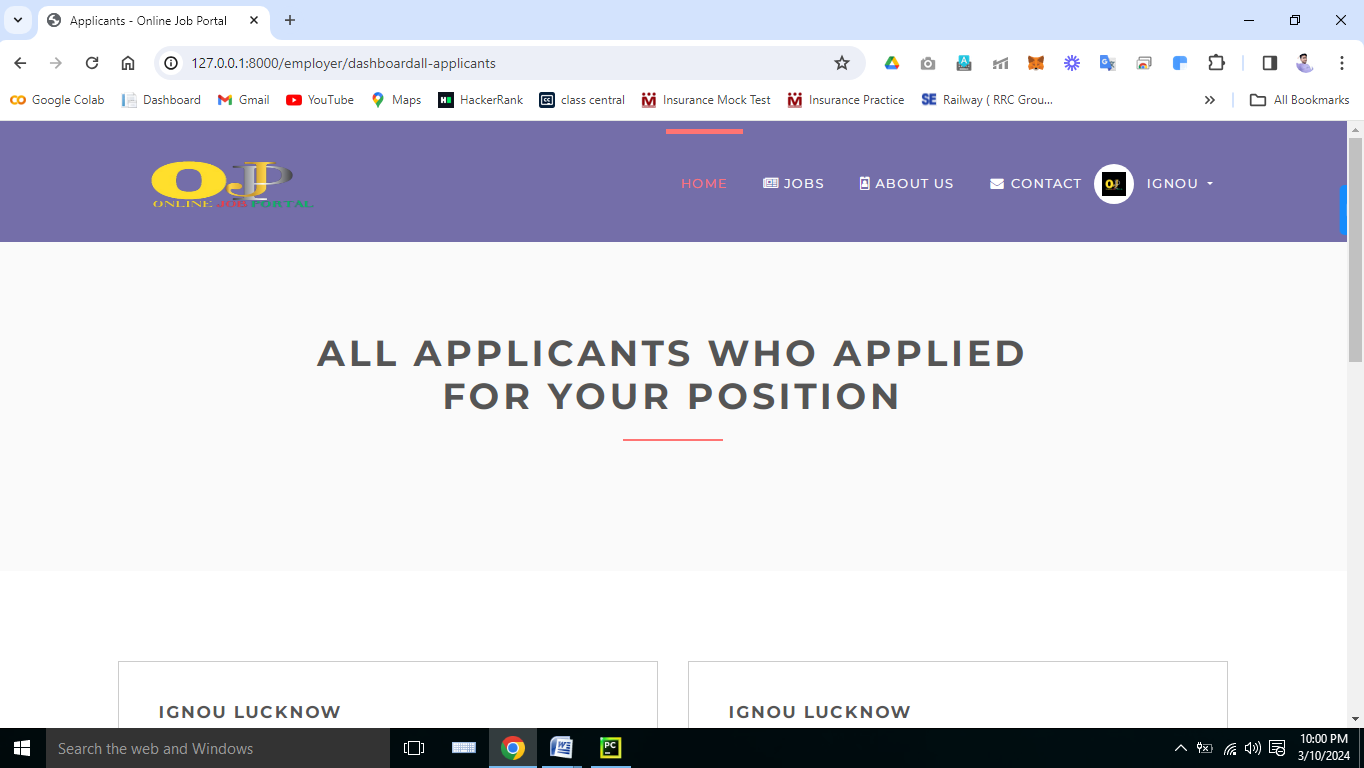
**Interface: Employer Dashboard**

****

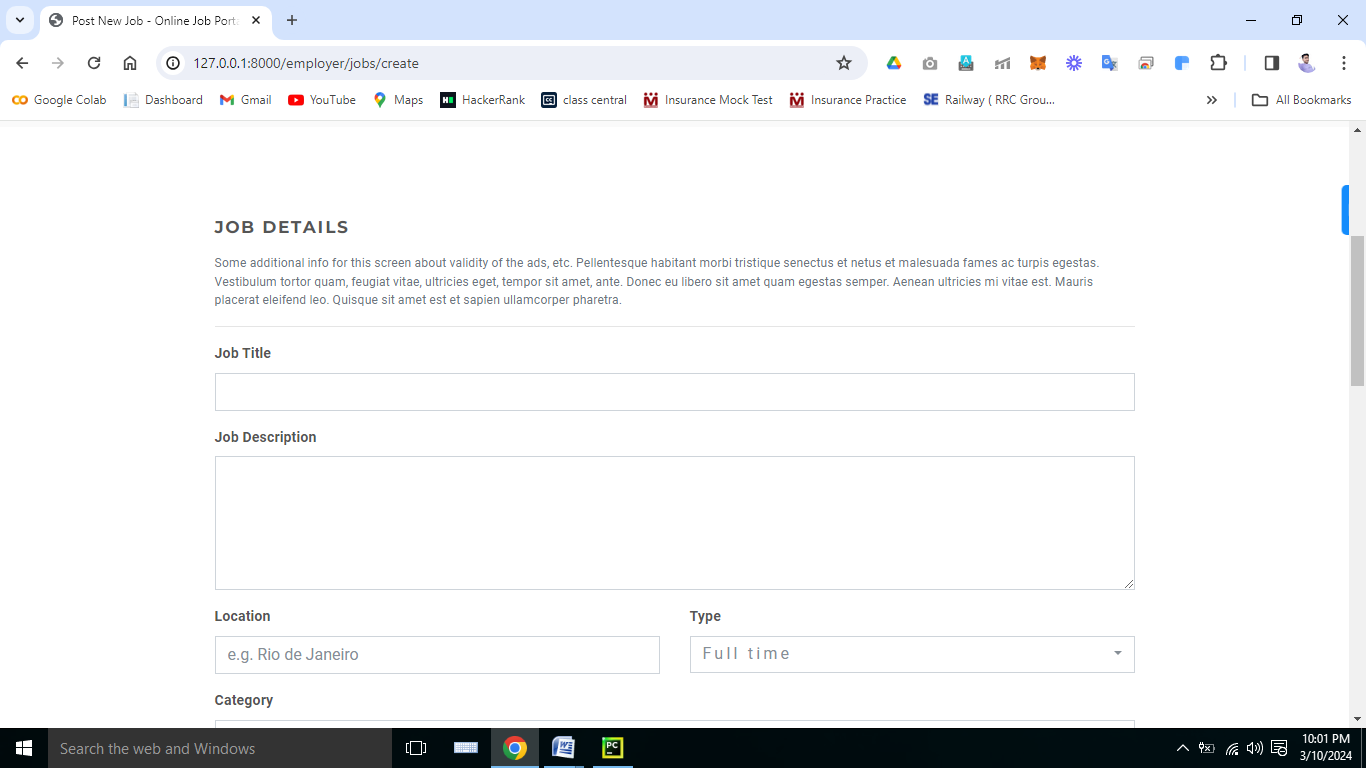
****

****

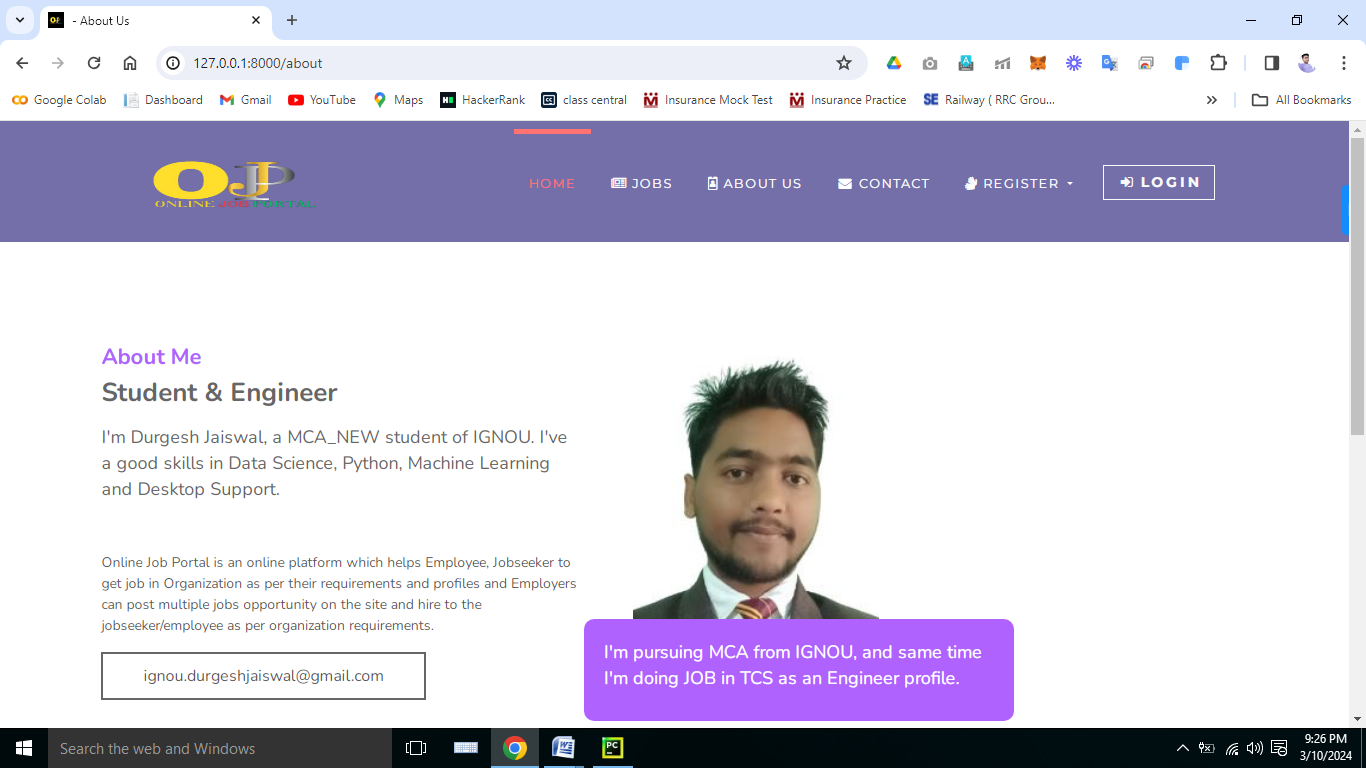
**Interface: Applicants (Employer)**

****

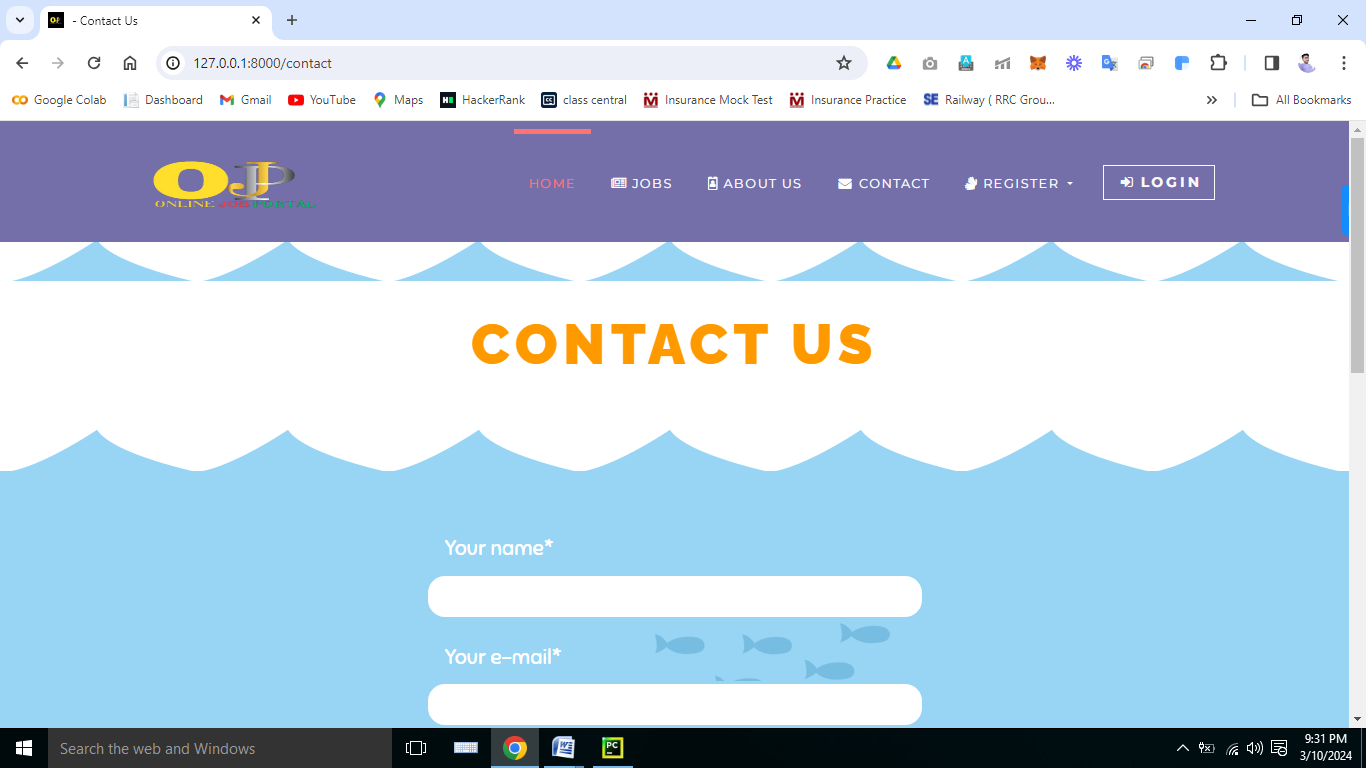
**Interface: Job Posting (Employer)**

****

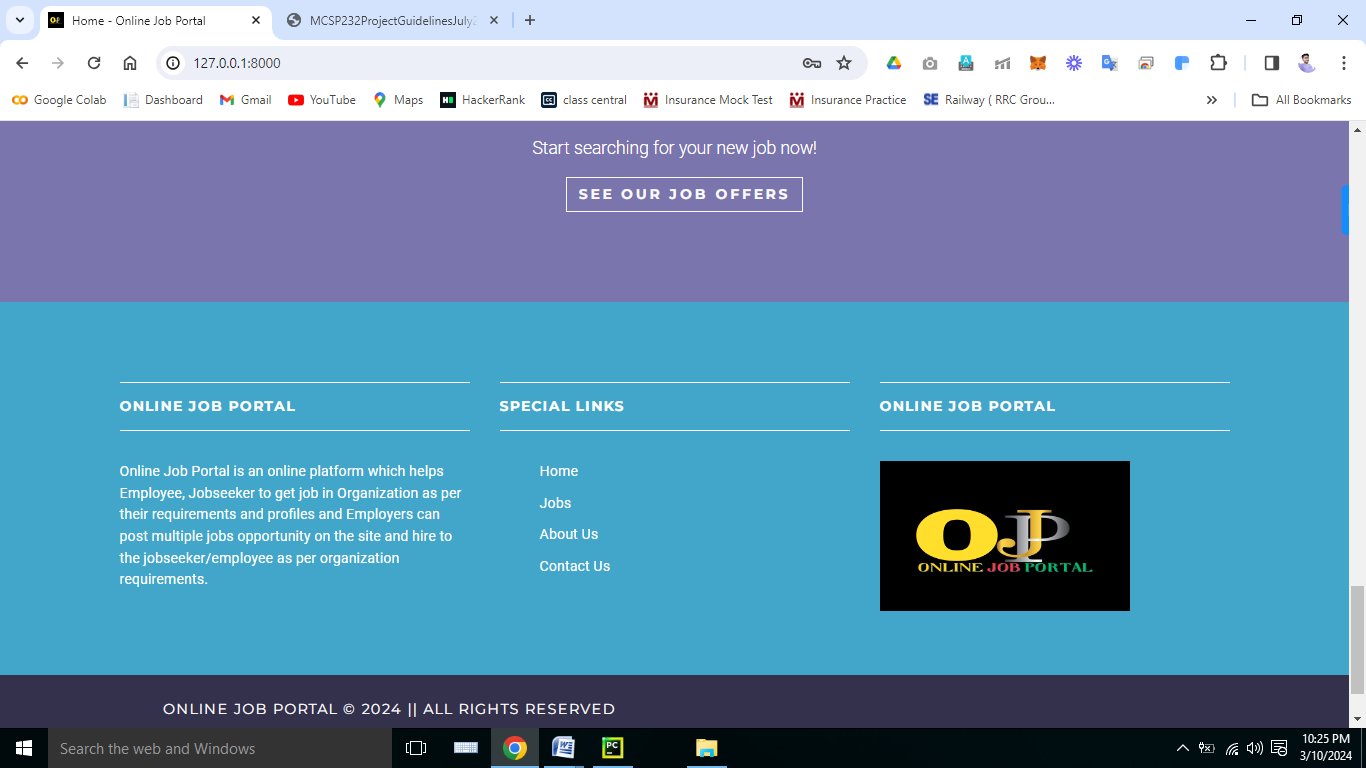
**Interface: About “Online Job Portal”**

****

**Interface: Contact Us**

****

**Interface: Footer**

****

### 3.5) Test Cases

**Unit testing**

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white-box style),to ensure that the specific unction is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

Depending on the organization's expectations for software development, unit testing might include static code analysis, data flow analysis metrics analysis, peer code reviews, code coverage analysis and other software verification practices.

Unit Testing is a software verification and validation method, where the programmer gains confidence that individual units of source codes are fit for use. A unit is the smallest testable part of an application. In procedural programming, a unit may be an individual program, function, procedure, etc., while in object-oriented programming, the smallest unit is a method, which may belong to a base/super class, abstract class or derived/child class. Unit testing can be done by something as simple as stepping through code in a debugger; modern applications include the use of a test framework such as xUnit.

Ideally, each test case is independent from the others: substitutes like method stubs, mock objects, fakes and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its requirements and behaves as intended. Its implementation can vary from being very manual (pencil and paper) to being formalized as part of build automation.

### System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification.

System Testing (sometimes called System and Testing, abbreviated **I&T**) is the activity of software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing.

System testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an System test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

The purpose of System testing, is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using Black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test that all components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing.

4. Coding

base.html

{% load static %}  
<!DOCTYPE html>  
<html>  
  
<head>  
 <meta charset="utf-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <title>{% block title %} {% endblock %} - Online Job Portal </title>  
 <meta name="description" content="">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css" >  
 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">  
 <meta name="robots" content="all,follow">  
 <!-- Bootstrap CSS-->  
 <link rel="stylesheet"  
 href="{% static 'vendor/bootstrap/css/bootstrap.min.css' %}">  
 <!-- Font Awesome CSS-->  
 <link rel="stylesheet"  
 href="{% static 'vendor/font-awesome/css/font-awesome.min.css' %}">  
 <!-- Google fonts - Roboto for copy, Montserrat for headings-->  
 <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Roboto:300,400,700">  
 <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Montserrat:400,700">  
 <!-- owl carousel-->  
 <link rel="stylesheet"  
 href="{% static 'vendor/owl.carousel/assets/owl.carousel.css' %}">  
 <link rel="stylesheet"  
 href="{% static 'vendor/owl.carousel/assets/owl.theme.default.css' %}">  
 <!-- theme stylesheet-->  
 <link rel="stylesheet" href="{% static 'css/style.default.css' %}"  
 id="theme-stylesheet">  
 <!-- Custom stylesheet - for your changes-->  
 <link rel="stylesheet" href="{% static 'css/custom.css' %}">  
 {% block styles %}{% endblock %}  
 <!-- Favicon-->  
 <link rel="shortcut icon" href="static/img/11.png">  
 <!-- Tweaks for older IEs--><!--[if lt IE 9]>  
 <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>  
 <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script><![endif]-->  
 <style>  
 .footer .footer-row {  
 display: flex;  
 flex-wrap: wrap;  
 justify-content: space-between;  
 gap: 3.5rem;  
 padding: 0px;  
}  
.footer-row .footer-col h4 {  
 color: #fff;  
 font-size: 1.2rem;  
 font-weight: 400;  
}  
.footer-col .links {  
 margin-top: 0px;  
}  
.footer-col .links li {  
 list-style: none;  
 margin-bottom: 10px;  
}  
  
 </style>  
</head>  
<body>  
<!-- navbar-->  
<header class="header">  
 <nav class="navbar navbar-expand-lg bg-blue">  
 <div class="container">  
 <a href="/" class="navbar-brand">  
 <img src="{% static 'img/22.png' %}" alt="logo" class="d-none d-lg-block" height="110" width="230">  
 <span class="sr-only">Home</span>  
 </a>  
 <button type="button" data-toggle="collapse" data-target="#navbarSupportedContent"  
 aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation"  
 class="navbar-toggler navbar-toggler-right">Menu<i class="fa fa-bars"></i>  
 </button>  
 <div id="navbarSupportedContent" class="collapse navbar-collapse">  
 <ul class="navbar-nav ml-auto">  
 <li class="nav-item active">  
 <a href="/" class="nav-link">Home <span class="sr-only">(current)</span></a>  
 </li>  
 <li class="nav-item ">  
 <a href="jobs" class="nav-link"><i class="fa fa-newspaper-o"></i>Jobs </a>  
 </li>  
 <li class="nav-item">  
 <a href="about" class="nav-link"><i class="fa fa-id-badge"></i>About Us</a>  
 </li>  
 <li class="nav-item">  
 <a href="contact" class="nav-link"><i class="fa fa-fw fa-envelope"></i>Contact</a>  
 </li>  
 {% if request.user.is\_authenticated and request.user.role == 'employer' %}  
 <li class="nav-item dropdown">  
 <a id="clientZone" href="#" data-toggle="dropdown"  
 aria-haspopup="true" aria-expanded="false"  
 class="nav-link dropdown-toggle has-img mb-3 mb-lg-0 mt-3 mt-lg-0">  
 <div class="client-menu-image">  
 <div class="inner">  
 <img src="{% static 'img/11.png' %}"  
 alt="company image"  
 class="img-fluid">  
 </div>  
 </div>  
 {% if request.user.role == 'employee' %}  
 {{ request.user.first\_name }} {{ request.user.last\_name }}  
 {% else %}  
 {{ request.user.first\_name }}  
 {% endif %}  
 </a>  
 <div aria-labelledby="clientZone" class="dropdown-menu">  
 <a href="{% url 'jobs:employer-dashboard' %}" class="dropdown-item">Dashboard</a>  
 <a href="{% url 'jobs:employer-all-applicants' %}" class="dropdown-item">Applicants </a>  
 <a href="{% url 'jobs:employer-jobs-create' %}" class="dropdown-item">Post a new  
 job </a>  
 <a href="{% url 'accounts:logout' %}" class="dropdown-item">Logout</a>  
 </div>  
 </li>  
 {% elif request.user.is\_authenticated and request.user.role == 'employee' %}  
 <li class="nav-item dropdown">  
 <a id="clientZone" href="#" data-toggle="dropdown"  
 aria-haspopup="true" aria-expanded="false"  
 class="nav-link dropdown-toggle has-img mb-3 mb-lg-0 mt-3 mt-lg-0">  
 <div class="client-menu-image">  
 <div class="inner">  
 <img src="{% static 'img/11.png' %}"  
 alt="company image"  
 class="img-fluid">  
 </div>  
 </div>  
 {% if request.user.role == 'employee' %}  
 {{ request.user.first\_name }} {{ request.user.last\_name }}  
 {% else %}  
 {{ request.user.first\_name }}  
 {% endif %}  
 </a>  
 <div aria-labelledby="clientZone" class="dropdown-menu">  
 <a href="{% url 'accounts:employer-profile-update' %}" class="dropdown-item">Edit Profile</a>  
 <a href="{% url 'accounts:logout' %}" class="dropdown-item">Logout</a>  
 </div>  
 </li>  
 {% else %}  
 <li class="nav-item dropdown">  
 <a id="pages" href="#" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false"  
 class="nav-link dropdown-toggle"><i class="fa fa-signing"></i>Register  
 </a>  
 <div aria-labelledby="pages" class="dropdown-menu">  
 <a href="{% url 'accounts:employee-register' %}" class="dropdown-item">Jobseeker</a>  
 <a href="{% url 'accounts:employer-register' %}" class="dropdown-item">Employer</a>  
 </div>  
 </li>  
 <li class="nav-item" style="color: #fff">  
 <a href="{% url 'accounts:login' %}" class="btn navbar-btn btn-outline-light mb-5 mb-lg-0">  
 <i class="fa fa-sign-in"></i>Login  
 </a>  
 </li>  
 {% endif %}  
 </ul>  
 </div>  
 </div>  
 </nav>  
</header>  
<div>  
 {% if messages %}  
  
 {% for message in messages %}  
 <div class="alert alert-{{ message.tags }} alert-dismissible fade show" role="alert">  
 {{ message }}  
 <button type="button" class="close" data-dismiss="alert" aria-label="Close">  
 <span aria-hidden="true">&times;</span>  
 </button>  
 </div>  
{# <li{% if message.tags %} class="alert alert-{{ message.tags }}"{% endif %}>{{ message }}</li>#}  
 {% endfor %}  
  
 {% endif %}  
 {% block content %} {% endblock %}  
</div>  
<footer class="footer">  
 <div class="footer\_\_block">  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-4 col-md-12 mb-5">  
 <h4 class="h5"> Online Job Portal </h4>  
 <p> Online Job Portal is an online platform which helps Employee, Jobseeker to get job in Organization as per their requirements and profiles and Employers can post multiple jobs opportunity on the site and hire to the jobseeker/employee as per organization requirements. </p>  
 </div>  
 <div class="col-lg-4 col-md-12 mb-5">  
 <h4 class="h5">Special Links </h4>  
 <div class="footer-row">  
 <div class="footer-col">  
 <ul class="links">  
 <li><a href="/"> Home </a></li>  
 <li><a href="jobs"> Jobs </a></li>  
 <li><a href="about"> About Us</a></li>  
 <li><a href="contact"> Contact Us </a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 <div class="col-lg-4 col-md-12 mb-5">  
 <h4 class="h5">Online Job Portal </h4>  
 <p><img src="{% static 'img/11.png' %}" height="150" width="250"></p>  
 </div>  
 </div>  
 </div>  
 <div class="footer\_\_copyright">  
 <div class="container">  
 <div class="row">  
 <div class="col-md-6 text-md-left text-center">  
 <p align="center"> Online Job Portal © 2024 || All rights reserved </p>  
 </div>  
 </div>  
 </div>  
 </div>  
</footer>  
  
<script src="{% static 'vendor/jquery/jquery.min.js' %}"></script>  
<script src="{% static 'vendor/popper.js/umd/popper.min.js' %}"></script>  
<script src="{% static 'vendor/bootstrap/js/bootstrap.min.js' %}"></script>  
<script src="{% static 'vendor/jquery.cookie/jquery.cookie.js' %}"></script>  
<script src="{% static 'vendor/owl.carousel/owl.carousel.min.js' %}"></script>  
<script src="{% static 'vendor/bootstrap-select/js/bootstrap-select.min.js' %}"></script>  
<script src="{% static 'js/front.js' %}"></script>  
  
{% block javascripts %}{% endblock %}  
  
</body>  
  
</html>

home.html

{% extends 'base.html' %}  
{% load static %}  
{% block title %}  
 Home  
{% endblock %}  
  
{% block content %}  
  
 <div id="loginModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true"  
 class="modal fade">  
 <div role="document" class="modal-dialog">  
 <div class="modal-content">  
 <div class="modal-header">  
 <h4 id="exampleModalLabel" class="modal-title">Customer Login</h4>  
 <button type="button" data-dismiss="modal" aria-label="Close" class="close">  
 <span aria-hidden="true">×</span>  
 </button>  
 </div>  
 <div class="modal-body">  
 <form action="" method="post">  
 <div class="form-group">  
 <input id="email\_modal" type="text" placeholder="email" class="form-control">  
 </div>  
 <div class="form-group">  
 <input id="password\_modal" type="password" placeholder="password" class="form-control">  
 </div>  
 <p class="text-center">  
 <button type="button" class="btn btn-outline-white-primary"><i class="fa fa-sign-in"></i>  
 Log in  
 </button>  
 </p>  
 </form>  
 <p class="text-center text-muted">Not registered yet?</p>  
 <p class="text-center text-muted"><a href="client-register.html"><strong>Register now</strong></a>!  
 It is easy and done in 1 minute and gives you access to special discounts and much more!</p>  
 </div>  
 </div>  
 </div>  
 </div>  
 <!-- \*\*\* LOGIN MODAL END \*\*\*-->  
 <section class="job-form-section job-form-section--image">  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-8 mx-auto">  
 <div class="job-form-box">  
 <h2 class="heading">Find a <span class="accent">job</span> you will  
 <span class="accent">love</span>.  
 </h2>  
 <form id="job-main-form" method="get" action="{% url 'jobs:searh' %}" class="job-main-form">  
 <div class="controls">  
 <div class="row align-items-center">  
 <div class="col-md-5">  
 <div class="form-group">  
 <label for="profession">Position</label>  
 <input type="text" id="profession" name="position"  
 placeholder="Position you are looking for" class="form-control">  
 </div>  
 </div>  
 <div class="col-md-5">  
 <div class="form-group">  
 <label for="location">Location</label>  
 <input type="text" id="location" name="location"  
 placeholder="Any particular location?" class="form-control">  
 </div>  
 </div>  
 <div class="col-md-2">  
 <button type="submit"  
 class="btn btn-outline-white-primary job-main-form\_\_button">  
 <i class="fa fa-search"></i>  
 </button>  
 </div>  
 </div>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
 </div>  
 </section>  
 <section class="bg-light-gray">  
 <div class="container">  
 <h3 class="heading">Featured jobs</h3>  
 <div class="row featured align-items-stretch">  
 {% for job in jobs %}  
 <div class="col-lg-4 mb-5 mb-lg-0">  
 <div class="box-image-text bg-visible full-height">  
 <div class="top">  
 <a href="#">  
 <div class="image">  
 <img src="{% static 'img/11.png' %}" alt="" class="img-fluid">  
 </div>  
 <div class="bg"></div>  
 <div class="logo">  
 <img src="{% static '#' %}" alt="" style="max-width: 80px;">  
 </div>  
 </a>  
 </div>  
 <div class="content">  
 <h5><a href="{% url 'jobs:jobs-detail' job.id %}">{{ job.title }}</a></h5>  
 <p class="featured\_\_details"><i class="fa fa-map-marker job\_\_location"></i>  
 {{ job.location }}  
 {% if job.type == '1' %}  
 <span class="badge featured-badge badge-success">Full time</span>  
 {% elif job.type == '2' %}  
 <span class="badge featured-badge badge-primary">Part time</span>  
 {% else %}  
 <span style="color: #ffffff;" class="badge featured-badge badge-warning">Internship</span>  
 {% endif %}  
 </p>  
 <p>{{ job.description }}</p>  
 </div>  
 </div>  
 </div>  
 {% endfor %}  
 </div>  
 </div>  
 </section>  
 <section>  
 <div class="container">  
 <h4 class="heading">Trending this month</h4>  
 {% for trending in trendings %}  
 <div class="job-listing job-listing--last">  
 <div class="row">  
 <div class="col-md-12 col-lg-6">  
 <div class="row">  
 <div class="col-2">  
 <img src="{% static 'img/11.png' %}"  
 alt="ShareBoardd " class="img-fluid">  
 </div>  
 <div class="col-10">  
 <h4 class="job\_\_title">  
 <a href="{% url 'jobs:jobs-detail' trending.id %}">{{ trending.title }}</a>  
 </h4>  
 <p class="job\_\_company">  
 {{ trending.company\_name }}  
 </p>  
 </div>  
 </div>  
 </div>  
 <div class="col-10 col-md-3 col-lg-2 ml-auto"><i class="fa fa-map-marker job\_\_location"></i>  
 {{ trending.location }}  
 </div>  
 <div class="col-10 col-md-3 col-lg-3 ml-auto">  
 <p>Posted {{ trending.created\_at|timesince }}</p>  
 </div>  
 <div class="col-sm-12 col-md-2 col-lg-1">  
 <div class="job\_\_star">  
 <a href="#" data-toggle="tooltip" data-placement="top"  
 title="Save to favourites" class="job\_\_star\_\_link">  
 <i class="fa fa-star"></i>  
 </a>  
 </div>  
 </div>  
 </div>  
 </div>  
 {% endfor %}  
 </div>  
 </section>  
 <section style="background-image: url({% static 'img/startup.jpg' %}); " class="section-divider">  
 <div class="overlay"></div>  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-12 text-center">  
 <p>Start searching for your new job now! </p>  
 <p><a href="{% url 'jobs:jobs' %}" class="btn btn-outline-light">See our job offers </a></p>  
 </div>  
 </div>  
 </div>  
 </section>  
  
{% endblock %}

jobs.html (jobs)

{% extends 'base.html' %}  
{% load static %}  
{% block title %}  
 All jobs  
{% endblock %}  
  
{% block content %}  
  
 <section class="job-form-section job-form-section--compressed">  
 <div class="container">  
 <h4 class="heading">Find a <span class="accent">job </span> you will <span class="accent">love</span></h4>  
 <form id="job-main-form" method="get" action="#" class="job-main-form">  
 <div class="controls">  
 <div class="row align-items-end">  
 <div class="col-sm-1 col-lg-2"></div>  
 <div class="col-sm-5 col-lg-4 form-group">  
 <label for="profession">Profession</label>  
 <input type="text" id="profession" name="profession"  
 placeholder="Profession you are looking for" class="form-control">  
 </div>  
 <div class="col-sm-5 col-lg-4 form-group">  
 <label for="location">Location</label>  
 <input type="text" id="location" name="location" placeholder="Any particular location?"  
 value="Bay Area" class="form-control">  
 </div>  
 <div class="col-sm-1 col-lg-2 form-group">  
 <button type="submit" name="name"  
 class="btn btn-outline-white-primary job-main-form\_\_button">  
 <i class="fa fa-search"></i>  
 </button>  
 </div>  
 </div>  
 </div>  
 </form>  
 </div>  
 </section>  
 <section>  
 <div class="container">  
 <h3 class="heading">We have found <span class="accent">{{ jobs.count }}</span> jobs</h3>  
 {% for job in jobs %}  
 <div class="job-listing ">  
 <div class="row">  
 <div class="col-md-12 col-lg-6">  
 <div class="row">  
 <div class="col-2">  
 <img src="{% static 'img/11.png' %}"  
 alt="Ipsioou " class="img-fluid">  
 </div>  
 <div class="col-10">  
 <h4 class="job\_\_title"><a  
 href="{% url 'jobs:jobs-detail' job.id %}">{{ job.title }}</a></h4>  
 <p class="job\_\_company">  
 {{ job.company\_name }}  
 </p>  
 </div>  
 </div>  
 </div>  
 <div class="col-10 col-md-3 col-lg-2 ml-auto"><i class="fa fa-map-marker job\_\_location"></i>  
 {{ job.location }}  
 </div>  
 <div class="col-10 col-md-3 col-lg-3 ml-auto">  
 <p>Posted {{ job.created\_at|timesince }}</p>  
 </div>  
 </div>  
 </div>  
 {% endfor %}  
 {% if is\_paginated %}  
 <div class="row">  
 <div class="col-md-12">  
 <nav aria-label="Page navigation example" class="d-flex justify-content-center mt-4 mb-4">  
 <ul class="pagination">  
 {% if page\_obj.has\_previous %}  
 <li class="page-item">  
 <a href="?page={{ page\_obj.previous\_page\_number }}"  
 class="page-link">&laquo;</a>  
 </li>  
 {% else %}  
 <li class="page-item disabled">  
 <a class="page-link">&laquo;</a>  
 </li>  
 {% endif %}  
 {% for i in page\_obj.paginator.page\_range %}  
 {% if page\_obj.number == i %}  
 <li class="page-item active">  
 <a class="page-link">{{ i }}</a>  
 </li>  
 {% else %}  
 <li class="page-item">  
 <a href="?page={{ i }}" class="page-link">{{ i }}</a>  
 </li>  
 {% endif %}  
 {% endfor %}  
 {% if page\_obj.has\_next %}  
 <li class="page-item">  
 <a href="?page={{ page\_obj.next\_page\_number }}" class="page-link">&raquo;</a>  
 </li>  
 {% else %}  
 <li class="page-item disabled">  
 <a class="page-link">&raquo;</a>  
 </li>  
 {% endif %}  
 </ul>  
 </nav>  
 </div>  
 </div>  
 {% endif %}  
 </div>  
 </section>  
  
{% endblock %}

create.html (jobs)

{% extends 'base.html' %}  
{% load static %}  
{% block title %}  
 {{ title }}  
{% endblock %}  
  
{% block styles %}  
 <link rel="stylesheet" href="{% static 'vendor/bootstrap-select/css/bootstrap-select.min.css' %}">  
{% endblock %}  
  
{% block content %}  
  
 <section class="bg-light-gray">  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-8 mx-auto">  
 <h1 class="heading">Add a new position</h1>  
 </div>  
 </div>  
 </div>  
 </section>  
 <section>  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-10 mx-auto">  
 {% if form.errors %}  
 {% for field in form %}  
 {% for error in field.errors %}  
 <div class="alert alert-danger alert-dismissable">  
 <a href="#" class="close" data-dismiss="alert" aria-label="close">&times;</a>  
 <strong>{{ field.name }} {{ error|escape }}</strong>  
 </div>  
 {% endfor %}  
 {% endfor %}  
 {% endif %}  
 <form id="job-main-form" method="post" action="" class="job-add-form">  
 {% csrf\_token %}  
 <div class="row">  
 <div class="col-lg-12">  
 <h4>Job details</h4>  
 <p class="text-muted text-small">Some additional info for this screen about validity of  
 the ads, etc. Pellentesque habitant morbi tristique senectus et netus et malesuada  
 fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget,  
 tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean  
 ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien  
 ullamcorper pharetra. </p>  
 <hr>  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-lg-12 form-group">  
 <label for="title">Job Title</label>  
 <input id="title" name="title" type="text" class="form-control">  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-lg-12 form-group">  
 <label for="text">Job Description</label>  
 <textarea id="text" name="description" rows="5" class="form-control"></textarea>  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-lg-6 form-group">  
 <label for="location">Location</label>  
 <input id="location" name="location" type="text" placeholder="e.g. Rio de Janeiro"  
 class="form-control">  
 </div>  
 <div class="col-lg-6 form-group">  
 <label for="type">Type</label>  
 <select id="type" name="type" placeholder="Choose job type"  
 class="form-control select2">  
 <option value="1">Full time</option>  
 <option value="2">Part time</option>  
 <option value="3">Internship</option>  
 </select>  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-lg-12 form-group">  
 <label for="category">Category</label>  
 <select id="category" name="category" placeholder="Choose category"  
 class="form-control select2">  
 <option value="web-design">Web design</option>  
 <option value="graphic-design">Graphic design</option>  
 <option value="web-development">Web development</option>  
 <option value="human-resource">Human Resources</option>  
 <option value="support">Support</option>  
 <option value="android">Android</option>  
 </select>  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-xl-6 form-group">  
 <label for="apply\_url">Apply URL <span  
 class="note">(users will apply on your website)</span></label>  
 <input id="apply\_url" type="url" class="form-control">  
 </div>  
 <div class="col-xl-6 form-group">  
 <label for="validity">Validity of the post</label>  
 <input id="validity" name="last\_date" type="date" class="form-control">  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-lg-12">  
 <hr class="margin-bottom--big">  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-lg-12">  
 <h4>Company details</h4>  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-lg-12 form-group">  
 <label for="company\_name">Company name</label>  
 <input id="company\_name" name="company\_name" type="text" class="form-control">  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-lg-12 form-group">  
 <label for="company\_description">Company Description</label>  
 <textarea id="company\_description" name="company\_description" rows="3"  
 class="form-control"></textarea>  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-lg-4 form-group">  
 <label for="company\_website">Website</label>  
 <input id="company\_website" name="website" type="url" class="form-control">  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-lg-12">  
 <hr>  
 <div class="checkbox text-center">  
 <label>  
 <input type="checkbox"> I agree with the <a href="#">Terms and conditions</a>.  
 </label>  
 </div>  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-lg-12 text-center">  
 <hr>  
 <button type="submit" class="btn btn-outline-white-primary"><i class="fa fa-magic"></i>  
 Save and publish  
 </button>  
 </div>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
 </section>  
  
{% endblock %}

details.html (jobs)

{% extends 'base.html' %}  
{% load static %}  
{% load is\_already\_applied %}  
{% block title %}  
 Job details  
{% endblock %}  
  
{% block content %}  
  
 <section class="bg-light-gray">  
 <div class="container">  
 <h1 class="heading">{{ job.title }}<br>  
 <small>at <a href="#"> {{ job.company\_name }}</a></small>  
 </h1>  
 <div class="job-detail-description"><i class="fa fa-map-marker job\_\_location"> </i>  
 {{ job.location }} | Posted {{ job.created\_at|timesince }} |  
 {% if job.type == '1' %}  
 <span class="badge featured-badge badge-success">Full time</span>  
 {% elif job.type == '2' %}  
 <span class="badge featured-badge badge-primary">Part time</span>  
 {% else %}  
 <span style="color: #ffffff;" class="badge featured-badge badge-warning">Internship</span>  
 {% endif %}  
 </div>  
 </div>  
 </section>  
 <section>  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-8">  
 <h3>Category: {{ job.category }}</h3>  
 <h5>Last date: {{ job.last\_date|date }}</h5>  
 <h5 class="text-info">Salary: Negotiable</h5>  
 <blockquote class="blockquote">  
 <p>{{ job.description }}</p>  
 </blockquote>  
 <div class="job-detail\_\_apply-bottom">  
 {% if request.user.is\_authenticated %}  
 {% is\_already\_applied job request.user as is\_applied %}  
 {% if is\_applied %}  
 <input type="button" class="btn btn-outline-white-primary" value="Already applied"/>  
 {% else %}  
 <form action="{% url 'jobs:apply-job' job.id %}" method="post">  
 {% csrf\_token %}  
 <input type="hidden" name="job" value="{{ job.id }}">  
 <input type="submit" class="btn btn-outline-white-primary"  
 value="Apply for this job"/>  
 </form>  
 {% endif %}  
 {% else %}  
 <form action="{% url 'jobs:apply-job' job.id %}" method="post">  
 {% csrf\_token %}  
 <input type="hidden" name="job" value="{{ job.id }}">  
 <input type="submit" class="btn btn-outline-white-primary"  
 value="Apply for this job"/>  
 </form>  
 {% endif %}  
 </div>  
 </div>  
 <div class="col-lg-1"></div>  
 <div class="col-lg-3">  
 <h4>About {{ job.company\_name }}</h4>  
 <p class="job-detail\_\_company-description">{{ job.company\_description }}</p>  
 <p class="job-detail\_\_social social social--outline">  
 <a href="{{ job.website }}" data-toggle="tooltip" data-placement="top" title="Website"  
 class="link">  
 <i class="fa fa-link"></i>  
 </a>  
 <a href="#" data-toggle="tooltip" data-placement="top" title="Twitter" class="twitter">  
 <i class="fa fa-twitter"></i>  
 </a>  
 <a href="#" data-toggle="tooltip" data-placement="top" title="Facebook" class="facebook">  
 <i class="fa fa-facebook"></i>  
 </a>  
 </p>  
 <div class="job-detail\_\_apply-top">  
 {% if is\_applied %}  
 <input type="button" class="btn btn-outline-white-primary" value="Already applied"/>  
 {% else %}  
 <form action="{% url 'jobs:apply-job' job.id %}" method="post">  
 {% csrf\_token %}  
 <input type="hidden" name="job" value="{{ job.id }}">  
 <input type="submit" class="btn btn-outline-white-primary" value="Apply for this job"/>  
 </form>  
 {% endif %}  
 </div>  
 </div>  
 </div>  
 </div>  
 </section>  
  
{% endblock %}

search.html (jobs)

{% extends 'base.html' %}  
{% load static %}  
{% block title %}  
 All jobs  
{% endblock %}  
  
{% block content %}  
  
 <section class="job-form-section job-form-section--compressed">  
 <div class="container">  
 <h4 class="heading">Find a <span class="accent">job </span> you will <span class="accent">love</span></h4>  
 <form id="job-main-form" method="get" action="#" class="job-main-form">  
 <div class="controls">  
 <div class="row align-items-end">  
 <div class="col-sm-1 col-lg-2"></div>  
 <div class="col-sm-5 col-lg-4 form-group">  
 <label for="profession">Position</label>  
 <input type="text" id="position" name="position"  
 placeholder="Position you are looking for" class="form-control">  
 </div>  
 <div class="col-sm-5 col-lg-4 form-group">  
 <label for="location">Location</label>  
 <input type="text" id="location" name="location" placeholder="Any particular location?"  
 value="" class="form-control">  
 </div>  
 <div class="col-sm-1 col-lg-2 form-group">  
 <button type="submit" name="name"  
 class="btn btn-outline-white-primary job-main-form\_\_button">  
 <i class="fa fa-search"></i>  
 </button>  
 </div>  
 </div>  
 </div>  
 </form>  
 </div>  
 </section>  
 <section>  
 <div class="container">  
 <h3 class="heading">We have found <span class="accent">{{ jobs.count }}</span> jobs</h3>  
 {% for job in jobs %}  
 <div class="job-listing ">  
 <div class="row">  
 <div class="col-md-12 col-lg-6">  
 <div class="row">  
 <div class="col-2">  
 <img src="{% static 'img/company-1.png' %}"  
 alt="Ipsioou " class="img-fluid">  
 </div>  
 <div class="col-10">  
 <h4 class="job\_\_title"><a  
 href="{% url 'jobs:jobs-detail' job.id %}">{{ job.title }}</a></h4>  
 <p class="job\_\_company">  
 {{ job.company\_name }}  
 </p>  
 </div>  
 </div>  
 </div>  
 <div class="col-10 col-md-3 col-lg-2 ml-auto"><i class="fa fa-map-marker job\_\_location"></i>  
 {{ job.location }}  
 </div>  
 <div class="col-10 col-md-3 col-lg-3 ml-auto">  
 <p>Posted {{ job.created\_at|timesince }}</p>  
 </div>  
 </div>  
 </div>  
 {% endfor %}  
 {% if is\_paginated %}  
 <div class="row">  
 <div class="col-md-12">  
 <nav aria-label="Page navigation example" class="d-flex justify-content-center mt-4 mb-4">  
 <ul class="pagination">  
 {% if page\_obj.has\_previous %}  
 <li class="page-item">  
 <a href="?page={{ page\_obj.previous\_page\_number }}"  
 class="page-link">&laquo;</a>  
 </li>  
 {% else %}  
 <li class="page-item disabled">  
 <a class="page-link">&laquo;</a>  
 </li>  
 {% endif %}  
 {% for i in page\_obj.paginator.page\_range %}  
 {% if page\_obj.number == i %}  
 <li class="page-item active">  
 <a class="page-link">{{ i }}</a>  
 </li>  
 {% else %}  
 <li class="page-item">  
 <a href="?page={{ i }}" class="page-link">{{ i }}</a>  
 </li>  
 {% endif %}  
 {% endfor %}  
 {% if page\_obj.has\_next %}  
 <li class="page-item">  
 <a href="?page={{ page\_obj.next\_page\_number }}" class="page-link">&raquo;</a>  
 </li>  
 {% else %}  
 <li class="page-item disabled">  
 <a class="page-link">&raquo;</a>  
 </li>  
 {% endif %}  
 </ul>  
 </nav>  
 </div>  
 </div>  
 {% endif %}  
 </div>  
 </section>  
  
{% endblock %}

about.html

{% load static %}  
<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <title>{% block title %} {% endblock %} - About Us </title>  
 <meta name="description" content="">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css" >  
 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">  
 <meta name="robots" content="all,follow">  
 <!-- Bootstrap CSS-->  
 <link rel="stylesheet"  
 href="{% static 'vendor/bootstrap/css/bootstrap.min.css' %}">  
 <!-- Font Awesome CSS-->  
 <link rel="stylesheet"  
 href="{% static 'vendor/font-awesome/css/font-awesome.min.css' %}">  
 <!-- Google fonts - Roboto for copy, Montserrat for headings-->  
 <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Roboto:300,400,700">  
 <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Montserrat:400,700">  
 <!-- owl carousel-->  
 <link rel="stylesheet"  
 href="{% static 'vendor/owl.carousel/assets/owl.carousel.css' %}">  
 <link rel="stylesheet"  
 href="{% static 'vendor/owl.carousel/assets/owl.theme.default.css' %}">  
 <!-- theme stylesheet-->  
 <link rel="stylesheet" href="{% static 'css/style.default.css' %}"  
 id="theme-stylesheet">  
 <!-- Custom stylesheet - for your changes-->  
 <link rel="stylesheet" href="{% static 'css/custom.css' %}">  
 {% block styles %}{% endblock %}  
 <!-- Favicon-->  
 <link rel="shortcut icon" href="static/img/11.png">  
 <!-- Tweaks for older IEs--><!--[if lt IE 9]>  
 <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>  
 <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script><![endif]-->  
 <style>  
@import url('https://fonts.googleapis.com/css2?family=Nunito:wght@200;300;400;600;700;800&amp;display=swap');  
  
\*,  
\*:before,  
\*:after {  
 -moz-box-sizing: border-box;  
 -webkit-box-sizing: border-box;  
 box-sizing: border-box;  
}  
  
body {  
 margin: 0;  
}  
  
.wk-desk-1 {  
 width: 8.333333%;  
}  
  
.wk-desk-2 {  
 width: 16.666667%;  
}  
  
.wk-desk-3 {  
 width: 25%;  
}  
  
.wk-desk-4 {  
 width: 33.333333%;  
}  
  
.wk-desk-5 {  
 width: 41.666667%;  
}  
  
.wk-desk-6 {  
 width: 50%;  
}  
  
.wk-desk-7 {  
 width: 58.333333%;  
}  
  
.wk-desk-8 {  
 width: 66.666667%;  
}  
  
.wk-desk-9 {  
 width: 75%;  
}  
  
.wk-desk-10 {  
 width: 83.333333%;  
}  
  
.wk-desk-11 {  
 width: 91.666667%;  
}  
  
.wk-desk-12 {  
 width: 100%;  
}  
  
@media (max-width: 1024px) {  
 .wk-ipadp-1 {  
 width: 8.333333%;  
 }  
  
 .wk-ipadp-2 {  
 width: 16.666667%;  
 }  
  
 .wk-ipadp-3 {  
 width: 25%;  
 }  
  
 .wk-ipadp-4 {  
 width: 33.333333%;  
 }  
  
 .wk-ipadp-5 {  
 width: 41.666667%;  
 }  
  
 .wk-ipadp-6 {  
 width: 50%;  
 }  
  
 .wk-ipadp-7 {  
 width: 58.333333%;  
 }  
  
 .wk-ipadp-8 {  
 width: 66.666667%;  
 }  
  
 .wk-ipadp-9 {  
 width: 75%;  
 }  
  
 .wk-ipadp-10 {  
 width: 83.333333%;  
 }  
  
 .wk-ipadp-11 {  
 width: 91.666667%;  
 }  
  
 .wk-ipadp-12 {  
 width: 100%;  
 }  
}  
  
@media (max-width: 768px) {  
 .wk-tab-1 {  
 width: 8.333333%;  
 }  
  
 .wk-tab-2 {  
 width: 16.666667%;  
 }  
  
 .wk-tab-3 {  
 width: 25%;  
 }  
  
 .wk-tab-4 {  
 width: 33.333333%;  
 }  
  
 .wk-tab-5 {  
 width: 41.666667%;  
 }  
  
 .wk-tab-6 {  
 width: 50%;  
 }  
  
 .wk-tab-7 {  
 width: 58.333333%;  
 }  
  
 .wk-tab-8 {  
 width: 66.666667%;  
 }  
  
 .wk-tab-9 {  
 width: 75%;  
 }  
  
 .wk-tab-10 {  
 width: 83.333333%;  
 }  
  
 .wk-tab-11 {  
 width: 91.666667%;  
 }  
  
 .wk-tab-12 {  
 width: 100%;  
 }  
}  
  
@media (max-width: 500px) {  
 .wk-mobile-1 {  
 width: 8.333333%;  
 }  
  
 .wk-mobile-2 {  
 width: 16.666667%;  
 }  
  
 .wk-mobile-3 {  
 width: 25%;  
 }  
  
 .wk-mobile-4 {  
 width: 33.333333%;  
 }  
  
 .wk-mobile-5 {  
 width: 41.666667%;  
 }  
  
 .wk-mobile-6 {  
 width: 50%;  
 }  
  
 .wk-mobile-7 {  
 width: 58.333333%;  
 }  
  
 .wk-mobile-8 {  
 width: 66.666667%;  
 }  
  
 .wk-mobile-9 {  
 width: 75%;  
 }  
  
 .wk-mobile-10 {  
 width: 83.333333%;  
 }  
  
 .wk-mobile-11 {  
 width: 91.666667%;  
 }  
  
 .wk-mobile-12 {  
 width: 100%;  
 }  
}  
\* {  
 font-family: Nunito, sans-serif;  
}  
  
.responsive-container-block {  
 min-height: 75px;  
 height: fit-content;  
 width: 100%;  
 padding-top: 10px;  
 padding-right: 10px;  
 padding-bottom: 10px;  
 padding-left: 10px;  
 display: flex;  
 flex-wrap: wrap;  
 margin-top: 0px;  
 margin-right: auto;  
 margin-bottom: 0px;  
 margin-left: auto;  
 justify-content: flex-start;  
}  
  
a {  
 text-decoration-line: none;  
 text-decoration-thickness: initial;  
 text-decoration-style: initial;  
 text-decoration-color: initial;  
}  
  
.text-blk {  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 padding-top: 10px;  
 padding-right: 10px;  
 padding-bottom: 10px;  
 padding-left: 10px;  
 line-height: 25px;  
}  
  
.responsive-container-block.bigContainer {  
 padding-top: 10px;  
 padding-right: 30px;  
 padding-bottom: 10px;  
 padding-left: 30px;  
 flex-direction: column;  
 align-items: center;  
 justify-content: center;  
 padding: 10px 50px 10px 50px;  
}  
  
.mainImg {  
 color: black;  
 width: 40%;  
 height: auto;  
 border-top-left-radius: 10px;  
 border-top-right-radius: 10px;  
 border-bottom-right-radius: 10px;  
 border-bottom-left-radius: 10px;  
}  
  
.text-blk.headingText {  
 font-size: 22px;  
 font-weight: 700;  
 line-height: 30px;  
 color: rgb(176, 98, 255);  
 padding-top: 0px;  
 padding-right: 10px;  
 padding-bottom: 0px;  
 padding-left: 0px;  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: 5px;  
 margin-left: 0px;  
}  
  
.allText {  
 padding-top: 0px;  
 padding-right: 0px;  
 padding-bottom: 0px;  
 padding-left: 0px;  
 width: 40%;  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
}  
  
.text-blk.subHeadingText {  
 color: rgb(102, 102, 102);  
 font-size: 26px;  
 line-height: 32px;  
 font-weight: 700;  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: 15px;  
 margin-left: 0px;  
 padding-top: 0px;  
 padding-right: 10px;  
 padding-bottom: 0px;  
 padding-left: 0px;  
}  
  
.text-blk.description {  
 font-size: 18px;  
 line-height: 26px;  
 color: rgb(102, 102, 102);  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: 50px;  
 margin-left: 0px;  
 font-weight: 400;  
 padding-top: 0px;  
 padding-right: 10px;  
 padding-bottom: 0px;  
 padding-left: 0px;  
}  
  
.explore {  
 font-size: 16px;  
 line-height: 28px;  
 color: rgb(102, 102, 102);  
 border-top-width: 2px;  
 border-right-width: 2px;  
 border-bottom-width: 2px;  
 border-left-width: 2px;  
 border-top-style: solid;  
 border-right-style: solid;  
 border-bottom-style: solid;  
 border-left-style: solid;  
 border-top-color: rgb(102, 102, 102);  
 border-right-color: rgb(102, 102, 102);  
 border-bottom-color: rgb(102, 102, 102);  
 border-left-color: rgb(102, 102, 102);  
 border-image-source: initial;  
 border-image-slice: initial;  
 border-image-width: initial;  
 border-image-outset: initial;  
 border-image-repeat: initial;  
 cursor: pointer;  
 background-color: white;  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 padding-top: 8px;  
 padding-right: 40px;  
 padding-bottom: 8px;  
 padding-left: 40px;  
}  
  
.explore:hover {  
 background-image: initial;  
 background-position-x: initial;  
 background-position-y: initial;  
 background-size: initial;  
 background-repeat-x: initial;  
 background-repeat-y: initial;  
 background-attachment: initial;  
 background-origin: initial;  
 background-clip: initial;  
 background-color: rgb(176, 98, 255);  
 color: white;  
 border-top-width: initial;  
 border-right-width: initial;  
 border-bottom-width: initial;  
 border-left-width: initial;  
 border-top-style: none;  
 border-right-style: none;  
 border-bottom-style: none;  
 border-left-style: none;  
 border-top-color: initial;  
 border-right-color: initial;  
 border-bottom-color: initial;  
 border-left-color: initial;  
 border-image-source: initial;  
 border-image-slice: initial;  
 border-image-width: initial;  
 border-image-outset: initial;  
 border-image-repeat: initial;  
}  
  
.responsive-container-block.Container {  
 margin-top: 80px;  
 margin-right: auto;  
 margin-bottom: 50px;  
 margin-left: auto;  
 justify-content: center;  
 align-items: center;  
 max-width: 1320px;  
 padding-top: 10px;  
 padding-right: 10px;  
 padding-bottom: 10px;  
 padding-left: 10px;  
}  
  
.responsive-container-block.Container.bottomContainer {  
 flex-direction: row-reverse;  
 margin-top: 80px;  
 margin-right: auto;  
 margin-bottom: 50px;  
 margin-left: auto;  
 position: static;  
}  
  
.allText.aboveText {  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 40px;  
}  
  
.allText.bottomText {  
 margin-top: 0px;  
 margin-right: 40px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 display: flex;  
 flex-direction: column;  
 align-items: flex-start;  
 justify-content: flex-start;  
 padding-top: 0px;  
 padding-right: 15px;  
 padding-bottom: 0px;  
 padding-left: 0px;  
}  
  
.purpleBox {  
 display: flex;  
 flex-direction: column;  
 align-items: flex-start;  
 max-width: 430px;  
 background-color: rgb(176, 98, 255);  
 padding-top: 20px;  
 padding-right: 20px;  
 padding-bottom: 20px;  
 padding-left: 20px;  
 border-top-left-radius: 10px;  
 border-top-right-radius: 10px;  
 border-bottom-right-radius: 10px;  
 border-bottom-left-radius: 10px;  
 position: absolute;  
 bottom: -35px;  
 left: -8%;  
}  
  
.purpleText {  
 font-size: 18px;  
 line-height: 26px;  
 color: white;  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: 10px;  
 margin-left: 0px;  
}  
  
.ultimateImg {  
 width: 50%;  
 position: relative;  
}  
  
@media (max-width: 1024px) {  
 .responsive-container-block.Container {  
 max-width: 850px;  
 }  
  
 .mainImg {  
 width: 55%;  
 height: auto;  
 }  
  
 .allText {  
 width: 40%;  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 20px;  
 }  
  
 .responsive-container-block.bigContainer {  
 padding-top: 10px;  
 padding-right: 10px;  
 padding-bottom: 10px;  
 padding-left: 10px;  
 }  
  
 .responsive-container-block.Container.bottomContainer {  
 margin-top: 80px;  
 margin-right: auto;  
 margin-bottom: 50px;  
 margin-left: auto;  
 }  
  
 .responsive-container-block.Container {  
 max-width: 830px;  
 }  
  
 .allText.aboveText {  
 margin-top: 30px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 40px;  
 }  
  
 .allText.bottomText {  
 margin-top: 30px;  
 margin-right: 40px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 text-align: left;  
 }  
  
 .text-blk.headingText {  
 text-align: center;  
 }  
  
 .allText.aboveText {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
 justify-content: center;  
 margin-top: 30px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 }  
  
 .text-blk.subHeadingText {  
 text-align: left;  
 font-size: 26px;  
 line-height: 32px;  
 }  
  
 .text-blk.description {  
 text-align: left;  
 line-height: 24px;  
 }  
  
 .explore {  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 }  
  
 .responsive-container-block.bigContainer {  
 padding-top: 10px;  
 padding-right: 30px;  
 padding-bottom: 10px;  
 padding-left: 30px;  
 }  
  
 .responsive-container-block.Container {  
 justify-content: space-evenly;  
 }  
  
 .purpleBox {  
 bottom: 10%;  
 }  
  
 .responsive-container-block.Container.bottomContainer {  
 padding-top: 10px;  
 padding-right: 0px;  
 padding-bottom: 10px;  
 padding-left: 0px;  
 max-width: 930px;  
 }  
  
 .allText.bottomText {  
 width: 40%;  
 }  
  
 .purpleBox {  
 bottom: auto;  
 left: -10%;  
 top: 70%;  
 }  
  
 .mainImg {  
 width: 100%;  
 }  
  
 .text-blk.headingText {  
 text-align: left;  
 }  
}  
  
@media (max-width: 768px) {  
 .allText {  
 width: 100%;  
 display: flex;  
 flex-direction: column;  
 justify-content: center;  
 align-items: center;  
 padding-top: 0px;  
 padding-right: 0px;  
 padding-bottom: 0px;  
 padding-left: 0px;  
 }  
  
 .responsive-container-block.Container {  
 flex-direction: column;  
 height: auto;  
 }  
  
 .text-blk.headingText {  
 text-align: center;  
 }  
  
 .text-blk.subHeadingText {  
 text-align: center;  
 font-size: 24px;  
 }  
  
 .text-blk.description {  
 text-align: center;  
 font-size: 18px;  
 }  
  
 .allText {  
 margin-top: 40px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 }  
  
 .allText.aboveText {  
 margin-top: 40px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 }  
  
 .responsive-container-block.Container {  
 margin-top: 80px;  
 margin-right: auto;  
 margin-bottom: 50px;  
 margin-left: auto;  
 }  
  
 .responsive-container-block.Container.bottomContainer {  
 margin-top: 50px;  
 margin-right: auto;  
 margin-bottom: 50px;  
 margin-left: auto;  
 }  
  
 .allText.bottomText {  
 margin-top: 40px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 }  
  
 .mainImg {  
 width: 100%;  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: -70px;  
 margin-left: 0px;  
 }  
  
 .responsive-container-block.Container.bottomContainer {  
 flex-direction: column;  
 }  
  
 .ultimateImg {  
 width: 100%;  
 }  
  
 .purpleBox {  
 position: static;  
 }  
  
 .allText.bottomText {  
 width: 100%;  
 align-items: flex-start;  
 }  
  
 .text-blk.headingText {  
 text-align: left;  
 }  
  
 .text-blk.subHeadingText {  
 text-align: left;  
 }  
  
 .text-blk.description {  
 text-align: left;  
 }  
  
 .ultimateImg {  
 position: static;  
 }  
  
 .mainImg {  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 }  
  
 .ultimateImg {  
 position: relative;  
 }  
  
 .purpleBox {  
 margin-top: 0px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 position: absolute;  
 left: 0px;  
 top: 80%;  
 }  
  
 .allText.bottomText {  
 margin-top: 100px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 }  
}  
  
@media (max-width: 500px) {  
 .responsive-container-block.Container {  
 padding-top: 10px;  
 padding-right: 0px;  
 padding-bottom: 10px;  
 padding-left: 0px;  
 width: 100%;  
 max-width: 100%;  
 }  
  
 .mainImg {  
 width: 100%;  
 }  
  
 .responsive-container-block.bigContainer {  
 padding-top: 10px;  
 padding-right: 25px;  
 padding-bottom: 10px;  
 padding-left: 25px;  
 }  
  
 .text-blk.subHeadingText {  
 font-size: 24px;  
 padding-top: 0px;  
 padding-right: 0px;  
 padding-bottom: 0px;  
 padding-left: 0px;  
 line-height: 28px;  
 }  
  
 .text-blk.description {  
 font-size: 16px;  
 padding-top: 0px;  
 padding-right: 0px;  
 padding-bottom: 0px;  
 padding-left: 0px;  
 line-height: 22px;  
 }  
  
 .allText {  
 padding-top: 0px;  
 padding-right: 0px;  
 padding-bottom: 0px;  
 padding-left: 0px;  
 width: 100%;  
 }  
  
 .allText.bottomText {  
 margin-top: 50px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 padding: 0 0 0 0;  
 margin: 30px 0 0 0;  
 }  
  
 .ultimateImg {  
 position: static;  
 }  
  
 .purpleBox {  
 position: static;  
 }  
  
 .stars {  
 width: 55%;  
 }  
  
 .allText.bottomText {  
 margin-top: 75px;  
 margin-right: 0px;  
 margin-bottom: 0px;  
 margin-left: 0px;  
 }  
  
 .responsive-container-block.bigContainer {  
 padding-top: 10px;  
 padding-right: 20px;  
 padding-bottom: 10px;  
 padding-left: 20px;  
 }  
  
 .purpleText {  
 font-size: 16px;  
 line-height: 22px;  
 }  
  
 .explore {  
 padding: 6px 35px 6px 35px;  
 font-size: 15px;  
 }  
}  
  
 .footer .footer-row {  
 display: flex;  
 flex-wrap: wrap;  
 justify-content: space-between;  
 gap: 3.5rem;  
 padding: 0px;  
}  
.footer-row .footer-col h4 {  
 color: #fff;  
 font-size: 1.2rem;  
 font-weight: 400;  
}  
.footer-col .links {  
 margin-top: 0px;  
}  
.footer-col .links li {  
 list-style: none;  
 margin-bottom: 10px;  
}  
</style>  
</head>  
<body>  
<!-- navbar-->  
<header class="header">  
 <nav class="navbar navbar-expand-lg bg-blue">  
 <div class="container">  
 <a href="/" class="navbar-brand">  
 <img src="{% static 'img/22.png' %}" alt="logo" class="d-none d-lg-block" height="110" width="230">  
 <span class="sr-only">Home</span>  
 </a>  
 <button type="button" data-toggle="collapse" data-target="#navbarSupportedContent"  
 aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation"  
 class="navbar-toggler navbar-toggler-right">Menu<i class="fa fa-bars"></i>  
 </button>  
 <div id="navbarSupportedContent" class="collapse navbar-collapse">  
 <ul class="navbar-nav ml-auto">  
 <li class="nav-item active">  
 <a href="/" class="nav-link">Home <span class="sr-only">(current)</span></a>  
 </li>  
 <li class="nav-item ">  
 <a href="jobs" class="nav-link"><i class="fa fa-newspaper-o"></i>Jobs </a>  
 </li>  
 <li class="nav-item">  
 <a href="about" class="nav-link"><i class="fa fa-id-badge"></i>About Us</a>  
 </li>  
 <li class="nav-item">  
 <a href="contact" class="nav-link"><i class="fa fa-fw fa-envelope"></i>Contact</a>  
 </li>  
 {% if request.user.is\_authenticated and request.user.role == 'employer' %}  
 <li class="nav-item dropdown">  
 <a id="clientZone" href="#" data-toggle="dropdown"  
 aria-haspopup="true" aria-expanded="false"  
 class="nav-link dropdown-toggle has-img mb-3 mb-lg-0 mt-3 mt-lg-0">  
 <div class="client-menu-image">  
 <div class="inner">  
 <img src="{% static 'img/11.png' %}"  
 alt="company image"  
 class="img-fluid">  
 </div>  
 </div>  
 {% if request.user.role == 'employee' %}  
 {{ request.user.first\_name }} {{ request.user.last\_name }}  
 {% else %}  
 {{ request.user.first\_name }}  
 {% endif %}  
 </a>  
 <div aria-labelledby="clientZone" class="dropdown-menu">  
 <a href="{% url 'jobs:employer-dashboard' %}" class="dropdown-item">Dashboard</a>  
 <a href="{% url 'jobs:employer-all-applicants' %}" class="dropdown-item">Applicants </a>  
 <a href="{% url 'jobs:employer-jobs-create' %}" class="dropdown-item">Post a new  
 job </a>  
 <a href="{% url 'accounts:logout' %}" class="dropdown-item">Logout</a>  
 </div>  
 </li>  
 {% elif request.user.is\_authenticated and request.user.role == 'employee' %}  
 <li class="nav-item dropdown">  
 <a id="clientZone" href="#" data-toggle="dropdown"  
 aria-haspopup="true" aria-expanded="false"  
 class="nav-link dropdown-toggle has-img mb-3 mb-lg-0 mt-3 mt-lg-0">  
 <div class="client-menu-image">  
 <div class="inner">  
 <img src="{% static 'img/11.png' %}"  
 alt="company image"  
 class="img-fluid">  
 </div>  
 </div>  
 {% if request.user.role == 'employee' %}  
 {{ request.user.first\_name }} {{ request.user.last\_name }}  
 {% else %}  
 {{ request.user.first\_name }}  
 {% endif %}  
 </a>  
 <div aria-labelledby="clientZone" class="dropdown-menu">  
 <a href="{% url 'accounts:employer-profile-update' %}" class="dropdown-item">Edit Profile</a>  
 <a href="{% url 'accounts:logout' %}" class="dropdown-item">Logout</a>  
 </div>  
 </li>  
 {% else %}  
 <li class="nav-item dropdown">  
 <a id="pages" href="#" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false"  
 class="nav-link dropdown-toggle"><i class="fa fa-signing"></i>Register  
 </a>  
 <div aria-labelledby="pages" class="dropdown-menu">  
 <a href="{% url 'accounts:employee-register' %}" class="dropdown-item">Employee</a>  
 <a href="{% url 'accounts:employer-register' %}" class="dropdown-item">Employers</a>  
 </div>  
 </li>  
 <li class="nav-item" style="color: #fff">  
 <a href="{% url 'accounts:login' %}" class="btn navbar-btn btn-outline-light mb-5 mb-lg-0">  
 <i class="fa fa-sign-in"></i>Login  
 </a>  
 </li>  
 {% endif %}  
 </ul>  
 </div>  
 </div>  
 </nav>  
</header>  
<div>  
 {% if messages %}  
  
 {% for message in messages %}  
 <div class="alert alert-{{ message.tags }} alert-dismissible fade show" role="alert">  
 {{ message }}  
 <button type="button" class="close" data-dismiss="alert" aria-label="Close">  
 <span aria-hidden="true">&times;</span>  
 </button>  
 </div>  
{# <li{% if message.tags %} class="alert alert-{{ message.tags }}"{% endif %}>{{ message }}</li>#}  
 {% endfor %}  
  
 {% endif %}  
 {% block content %} {% endblock %}  
</div>  
  
<div class="responsive-container-block bigContainer">  
 <div class="responsive-container-block Container bottomContainer">  
 <div class="ultimateImg">  
 <img class="mainImg" src="{% static 'img/Capture.jpg' %}" height="100" width="110">  
 <div class="purpleBox">  
 <p class="purpleText">  
 I'm pursuing MCA from IGNOU, and same time I'm doing JOB in TCS as an Engineer profile.  
 </p>  
  
 </div>  
 </div>  
 <div class="allText bottomText">  
 <p class="text-blk headingText">  
 About Me  
 </p>  
 <p class="text-blk subHeadingText">  
 Student & Engineer  
 </p>  
 <p class="text-blk description">  
 I'm Durgesh Jaiswal, a MCA\_NEW student of IGNOU. I've a good skills in Data Science, Python, Machine Learning and Desktop Support.  
 </p>  
 <p>  
 Online Job Portal is an online platform which helps Employee, Jobseeker to get job in Organization as per their requirements and profiles and Employers can post multiple jobs opportunity on the site and hire to the jobseeker/employee as per organization requirements.  
 </p>  
 <a class="explore">  
 ignou.durgeshjaiswal@gmail.com  
 </a>  
 </div>  
  
 </div>  
</div>  
<footer class="footer">  
 <div class="footer\_\_block">  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-4 col-md-12 mb-5">  
 <h4 class="h5"> Online Job Portal </h4>  
 <p> Online Job Portal is an online platform which helps Employee, Jobseeker to get job in Organization as per their requirements and profiles and Employers can post multiple jobs opportunity on the site and hire to the jobseeker/employee as per organization requirements. </p>  
 </div>  
 <div class="col-lg-4 col-md-12 mb-5">  
 <h4 class="h5">Special Links </h4>  
 <div class="footer-row">  
 <div class="footer-col">  
 <ul class="links">  
 <li><a href="/"> Home </a></li>  
 <li><a href="jobs"> Jobs </a></li>  
 <li><a href="about"> About Us</a></li>  
 <li><a href="contact"> Contact Us </a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 <div class="col-lg-4 col-md-12 mb-5">  
 <h4 class="h5">Online Job Portal </h4>  
 <p><img src="{% static 'img/11.png' %}" height="150" width="250"></p>  
 </div>  
 </div>  
 </div>  
 <div class="footer\_\_copyright">  
 <div class="container">  
 <div class="row">  
 <div class="col-md-6 text-md-left text-center">  
 <p align="center"> Online Job Portal © 2024 || All rights reserved </p>  
 </div>  
 </div>  
 </div>  
 </div>  
</footer>  
</body>  
</html>

contact.html

{% load static %}  
<!DOCTYPE html>  
<html>  
  
<head>  
 <meta charset="utf-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <title>{% block title %} {% endblock %} - Contact Us </title>  
 <meta name="description" content="">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css" >  
 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">  
 <meta name="robots" content="all,follow">  
 <!-- Bootstrap CSS-->  
 <link rel="stylesheet"  
 href="{% static 'vendor/bootstrap/css/bootstrap.min.css' %}">  
 <!-- Font Awesome CSS-->  
 <link rel="stylesheet"  
 href="{% static 'vendor/font-awesome/css/font-awesome.min.css' %}">  
 <!-- Google fonts - Roboto for copy, Montserrat for headings-->  
 <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Roboto:300,400,700">  
 <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Montserrat:400,700">  
 <!-- owl carousel-->  
 <link rel="stylesheet"  
 href="{% static 'vendor/owl.carousel/assets/owl.carousel.css' %}">  
 <link rel="stylesheet"  
 href="{% static 'vendor/owl.carousel/assets/owl.theme.default.css' %}">  
 <!-- theme stylesheet-->  
 <link rel="stylesheet" href="{% static 'css/style.default.css' %}"  
 id="theme-stylesheet">  
 <!-- Custom stylesheet - for your changes-->  
 <link rel="stylesheet" href="{% static 'css/custom.css' %}">  
 {% block styles %}{% endblock %}  
 <!-- Favicon-->  
 <link rel="shortcut icon" href="static/img/11.png">  
 <!-- Tweaks for older IEs--><!--[if lt IE 9]>  
 <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>  
 <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script><![endif]-->  
 <style>  
@import url(https://fonts.googleapis.com/css?family=Sniglet|Raleway:900);  
  
  
body, html{  
 height: 100%;  
 padding: 0;  
 margin: 0;  
 font-family: 'Sniglet', cursive;  
}  
h1{  
 font-weight: normal;  
 font-size: 4em;  
 font-family: 'Raleway', sans-serif;  
 margin: 0 auto;  
 margin-top: 30px;  
 width: 500px;  
 color: #F90;  
 text-align: center;  
  
}  
  
/\* Animation webkit \*/  
@-webkit-keyframes myfirst  
{  
 0% {margin-left: -235px}  
 90% {margin-left: 100%;}  
 100% {margin-left: 100%;}  
}  
  
/\* Animation \*/  
@keyframes myfirst  
{  
 0% {margin-left: -235px}  
 70% {margin-left: 100%;}  
 100% {margin-left: 100%;}  
}  
  
.fish{  
 background-image: url('http://www.geertjanhendriks.nl/codepen/form/fish.png');  
 width: 235px;  
 height: 104px;  
 margin-left: -235px;  
 position: absolute;  
 animation: myfirst 24s;  
 -webkit-animation: myfirst 24s;  
 animation-iteration-count: infinite;  
 -webkit-animation-iteration-count: infinite;  
 animation-timing-function: linear;  
 -webkit-animation-timing-function: linear;  
}  
  
#fish{  
 top: 120px;  
}  
  
#fish2{  
 top: 260px;  
 animation-delay: 12s;  
 -webkit-animation-delay: 12s;  
}  
  
  
header{  
 height: 160px;  
 background: url('http://www.geertjanhendriks.nl/codepen/form/golf.png') repeat-x bottom;  
}  
  
#form{  
 height: 100%;  
 background-color: #98d4f3;  
 overflow: hidden;  
 position: relative;  
  
}  
form{  
 margin: 0 auto;  
 width: 500px;  
 padding-top: 40px;  
 color: white;  
 position: relative;  
  
  
}  
label, input, textarea{  
 display: block;  
}  
input, textarea{  
 width: 500px;  
 border: none;  
 border-radius: 20px;  
 outline: none;  
 padding: 10px;  
 font-family: 'Sniglet', cursive;  
 font-size: 1em;  
 color: #676767;  
 transition: border 0.5s;  
 -webkit-transition: border 0.5s;  
 -moz-transition: border 0.5s;  
 -o-transition: border 0.5s;  
 border: solid 3px #98d4f3;  
 -webkit-box-sizing:border-box;  
 -moz-box-sizing:border-box;  
 box-sizing:border-box;  
  
}  
input:focus, textarea:focus{  
 border: solid 3px #77bde0;  
}  
  
textarea{  
 height: 100px;  
 resize: none;  
 overflow: auto;  
}  
input[type="submit"]{  
 background-color: #F90;  
 color: white;  
 height: 50px;  
 cursor: pointer;  
 margin-top: 30px;  
 font-size: 1.29em;  
 font-family: 'Sniglet', cursive;  
 -webkit-transition: background-color 0.5s;  
 -moz-transition: background-color 0.5s;  
 -o-transition: background-color 0.5s;  
 transition: background-color 0.5s;  
}  
input[type="submit"]:hover{  
 background-color: #e58f0e;  
  
}  
label{  
 font-size: 1.5em;  
 margin-top: 20px;  
 padding-left: 20px;  
}  
.formgroup, .formgroup-active, .formgroup-error{  
 background-repeat: no-repeat;  
 background-position: right bottom;  
 background-size: 10.5%;  
 transition: background-image 0.7s;  
 -webkit-transition: background-image 0.7s;  
 -moz-transition: background-image 0.7s;  
 -o-transition: background-image 0.7s;  
 width: 566px;  
 padding-top: 2px;  
}  
  
.formgroup{  
 background-image: url('http://www.geertjanhendriks.nl/codepen/form/pixel.gif');  
}  
.formgroup-active{  
 background-image: url('http://www.geertjanhendriks.nl/codepen/form/octo.png');  
}  
.formgroup-error{  
 background-image: url('http://www.geertjanhendriks.nl/codepen/form/octo-error.png');  
 color: red;  
}  
 .footer .footer-row {  
 display: flex;  
 flex-wrap: wrap;  
 justify-content: space-between;  
 gap: 3.5rem;  
 padding: 0px;  
}  
.footer-row .footer-col h4 {  
 color: #fff;  
 font-size: 1.2rem;  
 font-weight: 400;  
}  
.footer-col .links {  
 margin-top: 0px;  
}  
.footer-col .links li {  
 list-style: none;  
 margin-bottom: 10px;  
}  
</style>  
  
<script>  
 $('document').ready(function(){  
 $('input[type="text"], input[type="email"], textarea').focus(function(){  
 var background = $(this).attr('id');  
 $('#' + background + '-form').addClass('formgroup-active');  
 $('#' + background + '-form').removeClass('formgroup-error');  
 });  
 $('input[type="text"], input[type="email"], textarea').blur(function(){  
 var background = $(this).attr('id');  
 $('#' + background + '-form').removeClass('formgroup-active');  
 });  
  
function errorfield(field){  
 $(field).addClass('formgroup-error');  
 console.log(field);  
}  
  
$("#waterform").submit(function() {  
 var stopsubmit = false;  
  
if($('#name').val() == "") {  
 errorfield('#name-form');  
 stopsubmit=true;  
}  
if($('#email').val() == "") {  
 errorfield('#email-form');  
 stopsubmit=true;  
}  
 if(stopsubmit) return false;  
});  
  
});  
</script>  
</head>  
<body>  
<!-- navbar-->  
<header class="header">  
 <nav class="navbar navbar-expand-lg bg-blue">  
 <div class="container">  
 <a href="/" class="navbar-brand">  
 <img src="{% static 'img/22.png' %}" alt="logo" class="d-none d-lg-block" height="110" width="230">  
 <span class="sr-only">Home</span>  
 </a>  
 <button type="button" data-toggle="collapse" data-target="#navbarSupportedContent"  
 aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation"  
 class="navbar-toggler navbar-toggler-right">Menu<i class="fa fa-bars"></i>  
 </button>  
 <div id="navbarSupportedContent" class="collapse navbar-collapse">  
 <ul class="navbar-nav ml-auto">  
 <li class="nav-item active">  
 <a href="/" class="nav-link">Home <span class="sr-only">(current)</span></a>  
 </li>  
 <li class="nav-item ">  
 <a href="jobs" class="nav-link"><i class="fa fa-newspaper-o"></i>Jobs </a>  
 </li>  
 <li class="nav-item">  
 <a href="about" class="nav-link"><i class="fa fa-id-badge"></i>About Us</a>  
 </li>  
 <li class="nav-item">  
 <a href="contact" class="nav-link"><i class="fa fa-fw fa-envelope"></i>Contact</a>  
 </li>  
 {% if request.user.is\_authenticated and request.user.role == 'employer' %}  
 <li class="nav-item dropdown">  
 <a id="clientZone" href="#" data-toggle="dropdown"  
 aria-haspopup="true" aria-expanded="false"  
 class="nav-link dropdown-toggle has-img mb-3 mb-lg-0 mt-3 mt-lg-0">  
 <div class="client-menu-image">  
 <div class="inner">  
 <img src="{% static 'img/1.jpeg' %}"  
 alt="company image"  
 class="img-fluid">  
 </div>  
 </div>  
 {% if request.user.role == 'employee' %}  
 {{ request.user.first\_name }} {{ request.user.last\_name }}  
 {% else %}  
 {{ request.user.first\_name }}  
 {% endif %}  
 </a>  
 <div aria-labelledby="clientZone" class="dropdown-menu">  
 <a href="{% url 'jobs:employer-dashboard' %}" class="dropdown-item">Dashboard</a>  
 <a href="{% url 'jobs:employer-all-applicants' %}" class="dropdown-item">Applicants </a>  
 <a href="{% url 'jobs:employer-jobs-create' %}" class="dropdown-item">Post a new  
 job </a>  
 <a href="{% url 'accounts:logout' %}" class="dropdown-item">Logout</a>  
 </div>  
 </li>  
 {% elif request.user.is\_authenticated and request.user.role == 'employee' %}  
 <li class="nav-item dropdown">  
 <a id="clientZone" href="#" data-toggle="dropdown"  
 aria-haspopup="true" aria-expanded="false"  
 class="nav-link dropdown-toggle has-img mb-3 mb-lg-0 mt-3 mt-lg-0">  
 <div class="client-menu-image">  
 <div class="inner">  
 <img src="{% static 'img/1.jpeg' %}"  
 alt="company image"  
 class="img-fluid">  
 </div>  
 </div>  
 {% if request.user.role == 'employee' %}  
 {{ request.user.first\_name }} {{ request.user.last\_name }}  
 {% else %}  
 {{ request.user.first\_name }}  
 {% endif %}  
 </a>  
 <div aria-labelledby="clientZone" class="dropdown-menu">  
 <a href="{% url 'accounts:employer-profile-update' %}" class="dropdown-item">Edit Profile</a>  
 <a href="{% url 'accounts:logout' %}" class="dropdown-item">Logout</a>  
 </div>  
 </li>  
 {% else %}  
 <li class="nav-item dropdown">  
 <a id="pages" href="#" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false"  
 class="nav-link dropdown-toggle"><i class="fa fa-signing"></i>Register  
 </a>  
 <div aria-labelledby="pages" class="dropdown-menu">  
 <a href="{% url 'accounts:employee-register' %}" class="dropdown-item">Employee</a>  
 <a href="{% url 'accounts:employer-register' %}" class="dropdown-item">Employers</a>  
 </div>  
 </li>  
 <li class="nav-item" style="color: #fff">  
 <a href="{% url 'accounts:login' %}" class="btn navbar-btn btn-outline-light mb-5 mb-lg-0">  
 <i class="fa fa-sign-in"></i>Login  
 </a>  
 </li>  
 {% endif %}  
 </ul>  
 </div>  
 </div>  
 </nav>  
</header>  
<div>  
 {% if messages %}  
  
 {% for message in messages %}  
 <div class="alert alert-{{ message.tags }} alert-dismissible fade show" role="alert">  
 {{ message }}  
 <button type="button" class="close" data-dismiss="alert" aria-label="Close">  
 <span aria-hidden="true">&times;</span>  
 </button>  
 </div>  
{# <li{% if message.tags %} class="alert alert-{{ message.tags }}"{% endif %}>{{ message }}</li>#}  
 {% endfor %}  
  
 {% endif %}  
 {% block content %} {% endblock %}  
</div>  
  
<header>  
 <h1>Contact us</h1>  
</header>  
  
<div id="form">  
  
<div class="fish" id="fish"></div>  
<div class="fish" id="fish2"></div>  
  
<form id="waterform" method="post" action="https://docs.google.com/forms/d/e/1FAIpQLScdbxoBjw2NmP7ljzJOYfO77jGKg1cb0Ck1lqVzebE-WxkIyw/viewform">  
  
<div class="formgroup" id="name-form">  
 <label for="name">Your name\*</label>  
 <input type="text" id="name" name="name" />  
</div>  
  
<div class="formgroup" id="email-form">  
 <label for="email">Your e-mail\*</label>  
 <input type="email" id="email" name="email" />  
</div>  
  
<div class="formgroup" id="message-form">  
 <label for="message">Your message</label>  
 <textarea id="message" name="message"></textarea>  
</div>  
  
 <input type="submit" value="Send your message!" />  
</form>  
</div>  
  
  
 </div>  
</div>  
<footer class="footer">  
 <div class="footer\_\_block">  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-4 col-md-12 mb-5">  
 <h4 class="h5"> Online Job Portal </h4>  
 <p> Online Job Portal is an online platform which helps Employee, Jobseeker to get job in Organization as per their requirements and profiles and Employers can post multiple jobs opportunity on the site and hire to the jobseeker/employee as per organization requirements. </p>  
 </div>  
 <div class="col-lg-4 col-md-12 mb-5">  
 <h4 class="h5">Special Links </h4>  
 <div class="footer-row">  
 <div class="footer-col">  
 <ul class="links">  
 <li><a href="/"> Home </a></li>  
 <li><a href="jobs"> Jobs </a></li>  
 <li><a href="about"> About Us</a></li>  
 <li><a href="contact"> Contact Us </a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 <div class="col-lg-4 col-md-12 mb-5">  
 <h4 class="h5">Online Job Portal </h4>  
 <p><img src="{% static 'img/11.png' %}" height="150" width="250"></p>  
 </div>  
 </div>  
 </div>  
 <div class="footer\_\_copyright">  
 <div class="container">  
 <div class="row">  
 <div class="col-md-6 text-md-left text-center">  
 <p align="center"> Online Job Portal © 2024 || All rights reserved </p>  
 </div>  
 </div>  
 </div>  
 </div>  
</footer>  
</body>  
</html>

edit-profile.html (Jobseeker)

{% extends 'base.html' %}  
{% load static %}  
{% block title %}  
 Edit Profile  
{% endblock %}  
  
{% block content %}  
 <div class="col-lg-6 offset-3">  
 <div class="box">  
 <h3 class="heading">Edit Profile</h3>  
 {% if form.errors %}  
 {% for field in form %}  
 {% for error in field.errors %}  
 <div class="alert alert-danger alert-dismissable">  
 <a href="#" class="close" data-dismiss="alert" aria-label="close">&times;</a>  
 <strong>{{ error|escape }}</strong>  
 </div>  
 {% endfor %}  
 {% endfor %}  
 {% for error in form.non\_field\_errors %}  
 <div class="alert alert-danger alert-dismissable">  
 <a href="#" class="close" data-dismiss="alert" aria-label="close">&times;</a>  
 <strong>{{ error|escape }}</strong>  
 </div>  
 {% endfor %}  
 {% endif %}  
 <form action="" method="post">  
 {% csrf\_token %}  
 {% for field in form %}  
 {% if field.name == 'gender' %}  
  
 <div class="form-group">  
 <label for="gender">Gender</label>  
 <br>  
 <div class="form-check form-check-inline">  
 <input class="form-check-input" {% if field.value == 'male' %} checked {% endif %}  
 type="radio" name="gender" id="male gender"  
 value="male">  
 <label class="form-check-label" for="male">Male</label>  
 </div>  
 <div class="form-check form-check-inline">  
 <input class="form-check-input" {% if field.value == 'female' %} checked {% endif %}  
 type="radio" name="gender" id="female"  
 value="female">  
 <label class="form-check-label" for="female">Female</label>  
 </div>  
 </div>  
  
 {% else %}  
  
 <div class="form-group">  
 <label for="id\_{{ field.name }}">{{ field.label }}</label>  
 <input type="{{ field.field.widget.input\_type }}"  
 class="form-control"  
 name="{{ field.name }}"  
 id="id\_{{ field.name }}"  
 value="{{ field.value }}"  
 placeholder="{{ field.field.widget.attrs.placeholder }}">  
 </div>  
  
 {% endif %}  
 {% endfor %}  
 <div class="text-center">  
 <button type="submit" class="btn btn-outline-white-primary">  
 <i class="fa fa-signing"></i> Update  
 </button>  
 </div>  
 </form>  
 </div>  
 </div>  
{% endblock %}

dashboard.html (Employer)

{% extends 'base.html' %}  
{% load static %}  
{% block title %}  
 Dashboard  
{% endblock %}  
  
{% block content %}  
  
 <section class="bg-light-gray">  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-8 mx-auto">  
 <h1 class="heading">dashboard</h1>  
 <p class="lead text-center">All created jobs</p>  
 </div>  
 </div>  
 </div>  
 </section>  
 <section>  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-12 text-right mb-5"><a href="{% url 'jobs:employer-jobs-create' %}"  
 class="btn btn-outline-white-primary">  
 <i class="fa fa-plus"></i>Add new position</a></div>  
 <div class="col-lg-12">  
 <div class="table-responsive">  
 <table class="table table-striped table-hover table-client-dashboard">  
 <thead>  
 <tr>  
 <th>Job title</th>  
 <th>Position filled</th>  
 <th>Date posted</th>  
 <th>Date expiring</th>  
 <th>Applicants</th>  
 <th>Actions</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for job in jobs %}  
  
 <tr>  
 <th><a href="{% url 'jobs:jobs-detail' job.id %}">{{ job.title }}</a>  
 </th>  
 <td>  
 {% if job.filled %}  
 <span class="badge badge-success">Filled</span>  
 {% else %}  
 <span class="badge badge-secondary">Not Filled</span>  
 {% endif %}  
 </td>  
 <td>{{ job.created\_at }}</td>  
 <td>{{ job.last\_date }}</td>  
 <td>  
 <a href="{% url 'jobs:employer-dashboard-applicants' job.id %}"  
 class="btn btn-success">  
 <i class="fa fa-users"></i>{{ job.applicants.count }}  
 <span class="hidden-xs hidden-sm">Applicants</span>  
 </a>  
 </td>  
 <td>  
 <a href="#" class="btn btn-outline-white-secondary">  
 <i class="fa fa-edit"></i>Edit  
 </a>  
 <br>  
 {% if job.filled %}  
 <a href="{% url 'jobs:job-mark-filled' job.id %}"  
 class="btn btn-outline-white-secondary">  
 <i class="fa fa-check-circle-o"></i>  
 <span class="hidden-xs hidden-sm">Filled</span>  
 </a><br>  
 {% else %}  
 <a href="{% url 'jobs:job-mark-filled' job.id %}"  
 class="btn btn-outline-white-secondary">  
 <i class="fa fa-check-circle-o"></i>  
 <span class="hidden-xs hidden-sm">Mark as </span>filled  
 </a><br>  
 {% endif %}  
 <a href="#" class="btn btn-outline-white-secondary">  
 <i class="fa fa-times-circle-o"></i>Delete  
 </a>  
 </td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
 </div>  
 <div class="pages">  
 <nav aria-label="Page navigation example" class="d-flex justify-content-center mt-4 mb-4">  
 <ul class="pagination">  
 <li class="page-item"><a href="#" aria-label="Previous" class="page-link"><span  
 aria-hidden="true">«</span><span class="sr-only">Previous</span></a></li>  
 <li class="page-item active"><a href="#" class="page-link">1</a></li>  
 <li class="page-item"><a href="#" class="page-link">2</a></li>  
 <li class="page-item"><a href="#" class="page-link">3</a></li>  
 <li class="page-item"><a href="#" class="page-link">4</a></li>  
 <li class="page-item"><a href="#" aria-label="Next" class="page-link"><span  
 aria-hidden="true">»</span><span class="sr-only">Next</span></a></li>  
 </ul>  
 </nav>  
 </div>  
 </div>  
 </div>  
 </div>  
 </section>  
  
{% endblock %}

applicants.html (Employer)

{% extends 'base.html' %}  
{% block title %}  
 Applicants  
{% endblock %}  
{% load static %}  
  
{% block content %}  
  
 <section class="bg-light-gray">  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-8 mx-auto">  
 <h1 class="heading">Applicants for the <span class="accent">{{ job.title }}</span> position</h1>  
 </div>  
 </div>  
 </div>  
 </section>  
 <section>  
 <div class="container">  
 <div class="row">  
 {% for applicant in applicants %}  
 <div class="col-lg-6">  
 <div class="box applicant">  
 <h4>{{ applicant.user.get\_full\_name }}</h4>  
 <a href="#" data-toggle="modal" data-target="#contact-modal"  
 class="btn btn-outline-white-secondary btn-sm">  
 <i class="fa fa-envelope"></i>Contact  
 </a>  
 <p>  
 <span class="badge badge-info">Applied {{ applicant.created\_at.astimezone }}</span>  
 </p>  
 <div class="d-flex justify-content-between">  
 <p class="intro"></p>  
 <div class="image">  
 <img src="{% static 'img/avatar.png' %}" alt=""  
 class="img-fluid rounded-circle">  
 </div>  
 </div>  
 </div>  
 </div>  
 {% endfor %}  
 {% if is\_paginated %}  
 <div class="row">  
 <div class="col-md-12">  
 <nav aria-label="Page navigation example" class="d-flex justify-content-center mt-4 mb-4">  
 <ul class="pagination">  
 {% if page\_obj.has\_previous %}  
 <li class="page-item">  
 <a href="?page={{ page\_obj.previous\_page\_number }}"  
 class="page-link">&laquo;</a>  
 </li>  
 {% else %}  
 <li class="page-item disabled">  
 <a class="page-link">&laquo;</a>  
 </li>  
 {% endif %}  
 {% for i in page\_obj.paginator.page\_range %}  
 {% if page\_obj.number == i %}  
 <li class="page-item active">  
 <a class="page-link">{{ i }}</a>  
 </li>  
 {% else %}  
 <li class="page-item">  
 <a href="?page={{ i }}" class="page-link">{{ i }}</a>  
 </li>  
 {% endif %}  
 {% endfor %}  
 {% if page\_obj.has\_next %}  
 <li class="page-item">  
 <a href="?page={{ page\_obj.next\_page\_number }}"  
 class="page-link">&raquo;</a>  
 </li>  
 {% else %}  
 <li class="page-item disabled">  
 <a class="page-link">&raquo;</a>  
 </li>  
 {% endif %}  
 </ul>  
 </nav>  
 </div>  
 </div>  
 {% else %}  
 <h3>You are end</h3>  
 {% endif %}  
 </div>  
 </div>  
 </section>  
  
{% endblock %}

all-applicants.html (Employer)

{% extends 'base.html' %}  
{% block title %}  
 Applicants  
{% endblock %}  
{% load static %}  
  
{% block content %}  
  
 <section class="bg-light-gray">  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-8 mx-auto">  
 <h1 class="heading">All applicants who applied for your position</h1>  
 </div>  
 </div>  
 </div>  
 </section>  
 <section>  
 <div class="container">  
 <div class="row">  
 {% for applicant in applicants %}  
 <div class="col-lg-6">  
 <div class="box applicant">  
 <h4>{{ applicant.user.get\_full\_name }}</h4>  
 <p><strong>Applied position: {{ applicant.job.title }}</strong></p>  
 <a href="#" data-toggle="modal" data-target="#contact-modal"  
 class="btn btn-outline-white-secondary btn-sm">  
 <i class="fa fa-envelope"></i>Contact  
 </a>  
 <p>  
 <span class="badge badge-info">Applied {{ applicant.created\_at.astimezone }}</span>  
 </p>  
 <div class="d-flex justify-content-between">  
 <p class="intro"></p>  
 <div class="image">  
 <img src="{% static 'img/avatar.png' %}" alt=""  
 class="img-fluid rounded-circle">  
 </div>  
 </div>  
 </div>  
 </div>  
 {% endfor %}  
 {% if is\_paginated %}  
 <div class="row">  
 <div class="col-md-12">  
 <nav aria-label="Page navigation example" class="d-flex justify-content-center mt-4 mb-4">  
 <ul class="pagination">  
 {% if page\_obj.has\_previous %}  
 <li class="page-item">  
 <a href="?page={{ page\_obj.previous\_page\_number }}"  
 class="page-link">&laquo;</a>  
 </li>  
 {% else %}  
 <li class="page-item disabled">  
 <a class="page-link">&laquo;</a>  
 </li>  
 {% endif %}  
 {% for i in page\_obj.paginator.page\_range %}  
 {% if page\_obj.number == i %}  
 <li class="page-item active">  
 <a class="page-link">{{ i }}</a>  
 </li>  
 {% else %}  
 <li class="page-item">  
 <a href="?page={{ i }}" class="page-link">{{ i }}</a>  
 </li>  
 {% endif %}  
 {% endfor %}  
 {% if page\_obj.has\_next %}  
 <li class="page-item">  
 <a href="?page={{ page\_obj.next\_page\_number }}"  
 class="page-link">&raquo;</a>  
 </li>  
 {% else %}  
 <li class="page-item disabled">  
 <a class="page-link">&raquo;</a>  
 </li>  
 {% endif %}  
 </ul>  
 </nav>  
 </div>  
 </div>  
 {% else %}  
 <h3>You are end</h3>  
 {% endif %}  
 </div>  
 </div>  
 </section>  
  
{% endblock %}

settings.py

*"""  
Django settings for jobs project.  
  
Generated by 'django-admin startproject' using Django 2.1.7.  
  
For more information on this file, see  
https://docs.djangoproject.com/en/2.1/topics/settings/  
  
For the full list of settings and their values, see  
https://docs.djangoproject.com/en/2.1/ref/settings/  
"""*import os  
  
# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)  
BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  
  
# Quick-start development settings - unsuitable for production  
# See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/  
  
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET\_KEY = '@pzqp#x^+#(olu#wy(6=mi9&a8n+g&x#af#apn07@j=5oin=xb'  
  
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True  
  
# Application definition  
  
INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'jobsapp',  
 'accounts',  
 'jobs',  
]  
  
MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
 'whitenoise.middleware.WhiteNoiseMiddleware',  
]  
  
ROOT\_URLCONF = 'jobs.urls'  
  
TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [os.path.join(BASE\_DIR, "templates")],  
 'APP\_DIRS': True,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug',  
 'django.template.context\_processors.request',  
 'django.contrib.auth.context\_processors.auth',  
 'django.contrib.messages.context\_processors.messages',  
 ],  
 },  
 },  
]  
  
WSGI\_APPLICATION = 'jobs.wsgi.application'  
  
# Database  
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.sqlite3',  
 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),  
 }  
}  
  
# Password validation  
# https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators  
  
AUTH\_PASSWORD\_VALIDATORS = [  
 # {  
 # 'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',  
 # },  
 # {  
 # 'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',  
 # },  
 # {  
 # 'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',  
 # },  
 # {  
 # 'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',  
 # },  
]  
  
# Internationalization  
# https://docs.djangoproject.com/en/2.1/topics/i18n/  
  
LANGUAGE\_CODE = 'en-us'  
  
TIME\_ZONE = 'UTC'  
  
USE\_I18N = True  
  
USE\_L10N = True  
  
USE\_TZ = True  
  
# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/2.1/howto/static-files/  
  
PROJECT\_ROOT = os.path.dirname(os.path.abspath(\_\_file\_\_))  
  
STATIC\_ROOT = os.path.join(PROJECT\_ROOT, 'staticfiles')  
ALLOWED\_HOSTS = ['django-portal.herokuapp.com', 'localhost', '\*']  
  
STATIC\_URL = '/static/'  
STATICFILES\_DIRS = [  
 os.path.join(BASE\_DIR, "static"),  
]  
STATICFILES\_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'  
  
MEDIA\_URL = "/media/"  
MEDIA\_ROOT = os.path.join(BASE\_DIR, "media")  
  
AUTH\_USER\_MODEL = "accounts.user"

urls.py

from django.contrib import admin  
from django.urls import path, include  
  
#from jobs import views  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', include('jobsapp.urls')),  
 path('', include('accounts.urls')),  
]

employee.py (views)

from django.contrib.auth.decorators import login\_required  
from django.http import Http404  
from django.urls import reverse\_lazy  
from django.utils.decorators import method\_decorator  
from django.views.generic import UpdateView  
  
from accounts.forms import EmployeeProfileUpdateForm  
from accounts.models import User  
from jobsapp.decorators import user\_is\_employee  
  
  
class EditProfileView(UpdateView):  
 model = User  
 form\_class = EmployeeProfileUpdateForm  
 context\_object\_name = 'employee'  
 template\_name = 'jobs/employee/edit-profile.html'  
 success\_url = reverse\_lazy('accounts:employer-profile-update')  
  
 @method\_decorator(login\_required(login\_url=reverse\_lazy('accounts:login')))  
 @method\_decorator(user\_is\_employee)  
 def dispatch(self, request, \*args, \*\*kwargs):  
 return super().dispatch(self.request, \*args, \*\*kwargs)  
  
 def get(self, request, \*args, \*\*kwargs):  
 try:  
 self.object = self.get\_object()  
 except Http404:  
 raise Http404("User doesn't exists")  
 # context = self.get\_context\_data(object=self.object)  
 return self.render\_to\_response(self.get\_context\_data())  
  
 def get\_object(self, queryset=None):  
 obj = self.request.user  
 print(obj)  
 if obj is None:  
 raise Http404("Job doesn't exists")  
 return obj

employer.py (views)

from django.contrib.auth.decorators import login\_required  
from django.http import HttpResponseRedirect  
from django.urls import reverse\_lazy  
from django.utils.decorators import method\_decorator  
from django.views.generic import CreateView, ListView  
  
from jobsapp.decorators import user\_is\_employer  
from jobsapp.forms import CreateJobForm  
from jobsapp.models import Job, Applicant  
  
  
class DashboardView(ListView):  
 model = Job  
 template\_name = 'jobs/employer/dashboard.html'  
 context\_object\_name = 'jobs'  
  
 @method\_decorator(login\_required(login\_url=reverse\_lazy('accounts:login')))  
 @method\_decorator(user\_is\_employer)  
 def dispatch(self, request, \*args, \*\*kwargs):  
 return super().dispatch(self.request, \*args, \*\*kwargs)  
  
 def get\_queryset(self):  
 return self.model.objects.filter(user\_id=self.request.user.id)  
  
  
class ApplicantPerJobView(ListView):  
 model = Applicant  
 template\_name = 'jobs/employer/applicants.html'  
 context\_object\_name = 'applicants'  
 paginate\_by = 1  
  
 @method\_decorator(login\_required(login\_url=reverse\_lazy('accounts:login')))  
 @method\_decorator(user\_is\_employer)  
 def dispatch(self, request, \*args, \*\*kwargs):  
 return super().dispatch(self.request, \*args, \*\*kwargs)  
  
 def get\_queryset(self):  
 return Applicant.objects.filter(job\_id=self.kwargs['job\_id']).order\_by('id')  
  
 def get\_context\_data(self, \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 context['job'] = Job.objects.get(id=self.kwargs['job\_id'])  
 return context  
  
  
class JobCreateView(CreateView):  
 template\_name = 'jobs/create.html'  
 form\_class = CreateJobForm  
 extra\_context = {  
 'title': 'Post New Job'  
 }  
 success\_url = reverse\_lazy('jobs:employer-dashboard')  
  
 @method\_decorator(login\_required(login\_url=reverse\_lazy('accounts:login')))  
 def dispatch(self, request, \*args, \*\*kwargs):  
 if not self.request.user.is\_authenticated:  
 return reverse\_lazy('accounts:login')  
 if self.request.user.is\_authenticated and self.request.user.role != 'employer':  
 return reverse\_lazy('accounts:login')  
 return super().dispatch(self.request, \*args, \*\*kwargs)  
  
 def form\_valid(self, form):  
 form.instance.user = self.request.user  
 return super(JobCreateView, self).form\_valid(form)  
  
 def post(self, request, \*args, \*\*kwargs):  
 form = self.get\_form()  
 if form.is\_valid():  
 return self.form\_valid(form)  
 else:  
 return self.form\_invalid(form)  
  
  
class ApplicantsListView(ListView):  
 model = Applicant  
 template\_name = 'jobs/employer/all-applicants.html'  
 context\_object\_name = 'applicants'  
  
 def get\_queryset(self):  
 # jobs = Job.objects.filter(user\_id=self.request.user.id)  
 return self.model.objects.filter(job\_\_user\_id=self.request.user.id)  
  
  
@login\_required(login\_url=reverse\_lazy('accounts:login'))  
def filled(request, job\_id=None):  
 job = Job.objects.get(user\_id=request.user.id, id=job\_id)  
 job.filled = True  
 job.save()  
 return HttpResponseRedirect(reverse\_lazy('jobs:employer-dashboard'))

home.py (views)

from django.contrib import messages  
from django.contrib.auth.decorators import login\_required  
from django.http import Http404, HttpResponseRedirect  
from django.urls import reverse\_lazy  
from django.utils import timezone  
from django.utils.decorators import method\_decorator  
from django.views.generic import ListView, DetailView, CreateView  
  
from jobsapp.forms import ApplyJobForm  
from jobsapp.models import Job, Applicant  
  
from django.views.generic import TemplateView  
  
  
class AboutView(TemplateView):  
 template\_name = "about.html"  
  
  
from django.views.generic import TemplateView  
  
  
class AboutView(TemplateView):  
 template\_name = "about.html"  
  
class ContactView(TemplateView):  
 template\_name = "contact.html"  
  
class HomeView(ListView):  
 model = Job  
 template\_name = 'home.html'  
 context\_object\_name = 'jobs'  
  
 def get\_queryset(self):  
 return self.model.objects.all()[:6]  
  
 def get\_context\_data(self, \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 context['trendings'] = self.model.objects.filter(created\_at\_\_month=timezone.now().month)[:3]  
 return context  
  
  
class SearchView(ListView):  
 model = Job  
 template\_name = 'jobs/search.html'  
 context\_object\_name = 'jobs'  
  
 def get\_queryset(self):  
 return self.model.objects.filter(location\_\_contains=self.request.GET['location'],  
 title\_\_contains=self.request.GET['position'])  
  
  
class JobListView(ListView):  
 model = Job  
 template\_name = 'jobs/jobs.html'  
 context\_object\_name = 'jobs'  
 paginate\_by = 5  
  
  
class JobDetailsView(DetailView):  
 model = Job  
 template\_name = 'jobs/details.html'  
 context\_object\_name = 'job'  
 pk\_url\_kwarg = 'id'  
  
 def get\_object(self, queryset=None):  
 obj = super(JobDetailsView, self).get\_object(queryset=queryset)  
 if obj is None:  
 raise Http404("Job doesn't exists")  
 return obj  
  
 def get(self, request, \*args, \*\*kwargs):  
 try:  
 self.object = self.get\_object()  
 except Http404:  
 # redirect here  
 raise Http404("Job doesn't exists")  
 context = self.get\_context\_data(object=self.object)  
 return self.render\_to\_response(context)  
  
  
class ApplyJobView(CreateView):  
 model = Applicant  
 form\_class = ApplyJobForm  
 slug\_field = 'job\_id'  
 slug\_url\_kwarg = 'job\_id'  
  
 @method\_decorator(login\_required(login\_url=reverse\_lazy('accounts:login')))  
 def dispatch(self, request, \*args, \*\*kwargs):  
 return super().dispatch(self.request, \*args, \*\*kwargs)  
  
 def post(self, request, \*args, \*\*kwargs):  
 form = self.get\_form()  
 if form.is\_valid():  
 messages.info(self.request, 'Successfully applied for the job!')  
 return self.form\_valid(form)  
 else:  
 return HttpResponseRedirect(reverse\_lazy('jobs:home'))  
  
 def get\_success\_url(self):  
 return reverse\_lazy('jobs:jobs-detail', kwargs={'id': self.kwargs['job\_id']})  
  
 # def get\_form\_kwargs(self):  
 # kwargs = super(ApplyJobView, self).get\_form\_kwargs()  
 # print(kwargs)  
 # kwargs['job'] = 1  
 # return kwargs  
  
 def form\_valid(self, form):  
 # check if user already applied  
 applicant = Applicant.objects.filter(user\_id=self.request.user.id, job\_id=self.kwargs['job\_id'])  
 if applicant:  
 messages.info(self.request, 'You already applied for this job')  
 return HttpResponseRedirect(self.get\_success\_url())  
 # save applicant  
 form.instance.user = self.request.user  
 form.save()  
 return super().form\_valid(form)

forms.py (jobsapp)

from django import forms  
  
from jobsapp.models import Job, Applicant  
  
  
class CreateJobForm(forms.ModelForm):  
 class Meta:  
 model = Job  
 exclude = ('user', 'created\_at',)  
  
 def is\_valid(self):  
 valid = super(CreateJobForm, self).is\_valid()  
  
 # if already valid, then return True  
 if valid:  
 return valid  
 return valid  
  
 def save(self, commit=True):  
 job = super(CreateJobForm, self).save(commit=False)  
 if commit:  
 job.save()  
 return job  
  
  
class ApplyJobForm(forms.ModelForm):  
 class Meta:  
 model = Applicant  
 fields = ('job',)

models.py (jobsapp)

from django.db import models  
from django.utils import timezone  
  
from accounts.models import User  
  
JOB\_TYPE = (  
 ('1', "Full time"),  
 ('2', "Part time"),  
 ('3', "Internship"),  
)  
  
  
class Job(models.Model):  
 user = models.ForeignKey(User, on\_delete=models.CASCADE)  
 title = models.CharField(max\_length=300)  
 description = models.TextField()  
 location = models.CharField(max\_length=150)  
 type = models.CharField(choices=JOB\_TYPE, max\_length=10)  
 category = models.CharField(max\_length=100)  
 last\_date = models.DateTimeField()  
 company\_name = models.CharField(max\_length=100)  
 company\_description = models.CharField(max\_length=300)  
 website = models.CharField(max\_length=100, default="")  
 created\_at = models.DateTimeField(default=timezone.now)  
 filled = models.BooleanField(default=False)  
  
 def \_\_str\_\_(self):  
 return self.title  
  
  
class Applicant(models.Model):  
 user = models.ForeignKey(User, on\_delete=models.CASCADE)  
 job = models.ForeignKey(Job, on\_delete=models.CASCADE, related\_name='applicants')  
 created\_at = models.DateTimeField(default=timezone.now)  
  
 def \_\_str\_\_(self):  
 return self.user.get\_full\_name()

urls.py (jobsapp)

from django.urls import path, include  
  
from .views import \*  
  
app\_name = "jobs"  
  
urlpatterns = [  
 path('', HomeView.as\_view(), name='home'),  
 path('about', AboutView.as\_view()),  
 path('contact', ContactView.as\_view()),  
 path('search', SearchView.as\_view(), name='searh'),  
 path('employer/dashboard', include([  
 path('', DashboardView.as\_view(), name='employer-dashboard'),  
 path('all-applicants', ApplicantsListView.as\_view(), name='employer-all-applicants'),  
 path('applicants/<int:job\_id>', ApplicantPerJobView.as\_view(), name='employer-dashboard-applicants'),  
 path('mark-filled/<int:job\_id>', filled, name='job-mark-filled'),  
 ])),  
 path('apply-job/<int:job\_id>', ApplyJobView.as\_view(), name='apply-job'),  
 path('jobs', JobListView.as\_view(), name='jobs'),  
 path('jobs/<int:id>', JobDetailsView.as\_view(), name='jobs-detail'),  
 path('employer/jobs/create', JobCreateView.as\_view(), name='employer-jobs-create'),  
]

forms.py (accounts)

from django import forms  
from django.contrib.auth import authenticate  
from django.contrib.auth.forms import UserCreationForm  
  
from accounts.models import User  
  
GENDER\_CHOICES = (  
 ('male', 'Male'),  
 ('female', 'Female'))  
  
  
class EmployeeRegistrationForm(UserCreationForm):  
 # gender = forms.MultipleChoiceField(widget=forms.CheckboxSelectMultiple, choices=GENDER\_CHOICES)  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(EmployeeRegistrationForm, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['gender'].required = True  
 self.fields['first\_name'].label = "First Name"  
 self.fields['last\_name'].label = "Last Name"  
 self.fields['password1'].label = "Password"  
 self.fields['password2'].label = "Confirm Password"  
  
 # self.fields['gender'].widget = forms.CheckboxInput()  
  
 self.fields['first\_name'].widget.attrs.update(  
 {  
 'placeholder': 'Enter First Name',  
 }  
 )  
 self.fields['last\_name'].widget.attrs.update(  
 {  
 'placeholder': 'Enter Last Name',  
 }  
 )  
 self.fields['email'].widget.attrs.update(  
 {  
 'placeholder': 'Enter Email',  
 }  
 )  
 self.fields['password1'].widget.attrs.update(  
 {  
 'placeholder': 'Enter Password',  
 }  
 )  
 self.fields['password2'].widget.attrs.update(  
 {  
 'placeholder': 'Confirm Password',  
 }  
 )  
  
 class Meta:  
 model = User  
 fields = ['first\_name', 'last\_name', 'email', 'password1', 'password2', 'gender']  
 error\_messages = {  
 'first\_name': {  
 'required': 'First name is required',  
 'max\_length': 'Name is too long'  
 },  
 'last\_name': {  
 'required': 'Last name is required',  
 'max\_length': 'Last Name is too long'  
 },  
 'gender': {  
 'required': 'Gender is required'  
 }  
 }  
  
 def clean\_gender(self):  
 gender = self.cleaned\_data.get('gender')  
 if not gender:  
 raise forms.ValidationError("Gender is required")  
 return gender  
  
 def save(self, commit=True):  
 user = super(UserCreationForm, self).save(commit=False)  
 user.role = "employee"  
 if commit:  
 user.save()  
 return user  
  
  
class EmployerRegistrationForm(UserCreationForm):  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(EmployerRegistrationForm, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['first\_name'].label = "Company Name"  
 self.fields['last\_name'].label = "Company Address"  
 self.fields['password1'].label = "Password"  
 self.fields['password2'].label = "Confirm Password"  
  
 self.fields['first\_name'].widget.attrs.update(  
 {  
 'placeholder': 'Enter Company Name',  
 }  
 )  
 self.fields['last\_name'].widget.attrs.update(  
 {  
 'placeholder': 'Enter Company Address',  
 }  
 )  
 self.fields['email'].widget.attrs.update(  
 {  
 'placeholder': 'Enter Email',  
 }  
 )  
 self.fields['password1'].widget.attrs.update(  
 {  
 'placeholder': 'Enter Password',  
 }  
 )  
 self.fields['password2'].widget.attrs.update(  
 {  
 'placeholder': 'Confirm Password',  
 }  
 )  
  
 class Meta:  
 model = User  
 fields = ['first\_name', 'last\_name', 'email', 'password1', 'password2']  
 error\_messages = {  
 'first\_name': {  
 'required': 'First name is required',  
 'max\_length': 'Name is too long'  
 },  
 'last\_name': {  
 'required': 'Last name is required',  
 'max\_length': 'Last Name is too long'  
 }  
 }  
  
 def save(self, commit=True):  
 user = super(UserCreationForm, self).save(commit=False)  
 user.role = "employer"  
 if commit:  
 user.save()  
 return user  
  
  
class UserLoginForm(forms.Form):  
 email = forms.EmailField()  
 password = forms.CharField(  
 label="Password",  
 strip=False,  
 widget=forms.PasswordInput,  
 )  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.user = None  
 self.fields['email'].widget.attrs.update({'placeholder': 'Enter Email'})  
 self.fields['password'].widget.attrs.update({'placeholder': 'Enter Password'})  
  
 def clean(self, \*args, \*\*kwargs):  
 email = self.cleaned\_data.get("email")  
 password = self.cleaned\_data.get("password")  
  
 if email and password:  
 self.user = authenticate(email=email, password=password)  
  
 if self.user is None:  
 raise forms.ValidationError("User Does Not Exist.")  
 if not self.user.check\_password(password):  
 raise forms.ValidationError("Password Does not Match.")  
 if not self.user.is\_active:  
 raise forms.ValidationError("User is not Active.")  
  
 return super(UserLoginForm, self).clean(\*args, \*\*kwargs)  
  
 def get\_user(self):  
 return self.user  
  
  
class EmployeeProfileUpdateForm(forms.ModelForm):  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(EmployeeProfileUpdateForm, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['first\_name'].widget.attrs.update(  
 {  
 'placeholder': 'Enter First Name',  
 }  
 )  
 self.fields['last\_name'].widget.attrs.update(  
 {  
 'placeholder': 'Enter Last Name',  
 }  
 )  
  
 class Meta:  
 model = User  
 fields = ["first\_name", "last\_name", "gender"]

managers.py (accounts)

from django.contrib.auth.models import BaseUserManager  
  
  
class UserManager(BaseUserManager):  
 *"""Define a model manager for User model with no username field."""* use\_in\_migrations = True  
  
 def \_create\_user(self, email, password, \*\*extra\_fields):  
 *"""Create and save a User with the given email and password."""* if not email:  
 raise ValueError('The given email must be set')  
 email = self.normalize\_email(email)  
 user = self.model(email=email, \*\*extra\_fields)  
 user.set\_password(password)  
 user.save(using=self.\_db)  
 return user  
  
 def create\_user(self, email, password=None, \*\*extra\_fields):  
 *"""Create and save a regular User with the given email and password."""* extra\_fields.setdefault('is\_staff', False)  
 extra\_fields.setdefault('is\_superuser', False)  
 return self.\_create\_user(email, password, \*\*extra\_fields)  
  
 def create\_superuser(self, email, password, \*\*extra\_fields):  
 *"""Create and save a SuperUser with the given email and password."""* extra\_fields.setdefault('is\_staff', True)  
 extra\_fields.setdefault('is\_superuser', True)  
  
 if extra\_fields.get('is\_staff') is not True:  
 raise ValueError('Superuser must have is\_staff=True.')  
 if extra\_fields.get('is\_superuser') is not True:  
 raise ValueError('Superuser must have is\_superuser=True.')  
  
 return self.\_create\_user(email, password, \*\*extra\_fields)

models.py (accounts)

from django.contrib.auth.models import AbstractUser  
from django.db import models  
  
from accounts.managers import UserManager  
  
GENDER\_CHOICES = (  
 ('male', 'Male'),  
 ('female', 'Female'))  
  
  
class User(AbstractUser):  
 username = None  
 role = models.CharField(max\_length=12, error\_messages={  
 'required': "Role must be provided"  
 })  
 gender = models.CharField(max\_length=10, blank=True, null=True, default="")  
 email = models.EmailField(unique=True, blank=False,  
 error\_messages={  
 'unique': "A user with that email already exists.",  
 })  
  
 USERNAME\_FIELD = "email"  
 REQUIRED\_FIELDS = []  
  
 def \_\_unicode\_\_(self):  
 return self.email  
  
 objects = UserManager()

urls.py (accounts)

from django.urls import path, include  
  
from django.conf import settings  
from django.conf.urls.static import static  
  
from jobsapp.views import EditProfileView  
from .views import \*  
  
app\_name = "accounts"  
  
urlpatterns = [  
 path('employee/register', RegisterEmployeeView.as\_view(), name='employee-register'),  
 path('employer/register', RegisterEmployerView.as\_view(), name='employer-register'),  
 path('employee/profile/update', EditProfileView.as\_view(), name='employer-profile-update'),  
 path('logout', LogoutView.as\_view(), name='logout'),  
 path('login', LoginView.as\_view(), name='login'),  
]+static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

views.py (accounts)

from django.contrib import messages, auth  
from django.http import HttpResponseRedirect  
from django.shortcuts import render, redirect  
from django.views.generic import CreateView, FormView, RedirectView  
from accounts.forms import \*  
from accounts.models import User  
  
  
class RegisterEmployeeView(CreateView):  
 model = User  
 form\_class = EmployeeRegistrationForm  
 template\_name = 'accounts/admin/register.html'  
 success\_url = '/'  
  
 extra\_context = {  
 'title': 'Register'  
 }  
  
 def dispatch(self, request, \*args, \*\*kwargs):  
 if self.request.user.is\_authenticated:  
 return HttpResponseRedirect(self.get\_success\_url())  
 return super().dispatch(self.request, \*args, \*\*kwargs)  
  
 def post(self, request, \*args, \*\*kwargs):  
  
 form = self.form\_class(data=request.POST)  
  
 if form.is\_valid():  
 user = form.save(commit=False)  
 password = form.cleaned\_data.get("password1")  
 user.set\_password(password)  
 user.save()  
 return redirect('accounts:login')  
 else:  
 return render(request, 'accounts/employee/register.html', {'form': form})  
  
class RegisterEmployeeView(CreateView):  
 model = User  
 form\_class = EmployeeRegistrationForm  
 template\_name = 'accounts/employee/register.html'  
 success\_url = '/'  
  
 extra\_context = {  
 'title': 'Register'  
 }  
  
 def dispatch(self, request, \*args, \*\*kwargs):  
 if self.request.user.is\_authenticated:  
 return HttpResponseRedirect(self.get\_success\_url())  
 return super().dispatch(self.request, \*args, \*\*kwargs)  
  
 def post(self, request, \*args, \*\*kwargs):  
  
 form = self.form\_class(data=request.POST)  
  
 if form.is\_valid():  
 user = form.save(commit=False)  
 password = form.cleaned\_data.get("password1")  
 user.set\_password(password)  
 user.save()  
 return redirect('accounts:login')  
 else:  
 return render(request, 'accounts/employee/register.html', {'form': form})  
  
  
class RegisterEmployerView(CreateView):  
 model = User  
 form\_class = EmployerRegistrationForm  
 template\_name = 'accounts/employer/register.html'  
 success\_url = '/'  
  
 extra\_context = {  
 'title': 'Register'  
 }  
  
 def dispatch(self, request, \*args, \*\*kwargs):  
 if self.request.user.is\_authenticated:  
 return HttpResponseRedirect(self.get\_success\_url())  
 return super().dispatch(self.request, \*args, \*\*kwargs)  
  
 def post(self, request, \*args, \*\*kwargs):  
  
 form = self.form\_class(data=request.POST)  
  
 if form.is\_valid():  
 user = form.save(commit=False)  
 password = form.cleaned\_data.get("password1")  
 user.set\_password(password)  
 user.save()  
 return redirect('accounts:login')  
 else:  
 return render(request, 'accounts/employer/register.html', {'form': form})  
  
  
class LoginView(FormView):  
 *"""  
 Provides the ability to login as a user with an email and password  
 """* success\_url = '/'  
 form\_class = UserLoginForm  
 template\_name = 'accounts/login.html'  
  
 extra\_context = {  
 'title': 'Login'  
 }  
  
 def dispatch(self, request, \*args, \*\*kwargs):  
 if self.request.user.is\_authenticated:  
 return HttpResponseRedirect(self.get\_success\_url())  
 return super().dispatch(self.request, \*args, \*\*kwargs)  
  
 def get\_success\_url(self):  
 if 'next' in self.request.GET and self.request.GET['next'] != '':  
 return self.request.GET['next']  
 else:  
 return self.success\_url  
  
 def get\_form\_class(self):  
 return self.form\_class  
  
 def form\_valid(self, form):  
 auth.login(self.request, form.get\_user())  
 return HttpResponseRedirect(self.get\_success\_url())  
  
 def form\_invalid(self, form):  
 *"""If the form is invalid, render the invalid form."""* return self.render\_to\_response(self.get\_context\_data(form=form))  
  
  
class LogoutView(RedirectView):  
 *"""  
 Provides users the ability to logout  
 """* url = '/login'  
  
 def get(self, request, \*args, \*\*kwargs):  
 auth.logout(request)  
 messages.success(request, 'You are now logged out')  
 return super(LogoutView, self).get(request, \*args, \*\*kwargs)

manage.py

#!/usr/bin/env python  
*"""Django's command-line utility for administrative tasks."""*import os  
import sys  
  
  
def main():  
 *"""Run administrative tasks."""* os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'jobs.settings')  
 try:  
 from django.core.management import execute\_from\_command\_line  
 except ImportError as exc:  
 raise ImportError(  
 "Couldn't import Django. Are you sure it's installed and "  
 "available on your PYTHONPATH environment variable? Did you "  
 "forget to activate a virtual environment?"  
 ) from exc  
 execute\_from\_command\_line(sys.argv)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

5. Standardization of the coding

**5.1) Code Efficiency**

Efficiency of any system is concern with the proper use of scare recourses by the system. The need of efficiency arises due to cost considerations. If some resources are scarce and expensive, it is desirable that those resources be used efficiency are processor time and memory. An efficient system is concern with less CPU interaction and less memory. It is important to make my software as a quick and intelligent to solve the desire purpose with in some fraction of second. Code efficiency means that especially code that is executed often with in tight loop.

Simplicity of the code an also consideration factor. Efficient the coding structures handle the entire problem to arrive at required output. In the system i.e. **“Online Job Portal ”** is designed in well-structured manner to suit “Code Efficiency” in such a way that can prove by default. I use some tools to make the code efficient. If the code is not simple then they maintain and up- gradation will be a tedious job for the user of the system

The said criteria are not nor may independent and increase of one have unfavorable for another.

A code should be design clearly that can be verifiable, complete (implements all the specification) and traceable (all design elements can be traced to some requirements).

**Code Optimization**

In the design methodology used, the basic structure of the design was created during analysis. As the analysis is concerned with capturing and representing various aspects of the problem, some inefficiency may have crept in. In this final step, the issue of efficiency is considered, keeping in mind that the final structures should not deviate too much from the logical structure produced by the analysis time.

Optimization of the code is the intermediate process of the system design. After completion of the coding, Optimization of the code is evaluated. There are number of different steps through which we can design the system and maintain the entire system with the application.

**While developing the system ,I have sincerely taken the following factors:-**

* Minimize the code size and accurate code forget desire output in less execution time.
* Internal relation between modules is managed very efficiently;also try to decrease the interdependency between modules.
* Normalize the table as normalize as possible for preventing redundancy of data.
* Easily modifiable to rectify and type of errors found during testing or actual operations.
* Write procedures/function for same type of models execution.
* Inclusion of entire new modules (System Expandability) becomes easy and simple.
* Flexibility for add new module at any time.

So, I think that my **“Online Job Portal”** system wills easily shootout any times of real life problems in an efficient manner.

**5.2) Error Handling**

Error Handling refers to the programming practice of anticipating and coding for error conditions that may arise when our program runs. Errors in general come in three flavors:-

* **Compiler errors**

Such as undeclared variables that prevent our code from compiling; user data entry error such as a user entering a negative value where only a positive number is acceptable; and run time errors, that occur when system cannot correctly execute a program statement.

* **Run time errors**

Run time errors, that occur when system cannot correctly execute a program statement.

* **Typical Run time errors**

Typical run time errors include attempting to access a non-existent worksheet or workbook, or attempting to divide by zero. The example code in this article will use the division by zero error when we want to deliberately raise an error

6. Testing

Testing is a process of executing a program with the goal of finding errors. So, testing means that one inspects behavior of a program on a finite set of test cases(a set of inputs, execution pre conditions, and expected out comes developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement, for which valued inputs always exist. In practice, the whole set of test cases is considered as infinite, therefore the erotically there are too many test cases even for the simplest programs. In this case, testing could require months and months to execute. So, how to select the most proper set of test cases? In practice, various techniques are used for that, and some of them are correlated with risk analysis, while others with test engineering expertise. Testing is an activity performed for evaluating software quality and for improving it. Hence,the goal of testing is systematic detection of different classes of errors (error can be defined as a human action that produces an incorrect result, in a minimum amount of time and with a minimum amount of effort).

Once the account is activated, jobseekers can search, view and apply for active job openings. All the applied jobs details are stored in the Applied Jobs table of SQLite database. The applicants can also write/update reviews for the companies.

**6.1) Testing techniques and Testing strategies used**

# Testing Techniques

A testing technique specifies the strategy used in testing to select input test cases and analyses test results. Different techniques reveal different quality aspects of a software system, and there are two major categories of testing techniques, **functional** and **structural**.

**Functional Testing**

The software program or system under test is viewed as a **“black box”**. The selection of test cases for functional testing is based on the requirement or design specification of the software entity under test. Examples of expected results, sometimes are called test oracles, includes requirement /design specifications, hand calculated values, and simulated results. Functional testing emphasizes on the external behavior of the software entity.

**Structural Testing**

The software entity is viewed as a **“white box”**. The selection of test cases is based on the implementation of the software entity. The goal of selecting such test cases is to cause the execution of specific spots in the software entity, such as specific statements, program branches or paths. The expected results are evaluated on a set of coverage criteria. Examples of coverage criteria include path coverage, branch coverage, and data-flow coverage. Structural testing emphasizes on the internal structure of the software entity.

### Testing Strategies

**Unit testing**

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white-box style),to ensure that the specific unction is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

Depending on the organization's expectations for software development, unit testing might include static code analysis, data flow analysis metrics analysis, peer code reviews, code coverage analysis and other software verification practices.

### Integration testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be localized more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

### System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification.

### Acceptance testing

Acceptance testing is a test conducted to determine if the requirements of a specification or contract are met. It may involve chemical tests, physical tests, or performance tests.

In systems engineering it may involve black-box testing performed on a system (for example: a piece of software, lots of manufactured mechanical parts, or batches of chemical products) prior to its delivery.

Software developers often distinguish acceptance testing by the system provider from acceptance testing by the customer (the user or client) prior to accepting transfer of ownership. In the case of software, acceptance testing performed by the customer is known as **user acceptance testing (UAT)**, end-user testing, site (acceptance) testing, or field(acceptance) testing. A smoke test is used as an acceptance test prior to introducing a build to the main testing process.

**6.2) Testing Plan used**

**Static Testing**

Static testing is the testing of the objects in a web browser that do not change, or are not transaction based. This type of testing is done on a web page that has already been loaded into a web browser. There are several types of static testing, and they will be discussed in this section.

**Content Checking**

Once the web page has been loaded, it has to be tested for accuracy, completeness, consistency, spelling and accessibility.

These terms have the traditional meanings, and the tests are as elementary as they sound. However, it is in areas like these where the site is first judged by the website visitor. For example, if the site has numerous misspellings, the product that the website is offering may come into question as the visitor may feel that if the attention to detail is not given to the site, it may not be given to the product either. These tests are mentioned in this research paper as these are simple things that may not automatically be on a web tester’s test plan, as most of these are unique to the web.

**Browser Syntax Compatibility**

This test is one level below the actual content. It is the technology of how tore present the content, whether that content consists of text, graphics, or other web objects. This is an important test as it determines whether or not the page under test works in various browsers. Even regarding only Microsoft’s Internet Explorer and Netscape’s Navigator web browsers ,this is a significant issue due to the fact that there are many versions of both still in use. These versions do not work the same, and depending on what the minimum version and browser type requirements are, the pages need to be tested in each supported browser

**Visual Browser Validation**

Simply, does the content look the same, regardless of supported browser used? If the requirements are that only one browser and version will be supported by the application, this test is not necessary. However, even if more than one version of the same browser will be supported, the page under test should loaded into both browsers, and they should be visually checked to see if there are any differences in the physical appearance of the objects in the page. If there are, they may be things such as the centering of objects, table layouts ,etc.

The differences should be reviewed by the users to see if there is any need to change the page so that it appears exactly the same (if possible) in all of the supported browsers.

**Test Browsing**

Test browsing tests aim to find the defects regarding navigation through web pages, including the availability of linked pages, and other objects, as well as the download speed of the individual page under test. The integration of web pages to server- based components is tested, to ensure that the correct components are called from the correct pages.

**Browsing the Site**

When traversing links and opening new pages, when a new page is opened , several questions should be addressee done ached every page the system links to. Can the page be downloaded and displayed? Do all objects load in an acceptable time (“acceptable” would be based on the business requirements)? When user turns the browser option of “images-load” to “off” – does the page still work? Do all of the text and graphical links work? All of these questions are important, as if the answer to any of them is in the negative, it would be considered defect.

Other issues to validate are whether the site still works if JavaScript or Java is disabled, or if a certain plug-in is not loaded or disabled. A good test case is to use a browser with no plug-ins loaded during testing, and when the tester is queried to download a plug-in, they should not load them, and see how the site reacts without the plug-in.

### Functional Testing

**Browser-Page Tests**

This type of test covers the objects and code that executes with in the browser, but does not execute the server-based components. For example, JavaScript and VBScript code within HTML that does rollovers, and other special effects. This type of test also includes field validations that are done at the HTML level. Additionally, browser-page tests include Java applets that implement screen functionality or graphical output.

### Non-Functional Testing

**Configuration Testing**

Beyond the browser validation, this type of test takes into consideration the operating system platforms used, the type of network connection,

internet service provider type, and browser used(including version).There all work for this type of test is ensuring that the requirements and assumptions are understood by the development team, and that a test environment with those choices is put in place to properly testis.

**Usability**

For usability, the tests can be subjective, but there are standards and guidelines that have been established throughout the industry and it would be easy for a project exam to blindly follow them, and feel that the site will be acceptable since the standards are followed.

However, human-computer interaction standards and guidelines can not guarantee usable website. Designers should not rely on them for all or even most of the design decisions, and project managers should not let standards compliance lull the team into complacency, thinking that since the standards are followed, that the site will automatically meet the needs of the users, their tasks, and their work environment.

A proactive suggestion is that while establishing the design guidelines, to define requirements that can be positively identified and measured .A way to do this is to capture and quantify the meaning of learn ability, understand ability, and operability in a testable form.

**Performance**

Performance testing is the validation that the system meets performance requirements. This can be as simplistic as ensuring at a web page load sin less than eight seconds, or can be as complex as requiring the system to handle 10,000 transactions per minute, while still being able to load a web page within eight seconds. The section below offers best practices for the execution of performance testing.

During research, one topic that was repeated was the importance that the performance- testing server is exactly like the production server; in fact, it ideally should be an exact replica in every way. It is very important to make sure every component (networks, firewalls, servers, mainframes) matches the production equipment.

Performance testing can bed one through the “window” of the browser, or directly on the server. If done on the server, some of the performance time that the browser takes is not accounted for. Scripting GUI orientated transactions to drive the browser interface can be much more complicated and the synchronization between test tool and browser it not always reliable. Therefore, if testers decide to ignore the performance time taken by the browsers, it is important to get “buy in” from project team members and users to understand the compromise. If there are issues, testers should advise management that

performance-testing using the GUI will introduce a time-intensive effort that may or may not impact the project timeline.

To assist with load and performance testing, testers should use the test scripts that have been created early in the project as a basis for initial load testing. By using the existing scripts, this avoids rework and allows the scripts to be used at different times by different virtual users when validating system performance.

**Scalability**

The term “scalability” can be defined as a web application’s ability to sustain its required number of simultaneous users and/or transactions, while maintaining adequate response times to its end users.

When testing’s capability, configuration of the server under test is critical. All logging levels, server timeouts, etc. need to be configured just like production. In an ideal situation, all of the configuration files should be simply copied from test environment to the production environment, with only minor changes to the global variables.

In order to test scalability, the web traffic loads must be determined to know what the threshold requirement for scalability should be.

**Security**: *-* Security is a critical part of an e-commerce website. Best practices to best test how secure the site is being in this section.

**Data Collection**: *-* Web sites collect data in log files, as well as through forms in which users supply to the website information that is saved on the web server. The web server should be setup so that users cannot browse directories and obtain file names.

**Cookies**: A cookie is a text file that is placed on a website visitor’s system that identifies the user’s “identity.” The cookie is retrieved when the user re-visits thesite at a later time. Cookies can expire in a short period of time, such as minutes or hours (session cookie) or can last for months or years (persistent cookie). Cookies can be controlled by the user, regarding whether they want to allow them or not.

**6.3) Testing Cases**

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 Online Job Portal web application.

|  |  |  |  |
| --- | --- | --- | --- |
| **TESTMODULE** | **TESTCASE** | **EXPECTEDRESULT** | **TESTRESULT** |
| ADMIN | Providevalidlogincredentials | User successfully logged in anddirectedtotheadmindashboard  page | PASS |
| ADMIN | Entersinvalidlogincredentials | DisplaysErrormessage | PASS |
| ADMIN | Uponsuccessfullogin,  click on the ‘List ofEmployers’tab. | Displaysthedetails oflistof  activeemployersregisteredwiththeapplication | PASS |
| ADMIN | Click on‘Active/Deactivate’tabunderstatusof the  employer | Thestatusoftheemployerwillbechangedtoactive/deactivate. | PASS |
| EMPLOYER | Provide details forregistration | Employersuccessfullyregisteredwiththeapplication | PASS |
| EMPLOYER | Uponsuccessfullogin,  click on ‘Post New Job’tab | Employer posts jobs with therequireddetails | PASS |
| EMPLOYER | Employertryingtopostjobwith insufficient  details | Prompts to fill in all the necessarydetailsof thejob | PASS |
| EMPLOYER | Employerclicksonthe‘List PostedJobs’ tab | All the jobs posted by theemployerwillbedisplayed. | PASS |

|  |  |  |  |
| --- | --- | --- | --- |
| EMPLOYER | Employer clicks on‘Active/deactivate’under  Status | Thestatusofthejobpostingwillchangedtoactive/deactivated. | PASS |
| EMPLOYER | Employerclicksonthe  ‘view’ tab undercandidatescolumn | Thelist of thedetails of  applicantsforaparticularjobpostingaredisplayed. | PASS |
| JOBSEEKER | Provide details forregistration | Jobseekersuccessfullyregisteredwiththeapplication | PASS |
| JOBSEEKER | Entersinvalidlogincredentials | Errormessagedisplayed | PASS |
| JOBSEEKER | Upon successful login,clickon‘MyProfile’tab | Listdetailsofjobseeker | PASS |
| JOBSEEKER | Upon successful login,clickon‘SearchJobs’tab | Detailsoftheactivejobpostingsaredisplayed. | PASS |
| JOBSEEKER | Upon successful login,clickon‘AppliedJobs’tab | Detailsofthejobsthatareappliedbythe jobseekeraredisplayed | PASS |
| JOBSEEKER | Clickon‘AddReview’tab | Displays a form to fill in thereviewdetailsofthecompany | PASS |
| JOBSEEKER | Logout | Redirects to the Home page of theapplication | PASS |

The test cases for the android application areas follows

**Test Objectives:** Navigation from Splash screen to Jobs screen

|  |  |  |  |
| --- | --- | --- | --- |
| **TEST**  **CONDITION** | **INPUT**  **SPECIFICATION** | **OUTPUT**  **SPECIFICATION** | **PASS/FAIL** |
| The user iscurrentlyonthe  Splashscreen | User enterscredentialsandclicks  onloginbutton | DirectstoJobsscreen | PASS |

|  |  |  |  |
| --- | --- | --- | --- |
| **TEST**  **CONDITION** | **INPUT**  **SPECIFICATION** | **OUTPUT**  **SPECIFICATION** | **PASS/FAIL** |
| The user iscurrentlyonthe  Jobsscreen | User clicks on the viewagainstaparticularjob | Directs to Job detailsscreen | PASS |

|  |  |  |  |
| --- | --- | --- | --- |
| **TEST**  **CONDITION** | **INPUT**  **SPECIFICATION** | **OUTPUT**  **SPECIFICATION** | **PASS/FAIL** |
| The user iscurrentlyontheJobspage | Userclicksonthe‘Apply’tabagainst ajobpost | Prompts a message as“Successfully applied to thisjob” | PASS |

|  |  |  |  |
| --- | --- | --- | --- |
| **TEST**  **CONDITION** | **INPUT**  **SPECIFICATION** | **OUTPUT**  **SPECIFICATION** | **PASS/FAIL** |
| The user is currentlyontheJobs page | Userclickson‘AppliedJobs’ tab | Listsallthejobsthatareapplied | PASS |

**Performance Testing**

Performance testing is performed to determine how well the system can perform in terms of responsiveness under all kinds of load. The web application is tested to see if it can sustain huge amount of requests providing higher throughput under different loads. I have simulated multiple hits on various pages of the application to evaluate the over all performance.

## Performance Analysis Tools

I have used Apache JMeter to test functional and performance both on static and dynamic resources(files, Views, Java scripts, Python Objects, Data Bases and Queries, FTP Servers and more).It can be used to simulate a heavy load on a server, network or object to test it’s strength or to analyze overall performance under different load types.

### System Configuration

The details of the configuration of the system that has been used for testing the application are listed below:

|  |  |
| --- | --- |
| OperatingSystem | Windows 10 (64 bit) |
| RAM | 8GB |
| Processor | Intel corei7 |
| ProcessorSpeed | 3.40GHz |

**6.4) Debugging**

# Debugging

After I have created my website and resolved the build errors. I must now correct those logic errors that keeps my application or stored procedures from running correctly. I have done this with the development environment's integrated debugging functions. These allowed me to stop at procedure locations, inspect memory and register values, change variables, observe message traffic, and get a close look at what my code does.

Visual Studio Code supports various types of debugging which helped me to debug my website.

### Parallel Debugging

Two new windows have been added for debugging parallel applications:

The GPU Threads window displays the status and the details of the threads running on the GPU.

The Parallel Watch window displays values of a single expression across multiple threads at the same time.

You can sort, reorder, configure, and group on the columns in the GPU Threads, Threads, Parallel Tasks, and Parallel Watch windows.

### Intelli Trace Debugging

You can record diagnostic events with the IntelliTrace collector for SharePoint 2010 applications running outside Visual Studio. This lets you save user profile events, Unified Logging System (ULS) events, and IntelliTrace events to an iTrace file. You can open this file in Visual Studio Ultimate to start diagnosing SharePoint 2010 applications in production or other environments.

**7. System Security measures**

**Database Security:**Database security is addressed at several levels:

* Data base file protection. All files stored within the database are protected from reading by any account other than the Post gores super user account.
* Connections from a client to the database server are, by default, allowed only via a local Unix socket, not via TCP/IP sockets. The backend must be started with the-i option to allow non-local clients to connect.
* Client connections can be restricted by IP address and/or user name via the pg h ba. conf file in PG DATA.
* Client connections may be authenticated vi other external packages.
* Each user in Postgres is assigned a username and (optionally) a password. By default, users do not have write access to databases they did not create.
* Users may be assigned to *groups*, and table access may be restricted based on group privileges.

# Creation Of User Profiles And Accessrights

Django provides an authentication and authorization ("permission") system, built on top of the session framework discussed in the previous tutorial, that allows you to verify user credentials and define what actions each user is allowed to perform. The framework includes built-in models for Users and Groups (a generic way of applying permissions to more than one user at a time), permissions/flags that designate whether a user may perform a task, forms and views for logging in users, and view tools for restricting content.

The authentication was enabled automatically when we created the skeleton website so you don't need to do anything more at this point.

The necessary configuration was all done for us when wee created the app using django- admin start project command. The database tables for users and model permissions were created when we first called python manage.py migrate

The configuration is set up in the INSTALLED\_APPS and MIDDLEWARE sections of the project file

The first user was a super user, created with the command python manage.py create super user). Our super user is already authenticated and has all permissions, so we'll need to create a test user to represent a normal site user. We'll be using the admin site to create our *local library* groups and website logins, as it is one of the quickest ways to do so.

Django provides almost everything you need to create authentication pages to handle login, log out, and password management "out of the box". This includes a URL mapper, views and forms, but it does not include the templates — we have to create our own!

**8. Cost Estimation of the Project**

### Process main phases:

In this simple top-down estimate process you can identify five main phases.

1. Define Activities
2. Define Task
3. Define Human Resources
4. Assign Human Resources to Tasks
5. Estimate times and costs

The process start with a general definition of macro-activities and with a detailed definition of tasks, human resources used, times and costs related to each task.

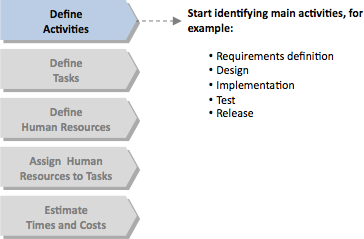
# 

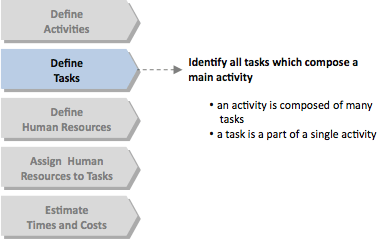
**Define Activities**: In this first phase you have to define the main activities which compose your project:

For example, in a generic web project you can identify the following main activities:

1. Requirements definition
2. Design
3. Implementation
4. Test
5. Release

Next step is to detail each activity with a certain number of specific tasks.

****

***Define Tasks****:* Each activity is composed from some tasks. Each task is a smaller piece of work which composes a main activity:

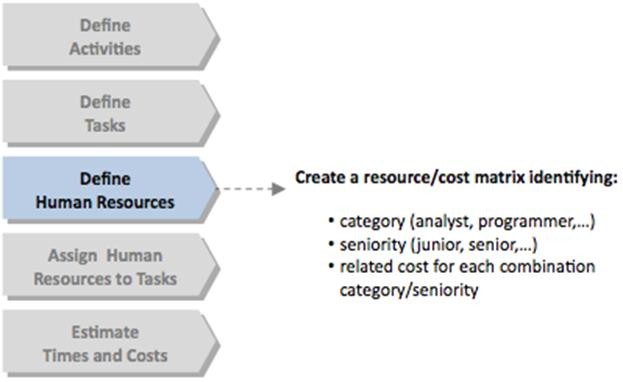
1. Requirementsdefinition
   1. Define application scope
   2. Define technical requirements
2. Design
   1. Application Map
   2. Database Entity relationship model

...

1. Implementation
   1. SQLcode
   2. HTML code

*CSScode*

**Define Human Resources**: Next step is defining human resources in terms of category, seniority and hourly cost:



Each category has a specific hourly cost related to specific seniority. You can organize these information using a simple category/seniority matrix. For example if you have to estimate a big/medium size project you can identify the following categories:

* Analyst
* Programmer
* Project manager

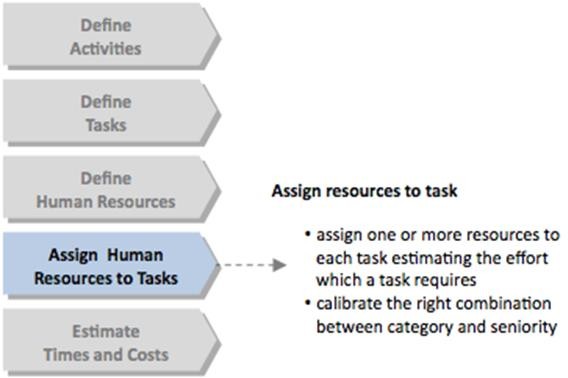
and the following seniorities

* Junior
* Senior

Now, define hourly cost for each category/seniority combination (in a more complex project you can also define a standard rate and an over time rate for each combination). In the spreadsheet ,you can create the table above in a new sheet called Resources in the same spreadsheet. At this point you have two sheets:

A first sheet with activities and a second sheet with resources. In this way when you assign resources to tasks you can link the cost of a specific resource with a reference formula(=). This is a good practice because if you have to change the cost related to a specific combination category/seniority, you can do it only once in the sheet "Resources" and automatically all changes will be reported in all instances(task) which use that combination in the sheet "Activities".

**Assign Human Resources to Tasks**: Next step: assigning one or more resources to eachtaskestimatingtheeffortwhichataskrequires.Thisisaverydelicateactivitybecause you have to calibrate the right combination between category and seniority of resources you want to use in your project in order to estimate correctly project times and costs.

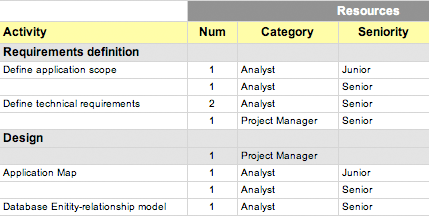


In the spreadsheet, in the sheet "Activities" create the following three columns:

Num (number of resources assigned to atask)

Category

Seniority

This is the result:

You can add different resources to each task (different category or different seniority) by simply adding a row below the task name (for example take a look at "Define application scope" where I added 1 analyst junior in the first row and 1 analyst senior in a new row below the task name).

**Estimate Times and Costs**: Now, for each resource, estimate the daily effort (Hours/day column), number of days(Days column),get cost related to category/seniority combinationfromthesheet"Resources"usingareferenceformula(HourlyCostcolumn), and calculate Total costs:



For each task (row)

***Total Cost is equal to: Total Cost = Hours/day \* Hourly Cost \* Days***

Take a mind some task could have specific costs which are independent from the number of resources you assign to that task. You can add this costs adding a new column to the left of the column Total Cost called "Additional Costs".

In this case Total Cost will be equal to:

# Total Cost = (Hours/day \* Hourly Cost \* Days) + Additional Cost

### COST ESTIMATION OF MY PROJECT

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Activities** | | | | | | | | |
|  |  |  |  |  |  |  |  | **Total** |
|  |  |  | **Resources** | |  |  |  | **Rs**  **30990.00** |
| **WBS** | **Activity** | **Num** | **Category** | **Seniority** | **Hours/day** | **Hourly**  **Cost** | **Days** | **Total Cost** |
| **1** | **Requirements**  **definition** |  |  |  |  |  |  | **Rs 4760.00** |
| 1.1 | Define application scope | 1 | Analyst | Junior | **8** | **Rs 45.00** | **4** | **Rs 1440.00** |
|  |  | 1 | Analyst | Senior | **4** | **Rs 55.00** | **4** | **Rs 880.00** |
| 1.2 | Define technical requirements | 2 | Analyst | Senior | **8** | **Rs 55.00** | **4** | **Rs 1760.00** |
|  |  | 1 | Project  Manager | Senior | **3** | **Rs 80.00** | **2** | **Rs 480.00** |
| **2** | **Design** |  |  |  |  |  |  | **Rs 6830.00** |
|  |  | 1 | Project Manager |  | 8 | Rs 80.00 | 5 | **Rs 3200.00** |
| 2.1 | Application Map | 1 | Analyst | Junior | 8 | Rs 55.00 | 4 | **Rs 1760.00** |
|  |  | 1 | Analyst | Senior | 5 | Rs 55.00 | 2 | **Rs 550.00** |
| 2.2 | Database Enitity- relationship model | 1 | Analyst | Senior | 8 | Rs 55.00 | 3 | **Rs 1320.00** |
| 2.3 | ... |  |  |  |  |  |  |  |
| **3** | **Implementation** |  |  |  |  |  |  | **Rs**  **19,400.00** |
|  |  | 1 | Project Manager |  | 8 | Rs 80.00 | 5 | **Rs 3200.00** |
| 3.1 | SQL code | 2 | Programmer | Senior | 8 | Rs 55.00 | 10 | **Rs 4400.00** |
| 3.2 | HTML code | 2 | Programmer | Senior | 8 | Rs 55.00 | 10 | **Rs 4400.00** |
| 3.3 | CSS code | 2 | Programmer | Senior | 8 | Rs 55.00 | 10 | **Rs 4400.00 SCREENSHOTS**  **SCREENSHOTS** |
|  |  |  |  |  |  |  |  |  |
| 3.4 | ... |  |  |  |  |  |  |  |

**9. Future Scope & Further Enhancement**

**Further Enhancement**

This project fulfills the primary requirements of the jobseekers 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 jobseeker, it can be extended to support functionalities of Employer as well.

**Conclusion**

Online Job Portal stands as a revolutionizing element in the sphere of recruitment. They act as a communication bridge between applicants and recruiters facilitating their requirements. This application helps organizations to have a greater exposure to the candidate pool and also jobseekers facilitating wide search of jobs matching their interests. This application provides an enhanced user experience for both employer and jobseeker. It provides user friendly interface which facilitates in reaching wide range of audience.

The application has achieved all the requirements that were initially set in the requirements gathering phase. This project taught me some best practices in the technology stack like Django MVT. Starting from requirement solicitation to design, construction, and implementation and testing, I have gained a very good experience working with various technologies a every phase. Development of this project boosted my confidence in mobile and web development.

**10. Bibliography**

Ref [Python Crash Course](https://geni.us/940d) ~ Eric Matthes

Ref [Python Programming](http://geni.us/BGc5Vs) ~ John M Zelle

Ref Software Engineering –Roger S. Pressman

Ref Build Websites with Python and Django ~ William S. Vincent.

Ref  [[**Essential SQLAlchemy**](https://www.amazon.com/dp/149191646X?tag=uuid10-20)](https://www.amazon.com/dp/149191646X?tag=uuid10-20) ~ [Jason Myers](https://bookauthority.org/author/Jason-Myers), [Rick Copeland](https://bookauthority.org/author/Rick-Copeland)

Ref RDBMS ~ Bipin C. Desai.

CD with Project Details