

ONLINE JOB PORTAL - EASYJOBS

A PROJECT REPORT

Submitted By

RUBY C POOVELIL

Reg.No: SJC17MCA027

to

the APJ Abdul Kalam Technological University
in partial fulfilment of the requirements for
the award of the degree

of

MASTER OF COMPUTER APPLICATIONS



Department of Computer Science and Applications
ST.JOSEPH'S COLLEGE OF ENGINEERING AND TECHNOLOGY
Bharananganam, Pravithanam Road, Kottayam, Palai,
Choondacherry, Kerala 686579

May, 2020

DECLARATION

I undersigned hereby declare that the project report “ONLINE JOB PORTAL - EASYJOBS”, submitted for partial fulfilment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bona fide work done by me under supervision of Mr. Jose George. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Palai

Date: 23-05-2020

RUBY C POOVELIL

Reg.No:SJC17MCA027

**DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS
ST.JOSEPH'S COLLEGE OF ENGINEERING AND TECHNOLOGY**

Palai, Choondacherry, Kottayam, 686579

(Approved by AICTE and affiliated to APJ Abdul Kalam Technological University)



CERTIFICATE

This is to certify that the report entitled **“ONLINE JOB PORTAL - EASYJOBS”** submitted by **“RUBY C POOVELIL , Reg.No: SJC17MCA027”** to the APJ Abdul Kalam Technological University in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications is a bona fide record of the project work carried out by her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Mr. Jose George
Internal Guide

Project Co-ordinator

Dr.T.D.Jainendrakumar
Head of the Department

Viva-voce held on:.....

External Examiner 1:

External Examiner 2:

ACKNOWLEDGEMENT

If words are considered as symbols of approval and tokens of acknowledgment, then let words play the heralding role in expressing my gratitude. To bring something into existence is truly a work of God. I would like to thank God for not letting me down and showing me the silver lining in the dark clouds.

I would like to thank Dr. J David, Principal, St. Joseph's College of Engineering & Technology for his support and encouragement. I convey my heartfelt thanks to Dr. T.D Jainendrakumar (Head of the Department - Master of Computer Applications, St. Joseph's College of Engineering & Technology,) for providing an opportunity for the project presentation. It is my pleasure to express my gratitude to the project coordinator Mr. Alex Jose, Asst.professor, Department of Computer Applications, St. Joseph's College of Engineering & Technology whose support and constructive criticism has led to the successful completion of the task.

With the biggest contribution to this report, I would like to thank Mr. Jose George Asst.proffessor, Department of Computer science and Applications who had given me full support in guiding me with stimulating suggestions and encouragement to go ahead in all the time of the this work.

I would also thank my institution and faculty, my family and friends without whom this project would have been a distant reality.

RUBY C POOVELIL

ABSTRACT

This project is about the recruitment process which is done online. This will allow the person to apply for a job in the company by making the payment. The person will be having the account after registration. This project is created for fulfilling the requests of the company managers. The users who visit the website can view the jobs in the company and will be able to apply directly from remote places. They can also search by location or by company name. This website will allow the candidates to upload their resume. Every time when the candidates apply for the job, their resume will be automatically sent to the company. The candidates have the facility to view notifications from companies, view and edit their profile etc.

This project will provide the registration facility to the job providers also. They can create their profile, post their jobs, view applications and resume, view shortlisted candidates, reject applications, send call letters, view application report, view profile, edit profile etc. Companies can register on this site by making payment

Administrator can manage the whole website. Admin can view the users of this website. He can view or edit or delete the job applications, selection list, users profile, posted jobs, notifications, statistics etc. Thus this project will help the job seekers to get better jobs and the job providers to get better employees.

System consist of 3 modules

- Admin
- Company
- Candidate

Technology : Python –Django

Database : PostgreSQL

CONTENTS

Contents	Page No.
ACKNOWLEDGEMENT	iv
ABSTRACT	v
LIST OF TABLES	ix
LIST OF FIGURES	ix
Chapter I: INTRODUCTION	1
I.1 Problem definition	1
I.2 About the organization	1
I.3 Objective of the Project	2
 Chapter II: LITERATURE SURVEY	 3
II.1 Initial Investigation	3
II.2 Existing System	3
II.3 Proposed System	3
II.3.1 Advantages of the Proposed System	4
II.3.2 Features of the Proposed System	4
II.4 Feasibility Study	4
II.4.1 Technical Feasibility	5
II.4.2 Economic Feasibility	5
II.4.3 Operational Feasibility	6
II.4.4 Behavioural Feasibility	6
 Chapter III: SYSTEM ANALYSIS AND DESIGN	 7
III.1 Software Requirement Specification	7
III.1.1 Hardware Requirement	7

III.1.2 Software Requirement	7
III.2 System Design	8
III.2.1 Non-Functional Requirements	8
III.3 Unified Modeling Language[UML]	8
III.3.1 Usecase Diagram	10
III.3.2 Sequence Diagram	16
III.3.3 Activity Diagram	18
III.4 System Design	20
III.4.1 Data Flow Diagram	26
III.4.2 Input Design	30
III.4.3 Output Design	30
III.4.4 Tables	31
III.5 Tools And Platforms	36
III.5.1 Python	36
III.5.2 Django Framework	38
III.5.3 Python – xhtml2pdf	40
III.5.4 Visual Studio Code IDE	41
 Chapter IV: SYSTEM TESTING	 42
IV.1 Testing Methodologies And Strategies	42
IV.1.1 Unit Testing	42
IV.1.2 Integration Testing	42
IV.1.3 system Testing	43
IV.1.4 User Acceptance Testing	43
IV.1.5 Sample Test Cases	44

Chapter V: SYSTEM IMPLEMENTATION	45
Chapter VI: CONCLUSION	46
REFERENCES	47
Chapter A: APPENDICES	48
A.1 Screenshots Input Form, Output Forms	48
A.2 Sample Code	54

LIST OF TABLES

No.	Title	Page No.
III.1	Table User	27
III.2	Table Com_pro	27
III.3	Table Can_pro	28
III.4	Table Notification	29
III.5	Table Jobpost.	29
III.6	Table Jobapply	30
III.7	Table Helpdesk	31
III.8	Table Reply	31
III.9	Table Payments	40
III.10	Login Test case	40
III.11	Registration Test case.	40
III.12	Job Post Test Case.	40

LIST OF FIGURES

No.	Title	Page No.
III.1	Actor	11
III.2	Use Case	11
III.3	Uml Diagram For Admin	13
III.4	Uml Diagram For Company	14
III.5	Uml Diagram For Candidate	15
III.6	Sequence Diagram For Admin.	17
III.7	Sequence Diagram For Company.	17
III.8	Sequence Diagram For Candidate.	18

III.9 Activity Diagram For Admin.	17
III.10 Activity Diagram For Company.	17
III.11 Activity Diagram For Candidate.	18
III.12 Level0	22
III.13 Level1-Admin	23
III.14 Level1-Company	24
III.15 Level1-Candidate	25
A.1 Home Page	42
A.2 Admin Home	43
A.3 Admin Activate/Reject candidate	43
A.4 Admin Activate/Reject company.	44
A.6 Admin –View Applications	44
A.5 Admin- Edit payment amount	44
A.7Company –Home Page.	44
A.8 Company –Notification.	45
A.9 Company –Post Job.	45
A.10 Candidate –Home Page	46
A.11 Candidate – View Job.	46
A.12 Candidate – Payment.	47
A.13 Candidate –Notification.	48

CHAPTER I

INTRODUCTION

I.1 PROBLEM DEFINITION

A job portal, also known as a career portal, is a modern name for an online job board that helps applicants to find jobs and helps the companies to get better employees. An employment website is a website that deals specifically with careers. This project is designed to allow companies to post job requirements for a position to be filled and are commonly known as job boards. Job Portal is a platform that joins recruiters and the job seekers to complete their goals and requirements. Recruiters look for a right candidate who has the right qualification to handle the responsibilities efficiently. Sending job applications through job portal is a quicker way to get the right candidate and the right job at the right time.

This project is about the recruitment process which is done online. This will allow the person to apply for a job in the company. This project is created for fulfilling the requests of the company managers. The users who visit the website can view the jobs in the company and will be able to apply directly from remote places. They can also search by location or by company name. This website will allow the candidates to upload their resume. Every time when the candidates apply for the job, their resume will be automatically sent to the company. The candidates have the facility to view notifications from companies, view and edit their profile etc.

This project will provide the registration facility to the job providers also. They can create their profile, post their jobs, view applications and resume, view shortlisted candidates, reject applications, send call letters, view application report, view profile, edit profile etc. Companies can register on this site by making payment

I.2 ABOUT THE ORGANIZATION

The college was founded by a group of well known educators. They are pioneering educators, having unmatched experience in the field of education with a belief that the continuous search for knowledge is the sole path to success. The Primary focus of the institution is to expose the

young minds to be world of technology, instilling in them confidence and fortitude to face new challenges that enables them to excel in their chosen field. The college inculcates the development of all facets of the mind culminating in an intellectual and balanced personality. Our team of dedicated and caring faculty strives to widen the students horizon of learning thereby achieving excellent results for every student. St. Joseph's college of Engineering & Technology (SJCET), has always been in the forefront with a wide spectrum of distinct features and facilities. The institution is a leader in the academia and its culture has set a benchmark in the region of quality in education. SJCET, Right from inception, has been maintaining high levels of standard in academic and extracurricular realms of activities. We offer BTECH degree courses in 6 engineering disciplines, and Masters Degree courses in Engineering, Computer Application and Business Administration. In the short span of a decade of its existence and among the six batches of students that have graduated, the college bagged several university ranks and has a remarkably high percentage of pass. The college is also the venue of national and state level seminars and symposiums and has emerged as the hub of technical education in the state. The placement scenario is also quite commendable, with several premier industries visiting ST. Joseph's college of engineering & technology for placement and recruitment.

I.3 OBJECTIVE OF THE PROJECT

“Online Job Portal-EasyJobs” is created for fulfilling the requests of the company managers. The person will be having the account after registration. The users who visit the website can view the jobs in the company and will be able to apply directly from remote places. They can also search by location or by company name. This website will allow the candidates to upload their resume. Every time when the candidates apply for the job, their resume will be automatically sent to the company. The candidates have the facility to view notifications from companies, view and edit their profile etc. This project will provide the registration facility to the job providers after making payment. They can create their profile, post their jobs, view applications and resume, view shortlisted candidates, reject applications, send call letters, view application report, view profile, edit profile etc. Administrator can manage the whole website. Admin can view the users of this website. He can view or edit or delete the job applications, users profile, posted jobs, notifications etc. Thus this project will help the job seekers to get better jobs and the job providers to get better employees.

CHAPTER II

LITERATURE SURVEY

II.1 INITIAL INVESTIGATION

The purpose of the preliminary investigation is to determine whether the problem or deficiency in the current system really exists. The project developer may reexamine some of the feasibility aspects of the project. At this point, the purpose is to make a “go” or “no-go” decision. The end result is a decision to proceed further or to abandon the project. In the preliminary investigation an initial picture about the system working is got from this study and data collection methods are identified. To launch a system investigation we need a master plan detailing the steps to be taken, the people to be questioned and the outcome expected.

II.2 EXISTING SYSTEM

Many employment websites are designed to allow employers to post job requirements for a position to be filled and are commonly known as job boards. In existing system the candidates suffer to find a job that matches their profile. And also there is a lot for the long application procedures as in the existing system.

II.3 PROPOSED SYSTEM

The proposed system is a web-based application where the users can perform all the arrangements easily and efficiently. This system provides a searching facility to the candidates based on company name and location. In this project the candidates can see only the job that matches them. And another specialty of this project is that the resume will be automatically send while applying for a job. This is the quick apply mode in this project. To overcome the drawbacks of the existing system, the proposed system has been evolved. The efficient reports can be generated by using this proposed system. This system allows the

companies to post their jobs in an easy manner. They can easily see the shortlisted candidates and can send the notification/call letter to all candidates at the same time or individually.

II.3.1 Advantages of the Proposed System

- The proposed system provides accurate data.
- An easily accessible, dynamic design.
- The proposed system is very much faster than existing issue
- Searching is taking small amount of computerized time.
- Easy editing, adding and deleting processes
- Attractive prominence & user friendly.
- Eliminate chances for errors and reduce effort

II.3.2 Features of the Proposed System

- Easy searching facility
- Feature to add selected candidates into selection list
- Easy to send call letter
- Easy procedures in job application
- Easy report availability
- Good validation checking

II.4 FEASIBILITY STUDY

During system analysis, a feasibility study of the proposed system was carried out to see whether it was beneficial to the organization. The main aim of the feasibility study is to determine whether it would be financially and technically feasible to develop the product. While evaluating the existing system, many advantages and disadvantages raised. Analyzing the problem thoroughly forms the vital part of the system study. Problematic areas are identified and information is collected.

The benefits of this site are users can easily interact and get the services without much complexity. It helps to make it possible that more users can interact with the site at a time. Feasibility study is to determine whether the proposed system is technically, economically and behaviourally feasible in all aspects.

The main aim of feasibility study is to evaluate alternative site and propose the most feasible and desirable site for development. If there is no loss for the organization then the proposed system is considered financially feasible. A feasibility study is carried out to select the best system that meets performance requirements.

The feasibility study activity involves the analysis of the problem and collection of all relevant information relating to the product such as the different data items which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system as well as various constraints on the behaviour of the system.

In this scenario, problems are identified. Essential data are being gathered for the existing problems. It is necessary that this analysis familiarizes the designer with objectives, activities, and the function of the organization in which the system is to be implemented. The feasibility study was divided into four:- Technical, Economical, Operational and Behavioural. It is summarized below:-

II.4.1 TECHNICAL FEASIBILITY

According to feasibility analysis procedure the technical feasibility of the system is analyzed and the technical requirements such as software facilities, procedure, inputs, are identified. While considering the problems of existing system, it is sufficient to implement the new system. The proposed system can be implemented to solve issues in the existing system. It includes the evaluation of and how it meets the proposed system. This system use Python Django framework and PostgreSQL as backend technology.

II.4.2 ECONOMIC FEASIBILITY

Economic analysis is most frequent used for evaluating of the effectiveness of the candidate system. More commonly known as cost/benefit analysis the procedure is to determine the benefit and saving that are expected from a candidate system and compare them with the existing system. Except for the initial capital amount and the amount after each financial year,

no other huge amount is needed. The expenses can be handles by any participants. So, the system is economically feasible.

This feasibility involves some questions such as whether the firm can afford to build the system, whether its benefits should substantially exceed its costs, and whether the project has higher priority and profits than other projects that might use the same re- sources. Here there is no problem. This firm has fully equipped hard ware, and fully fledged software, so no need to spend money on these issues. And as the client and the developer are one, there is no further problem in economic issues.

II.4.3 OPERATIONAL FEASIBILITY

Methods of processing and presentation are all according to the needs of clients since they can meet all user requirements here. The proposed system will not cause any problem under any circumstances and will work according to the specifications mentioned. Hence the proposed system is operationally feasible.

People are inherently resistant to change and computer has been known to facilitate changes. The system operation is the longest phase in the development life cycle of a system. So, Operational Feasibility should be given much importance. This system has a user-friendly interface. Thus it is easy to handle.

II.4.4 BEHAVIORAL FEASIBILITY

In today's world, computer is an inevitable entity. As per the definition of behaviour design, many valid points are recognized in this study. This system behaviour changes according to different environment. In order to ensure proper authentication and authorization and security of sensitive data of the admin or companies or candidates, login facilities are provided. These are the main feasibility studies tested in this application.

CHAPTER III

SYSTEM ANALYSIS AND DESIGN

III.1 SOFTWARE REQUIREMENT SPECIFICATION

The primary goal of the system analyst is to improve the efficiency of the existing system. For that study of specification of the requirement is very essential. For the development of the new system, a preliminary survey of the existing system will be conducted. An investigation is done whether the up gradation of the system into an application program could solve the problems and eradicate the inefficiency of the existing system. This gives an idea about the system specifications required to develop and install the project "EASYJOBS - ONLINE JOB PORTAL".

The System Requirements Specification is based on the System Definition. The requirement specifications are primarily concerned with functional and performance aspect of a software product and emphasis are placed on specifying product characteristics implying how the product will provide those characteristics. One of the most difficult tasks is selecting software, once the system requirement is find out then we have to determine whether a particular software package fits for those system requirements. This selection summarizes the application requirement.

III.1.1 HARDWARE REQUIREMENT

- CPU – INTEL CORE i3
- HARD DISKSPACE - 500 GB
- RAM - 2GB
- DISPLAY - 19 STANDARD RATIO LCDMONITOR
- KEYBOARD - 99-104 KEYS
- CLOCK SPEED - 1.99 GHZ

III.1.2 SOFTWARE REQUIREMENT

- OPERATING SYSTEM – WINDOWS
- IDE- VISUAL STUDIO CODE
- FRONT END – PYTHON DJANGO FRAMEWORK
- BACK END –POSTGRESQL

III.2 SYSTEM DESIGN

Designing the system in an effective way leads to the smooth working of any software's. System design is the process of developing specification for a candidate system that meet the criteria established in the system analysis. Major step in the system design is the preparation of the input forms and output reports in a form applicable to the user. The main objective of the system design is to use the package easily by any computer operator. System design is the creative act of invention, developing new inputs, and database, off-line files, method, procedure and output for processing business to meet an organization objective. System design builds information gathered during the system analysis. This system is designed neatly so that user will never get ambiguity while using the system.

III.2.1 NON-FUNCTIONAL REQUIREMENTS

Performance Requirements

For the efficient performance of the application, network must have high bandwidth so that the task of centralized management does not lead to network jam. Also the hard disk capability must be high so that data can be effectively stored and retrieved.

Security Requirements

Security requirements of this application involves authentication using username and password so that invalid users are restricted from data access.

III.3 UNIFIED MODELING LANGUAGE [UML]

UML is a way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar JAcobson and the RationalSoftware Corporation to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. Today, UML is accepted by the Object Management Group (OMG) as the standard for modelling software development.

UML stands for Unified Modeling Language. UML 2.0 helps extend the original UML specification to cover a wider portion of software development efforts including agile practices. Improved integration between structural models like class diagrams and behavior models like activity diagrams.

The original UML specified nine diagrams; UML 2.x brings that number up to 13. The four new diagrams are called: communication diagram, composite diagram, interaction overview diagram and timing diagram. It also renamed state chart diagrams to state machine diagrams, also known as state diagrams.

Types of UML diagrams

The current UML standards call for 13 different types of diagrams: class, activity, object, use case, sequence, package, state, component, communication, composite structure, interaction overview, timing and deployment. These diagrams are organized into two distinct groups: structural diagrams and behavioral or interaction diagrams.

Structural UML diagrams

- Class diagram
- Package diagram
- Object diagram
- Component diagram
- Composite structure diagram
- Deployment diagram

Behavioral UML diagrams

- Activity Diagram

- Sequence diagram
- Use case diagram
- State diagram
- Communication diagram
- Interaction overview diagram
- Timing diagram

III.3.1 Use case Diagram

To model a system the most important aspect is capture the dynamic behaviour. To modify a bit in details, dynamic behaviour of the system when it is running or operating. So only behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction. These internal and external agents are known as actors. So use case diagram consists of actors, use case and their relationships. The diagram is used to model the system of an application. A single use case diagram captures a particular functionality of a system.

Use case Diagram objects:

- Actor
- Use case
- System
- Package Actor

Actor

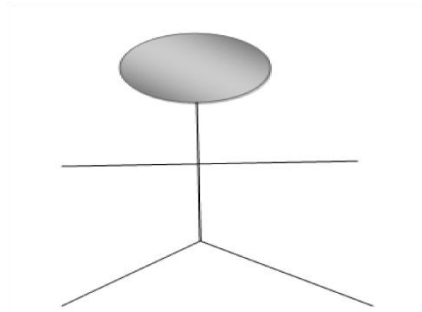


Figure III.1: Actor

Actor is a use case diagram in an entity that performs a role in one given system. This could be a person, organization or an external system usually drawn like skeleton.

Use case

A use case represents a function or an action within the system. It's drawn as an oval and named with the function.



Figure III.2: Use Case

System

System is used to define the scope of the use case and drawn as a rectangle. This is an optional element but useful when your visualizing large systems. For example you can create all the use cases and then use the system object to define the scope covered by your project. Or you can even use it to show the different areas covered in different releases.

Package

Package is another optional element that is extremely useful in complex diagrams. Similar to use class diagrams, packages are used to group together use cases.

The following is the UML diagram of this system:

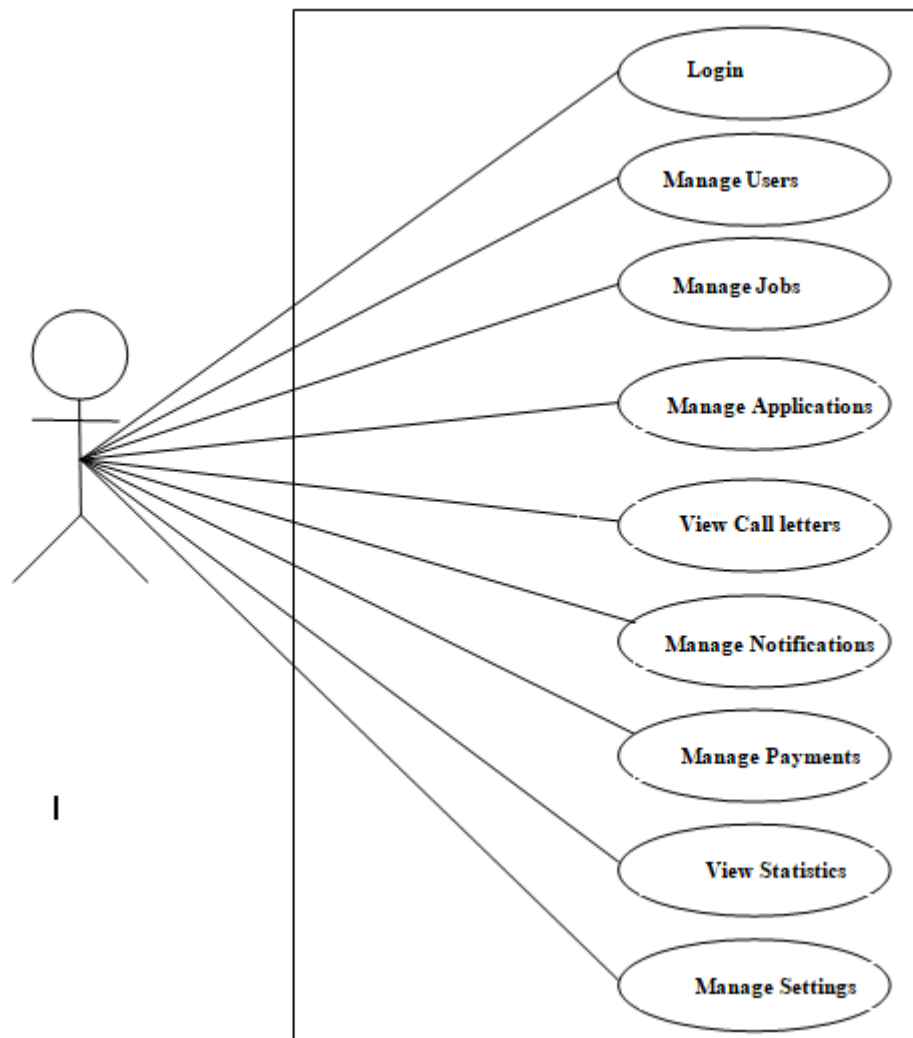


Figure III.3: UML DIAGRAM FOR ADMIN

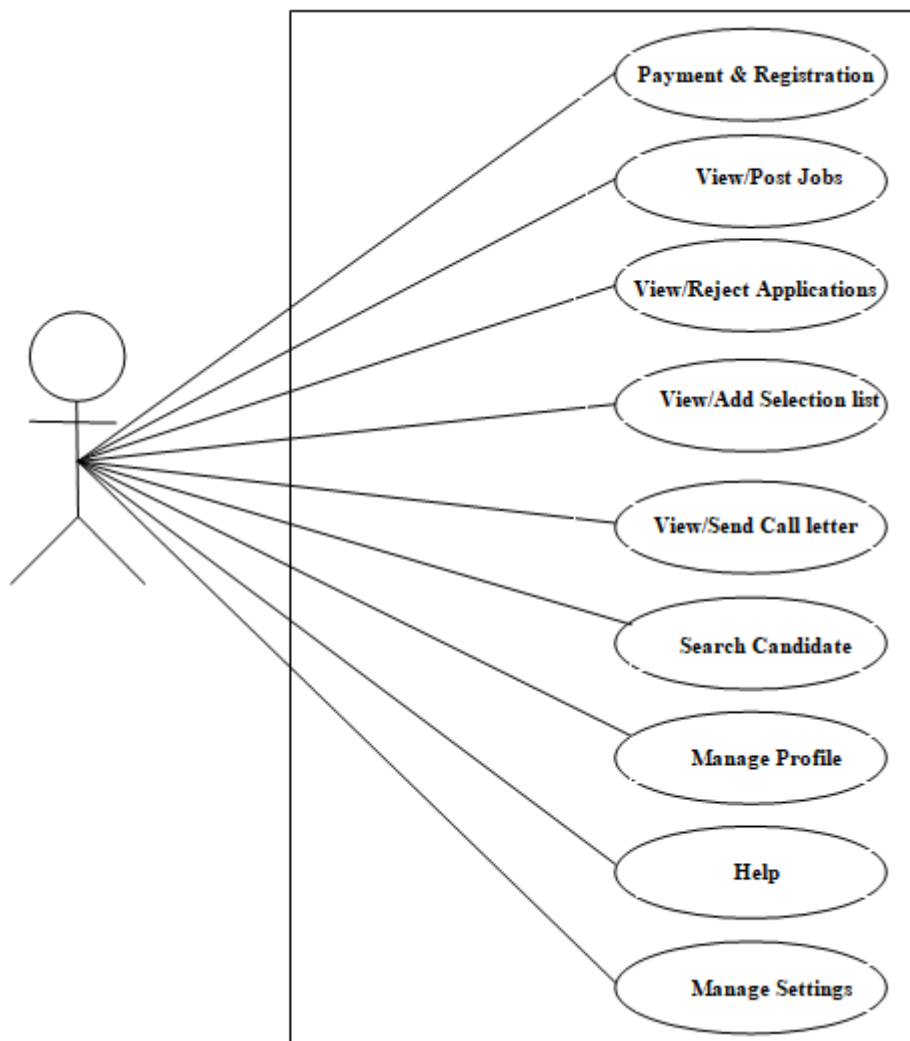


Figure III.4: UML DIAGRAM FOR COMPANY

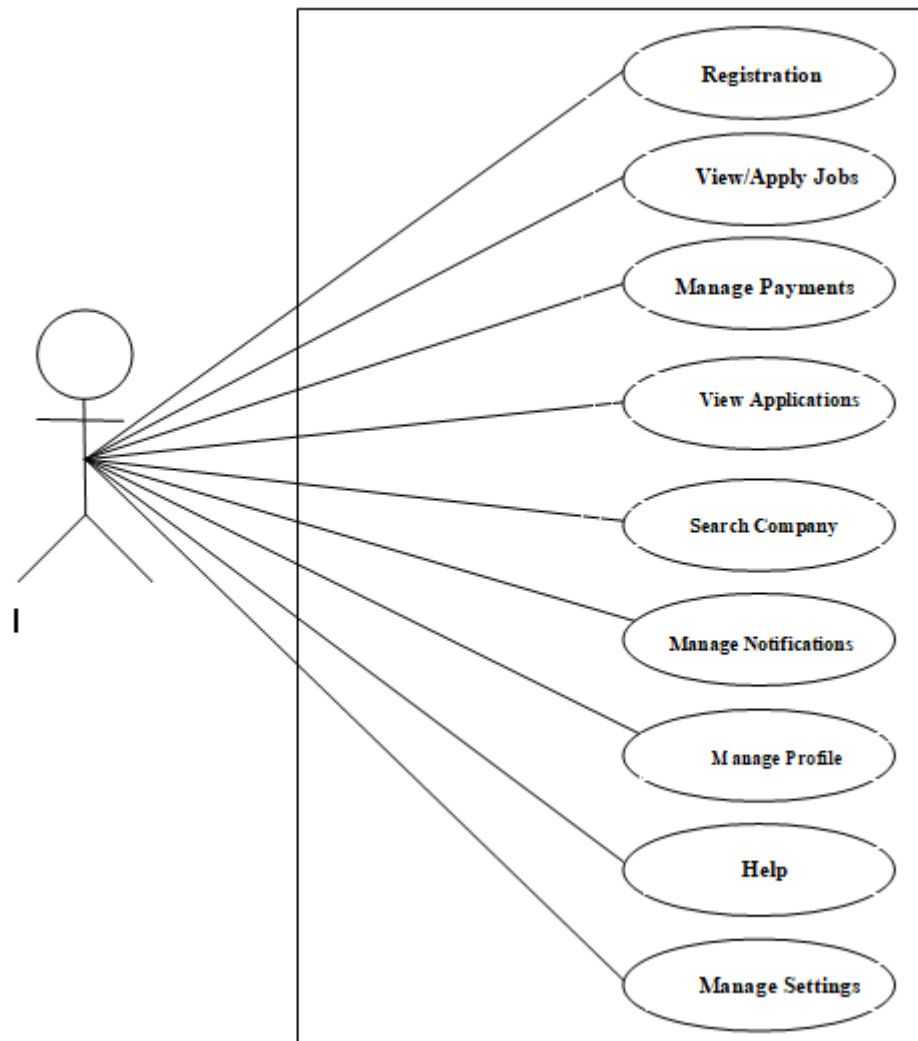


Figure III.5: UML DIAGRAM FOR CANDIDATE

Although UML sequence diagrams are typically used to describe object-oriented software systems, they are also extremely useful as system engineering tools to design system architectures in business process, as message sequence charts and call flows for telecoms or wireless system design, and for protocol stack design and analysis. A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence Diagrams are typically associated with use case realizations in the logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as

horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

III.3.2 Sequence Diagram

UML sequence diagrams are used to represent or model the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interaction of the header elements.

Sequence Diagrams are used primarily to design, document and validate the architecture, interfaces and logic of the system by describing the sequence of actions that need to be performed to complete a task. UML sequence diagrams are useful design tools because they provide a dynamic view of the system behavior which can be difficult to extract from static diagrams or specifications.

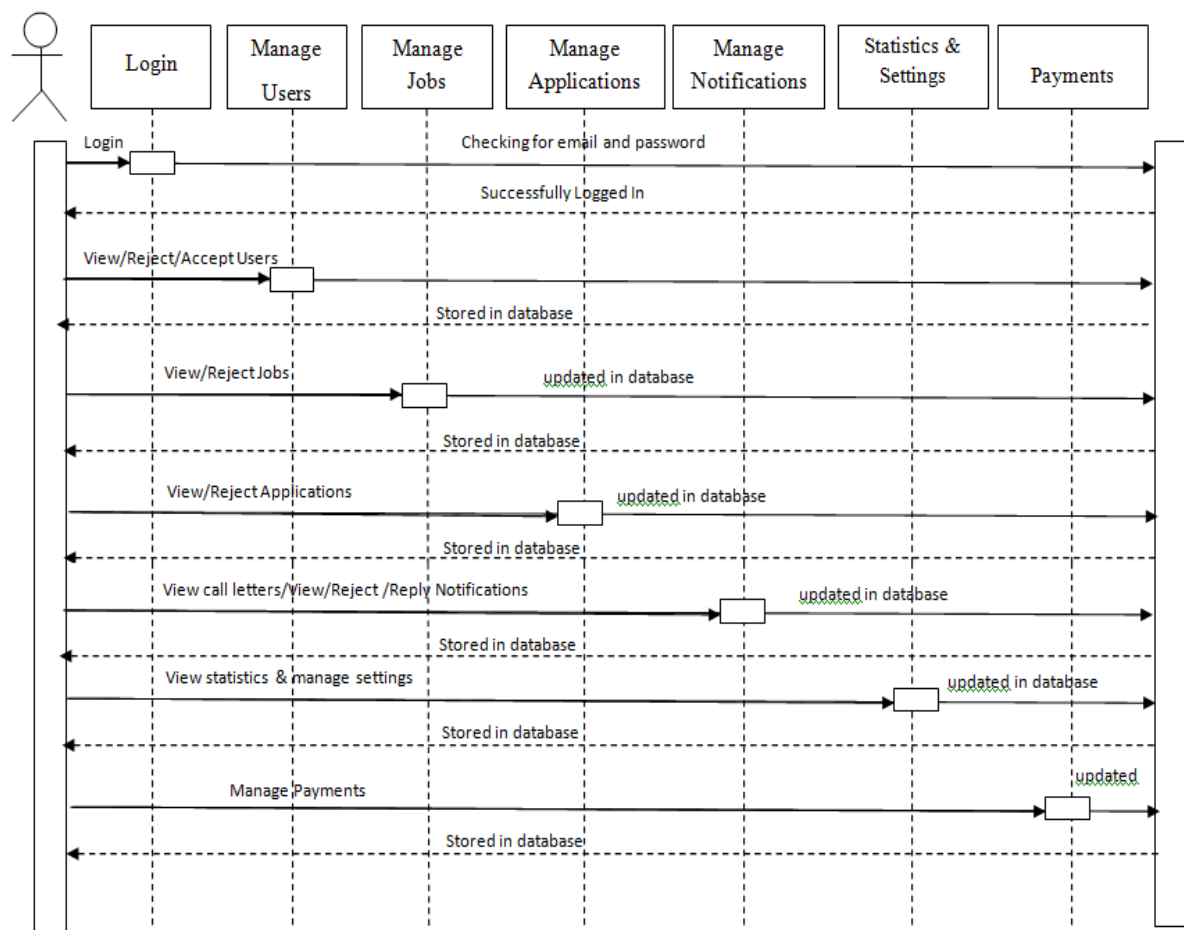


Figure III.6: Sequence Diagram For Admin

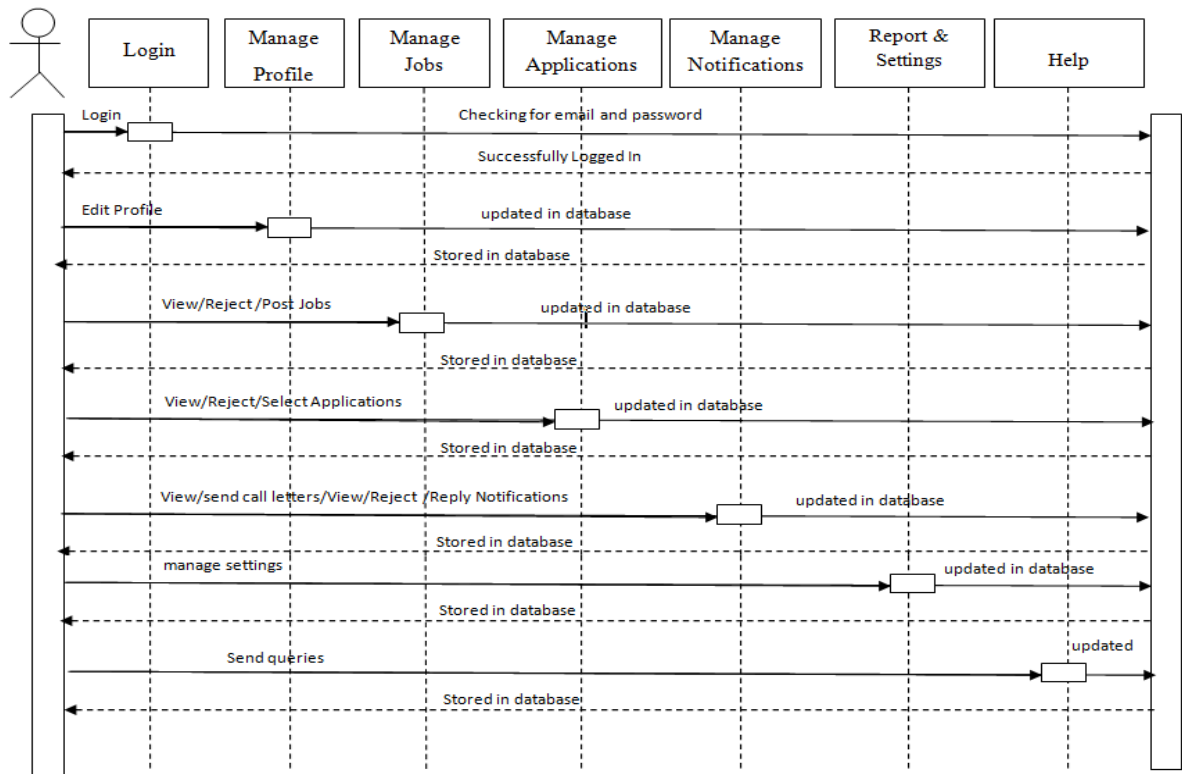


Figure III.7: Sequence Diagram For Company

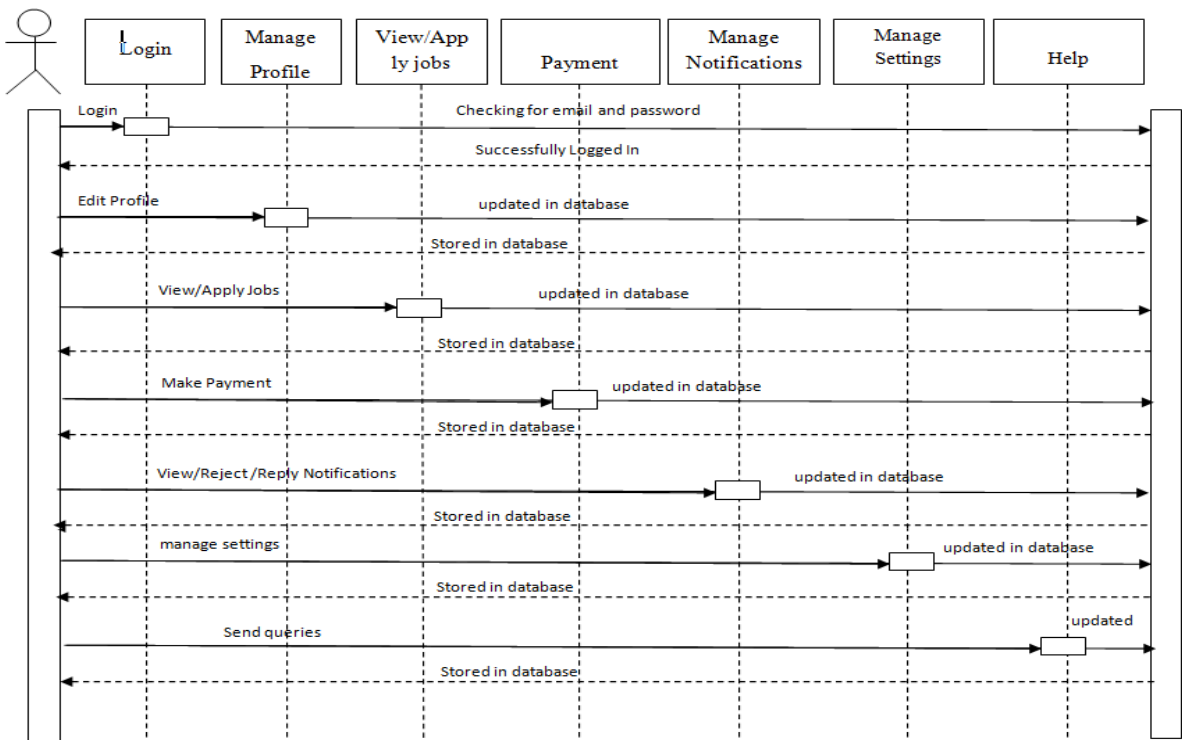


Figure III.8: Sequence Diagram For Candidate

III.3.3 Activity Diagram

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another. Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

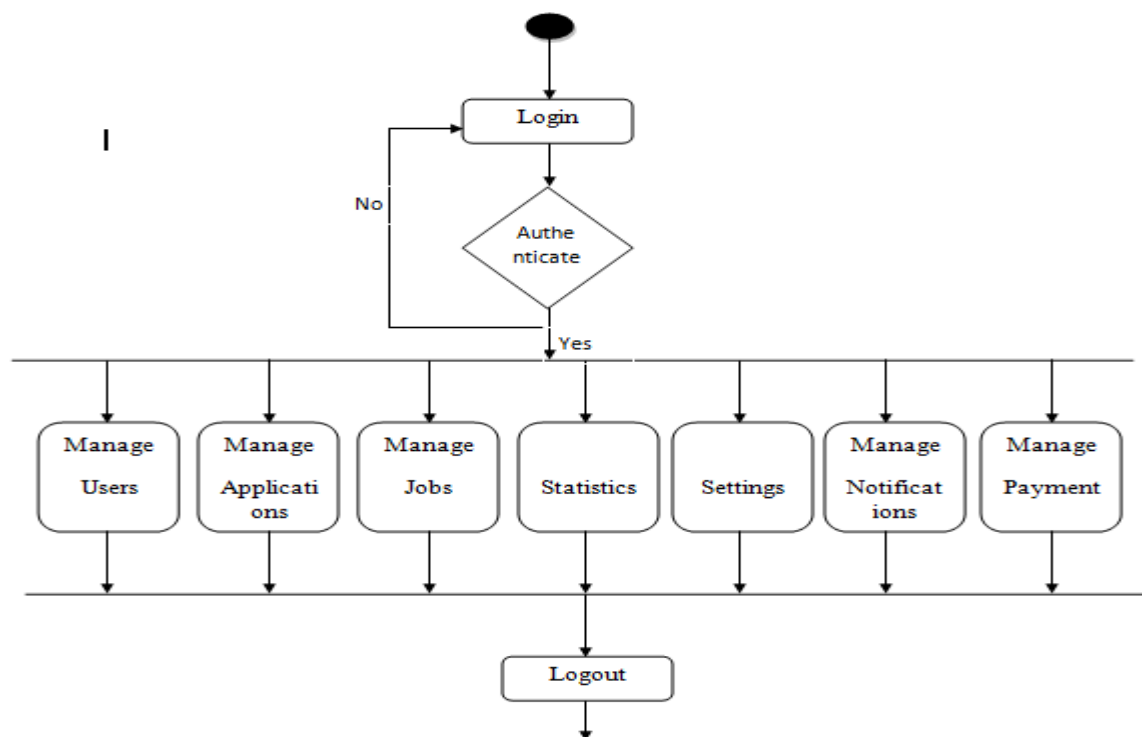


Figure III.9 Activity diagram- Admin

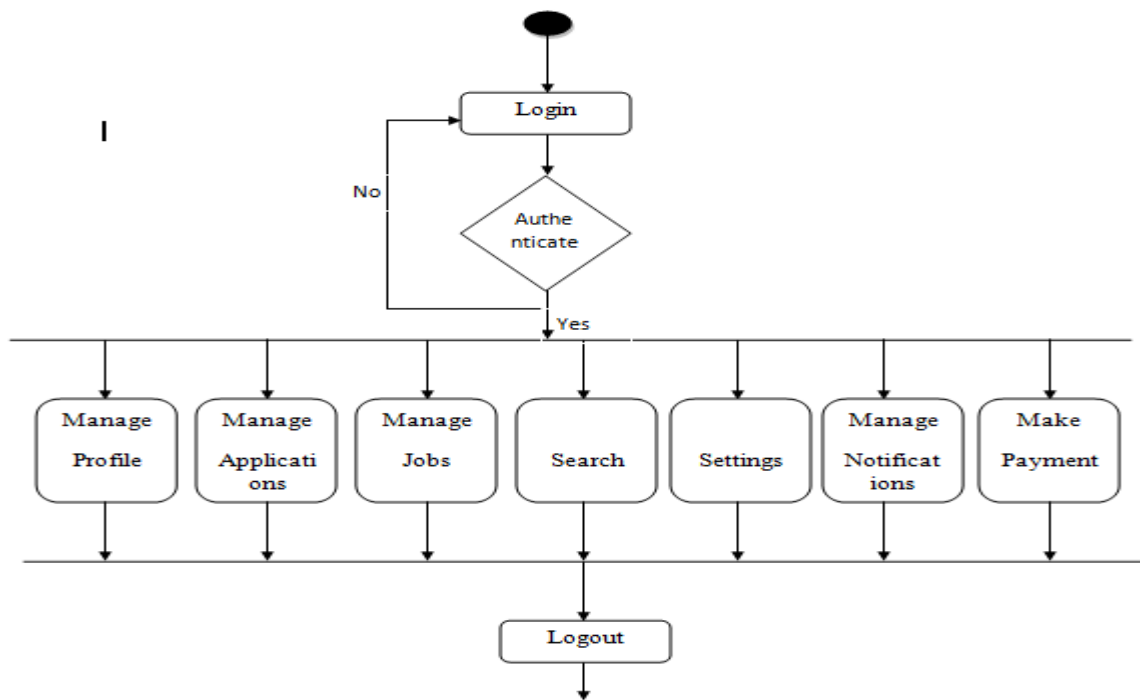


Figure III.10 Activity diagram- Company

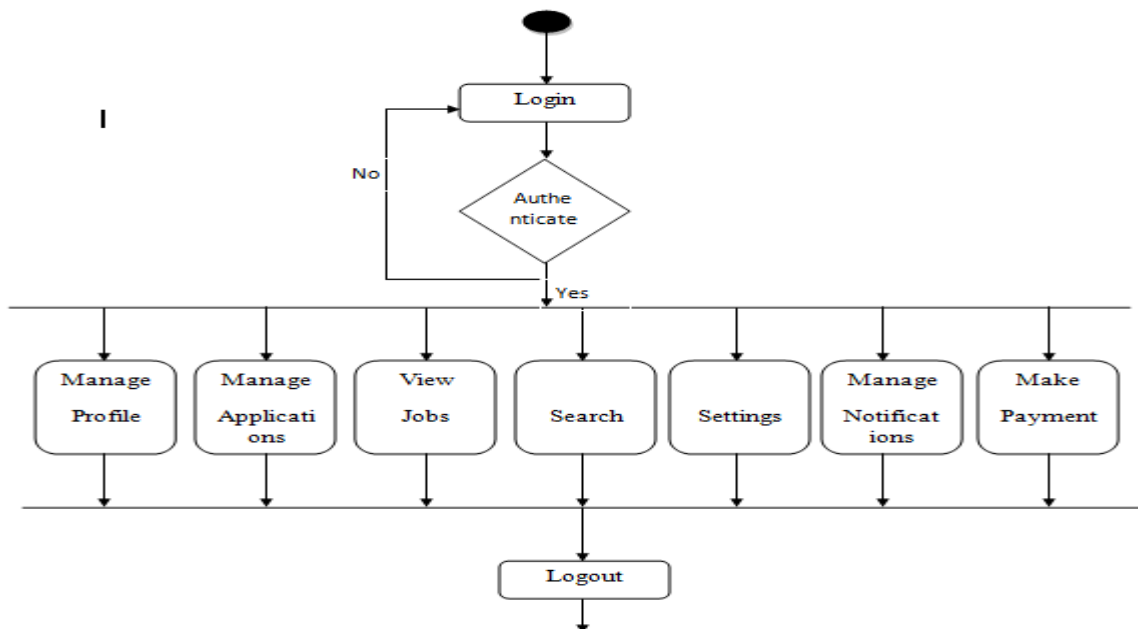


Figure III.11 Activity diagram- Candidate

III.4 SYSTEM DESIGN

The most creative and challenging phase of the system life cycle is the system design. The term design describes a final system and the process by which it is developed. It refers to the technical specification that will be applied in implementing the candidate system. In system design, we move from the logical to the physical aspects of the life cycle.

The first step is to determine how the output is to be produced and in what format. Then input data and master files have to be designed as the next step and finally the impact of the candidate system on the user and organization are documented and evaluated by the management. After identifying the problem and the limitation of the existing system, a detailed design of the proposed system is conducted.

Free flow personnel interview and reference to previous records prepared manually were the only methods taken to collect necessary information. At present, all organizations are on the path of computerization process. Design is the phase that indicates the final system. It is the solution, the translation of requirements into ways of meeting them. In this phase the following elements were designed namely, data flow, data stores, processes, procedures was formulated in a manner that meet the project requirements. After logical design physical construction of the system is done.

The database tables, input screens, output screens, output reports are designed. After analyzing the various functions involved in the system the database, labels as dictionaries designed. Care is taken for the field name to be in self-explanatory form. Unnecessary fields are avoiding so as not affecting the storage system.

Care must be taken to design the input screen in the most user-friendly way so as to help even the novice users to make entries approximately in the right place. This is being accomplished by the use of giving online help messages, which are brief and cleanly prompts users for appropriate action. Design is the only way that we can accurately translate a customer's requirements into a finished software product or system. Without design, risk of building an unstable system exist one that will fail when small changes are made, one that will be difficult to test. All input screens in the system are user friendly and are designed in such a way that even a layman can operate. The sizes of all screens are standardized.

The importance of the software design can be stated with a single word quality. Design is a place where quality is fostered in software development. Design is the only way where

requirements are actually translated into a finished software product or system.

Mainly this project consists of 3 Modules:

- **Admin**
- **Company**
- **Candidate**

Admin Module

Administrator is the main actor in this system. Administrator can manage the whole website. Admin can view the users of this website. He can view or edit or delete the job applications, selection list, users profile, posted jobs, notifications, statistics etc.

- **Admin Login**

By the Username and password admin can login to the system.

- **View/Reject/Activate Companies**

Admin is responsible for viewing, rejecting and activating the companies registered on the website

- **View/Reject/Activate Candidates**

Admin is responsible for viewing, rejecting and activating the candidates registered on the website

- **View/Reject/Activate Jobs**

Admin is responsible for viewing, rejecting and activating the Jobs which are posted on the website

- **View Applications**

Admin can view the applications for various companies and for various jobs which are applied through the website.

- **View Call Letters**

Admin can view the call letters which are send by the companies to the candidates

- **View Users**

Admin can view all the users which are registered in the website

- **Manage Settings**

Admin can change password

- **View Statistics**

Admin can view the statistics of the website. This can be called as a report.

- **Manage Payments**

Admin is responsible in managing the payment procedures during the registration and job application.

- **View/Reply/Remove Notification**

Admin is responsible for helping the users. Admin provides the reply for the queries that are submitted by the users.

Company Module

This project will provide the registration facility to the job providers also. They can create their profile, post their jobs, view applications and resume, view shortlisted candidates, reject applications, send call letters, view application report, view profile, edit profile etc. Companies can register on this site by making payment.

- **Company Login & Registration**

Company can register into the system by making payment and login into the system by using the username and the password.

- **Company View/Post/Reject Jobs**

Company can Post their jobs. They can view the jobs that are posted and also they can remove the posted jobs

- **Company View/Reject Applications**

Company can view and reject the applications

- **Company View/Reject Shortlist**

Company can view and reject the shortlist applications

- **Company View/add selection list**

Company can view and add the selected candidates into the selection list.

- **Company View Application Report**

Company can view full application report or monthly/weekly/yearly report.

- **Company View/Send call letter**

Company can view send call letter to the shortlisted candidates

- **Company Settings**

Company can delete their account and can change the password

- **Company Help**

Company can send queries to the admin

- **Company Search-candidate**

Company can search candidate by using the application id.

- **Company View/Remove Notification**

Company can view or remove the reply messages from the admin as a result of the queries submitted by the company.

Candidate Module

This will allow the person to apply for a job in the company by making the payment. The person will be having the account after registration. This project is created for fulfilling the requests of the company managers. The users who visit the website can view the jobs in the company and will be able to apply directly from remote places. They can also search by location or by company name. This website will allow the candidates to upload their resume. Every time when the candidates apply for the job, their resume will be automatically sent to the company. The candidates have the facility to view notifications from companies, view and edit their profile etc.

- **Candidate Login & Registration**

Candidate can register into the system and login into the system by using the username and the password.

- **Candidate View/Apply for job**

Candidate can view jobs posted by the companies and can apply for the jobs by making payment

- **Candidate View Applications**

Candidate can view their submitted applications

- **Candidate Search**

Candidate can search for a company by typing the company name or location

- **Candidate View/Remove Notification**

Candidate can view or remove the notifications comes from company or from admin

- **Candidate Settings**

Candidate can delete their account and can change the password

- **Candidate Manage Profile**

Candidate can view or edit their profile. They can also upload their resume and profile picture

- **Candidate Help**

Candidate can submit their queries to the admin using this functionality.

III.4.1 DATA FLOW DIAGRAM (DFD)

Context Level DFD

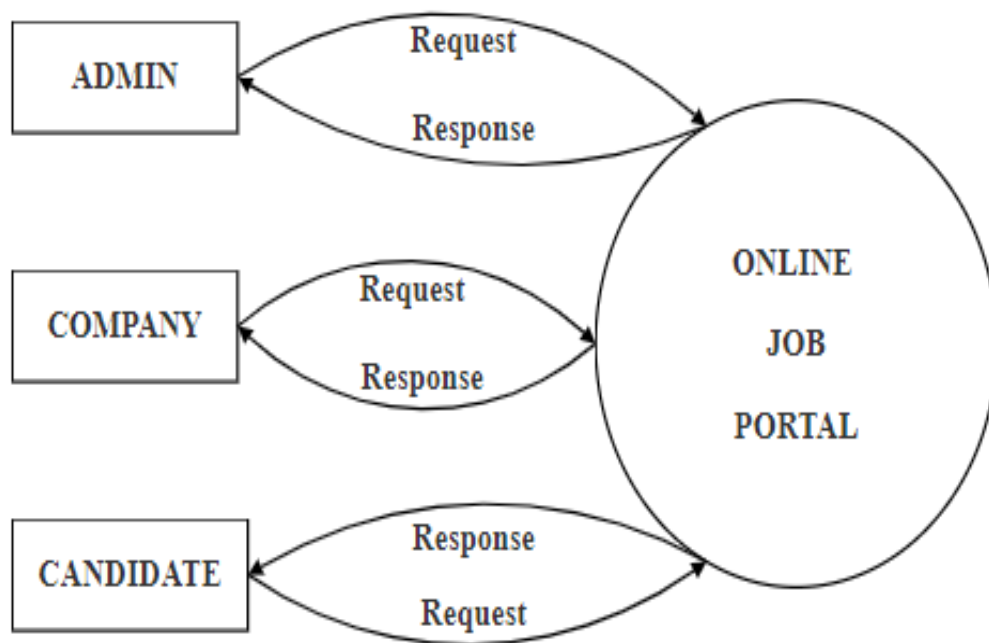


Figure III.12: Level 0

Level 1 DFD – Admin

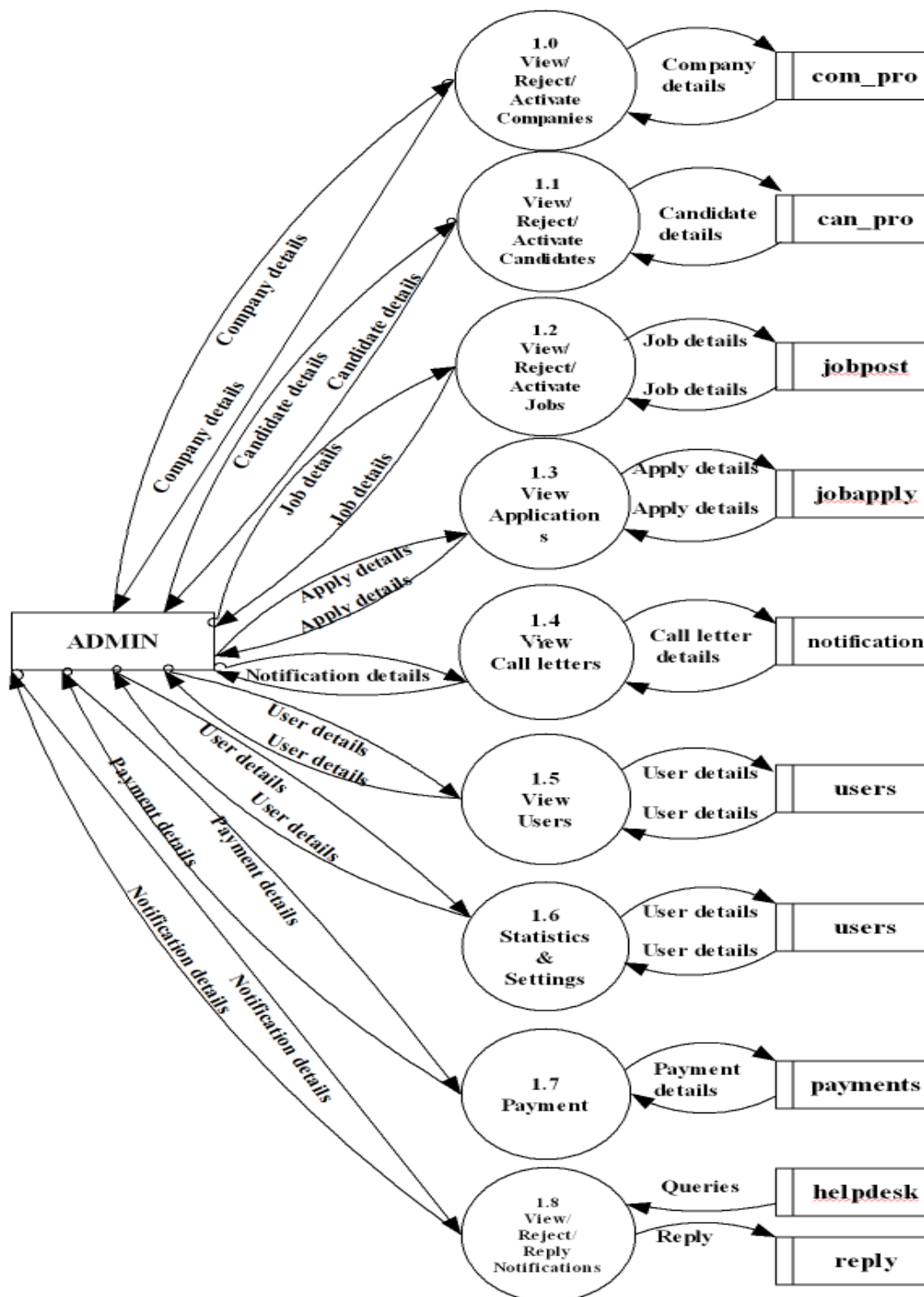


Figure III.13: Level 1 ADMIN

Level 1 DFD – Company

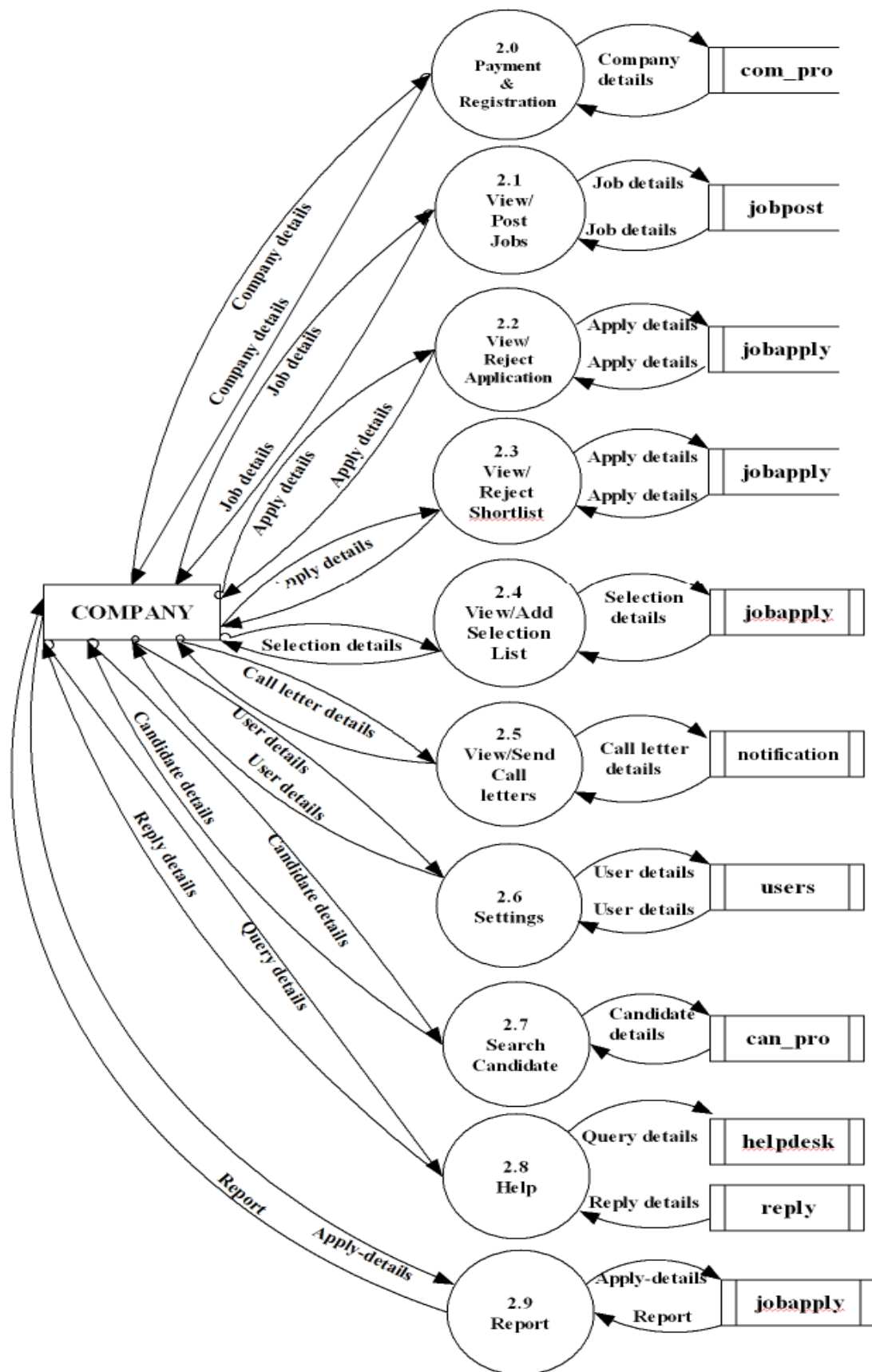


Figure III.14: Level 1 COMPANY

Level 1 DFD – Candidate

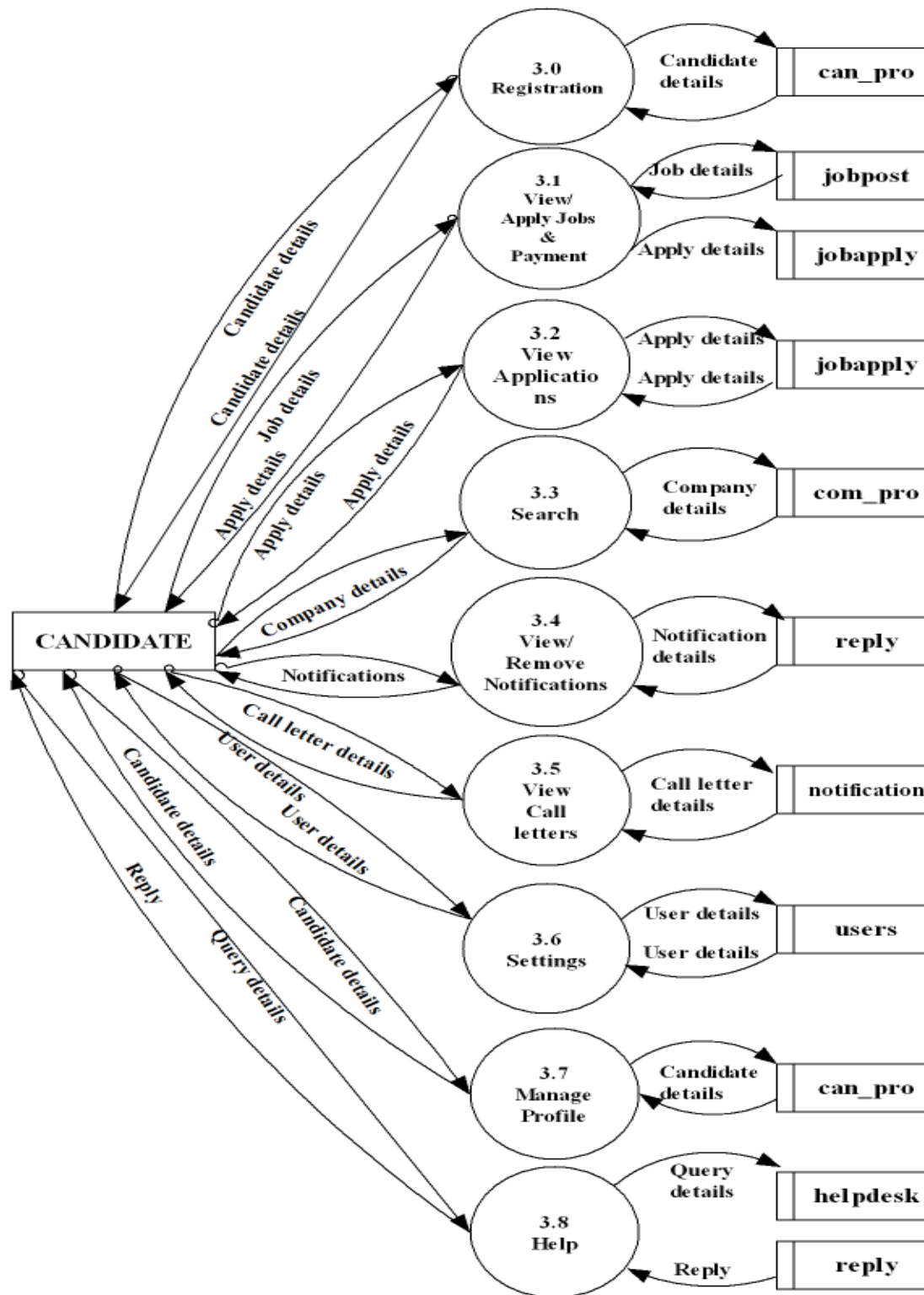


Figure III.15: Level 1 CANDIDATE

III.4.2 INPUT DESIGN

Input design is the process of converting user-oriented input into a computer based format. The goal of the designing input is to make data entry as easy and free from error. input to the system is entered through forms. A form is "any surface on which information is to be entered, the nature of which is determined by what is already on that surface." If the data going into the system is incorrect, then processing and output will magnify these errors. So designer should ensure that form is accessible and understandable by the user.

End-users are people who communicate to the system frequently through the user- interface , the design of the input screen should be according to their recommendations. The data is validated wherever it requires in the project. This ensures only correct data is entered to the system. GUI is the interface used in input design. All the input data are validated in the order and if any data violates any condition the use is warned by a message and asks to re-enter data. If the data satisfies all the conditions then it is transferred to the appropriate tables in the database. This project uses text boxes and drop down etc to accept user input. If user enters wrong format then it shows a message to the user.

III.4.3 OUTPUT DESIGN

A quality output is one, which meets the requirement of the end user and presents the information clearly. In any system results of processing are communicated to the user and to the other systems through outputs. In the output design it is determined how the information is to be displayed for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship and helps user decision making.

It generally refers to the results and the information that are generated by the system. Effective, descriptive and useful design will improve the relationship with the user and the system because it is the direct source of information to the user. The objective of the output design is to convey the information of all the past activities, required status and to emphasize important events.

III.4.4 TABLES

1.User

Field	Data type	Description
id	Integer	id
username	String	username of users
First_name	String	first name of users
Last_name	String	last name of users
Password	String	password of users
Email	String	email id of users
Last_login	Timestamp	Last login details
Is_superuser	Boolean	Super user status
Is_staff	Boolean	Staff status
Is_active	Boolean	Active status
Date_joined	Timestamp	Joined date

Primary Key : id

Unique Key : username

Table III.1: Table User

2.com_pro

Field	Data type	Description
Id	Integer	id
com_username	String	username of company
com_name	String	name of company
com_desc	String	description about company
com_place	String	place of company
com_pincode	Integer	pin code of company
com_dt	String	district of company
com_state	String	state of company
com_country	String	country of company
com_mob	String	contact number
com_email	String	email address
status	integer	status

logo	file	Company Logo
------	------	--------------

Primary Key : id

Foreign Key : com_username(USER)

Table III.2: Table com_pro

3.can_pro

Field	Data type	Description
id	Integer	id
can_uname	String	username of candidate
can_name	String	name of candidate
can_house	String	house name
can_place	String	place
can_pincode	Integer	pincode
can_gender	String	gender
can_email	String	email address
can_mob	Integer	contact number
can_dt	String	district
can_state	String	state
school	String	name of school
sc_board	String	board of school
sc_percent	Float	percentage score
sc_yop	Integer	year of passing
hss	String	name of hss
hss_board	String	board of hss
hss_stream	String	stream of study
hss_percent	Float	percentage score
hss_yop	Integer	year of passing
ug	String	ug college name
ug_uni	String	university
ug_course	String	course name
ug_percent	Float	percentage score
ug_yop	Integer	year of passing

pg	String	pg college
pg_uni	String	university
pg_course	String	course name
pg_yop	Integer	year of passing
pg_percent	Float	percentage score
skills	String	skills
status	Integer	status

Primary Key : id

Foreign Key : can_uname(USER)

Table III.3: Table can_pro

4.notification

Field	Data type	Description
id	Integer	id
job_id	String	id of job
com_username	String	username of company
can_uname	String	name of candidate
in_desc	String	notification
call	String	call status
call_date	Date	date
status	Integer	notification status

Primary Key : id

Foreign Key : com_username(USER) , can_uname(USER), job_id (JOBPOST)

Table III.4: Table notification

5.jobpost

Field	Data type	Description
id	Integer	id
com_username	String	username of company
job_name	String	job title
job_desc	String	job description
job_qua	String	qualification
job_place	String	job location

job_pin	Integer	pincode
job_dt	String	district
job_st	String	state
job_email	String	email address
job_phn	Integer	contact number
job_con	String	country
shrt_p_ten	Float	tenth percentage
shrt_p_tlw	Float	plus two percentage
shrt_p_ug	Float	UG percentage
shrt_p_pg	Float	PG percentage
shrt_skill	String	skills
post_date	Date	date
last_date	Date	date
Status	Integer	Status of job

Primary Key : id

Foreign Key : com_username(USER)

Table III.5: Table jobpost

6.jobapply

Field	Data type	Description
id	Integer	id
job_id	Integer	job id
com_une	String	company's username
can_une	String	candidate's username
apply_date	Date	applied date
short	String	shortlist status
short_date	Date	shortlisted date
status	Integer	application status
Selected	String	Selection status

Primary Key : id

Foreign Key : com_username (USER), can_username (USER), job_id (JOBPOST)

Table III.6: Table jobapply

7.helpdesk

Field	Data type	Description
id	Integer	id
username	String	username of users
message	String	Message
Help_date	Date	Date
status	Integer	status

Primary Key : id

Foreign Key : username (USER)

Table III.7: Table helpdesk

8.reply

Field	Data type	Description
id	Integer	id
Help_id	Integer	id
message	String	Message
reply_date	Date	Date
status	Integer	status

Primary Key : id

Foreign Key : help_id (HELPDESK)

Table III.8: Table reply

9.payments

Field	Data type	Description
id	Integer	id
Apply_amount	Float	Application amount
Reg_amount	Float	Registration amount

Primary Key : id

Table III.9: Table payments

III.5 TOOLS AND PLATFORMS

III.5.1 Python

Python is an interpreted high level programming language for general purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non profit Python Software Foundation.

Python Environment:

A Python environment is a child/offshoot of a parent python distribution which allows you to use the packages in the parent distribution as well as to install packages that are only visible to the child distribution.

Python Language:

Python is a powerful high level, object oriented programming language created by Guido van Rossum. It has simple easy to use syntax, making it the perfect language for someone trying to learn computer programming for the first time.

Python is a general purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D).

The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

Python Runtime Environment:

The runtime environment used to execute the code. It is made up of the Python language and Python interpreter. It is portable and it is platform neutral.

Python tools:

It is used by the developers to create Python code. They include Python compiler, Python interpreter, classes, libraries etc.

Python Application:

Applications are programs written in Python to carry out certain tasks on standalone local computer. Python source code is automatically compiled into Python byte code by the CPython interpreter. Compiled code is usually stored in PYC (or PYO) files, and is regenerated when the source is updated, or when otherwise necessary.

To distribute a program to people who already have Python installed, you can ship either the PY files or the PYC files. In recent versions, you can also create a ZIP archive containing PY or PYC files, and use a small “bootstrap script” to add that ZIP archive to the path.

III.5.2 Django Framework

Django is a high level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, 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 is free and open source, has a thriving and active community, great documentation, and many options for free and paid for support.

Django helps you write software that is:

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up to date documentation.

Versatile

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). The site you are currently reading is based on Django!

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

A password hash is a fixed length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password.

Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, crosssite request forgery and clickjacking (see [Website security](#) for more details of such attacks).

Scalable

Django uses a component based “shared-nothing” architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear

separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Portable

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

III.5.3 Python – xhtml2pdf

xhtml2pdf is a HTML/XHTML/CSS to PDF converter written in Python and based on Reportlab Toolkit, pyPDF, TechGame Networks CSS Library and HTML5lib. The primary focus is not on generating perfect printable webpages but to use HTML and CSS as commonly known tools to generate PDF files within Applications. For example generating documentations (like this one), generating invoices or other office documents etc.xhtml2pdf enables users to generate PDF documents from HTML content easily and with automated flow control such as pagination and keeping text together. The Python module can be used in any Python environment, including Django.

xhtml2pdf is a html2pdf converter using the ReportLab Toolkit, the HTML5lib and pyPdf. It supports HTML 5 and CSS 2.1 (and some of CSS 3). It is completely written in pure Python so it is platform independent. The main benefit of this tool is that a user with Web skills like HTML and CSS is able to generate PDF templates very quickly without learning new technologies. xhtml2pdf was previously developed as "pisa".

III.5.4 Visual Studio Code IDE

Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes embedded Git and support for debugging, syntax highlighting, intelligent code completion, snippets, and code refactoring. For serious coding, you'll often benefit from tools with more code understanding than just blocks of text. Visual Studio Code includes built-in support for IntelliSense code completion, rich semantic code understanding and navigation, and code refactoring. And when the coding gets tough, the tough get debugging. Debugging is often the one feature that developers miss most in a leaner coding experience, so we made it happen. Visual Studio Code includes an interactive debugger, so you can step through source code, inspect variables, view call stacks, and execute commands in the console.

VS Code also integrates with build and scripting tools to perform common tasks making everyday workflows faster. VS Code has support for Git so you can work with source control without leaving the editor including viewing pending changes.

CHAPTER IV SYSTEM TESTING

IV.1 TESTING METHODOLOGIES AND STRATEGIES

Software testing is an integral part of to ensure software quality, some software organizations are reluctant to include testing in their software cycle, because they are afraid of the high cost associated with the software testing .There are several factors that attribute the cost of software testing. Creating and maintaining large number of test cases is a time consuming process. Furthermore, it requires skilled and experienced testers to develop great quality test cases. Testing begins at the module level and work towards the integration of entire computer based system. No testing is completed without verification and validation part.

The goal of verification and validation activities are to access and improve the quality of work products generated during the development and modification of the software. Testing plays a vital role in determining the reliability and efficiency of the software and hence is very important stage in software development. Tests are to be conducted on the software to evaluate its performance under a number of conditions. Ideally, it should do so at the level of each module and also when all of them are integrated to form the completed system.

IV.1.1 Unit Testing

UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

IV.1.2 Integration Testing

After splitting the programs into units, the units were tested together to see the defects between each module and function. It is testing to one or more modules or functions together with the intent of finding interface defects between the modules or functions. Testing completed at as part of unit or

functional testing, integration testing can involve putting together of groups of modules and functions with the goal of completing and verifying meets the system requirements.

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements. It occurs after unit testing and before validation testing.

IV.1.3 System Testing

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing takes, as its input, all of the integrated components that have passed integration testing.

The System is tested to verify whether it meets the functional and technical requirements. The application/System is tested in an environment that closely resembles the production environment where the application will be finally deployed.

The prerequisites for System Testing are:-

- All the components should have been successfully Unit Tested.
- All the components should have been successfully integrated.
- Testing should be completed in an environment closely resembling the production environment. When necessary iterations of System Testing are done in multiple environments.

IV1.4 User Acceptance Testing

The system was tested by a small client community to see if the program met the requirements defined the analysis stage. It was found to be satisfactory. In this phase, the system is fully tested by the client community against the requirements defined in the analysis and design stages, corrections are made as required, and the production system is built. User acceptance of the system is key factor for success of the system.

IV1.5 SAMPLE TEST CASE

Figure III.10 Test case for Login

Test Step	Expected Result	Actual Result	Status
Click on the Login button without entering user name or password	Messages like "Please enter User Name" and "Please Enter Password" should appear.	Messages "Please enter User Name" and "Please Enter Password" appear.	Pass
Enter a non-existing user name/password and click on the Login button	Message like "Invalid User Name" should appear	A message "Invalid User Name" appears	Pass
Enter a valid user name and password and click on the Login button	The page should be navigated to the home page	The page is navigated to the home page	pass

Figure III.11 Test case for Registration

Test Step	Expected Result	Actual Result	Status
Enter all fields and click Register button	The page should navigated to the Login page	The page is navigated to the login page	Pass
Enter all fields, but some fields are invalid	Message like "Invalid entry"	A message "Invalid Entry" appears	Pass
Click on Register button without filling all fields that are required	Messages like "Please fill all required fields" should appear.	A message "Please fill all required fields" appears	pass

Figure III.12 Test case for Job Post

Test Step	Expected Result	Actual Result	Status
Enter all fields and click Job Post button	Message like "Success"	Will redirect the same page	Pass
Enter all fields, but some fields are invalid	Message like "Invalid entry"	A message "Invalid Entry" appears	Pass
Click on Job Post button without filling all fields that are required	Messages like "Please fill all required fields" should appear.	A message "Please fill all required fields" appears	pass

CHAPTER V

SYSTEM IMPLEMENTATION

The primary goal of implementation is to write the source code to its specification that can be achieved by making the source code clear and straight forward as possible. Implementation means the process of converting a new or revised system design into operational one.

The implementation is one phase of software development. Implementation is that stage in the project where theoretical design is turned into working system. Implementation involves placing the complete and tested software system into actual work environment. Implementation is concerned with translating design specification with source code. The three types of implementation are:-implementation of a computerized system to replace a manual system, implementation of a new system to replace existing one and implementation of a modified system to replace an existing one.

The implementation is the final stage and it is an important phase. It involves the individual programming; system testing, user training, and the operational running of developed proposed system that constitute the application subsystem. The implementation phase of the software development is concerned with translating design specification in the source code. The user tests the developed system and the changes are according to the needs. Before implementation, several tests have been conducted to ensure no errors encountered during the operation. The implementation phase ends with an evaluation of the system after placing it into operation of time. The validity and proper functionality of all the modules of the developed application is assured during the process of implementation. Implementation is the process of assuring that the information system is operational and then allowing user to take over its operation for use and evaluation. Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operated the new system. The most crucial stage in achieving a new successful system is that it works effectively and efficiently.

CHAPTER VI

CONCLUSION

“EasyJobs - Online Job Portal” is created for fulfilling the requests of the company managers. The person will be having the account after registration. The users who visit the website can view the jobs in the company and will be able to apply directly from remote places. They can also search by location or by company name. This website will allow the candidates to upload their resume. Every time when the candidates apply for the job, their resume will be automatically sent to the company. The candidates have the facility to view notifications from companies, view and edit their profile etc.

This project will provide the registration facility to the job providers after making payment. They can create their profile, post their jobs, view applications and resume, view shortlisted candidates, reject applications, send call letters, view application report, view profile, edit profile etc.

Administrator can manage the whole website. Admin can view the users of this website. He can view or edit or delete the job applications, users profile, posted jobs, notifications etc. Thus this project will help the job seekers to get better jobs and the job providers to get better employees.

Thus this project will help the job providers to get the right candidate at the right time to the right position. Also helps the job seekers to get the right job in the right company.

REFERENCES

1. <https://www.djangoproject.com/>
2. <https://docs.djangoproject.com/en/3.0/>
3. <https://docs.djangoproject.com/en/3.0/topics/db/>
4. <https://www.w3schools.com/>
5. <https://www.pgadmin.org/docs/>
6. <https://docs.python.org/3/tutorial/>
7. <https://stackoverflow.com/>

APPENDIX A APPENDICES

A.1 SCREEN SHOTS INPUT FORM, OUTPUT FORMS

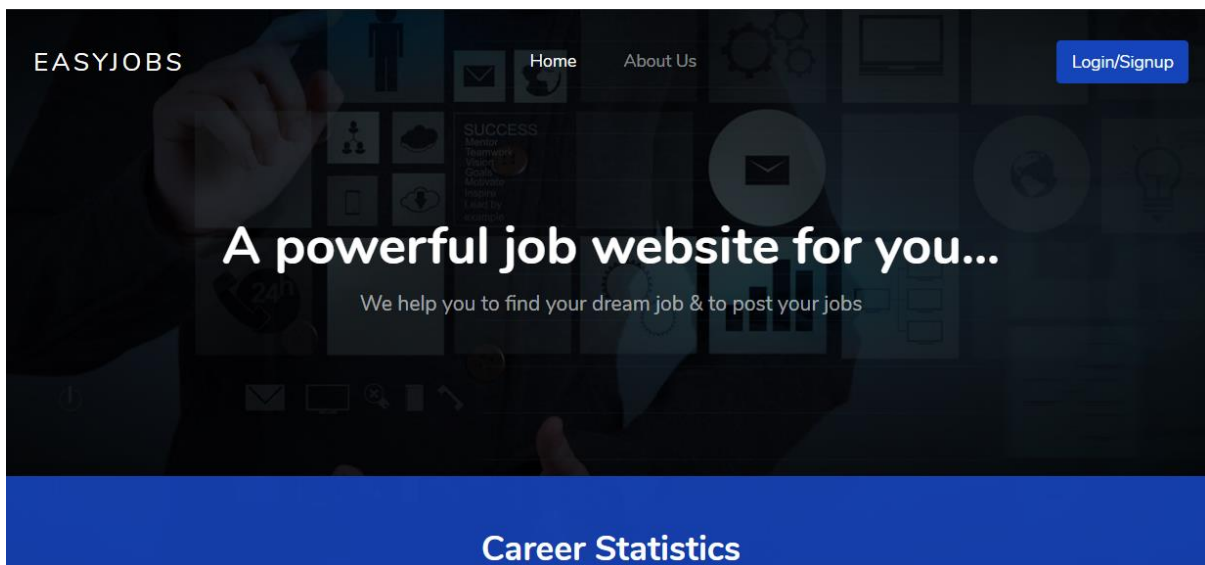


Figure A.1: Home Page

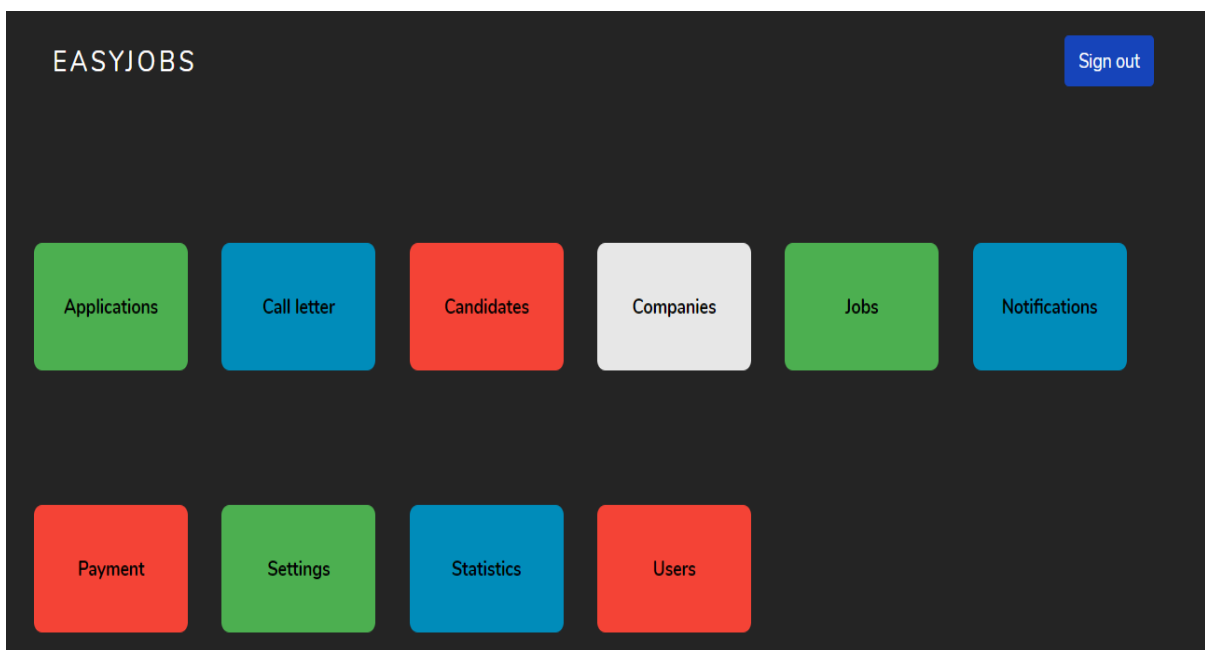


Figure A.2:Admin - Home Page

Back					Sign out	
Name	Address	Gender	Contact Number	Email Id		
ammu mariya	kannadiyil, kottayam , 90 kottayam , kerala	female	9878675645	a@gmail.com	View Resume	View More...
Ruby C	kannadiyil, kottayam , 123457 kottayam , kerala	male	6565786750	r@gmail.com	View Resume	View More...

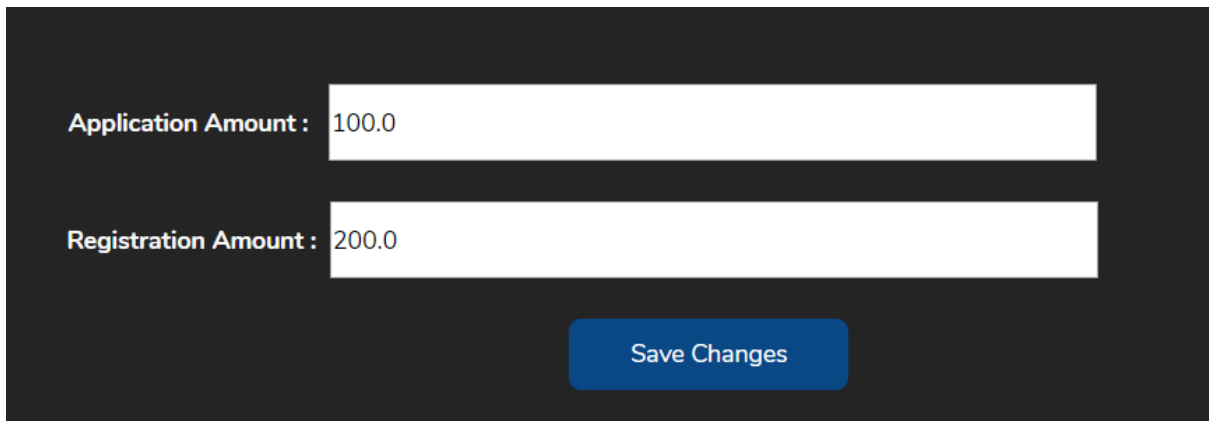
Figure A.3: Admin-Activate/Reject candidate

Back					Sign out	
Name	Description	Address	Contact Number	Email Id		
ruby	hsfdhs	kottayam , 675645 kottayam , kerala , india	1234567898	ru@gmail.com		Reject
ruby ltd	rubyy	tvm , 675645 tvm , keralaa , india	1234567898	rc@gmail.com		Reject

Figure A.4:Admin-Activate/Reject company

Active Applications : 1							
Shortlisted Applications : 1							
Selected Applications : 0							
Rejected Applications : 0							
Post	Company	Applicant	Id	Applied Date	Shortlist Status	Selection Status	Application Status (0-Rejected, 1-Active)
Android Developer	ruby ltd	Ruby C	11	May 11, 2020	yes	no	1

Figure A.5:Admin-View Application Report



Application Amount : 100.0

Registration Amount : 200.0

Save Changes

Figure A.6:Admin-Edit Amount for payment

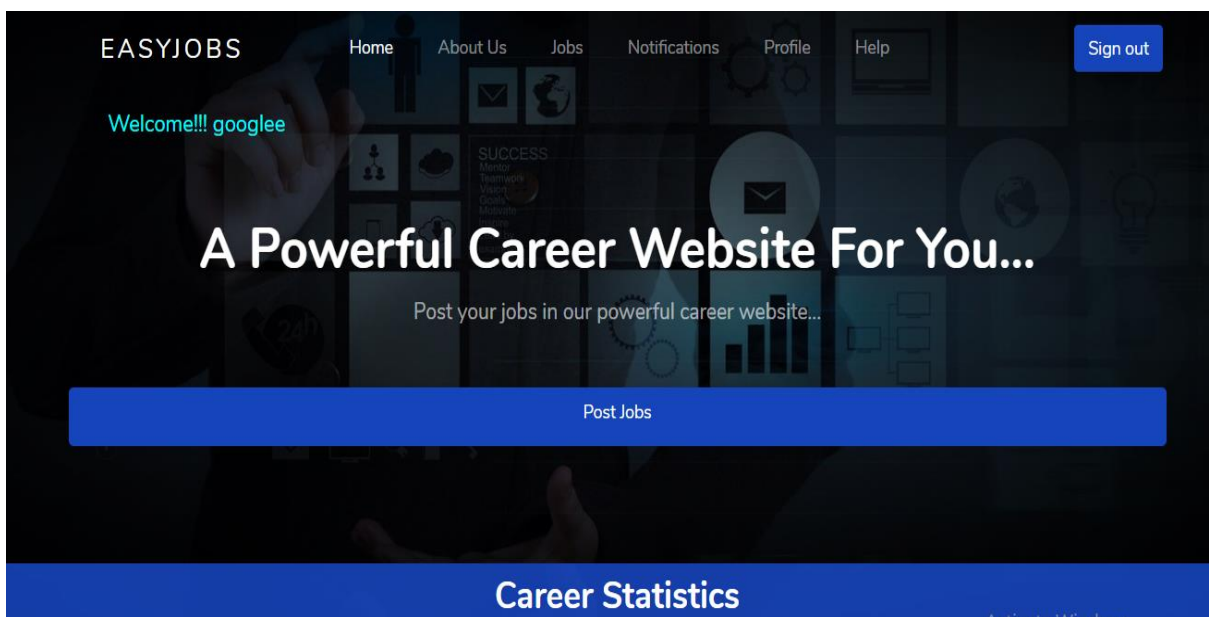
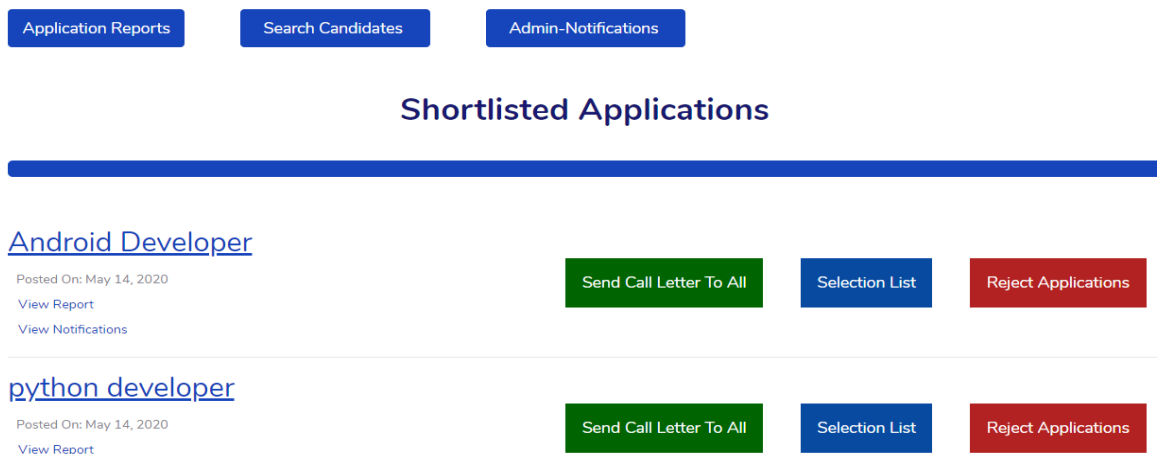


Figure A.7:Company-Home Page



Application Reports Search Candidates Admin-Notifications

Shortlisted Applications

Android Developer

Posted On: May 14, 2020

[View Report](#) [View Notifications](#)

Send Call Letter To All Selection List Reject Applications

python developer

Posted On: May 14, 2020

[View Report](#)

Send Call Letter To All Selection List Reject Applications

Figure A.8:Company-Notifications

[View Posted Jobs](#)

Post Your Jobs

Job Title:

Description :

Qualification:

Last Date :

dd-mm-yyyy

Job Location :

Pincode :

District :

State :

Email id :

Contact Number :

Country :

Criteria For Shortlist

0th Percentage :

12th Percentage :

Activate Wi

UG Percentage :

PG Percentage :

Required Skills :

submit

Activate Wi

Figure A.9:Company-Post Jobs

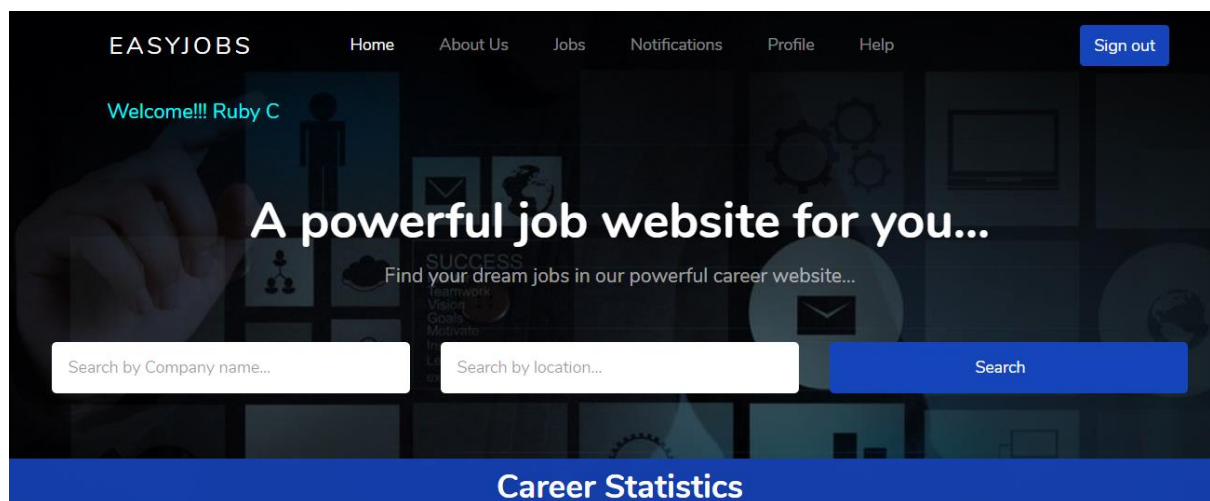


Figure A.10:Candidate Home Page

ruby ltd Android Developer

Posted On : May 3, 2020 **Last Date : May 24, 2020**

Description : an

Location : Trivandrum

District : trivandrum

Email Id : ibm@gmail.com

Country : india

12% : 50.0

PG% :

Qualification :bca/mca

Pincode : 876787

State : Kerala

Contact number : 9878

10 % : 50.0

UG% :

Skills Required : python, html

Quick
Apply

Activate Windows
Go to Settings to activate Windows.

Figure A.11:Candidate- Job Application

The screenshot shows a payment modal with a title bar 'Get Your Dream Job' and a close button. The form contains the following elements:

- An email input field with an envelope icon and the placeholder text 'Email'.
- A card number input field with a card icon and the placeholder text 'Card number'.
- Two input fields for the expiration date: 'MM / YY' with a calendar icon and 'CVC' with a lock icon.
- A checkbox labeled 'Remember me'.
- A large blue button at the bottom labeled 'Pay \$1.00'.

Figure A.12:Candidate- Payment

Notification From Companies

Figure A.13: Candidate- Notifications

A.2 SAMPLE CODE

views.py in Administrator app

```
from django.shortcuts import render, redirect

from django.http import HttpResponseRedirect

from company.models import com_pro, Jobpost, interview, notification

from candidate.models import can_pro, Jobapply

from .models import helpdesk,reply,payments

from datetime import date

import re


from django.contrib.auth.models import User,auth

from django.contrib import messages


from django.http import JsonResponse


from django.views.generic.base import View

# Create your views here.


def home(request):

    return render(request,'admin_home.html')


def admin_company(request):

    com = com_pro.objects.filter(status=1)

    c = com_pro.objects.filter(status=1).count()

    r = com_pro.objects.filter(status=0).count()

    return render(request,'admin_company.html',{'com':com,'c':c,'r':r})
```



```
def admin_reject_company(request):  
    com = com_pro.objects.filter(status=1)  
    rej = com_pro.objects.filter(status=0)  
    return render(request, 'admin_reject_company.html', {'com': com, 'rej': rej})
```

```
def reject_company(request, uname):  
    comp = com_pro.objects.get(com_username = uname)  
    comp.status = 0  
    comp.save()  
    u = User.objects.get(username=uname)  
    u.is_active = False  
    u.save()  
    return redirect('admin_reject_company')
```

```
def activate_company(request, uname):  
    comp = com_pro.objects.get(com_username = uname)  
    comp.status = 1  
    comp.save()  
    u = User.objects.get(username=uname)  
    u.is_active = True  
    u.save()  
    return redirect('admin_reject_company')
```

```
def admin_com_jobs(request, uname):  
    job = Jobpost.objects.filter(com_username = uname).order_by('-post_date')  
    return render(request, 'admin_com_jobs.html', {'job': job})
```

```
def admin_users(request):

    u = User.objects.filter(is_superuser="False",is_staff="False").order_by('-date_joined')

    return render(request,'admin_users.html',{'u':u})
```

```
def admin_candidate(request):

    can = can_pro.objects.filter(status=1)

    c = can_pro.objects.filter(status=1).count()

    r = can_pro.objects.filter(status=0).count()

    return render(request,'admin_candidate.html',{'can':can,'c':c,'r':r})
```

```
def admin_reject_candidate(request):

    can = can_pro.objects.filter(status=1)

    rej = can_pro.objects.filter(status=0)

    return render(request,'admin_reject_candidate.html',{'can':can,'rej':rej})
```

```
def reject_candidate(request,uname):

    canp = can_pro.objects.get(can_uname = uname)

    canp.status = 0

    canp.save()

    u = User.objects.get(username=uname)

    u.is_active = False

    u.save()

    return redirect('admin_reject_candidate')
```

```
def activate_candidate(request,uname):

    canp = can_pro.objects.get(can_uname = uname)

    canp.status = 1

    canp.save()
```

```
u = User.objects.get(username=uname)

u.is_active = True

u.save()

return redirect('admin_reject_candidate')
```

```
def candidate_details(request,uname):

    can = can_pro.objects.get(can_uname = uname)

    return render(request,'candidate_details.html',{'can':can})
```

```
def admin_jobs(request):

    job = Jobpost.objects.filter(status=1).order_by('-post_date')

    c = Jobpost.objects.filter(status=1).count()

    r = Jobpost.objects.filter(status=0).count()

    com =[]

    for e in job:

        cuname = e.com_username

        comp = com_pro.objects.get(com_username = cuname)

        cname = comp.com_name

        com.append(cname)

    return render(request,'admin_jobs.html',{'job':job,'com':com,'c':c,'r':r})
```

```
def reject_jobs(request):

    job = Jobpost.objects.filter(status=1).order_by('-post_date')

    com =[]

    for e in job:

        cuname = e.com_username

        comp = com_pro.objects.get(com_username = cuname)
```

```

        cname = comp.com_name

        com.append(cname)

    jobr = Jobpost.objects.filter(status=0)

    comr = []

    for e in jobr:

        cuname = e.com_username

        comp = com_pro.objects.get(com_username = cuname)

        cname = comp.com_name

        comr.append(cname)

    return render(request,'reject_jobs.html',{'job':job,'com':com,'jobr':jobr,'comr':comr})

def activate_jobs(request,jid):

    job = Jobpost.objects.get(id=jid)

    job.status=1

    job.save()

    return redirect('reject_jobs')

def delete_jobs(request,jid):

    job = Jobpost.objects.get(id=jid)

    job.status=0

    job.save()

    return redirect('reject_jobs')

def admin_applications(request):

    app = Jobapply.objects.filter().order_by('-apply_date')

    a = Jobapply.objects.filter(status=1).count()

    s = Jobapply.objects.filter(short="yes").count()

```

```

r = Jobapply.objects.filter(status=0).count()

sl = Jobapply.objects.filter(status=1,selected="yes").count()


can = []

com = []

job = []

for e in app:

    canuname = e.can_uname

    canpro = can_pro.objects.get(can_uname = canuname)

    canname = canpro.can_name

    can.append(canname)

for e in app:

    comuname = e.com_uname

    compro = com_pro.objects.get(com_username = comuname)

    comname = compro.com_name

    com.append(comname)


for e in app:

    jid = e.job_id

    jp = Jobpost.objects.get(id=jid)

    jname = jp.job_name

    job.append(jname)

return
render(request,'admin_applications.html',{'job':job,'com':com,'can':can,'app':app,'a':a,'s':s,'r':r,'sl':sl}
)


def admin_selection(request):

    app = Jobapply.objects.filter(status=1,selected="yes").order_by('-apply_date')

    sl = Jobapply.objects.filter(status=1,selected="yes").count()

```

```

can=[]

com=[]

job=[]

for e in app:

    canuname = e.can_uname

    canpro =can_pro.objects.get(can_uname = canuname)

    canname = canpro.can_name

    can.append(canname)

for e in app:

    comuname = e.com_uname

    compro = com_pro.objects.get(com_username = comuname)

    comname = compro.com_name

    com.append(comname)


for e in app:

    jid = e.job_id

    jp = Jobpost.objects.get(id=jid)

    jname =jp.job_name

    job.append(jname)

return render(request,'admin_selection.html',{'job':job,'com':com,'can':can,'app':app,'sl':sl})

```

```

def admin_notifications(request):

    n = notification.objects.filter(status=1).order_by('-call_date')

    post=[]

    com=[]

    can=[]

    for e in n:

        jid = e.job_id

```

```

    job = Jobpost.objects.get(id=jid)

    jname = job.job_name

    post.append(jname)

for e in n:

    comuname = e.com_username

    compro = com_pro.objects.get(com_username = comuname)

    comname = compro.com_name

    com.append(comname)

for e in n:

    canuname = e.can_uname

    canpro = can_pro.objects.get(can_uname = canuname)

    canname = canpro.can_name

    can.append(canname)

return render(request, 'admin_notifications.html', {'n':n, 'com':com, 'post':post, 'can':can})


def admin_pswd(request):

    return render(request, 'admin_pswd.html')


def change_password(request):

    if request.method == 'POST':

        pass1 = request.POST['pswd1']

        pass2 = request.POST['pswd2']

        pass3 = request.POST['pswd3']

        user = auth.authenticate(username = 'admin', password = pass1)

        if user is not None:

            u = User.objects.get(username = 'admin')

            if pass2 == pass3:

```

```

        u.set_password(pass3)

        u.save()

        messages.info(request,'Password Changed ,Please Login Again...')

        return render(request,'home.html')

    else:

        messages.info(request,'Password Missmatch...')

        return render(request,'admin_pswd.html')


    else:

        messages.info(request,'Invalid Password...')

        return render(request,'admin_pswd.html')

    else:

        return render(request,'admin_pswd.html')


def admin_helpdesk(request):

    hlp = helpdesk.objects.filter(status=1).order_by('-help_date')

    user =[]

    for e in hlp:

        uname = e.username

        c =com_pro.objects.filter(com_username=uname).count()

        if c != 0:

            compro =com_pro.objects.get(com_username = uname)

            comname = compro.com_name

            user.append(comname)

        else:

            canpro = can_pro.objects.get(can_uname=uname)

            canname = canpro.can_name

            user.append(canname)

```



```
return render(request,'admin_help.html',{'hlp':hlp,'user':user})
```

```
def reject_help(request,id):
```

```
    n = helpdesk.objects.get(id=id)
```

```
    n.status=0
```

```
    n.save()
```

```
    return redirect('admin_helpdesk')
```

```
def reply_help(request,id):
```

```
    return render(request,'reply_help.html',{'id':id})
```

```
def reply_submit(request,id):
```

```
    if request.method == 'POST':
```

```
        msg = request.POST['reply']
```

```
        r = reply(help_id=id,message=msg,reply_date=date.today(),status=1)
```

```
        r.save()
```

```
        return render(request,'reply_help.html',{'id':id})
```

```
    else:
```

```
        return render(request,'reply_help.html',{'id':id})
```

```
def pay_amount(request):
```

```
    pay = payments.objects.get(id=2)
```

```
    return render(request,'admin_payment.html',{'pay':pay})
```

```
def edit_amount(request):
```

```
    if request.method == 'POST':
```

```

ap_am = request.POST['app']

reg_am = request.POST['reg']

pay = payments.objects.get(id=2)

pay.apply_amount = ap_am

pay.save()

pay.reg_amount = reg_am

pay.save()

pay = payments.objects.get(id=2)

return render(request,'admin_payment.html',{'pay':pay})

```

```

def admin_stati(request):

```

```

    com_count = com_pro.objects.filter(status=1).count()

    can_count = can_pro.objects.filter(status=1).count()

    app_count = Jobapply.objects.filter(status=1).count()

    short_count = Jobapply.objects.filter(status=1,short="yes").count()

    sl_count = Jobapply.objects.filter(status=1,selected="yes").count()

    job_count = Jobpost.objects.filter(status=1).count()

    user_count = com_count + can_count

```

```

    comr_count = com_pro.objects.filter(status=0).count()

    canr_count = can_pro.objects.filter(status=0).count()

```

```

    return
    render(request,'admin_stati.html',{'com_count':com_count,'can_count':can_count,'app_count':app_count,

'short_count':short_count,'sl_count':sl_count,'job_count':job_count,'user_count':user_count,'comr_count':comr_count,'canr_count':canr_count

    })

```

View.py in Company app

```
from django.shortcuts import render
from django.http import HttpResponseRedirect

from company.forms import CompanyForm
from django.shortcuts import render, redirect
from django.contrib.auth.models import User,auth
from django.contrib import messages
from .models import com_pro, Jobpost, interview, notification
from candidate.models import can_pro, Jobapply
from administrator.models import helpdesk,reply
from datetime import date
from django.db.models import Q
from django.core.files.storage import FileSystemStorage

# Create your views here.

def home(request):

    cancount = can_pro.objects.filter(status=1).count()
    comcount = com_pro.objects.filter(status=1).count()
    jobcount = Jobpost.objects.filter(status=1).count()
    slt = Jobapply.objects.filter(selected="yes").count()

    return render(request,
'com_home.html',{ 'cancount':cancount,'comcount':comcount,'jobcount':jobcount,'slt':slt})
```

```

def com_about(request):

    cancount = can_pro.objects.filter(status=1).count()

    comcount = com_pro.objects.filter(status=1).count()

    jobcount = Jobpost.objects.filter(status=1).count()

    shrt = Jobapply.objects.filter(short="yes").count()

    slt = Jobapply.objects.filter(selected="yes").count()

    app = Jobapply.objects.all().count()

    return render(request,
'com_about.html',{ 'cancount':cancount,'comcount':comcount,'jobcount':jobcount,'slt':slt,'app':
app,'shrt':shrt})

```

```

def com_jobs(request, uname):

    jobs = Jobpost.objects.filter(com_username = uname,status=1).order_by('-post_date')

    return render(request, 'com_jobs.html', {'jobs':jobs})

```

```

def com_contact(request):

    return render(request, 'com_contact.html')

```

```

def com_noti(request, uname):

    apply = Jobpost.objects.filter(com_username = uname,status=1).order_by('-post_date')

    return render(request, 'com_noti.html', {'apply':apply})

```

```

def com_viewnoti(request,uname,id):

    c = Jobapply.objects.filter(com_uname=uname,job_id=id,short="yes").count()

    capp = Jobapply.objects.filter(com_uname=uname,job_id=id).count()

    jname = Jobpost.objects.get(id=id,status=1)

```

```

    # noti =
    Jobapply.objects.filter(com_uname=uname,job_id=id,short="yes",call="yes").order_by('-
    call_date')

    nc = notification.objects.filter(com_username=uname,job_id=id).count()

    if nc != 0:

        noti = notification.objects.filter(com_username=uname,job_id=id)

        can = []

        for e in noti:

            cuname = e.can_uname

            canpro=can_pro.objects.get(can_uname=cuname)

            canname = canpro.can_name

            can.append(canname)

        return
    render(request,'com_viewnoti.html',{ 'noti':noti,'c':c,'capp':capp,'jname':jname,'can':can })

    else:

        messages.info(request,'No Active Notifications...')

        return redirect('com_noti',uname)

```

```

def com_viewapp(request,uname,id):

    c = Jobapply.objects.filter(com_uname=uname,job_id=id,short="yes").count()

    capp = Jobapply.objects.filter(com_uname=uname,job_id=id).count()

    jname = Jobpost.objects.get(id=id,status=1)

    app = Jobapply.objects.filter(com_uname =uname,job_id=id)

    return render(request,'com_viewapp.html',{ 'app':app,'c':c,'capp':capp,'jname':jname })

```

```

def selection_list(request,uname,id):

    slt =
    Jobapply.objects.filter(com_uname=uname,job_id=id,status=1,selected="yes").count()

    jname = Jobpost.objects.get(id=id,status=1)

    jc = Jobapply.objects.filter(com_uname=uname,job_id=id,selected="yes").count()

    if jc != 0:

        j =
        Jobapply.objects.filter(com_uname=uname,job_id=id,selected="yes",status=1).order_by('-
        apply_date')

        can =[]

        for e in j:

            cuname = e.can_uname

            canpro=can_pro.objects.get(can_uname=cuname)

            canname = canpro.can_name

            can.append(canname)

        return render(request,'selection_list.html',{ 'jname':jname,'slt':slt,'j':j,'can':can })

    else:

        messages.info(request,'No Active Selection List...')

        return redirect('com_accept_applications',id,uname)

def noti_report(request,uname,id):

    c = Jobapply.objects.filter(com_uname=uname,job_id=id,short="yes").count()

    rjt = Jobapply.objects.filter(com_uname=uname,job_id=id,status=0).count()

    slt =
    Jobapply.objects.filter(com_uname=uname,job_id=id,status=1,selected="yes").count()

    capp = Jobapply.objects.filter(com_uname=uname,job_id=id).count()

    jname = Jobpost.objects.get(id=id,status=1)

```

```

jc = Jobapply.objects.filter(com_uname=username,job_id=id).count()
if jc != 0:
    j = Jobapply.objects.filter(com_uname=username,job_id=id).order_by('-apply_date')
    can = []
    for e in j:
        cuname = e.can_uname
        canpro=can_pro.objects.get(can_uname=cuname)
        canname = canpro.can_name
        can.append(canname)
    return
render(request,'noti_report.html',{ 'j':j,'c':c,'capp':capp,'jname':jname,'rjt':rjt,'slt':slt,'can':can })
else:
    messages.info(request,'No Active Report...')
    return redirect('com_noti',uname)

```

View.py in Candidate app

```

from django.shortcuts import render
from django.http import HttpResponseRedirect

from company.forms import CompanyForm
from django.shortcuts import render, redirect
from django.contrib.auth.models import User,auth
from django.contrib import messages
from company.models import com_pro, Jobpost, interview, notification
from candidate.models import can_pro, Jobapply
from administrator.models import helpdesk, reply
from datetime import date

```

```

import re

from django.db.models import Q

from django.core.files.storage import FileSystemStorage

import stripe

from django.conf import settings

# Create your views here.


def home(request):

    cancount = can_pro.objects.filter(status=1).count()

    comcount = com_pro.objects.filter(status=1).count()

    jobcount = Jobpost.objects.filter(status=1).count()

    slt = Jobapply.objects.filter(selected="yes").count()

    if request.method == 'POST':

        srch = request.POST['srch']

        loc = request.POST['loc']

        if srch != "" and loc != "":

            #comp =
            com_pro.objects.filter(Q(com_name__icontains=srch)|Q(com_place__icontains=loc)|Q(com_dt__ic
            ontains=loc)|Q(com_state__icontains=loc)|Q(com_country__icontains=loc))

            if
            com_pro.objects.filter(Q(com_name__icontains=srch)|Q(com_place__icontains=loc)|Q(com_dt__ic
            ontains=loc)|Q(com_state__icontains=loc)|Q(com_country__icontains=loc),status=1).exists():

                comp =
                com_pro.objects.filter(Q(com_name__icontains=srch)|Q(com_place__icontains=loc)|Q(com_dt__ic
                ontains=loc)|Q(com_state__icontains=loc)|Q(com_country__icontains=loc),status=1)

            return render(request,'can_search.html',{'comp':comp})

        else:

            messages.info(request, 'No Results...')

```



```

        return render(request, 'can_search.html')

elif srch != "" and loc == "":

    if com_pro.objects.filter(Q(com_name__icontains=srch),status=1).exists():

        comp = com_pro.objects.filter(Q(com_name__icontains=srch),status=1)

        return render(request, 'can_search.html',{'comp':comp})

    else:

        messages.info(request, 'No Results...')

        return render(request, 'can_search.html')

elif srch == "" and loc != "":

    if

com_pro.objects.filter(Q(com_place__icontains=loc)|Q(com_dt__icontains=loc)|Q(com_state__icon
tains=loc)|Q(com_country__icontains=loc),status=1).exists():

        comp =
com_pro.objects.filter(Q(com_place__icontains=loc)|Q(com_dt__icontains=loc)|Q(com_state__icon
tains=loc)|Q(com_country__icontains=loc),status=1)

        return render(request, 'can_search.html',{'comp':comp})

    else:

        messages.info(request, 'No Results...')

        return render(request, 'can_search.html')

else:

    return render(request, 'can_home.html')

return render(request,
'can_home.html',{'cancount':cancount,'comcount':comcount,'jobcount':jobcount,'slt':slt})

```

```

def can_about(request):

    cancount = can_pro.objects.filter(status=1).count()

    comcount = com_pro.objects.filter(status=1).count()

    jobcount = Jobpost.objects.filter(status=1).count()

    slt = Jobapply.objects.filter(selected="yes").count()

    shrt = Jobapply.objects.filter(short="yes").count()

    app = Jobapply.objects.all().count()

    if request.method == 'POST':

        srch = request.POST['srch']

        loc = request.POST['loc']


        if srch != "" and loc != "":

            #comp =
com_pro.objects.filter(Q(com_name__icontains=srch)|Q(com_place__icontains=loc)|Q(com_dt__ic
ontains=loc)|Q(com_state__icontains=loc)|Q(com_country__icontains=loc))

            if
com_pro.objects.filter(Q(com_name__icontains=srch)|Q(com_place__icontains=loc)|Q(com_dt__ic
ontains=loc)|Q(com_state__icontains=loc)|Q(com_country__icontains=loc),status=1).exists():

                comp =
com_pro.objects.filter(Q(com_name__icontains=srch)|Q(com_place__icontains=loc)|Q(com_dt__ic
ontains=loc)|Q(com_state__icontains=loc)|Q(com_country__icontains=loc),status=1)


            return render(request,'can_search.html',{'comp':comp})

        else:

            messages.info(request, 'No Results...')

            return render(request,'can_search.html')


    elif srch != "" and loc == "":

        if com_pro.objects.filter(Q(com_name__icontains=srch),status=1).exists():

```

```

        comp = com_pro.objects.filter(Q(com_name__icontains=srch),status=1)

        return render(request,'can_search.html',{'comp':comp})

    else:

        messages.info(request, 'No Results...')

        return render(request,'can_search.html')

elif srch == "" and loc != "":

    if
com_pro.objects.filter(Q(com_place__icontains=loc)|Q(com_dt__icontains=loc)|Q(com_state__icon
tains=loc)|Q(com_country__icontains=loc),status=1).exists():

        comp =
com_pro.objects.filter(Q(com_place__icontains=loc)|Q(com_dt__icontains=loc)|Q(com_state__icon
tains=loc)|Q(com_country__icontains=loc),status=1)

        return render(request,'can_search.html',{'comp':comp})

    else:

        messages.info(request, 'No Results...')

        return render(request,'can_search.html')

else:

    return
render(request,'can_about.html',{'cancount':cancount,'comcount':comcount,'jobcount':jobcount,'slt
':slt,'app':app,'sht':sht})

    return render(request,
'can_about.html',{'cancount':cancount,'comcount':comcount,'jobcount':jobcount,'slt':slt,'app':app,'s
hrt':shrt})

def can_jobs(request, uname):

    canpro = can_pro.objects.get(can_uname= uname,status=1)

    if request.method == 'POST':

```

```

srch = request.POST['srch']

loc = request.POST['loc']


if srch != "" and loc != "":

    #comp =
    com_pro.objects.filter(Q(com_name__icontains=srch)|Q(com_place__icontains=loc)|Q(com_dt__ic
    ontains=loc)|Q(com_state__icontains=loc)|Q(com_country__icontains=loc))

    if
    com_pro.objects.filter(Q(com_name__icontains=srch)|Q(com_place__icontains=loc)|Q(com_dt__ic
    ontains=loc)|Q(com_state__icontains=loc)|Q(com_country__icontains=loc),status=1).exists():

        comp =
        com_pro.objects.filter(Q(com_name__icontains=srch)|Q(com_place__icontains=loc)|Q(com_dt__ic
        ontains=loc)|Q(com_state__icontains=loc)|Q(com_country__icontains=loc),status=1)

        return render(request,'can_search.html',{'comp':comp})

    else:

        messages.info(request, 'No Results...')

        return render(request,'can_search.html')


elif srch != "" and loc == "":

    if com_pro.objects.filter(Q(com_name__icontains=srch),status=1).exists():

        comp = com_pro.objects.filter(Q(com_name__icontains=srch),status=1)

        return render(request,'can_search.html',{'comp':comp})

    else:

        messages.info(request, 'No Results...')

        return render(request,'can_search.html')


elif srch == "" and loc != "":

```

```

        if
com_pro.objects.filter(Q(com_place__icontains=loc)|Q(com_dt__icontains=loc)|Q(com_state__icon
tains=loc)|Q(com_country__icontains=loc),status=1).exists():

        comp =
com_pro.objects.filter(Q(com_place__icontains=loc)|Q(com_dt__icontains=loc)|Q(com_state__icon
tains=loc)|Q(com_country__icontains=loc),status=1)

        return render(request,'can_search.html',{'comp':comp})

    else:

        messages.info(request, 'No Results...')

        return render(request,'can_search.html')

    else:

        return redirect('can_jobs',uname)

#jobs = Jobpost.objects.filter(last_date__gte= date.today())

#jobs = Jobpost.objects.filter(last_date__gte= date.today(),
job_qua__in=[canpro.pg_course,canpro.ug_course, job])

if canpro.school != "" and canpro.sc_board != "" and canpro.sc_percent != "" and
canpro.ug_course == "" and canpro.pg_course == "" and canpro.hss_stream == "":

    jobs = Jobpost.objects.filter(last_date__gte= date.today(),
job_qua__in=["10","sslc","tenth","10th"],status=1).order_by('-post_date')

    comp=[]

    logo=[]

    for e in jobs:

        c = e.com_username

        com =com_pro.objects.get(com_username=c)

        company =com.com_name

        comp.append(company)

        img = com.logo

        logo.append(img)

```

```

return render(request, 'can_jobs.html', {'jobs':jobs,'comp':comp,'logo':logo})

elif canpro.school != "" and canpro.sc_board != "" and canpro.sc_percent != "" and
canpro.hss_stream != "" and canpro.ug_course == "" and canpro.pg_course == "":

    jobs = Jobpost.objects.filter( Q(job_qua__exact= "12") | Q(job_qua__exact= "twelveth") |
Q(job_qua__exact= "twelve")| Q(job_qua__icontains= "10") | Q(job_qua__icontains= "sslc") |
Q(job_qua__icontains= "tenth") | Q(job_qua__icontains= "10th") | Q(job_qua__icontains= "ten") |
Q(job_qua__icontains= canpro.hss_stream) , last_date__gte= date.today(),status=1).order_by('-
post_date')

    comp =[]

    logo=[]

    for e in jobs:

        c = e.com_username

        com =com_pro.objects.get(com_username=c)

        company =com.com_name

        comp.append(company)

        img = com.logo

        logo.append(img)

    return render(request, 'can_jobs.html', {'jobs':jobs,'comp':comp,'logo':logo})

elif canpro.school != "" and canpro.sc_board != "" and canpro.sc_percent != "" and
canpro.hss_stream != "" and canpro.ug_course != "" and canpro.pg_course == "":

    jobs = Jobpost.objects.filter(Q(job_qua__icontains=canpro.ug_course) | Q(job_qua__icontains=
canpro.hss_stream)| Q(job_qua__iexact="12") ,last_date__gte= date.today(),status=1).order_by('-
post_date')

    comp =[]

    logo=[]

    for e in jobs:

        c = e.com_username

        com =com_pro.objects.get(com_username=c)

```

```
company = com.com_name  
comp.append(company)  
  
img = com.logo  
logo.append(img)  
  
return render(request, 'can_jobs.html', {'jobs':jobs,'comp':comp,'logo':logo})
```

else:

```
jobs = Jobpost.objects.filter( Q(job_qua__icontains= canpro.pg_course) |  
Q(job_qua__icontains=canpro.ug_course) | Q(job_qua__icontains= canpro.hss_stream)|  
Q(job_qua__iexact="12") ,last_date__gte= date.today(),status=1).order_by('-post_date')
```

```
comp=[]  
logo=[]  
  
for e in jobs:  
  
    c = e.com_username  
  
    com = com_pro.objects.get(com_username=c)  
  
    company = com.com_name  
    comp.append(company)  
  
    img = com.logo  
    logo.append(img)  
  
return render(request, 'can_jobs.html', {'jobs':jobs,'comp':comp,'logo':logo})
```
