# RECRUITMENT SYSTEM

| | IDENTIFY A SOFTWARE SYSTEM THAT NEEDS TO BE DEVELOPED |
|---|---|

**AIM :**

To identify a software system that needs to be developed

**Recruitment System:**

A Recruitment System (RS) is a set of tools designed to automate and manage your organization's recruiting and staffing operations.

Recruitment systems have a range of components and functions. They include applicant tracking systems for managing job postings and applications, and customer relationship management-type functions to keep applicants connected and engaged

It helps the organization also to search for a profile with specific skills only, so they do not have to go through every job seeker asking him/her about the skill they needed.

The recruitment system is a platform to hold all the processes of employment for the organization. It is an automated system with various components to facilitate the processes involved in finding, attracting, assessing, interviewing, and hiring a new employee.

Presently the system includes a manual process through files, folders, and binders. Eventually, the job seeker has to find the company which is offering the required profile. Then check the eligibility of the post or do a search on eligibility basis by visiting the various company which he knows of.

The growth of recruitment system has been driven by combination of actual costssavings in the recruitment process,increased ease and efficienas for the emploter along with animproved experience for candidates.This software system reduce agency and processing costs,increase speed tohire,improve productivity and candidate quality.This recruitment system is an online website in which job seekerscan register themselves online and apply for job and attend the exam. This software producthave facilities, where prespective candidates can upload their curriculum vita and apply for jobssuited to them.

| | CREATION OF SOFTWARE REQUIREMENT SPECIFICATION |
|---|---|

**AIM :**

To Document the Software Requirements Specification (SRS) for the identified system.

**SRS (Software Requirement Specifications) Document**

A Software requirements specification (SRS), a requirements specification for a software system, is a complete description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software.

In addition it also contains non-functional requirements. Non-functional requirements impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

The software requirements specification document enlists all necessary requirements that are required for the development. To derive the requirements we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with customer.

- **Functionality:** It details all the features and functionalities the software needs to deliver. This includes user actions, system responses, and expected outputs.
- **Non-Functional Requirements:** Beyond functionalities, the SRS also specifies non-functional aspects like performance, security, usability, and reliability.
- **Target Audience:** The document caters to various stakeholders, including customers, developers, testers, and project managers. It ensures everyone involved has a clear understanding of the project's goals and expectations.

The purpose of the SRS is to:

1. Establish the basis for agreement between the customers and the suppliers on what thesoftware product is to do.
2. Provide a basis for developing the software design.
3. Reduce the development effort.
4. Provide a basis for estimating costs and schedules.
5. Provide a baseline for validation and verification.
6. Facilitate transfer.
7. Serve as a basis for enhancement.

**Problem statement:**

Indeed, Recruitment system landscape changed in recent years due to rapid development in technology. The year is 2024, and recruiters, hiring managers and recruitment teams are still struggling to attract the right candidate and engage with them effectively to hire them for growth.

The recruitment process is the most profound process from the inside, where the recruiter has to pass every stage along with the candidates.

The current recruitment process is manual and time-consuming, leading to several inefficiencies:

- **Lengthy hiring times:** Sifting through resumes, scheduling interviews, and managing communication is a slow process, delaying the filling of open positions.
- **Inconsistent candidate experience:** The application process can be confusing or lack transparency for candidates, leading to a negative employer brand.
- **Difficulty attracting top talent:** Limited visibility into the talent pool and a cumbersome application process make it hard to attract qualified candidates.
- **Data silos and bias:** Scattered resumes and manual screening can lead to information overload and unconscious bias in candidate selection.
- **Limited reporting and analytics:** Lack of data on recruitment metrics makes it difficult to measure the effectiveness of sourcing strategies and identify areas for improvement.

**Biggest Recruitment Challenges Faced by Recruiters:**

1. Talent shortage
2. Attracting the right candidates
3. Targeting passive candidates
4. Engaging with qualified candidates
5. Reducing Time-to-hire
6. Building a strong employer brand
7. Creating an efficient recruiting process
8. Eliminating bias in hiring
9. Securing a positive candidate experience
10. Implementing data-driven recruitment

## REQUIREMENT ANALYSIS

The purpose of system requirement analysis is to obtain a through and detailed understanding of business needs as defined in organization and captured in business case and to break if down into discrete requirements, which are then clearly defined, reviewed and agree upon with the customers and decision makers. During the system requirement analysis the framework for the application is developed and providing the foundation for future design and development efforts.

Requirement analysis is also called as requirement engineering which is the process of determining user expectation. These features are called requirement it must be quantifiable. In software engineering such requirement are often called as functional specification. **Modules of Recruitment System:**

The system has a different segment to process a specific task which is the modules. This will help the system to developed easily and makes it more user-friendly.

**Login Module:**

.This will help users to login into the system using id and password. A user who has the valid id and password can only log in to their respective accounts. It will help the authentication of the user who enters the system. The module provides a layer of security over the system as only authorized personal can login into the system.

This prevents any anonymous person to enter the system and mishandle the records. It is better than the manual method as they do not have any security measure of who can access the system and who cannot. **Registration Module:**

In this segment, we will register the new user of the system. As they are two different types of users i.e. employer and job seekers this module can be fragmented into two parts. Each part has its own interface and information required to get registered in the system.

**Post Requisition:**

In this interface, the employer who has registered themselves in the system can post the jobs. They can give the requisition specifying the post and skill needed for that post. It also implies the criteria of recruitment the employer has planned for the post.

**Job Search:**

This module is for the job seeker where they can search all the requisition present in the system. One can filter the search based on their skill and experience of the job. Once selected a requisition applicant can apply for the post.

**Hardware Requirements of Recruitment System :**

- Processor: Intel P-IV System
- Processor Speed: 250 MHz to 833 MHz
- Ram: 512 Mb
- Ram Hard Disk: 40 Gb

**Software Requirements of Online Recruitment System**

- Operating System: Windows 2000 Professional
- Environment: Visual Studio .NET 2002
- Net Framework: Version 1.0
- Language: Visual Basic .NET,PHP
- Backend: SQL Server 2000,APACHE SERVER

**Functional Requirement of Recruitment System:**

1.Job Posting Management: Effective job posting management ensures swift dissemination of job openings, optimizing visibility to attract diverse talent pools while maintaining consistency in branding and messaging.

2.Candidate Application Management: Streamlined candidate application management facilitates seamless tracking, organization, and communication with applicants, enhancing the candidate experience and expediting the screening process for recruiters.

3.Resume Parsing: Advanced resume parsing capabilities automate the extraction and categorization of pertinent candidate information from resumes, enhancing efficiency in candidate assessment and matching against job requirements.

4.Interview Scheduling: Efficient interview scheduling features simplify the coordination of interview times between candidates and hiring teams, reducing scheduling conflicts and enhancing overall recruitment process efficiency.

5.Candidate Evaluation: Comprehensive candidate evaluation tools enable structured assessment of applicant qualifications, skills, and fit for the role, facilitating informed hiring decisions and ensuring alignment with organizational objectives.

**Non-Functional Requirement of Recruitment System:**

- **Scalability:** This NFR describes how well the software can adapt to an increase in users, data, or workload.
- **Security:** This NFR is concerned with protecting the software system and data from unauthorized access, modification, or deletion. Security NFRs could include:
    - All user passwords must be encrypted using a strong hashing algorithm.
    - The system must implement two-factor authentication for all logins.
- **Usability:** This NFR focuses on how easy and intuitive the software is to use for the target audience. Usability NFRs might include:
    - The user interface should be simple and user-friendly, with clear navigation and consistent design patterns.
    - The system should provide comprehensive help documentation and tutorials.
- **Reliability:** This NFR describes how often the software functions correctly and consistently without failures or crashes. An NFR for reliability could be:
    - The system uptime must be 99.9% during business hours.
- **Availability:** This NFR describes how readily accessible the software is to users. An NFR for availability could be:
    - The system must be available 24/7 with minimal downtime for scheduled maintenance.
- **Maintainability:** This NFR refers to how easy it is to understand, modify, and fix the software code. An NFR for maintainability might be:
    - The code must be well-documented, using clear naming conventions and modular design principles.

**Conclusion**

Recruitment system can be used by employers to recruit the candidates based on their experience and the further interviews easily.It helps in maintaining the information of potential candidates in one place. It can be easily accessed by both applicants and employers. It is kept safe for a long period of time without any changes or omissions. It reduces the time employer take to make few recruitments in their firm.

# UNIFIED MODELING LANGUAGE

## DESIGN

Design has a great impact on the overall success of software development projects. A large payoff is associated with creating a good design up from before writing a single code, while this is due to all programming classes and objects understanding approach better design usually simplifies the implementation and maintenance. During the design phase we must evaluate the model to actual objects that can be perform the required tasks. There is a shift in emphasis from the application domain to implementation. The class during analysis provides as a framework for design phases.

## MODELING

Building a model for a software system prior to its construction is as essential as having a blueprint for building a large building. Good models are essential for communication among project teams. A modeling language must include

- Model elements-fundamental modeling concepts and semantics.  Notation-visual rendering of model elements.
- Guidelines-expression of usage within the trade

The use of visual notation to represent or model a problem can provide us with several benefits relating to clarity, familiarity, maintenance, and simplification.

## UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems and their components. UML is a graphical language with sets of rules and semantics. The rules and semantics of a model are expressed in English, in a form known as object constraint language (OCL). OCL is a specification language that uses simple logic for specifying the properties of a system. The UML is not intended to be a visual programming language in the sense of having all the necessary visual and semantic support to replace programming languages.

## UML DIAGRAMS

UML defines nine graphical diagrams

- Use-case diagram
- Class diagram
- Behavior Diagram
- Interaction Diagram
- Sequence Diagram
- Collaboration diagram
- State chart diagram
- Activity diagram
- Implementation diagram
- Component diagram

| | USE CASE DIAGRAM |
|---|---|

**AIM :**

To identify Use Cases and develop the Use Case model for the identified requirements.

**USE CASE DIAGRAM**

A use-case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) associations between the actors and the use cases, and generalization among the use cases. It shows the relationship among the actors and use cases within a system . They separate the system into actors and use cases. An actor represents the roles that can be played by the user of the system.

**Use case describes the behavior of the system.**

- 1. **Use cases.** A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.
- 2. **Actors.** An actor is a person, organization, or external system that plays a role in one or more interactions with your system. The actors are drawn as stick figures.
- 3. **Associations.** Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an inter action described by a use case.
- 4. **System boundary boxes (optional). A** rectangle around the use cases is called the system boundary box and is used to indicate the scope of the system.
- 5. **Packages (optional).** Packages are UML constructs that enable you to organize model elements (such as use cases) into groups.

**RELATIONSHIPS IN USE CASE DIAGRAM**

Three relationships are supported among use cases by the UML standard, which describes graphical notation for these relationships.

1. **Include(<<include>>)**

i. A given use case may *include* another. The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviors from multiple use cases into a single description.

2. **Extend(<<extend>>)**

i. A given use case, (the extension) may *extend* another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. This can be useful for dealing with special cases, or in accommodating new requirements during system maintenance and extension.

3. **Generalization**

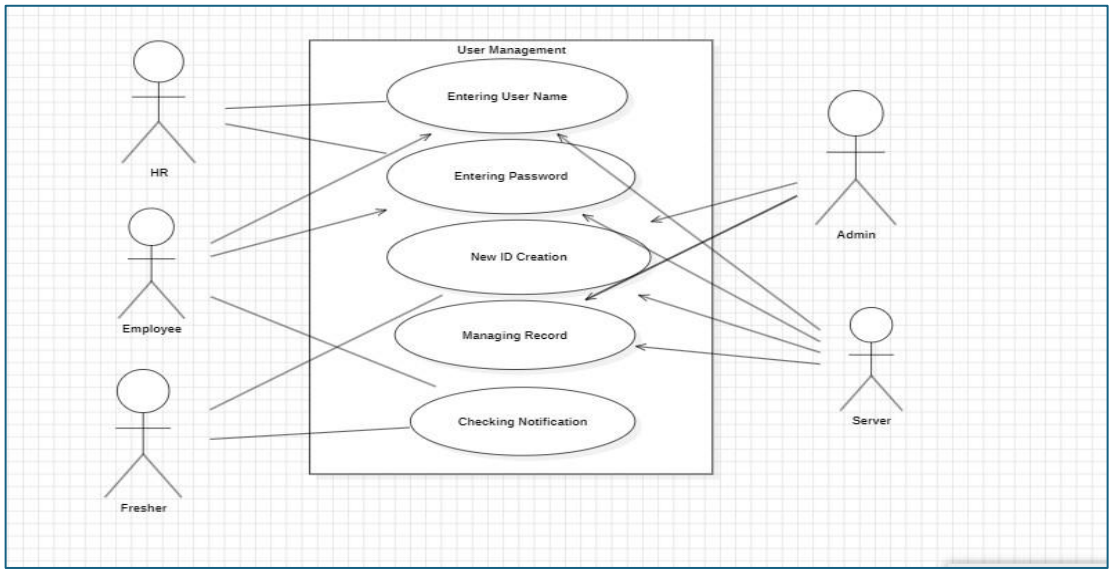
i. A generalization/ **specialization** relationship exists among use cases. A given use case may be a specialized form of an existing use case. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case. This resembles the object-oriented concept of sub-classing, in practice it can be both useful and effective to factor common behaviors,

constraints and assumptions to the general use case, describe them once, and deal same as except details in the specialized cases.

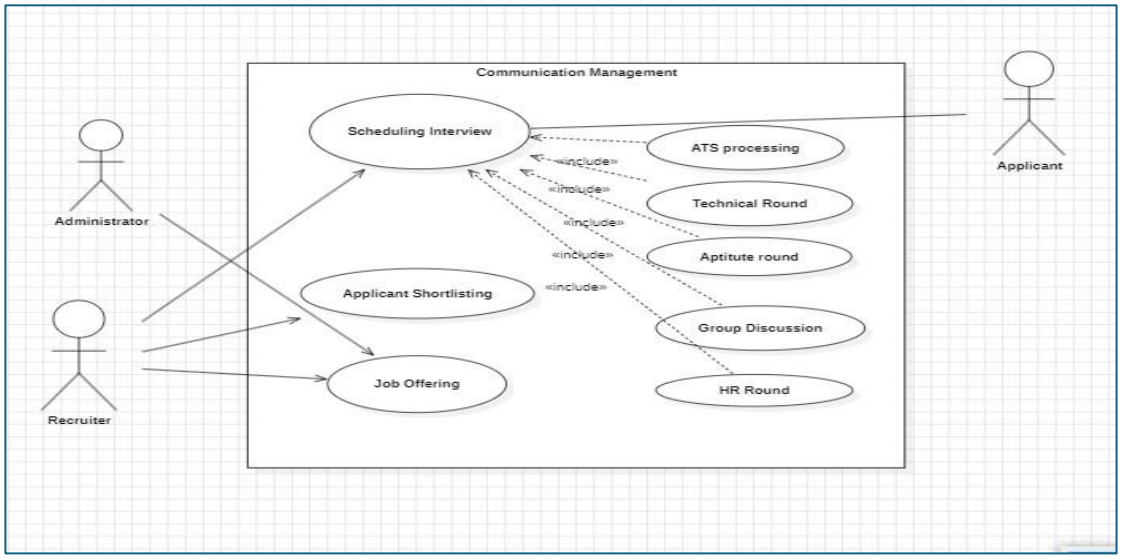
**PROCEDURE**

1. Identify distinct actors and use cases.
2. The external actors are placed to the left of the use case and the internal actors to the right.
3. Select Model | Add Diagram | Use Case Diagram in Menu Bar or select Add Diagram | Use Case Diagram in Context Menu in Star UML.
4. Name the diagram and draw the diagram using the symbols present in the Star UML toolbox according to the identified actors and use cases.
5. Draw the relationship between the use cases and actors as extend, include and generalization types using the symbols.

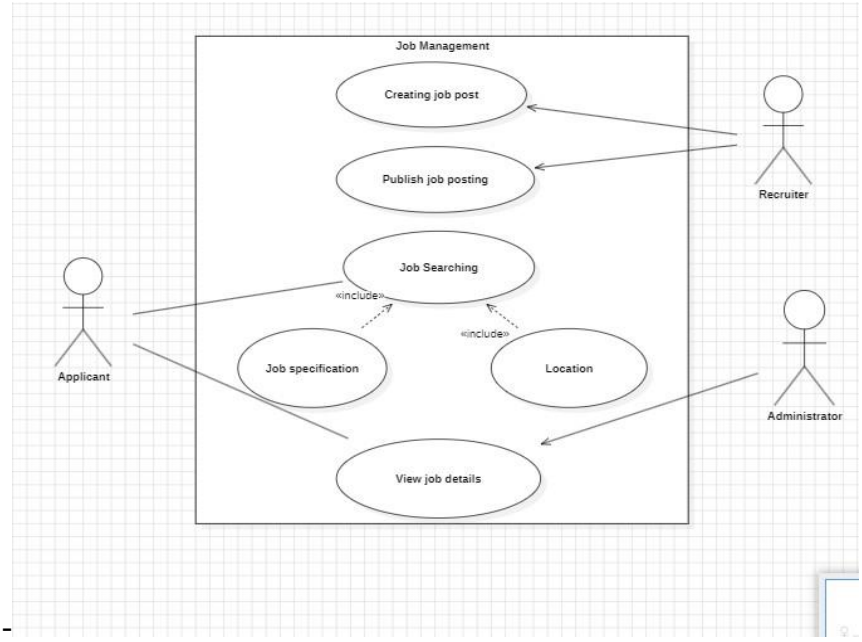
**User Management**



## Communication Management




## Job Management



| | UML-CLASS DIAGRAM |
|---|---|

**AIM :**

To identify conceptual classes and develop a domain model with UML Class diagram

**CLASS DIAGRAM :**

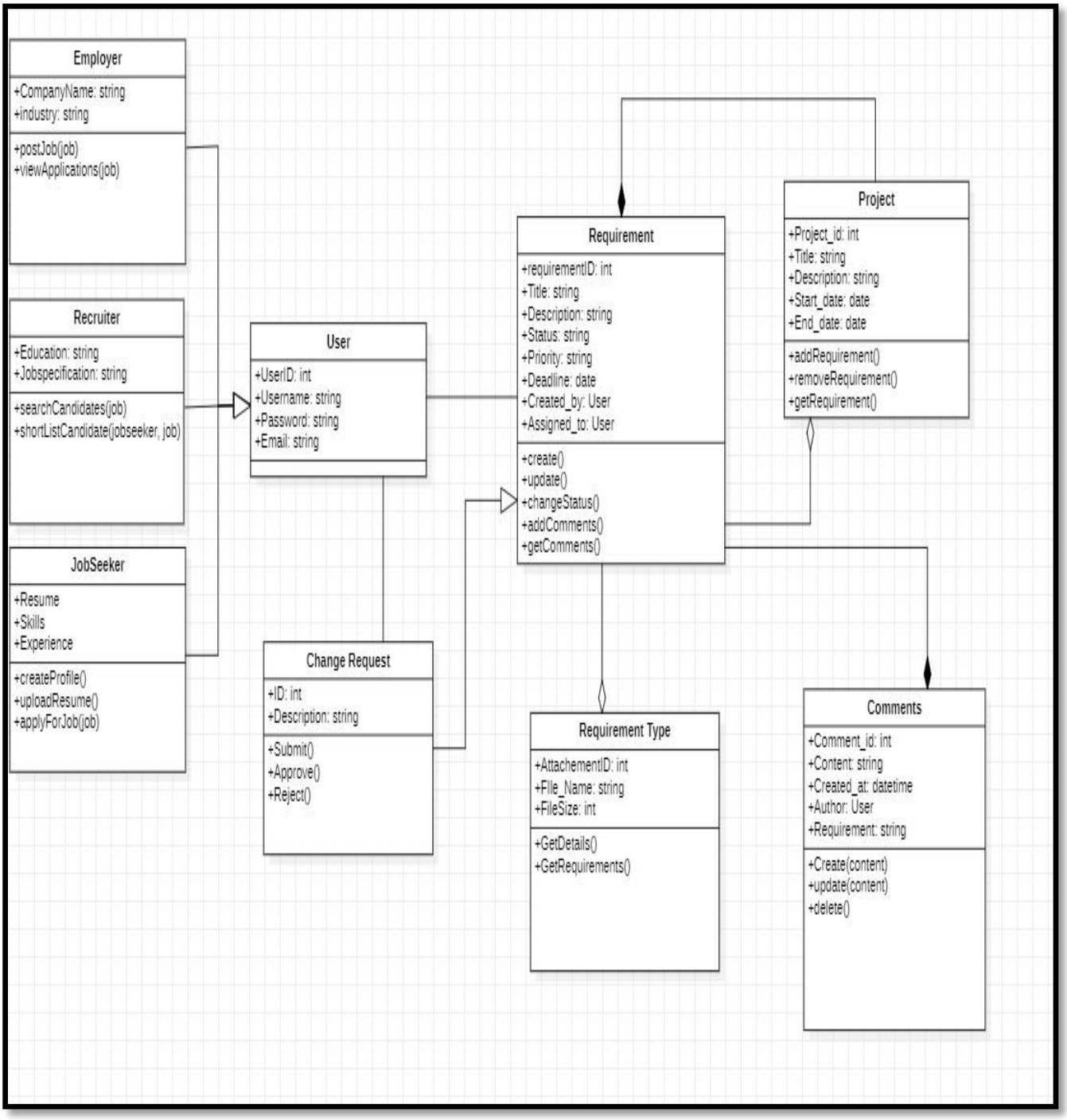
In the Unified Modeling Language (UML), a class diagram is a collection of static modeling elements, such as classes and their relationships, connected as a graph to each other and to their contents; their internal structures, and their relationships to other classes.

A class is drawn as a rectangle with three components separated by horizontal lines. The top compartment holds the class name, other general properties of the class, such as attributes, are in the middle compartment, and the bottom compartment holds a list of operations. Either or both the attribute and operation compartments may be suppressed.

A separator line is not drawn for a missing compartment if a compartment is suppressed; no inference can be drawn about the presence or absence of elements in it. The class name and other properties should be displayed in up to three sections. A stylistic convention of UML is to use an italic font for abstract classes and a normal (roman) font for concrete classes.

**PROCEDURE**

1. Identify the classes that are taking part in the diagram.
2. Select **Model | Add Diagram | Class Diagram** in the Menu Bar or select **Add Diagram | Class Diagram** in Context Menu.
3. Select class diagram, specify the attributes, operations or methods, parameters, datatype by selecting following.
   - Select **Model | Add | Class** in Menu Bar or **Add | Class** in Context Menu.
   - Select **Model | Add | Attribute** in Menu Bar or **Add | Attribute** in Context Menu.
   - Select **Model | Add | Operation** in Menu Bar or **Add | Operation** in Context Menu.
   - Select **Model | Add | Parameter** in Menu Bar or **Add | Parameter** in Context Menu.  Select **Model | Add | DataType**in Menu Bar or **Add | Datatype** in Context Menu.
4. Create the class diagram for the project.
5. Explicitly denote the relation between classes by generalizations, associations, multiplicities and cardinalities.
6. Save the file.


Employer
+CompanyName: string
+industry: string
+postJob(job)
+viewApplications(job)
Recruiter
+Education: string
+Jobspecification: string
+searchCandidates(job)
+shortListCandidate(jobseeker, job)
JobSeeker
+Resume
+Skills
+Experience
+createProfile()
+uploadResume()
+applyForJob(job)
User
+UserID: int
+Username: string
+Password: string
+Email: string
Requirement
+requirementID: int
+Title: string
+Description: string
+Status: string
+Priority: string
+Deadline: date
+Created_by: User
+Assigned_to: User
+create()
+update()
+changeStatus()
+addComments()
+getComments()
Project
+Project_id: int
+Title: string
+Description: string
+Start_date: date
+End_date: date
+addRequirement()
+removeRequirement()
+getRequirement()
Change Request
+ID: int
+Description: string
+Submit()
+Approve()
+Reject()
Requirement Type
+AttachementID: int
+File_Name: string
+FileSize: int
+GetDetails()
+GetRequirements()
Comments
+Comment_id: int
+Content: string
+Created_at: datetime
+Author: User
+Requirement: string
+Create(content)
+update(content)
+delete()

| | UML- SEQUENCE AND COLLABORATION DIAGRAMS |
|---|---|

**AIM :**

To draw sequence and collaboration diagrams for the identified scenarios and the interaction between objects.

**INTERACTION DIAGRAM :**

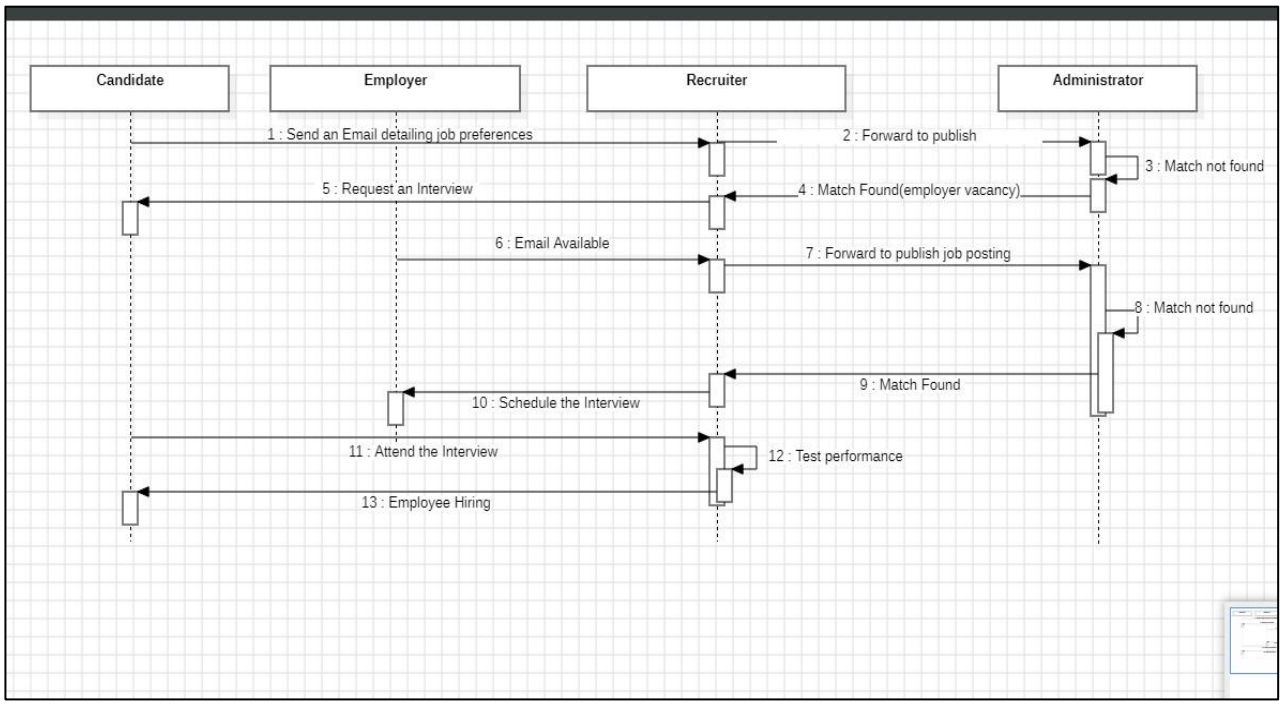
Sequence diagram describes the behavior of the system by viewing the interaction between the system and its environment. It represents the class at the top and their lifetime, their interactions as relations.

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).

**SEQUENCE DIAGRAM**

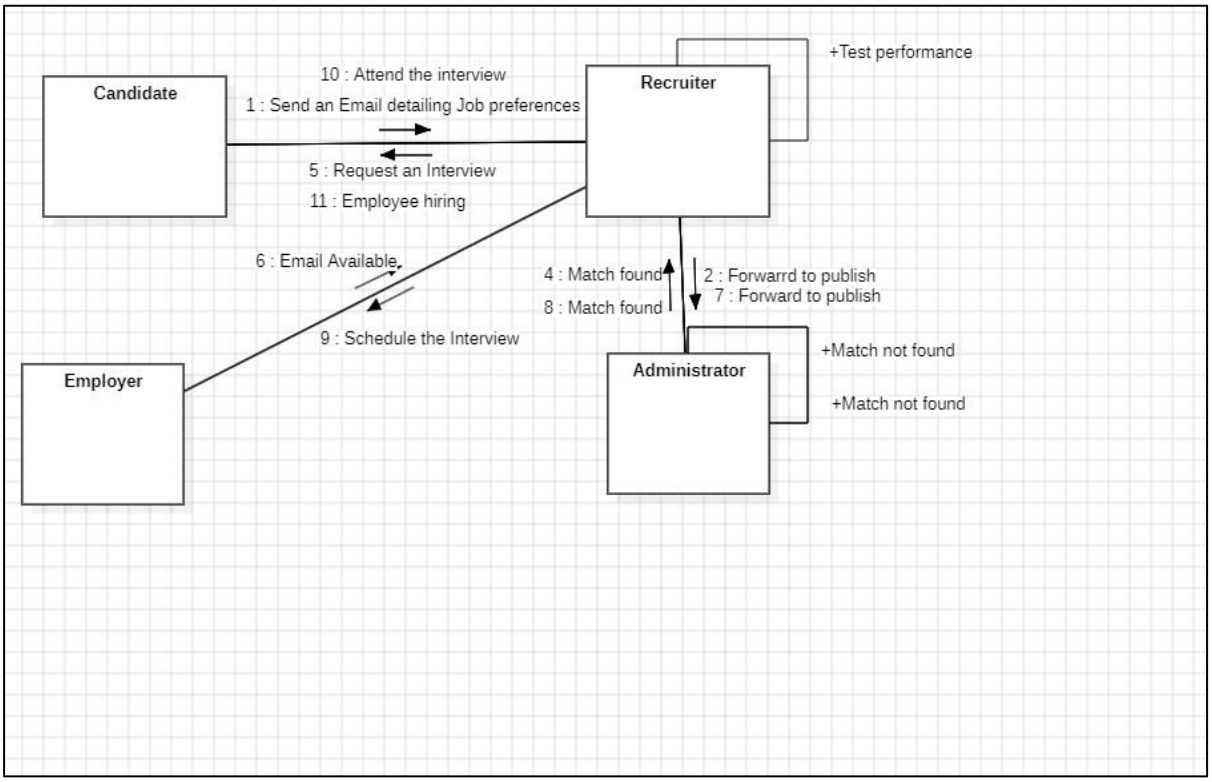
**PROCEDURE**

1. Identify the initiator of the process.
2. Associate each actor with a class whether it uses or provides services. 3. Each actor is represented as a rectangle.
4. The sequence of flow is denoted by the name of the operations to be done.
5. The actors are separated by vertical dashed lines and sequence flow is indicated through arrows.
6. Select Model | Add Diagram | Sequence Diagram in Menu Bar or select Add Diagram | Sequence Diagram in Context Menu in Star UML.
7. Create sequence diagrams, showing the interaction between the objects 8. Save the file.



## COLLABORATION DIAGRAM

1. This diagram is a numbered transition of sequence diagram in affixed order.
2. The transitions are denoted followed by the operation name.
3. Select **Model | Add Diagram | Communication Diagram** in Menu Bar or select **Add Diagram | Communication Diagram** in Context Menu in Star UML.
4. Create the collaboration diagram, showing numbered transition of sequence diagram between the objects
5. Save the file.




.

| | UML-STATE CHART DIAGRAM |
|---|---|

**AIM :**

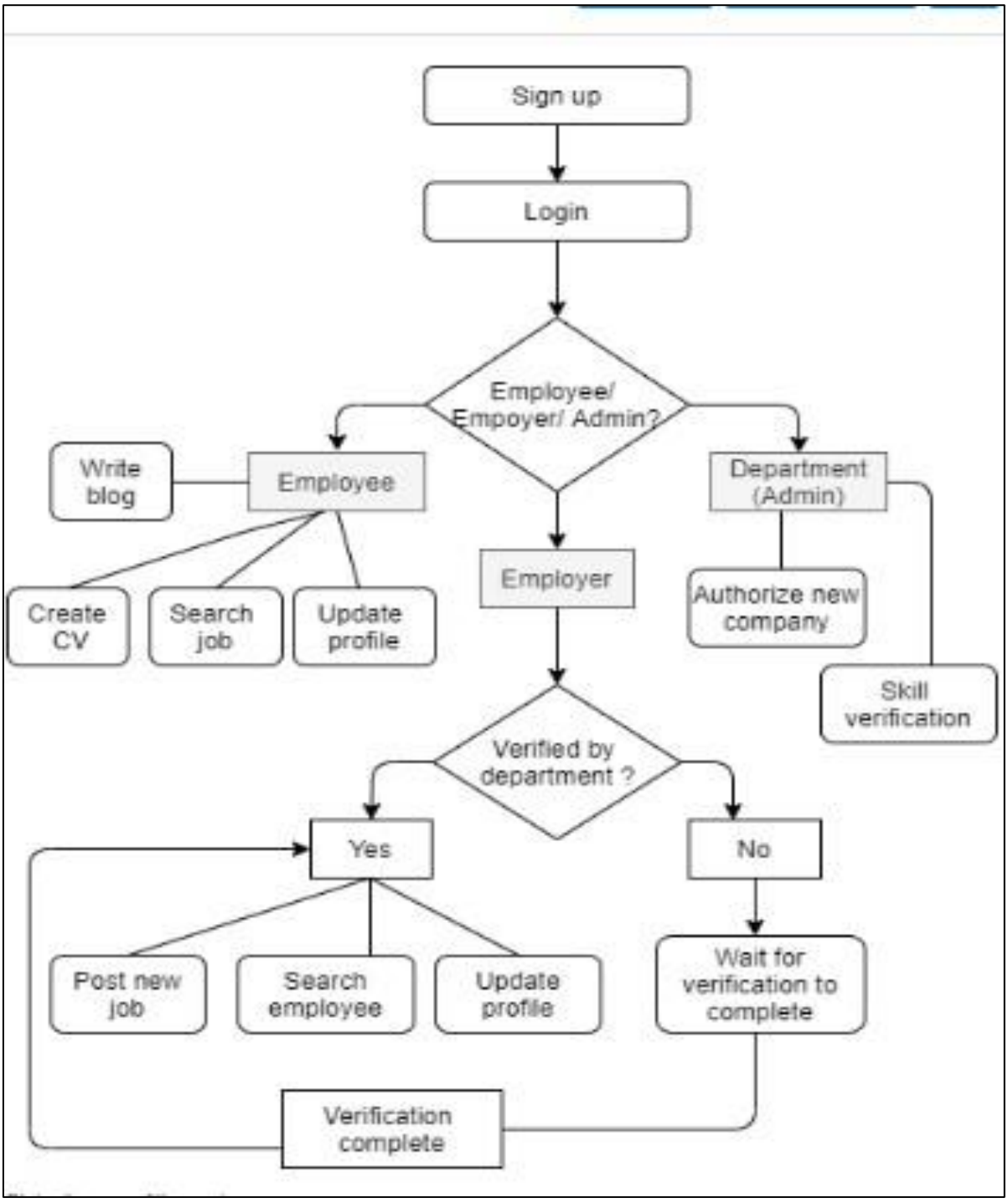
To draw the state chart diagram for the identified requirements

**STATE CHART DIAGRAM :**

A State chart diagram describes a state machine. A state machine can be defined as a machine which defines different states of an object, and these states are controlled by external or internal events. State chart diagram is used to describe the states of different objects in their life cycle. So, the emphasis is given on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately. States can be identified as the condition of objects when a particular event occurs. The state chart diagram shows the sequence of states that an object goes through during its life in response to outside message. Set of values that describe an object at a specific point in time.

**PROCEDURE**

1. Identify important objects to be analyzed.
2. Identify the states.
3. Identify the events.
4. Select **Model | Add Diagram | Statechart Diagram** in Menu Bar or select **Add Diagram | Statechart Diagram** in Context Menu.
5. Create state transition diagram for the project.
6. Use the tools and draw the overall diagram.
7. Save the file.


Sign up
Login
Employee/
Empoyer/ Admin?
Write
blog
Employee
Department
(Admin)
Employer
Create
CV
Search
job
Update
profile
Authorize new
company
Skill
verification
Verified by
department ?
Yes
No
Post new
job
Search
employee
Update
profile
Wait for
verification to
complete
Verification
complete

## UML-ACTIVITY DIAGRAM

**AIM :**

To draw activity the diagram for the identified requirements .

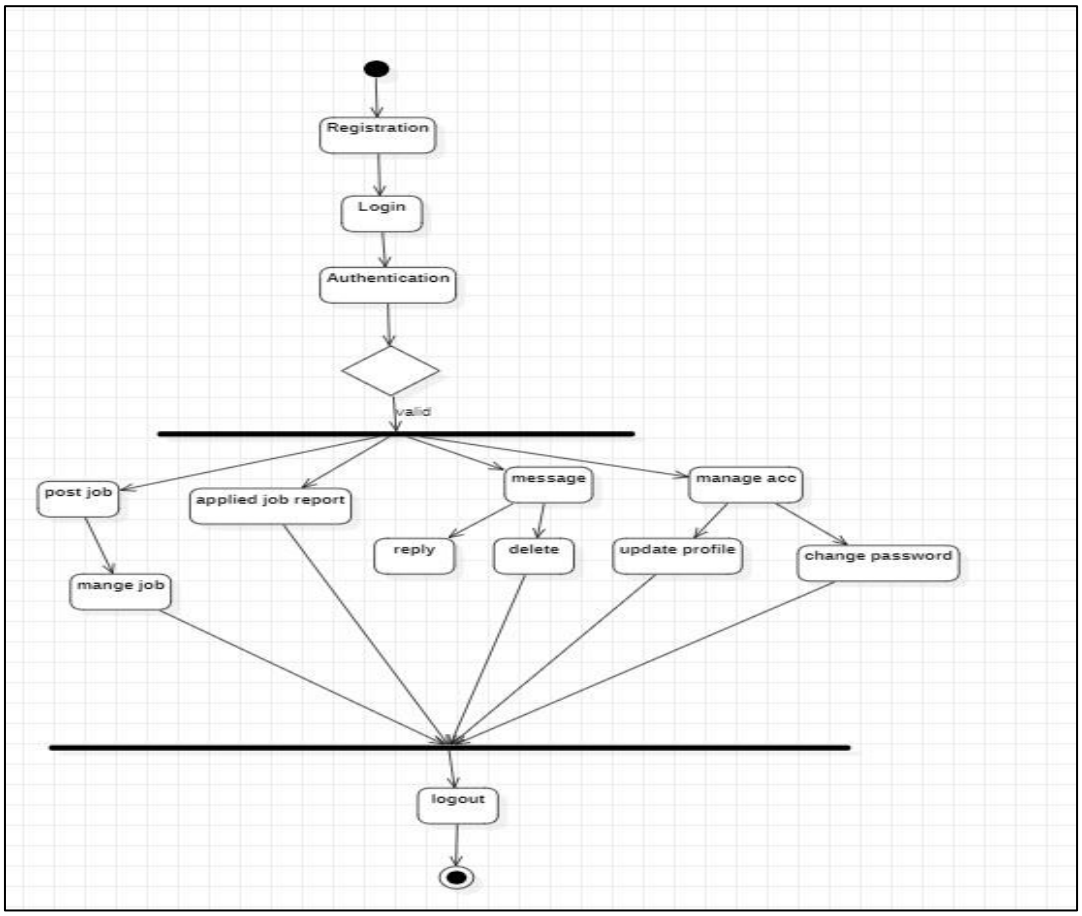
**ACTIVITY DIAGRAM :**

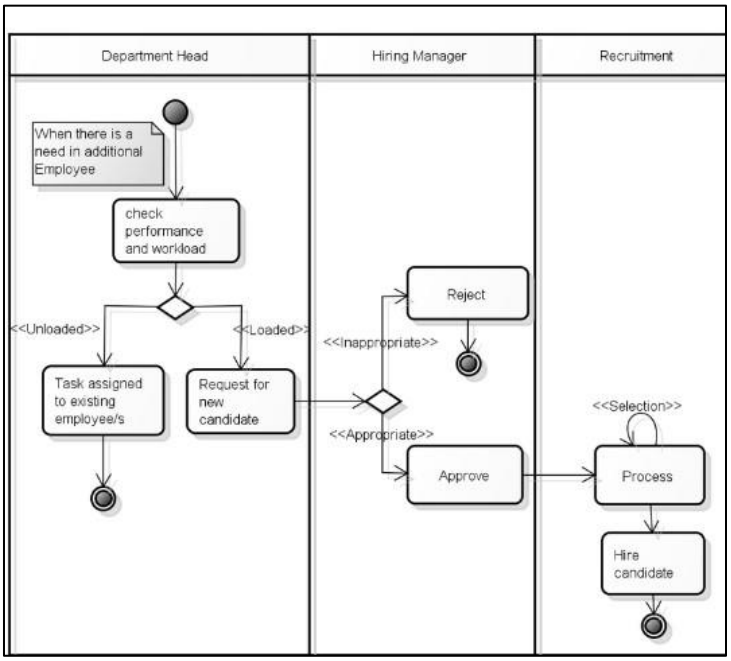
An activity diagram is a variation or special case of a state machine, in which the states are activities representing the performance of operations and the transitions are triggered by the completion of the operations. Unlike state diagrams that focus on the events occurring to a single object as it responds to messages, an activity diagram can be used to model an entire business process. The purpose of an activity diagram is to provide a view of flows and what is going on inside a use case or among several classes.

An activity is shown as a round box, containing the name of the operation. When an operation symbol appears within an activity diagram or other state diagram, it indicates the execution of the operation. Executing a particular step within the diagram represents a state within the execution of the overall method. It may be applied to any purpose such as visualizing the steps of a computer algorithm, but is considered especially useful for visualizing business workflows and processes, or use cases. Some of the outstanding notation includes parallel activities, swimlanes, and action-object flow relationship. An activity diagram allows the reader to see the system execution and how it changes direction based upon different conditions and stimuli.

**PROCEDURE**

1. Identify the activities, association, conditions and constraints.
2. Name the correct alternative types.
3. It is essential for the diagram to have a start and end points.
4. Select **Model | Add Diagram | Activity Diagram** in Menu Bar or select **Add Diagram | Activity Diagram** in Context Menu in Star UML.
5. Select activity diagram and create activity diagram for requirements of project using the tools.
6. Save the file.


Registration
Login
Authentication
valid
post job
applied job report
message
manage acc
reply
delete
update profile
change password
mange job
logout


Department Head
Hiring Manager
Recruitment
When there is a need in additional Employee
check performance and workload
<<Unloaded>>
<<Loaded>>
Task assigned to existing employee/s
Request for new candidate
Reject
<<Inappropriate>>
<<Appropriate>>
Approve
<<Selection>>
Process
Hire candidate

| | |
|---|---|
| | **Implement the system as per the detailed design** |

**Aim :**

To implement the Recruitment System with database connectivity.

**Source code :**

**Index.php :**

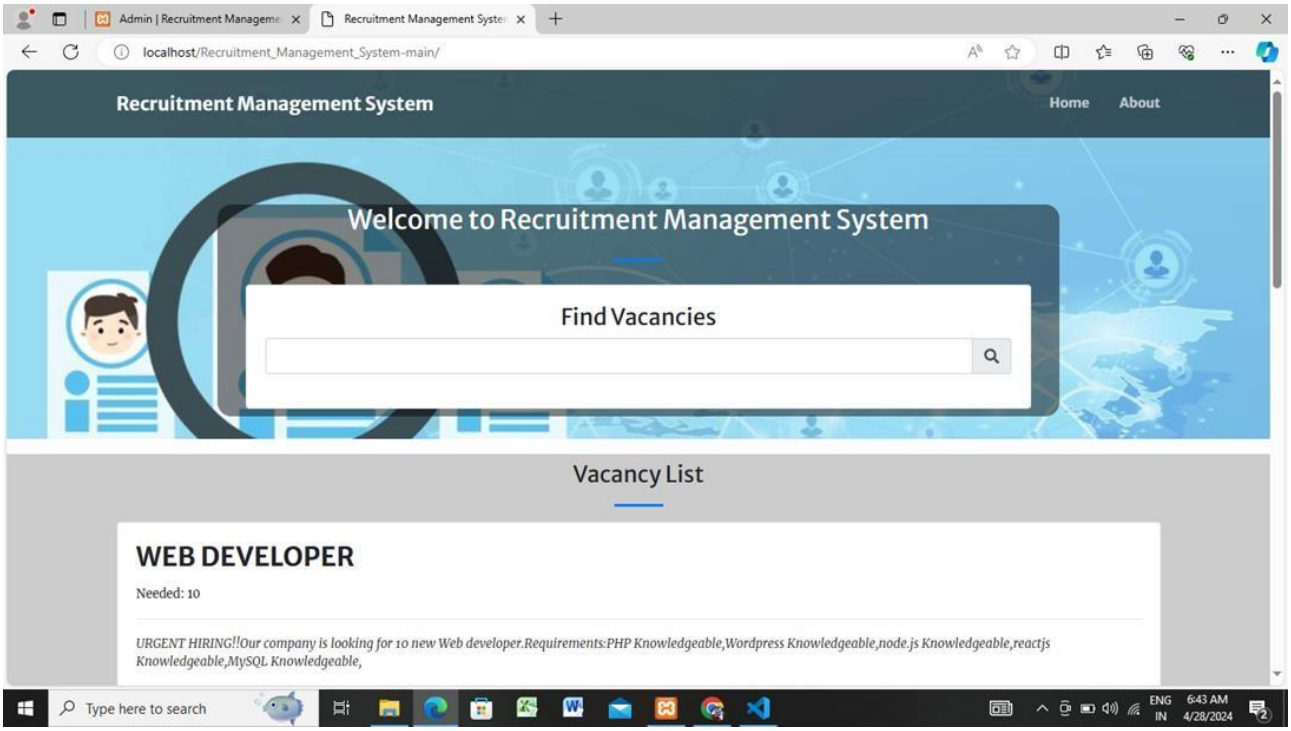
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta content="width=device-width, initial-scale=1.0" name="viewport">
<title>Admin | Recruitment system</title>
 <?php
 session_start();
if(!isset($_SESSION['login_id']))
header('location:login.php'); include('./header.php');
 // include('./auth.php');
 ?>
</head> <style>  body{
background: #80808045;
  }
  .modal-dialog.large { width:
80% !important; max-width:
unset;
  }
  .modal-dialog.mid-large { width:
50% !important; max-width:
unset;
  }
```

```
</style>
<body>
 <?php include 'topbar.php' ?>
 <?php include 'navbar.php' ?>
<div class="toast" id="alert_toast" role="alert" aria-live="assertive" aria-atomic="true">
<div class="toast-body text-white">
</div>
</div>
<main id="view-panel" >
<?php $page = isset($_GET['page']) ? $_GET['page'] :'home'; ?>
 <?php include $page.'.php' ?>
 </main>
<div id="preloader"></div>
<a href="#" class="back-to-top"><i class="icofont-simple-up"></i></a>
<div class="modal fade" id="confirm_modal" role='dialog'>
<div class="modal-dialog modal-md" role="document">
<div class="modal-content">
<div class="modal-header">
<h5 class="modal-title">Confirmation</h5>
</div>
<div class="modal-body">
<div id="delete_content"></div>
</div>
<div class="modal-footer">
<button type="button" class="btnbtn-primary" id='confirm' onclick="">Continue</button>
<button type="button" class="btnbtn-secondary" data-dismiss="modal">Close</button>
</div>
</div>
</div>
</div>
```

```
<div class="modal fade" id="uni_modal" role='dialog'>
<div class="modal-dialog modal-md" role="document">
<div class="modal-content">
<div class="modal-header">
<h5 class="modal-title"></h5>
</div>
<div class="modal-body">
</div>
<div class="modal-footer">
<button type="button" class="btnbtn-primary" id='submit' onclick="$('#uni_modal form').submit()">Save</button>
<button type="button" class="btnbtn-secondary" data-dismiss="modal">Cancel</button>
</div>
</div>
</div>
</div>
</body> <script>
 window.start_load = function(){
    $('body').prepend('<di id="preloader2"></di>')
  }
window.end_load = function(){
    $('#preloader2').fadeOut('fast', function() {
        $(this).remove();
      })
  }
window.uni_modal = function($title = '' , $url='',$size=""){
start_load()      $.ajax({ url:$url, error:err=>{ console.log()
alert("An error occured")
        },
success:function(resp){ if(resp){
```

```
            $('#uni_modal .modal-title').html($title)
$('#uni_modal  .modal-body').html(resp)  if($size
!= ''){
                $('#uni_modal .modal-dialog').addClass($size)
}else{
                $('#uni_modal              .modal-dialog').removeAttr("class").addClass("modal-dialog
modalmd")
            }
            $('#uni_modal').modal({
show:true, backdrop:'static',
keyboard:false, focus:true
            })
end_load()
          }
       }
   }) }
window._conf = function($msg='',$func='',$params = []){
    $('#confirm_modal #confirm').attr('onclick',$func+"("+$params.join(',')+")")
    $('#confirm_modal .modal-body').html($msg)
    $('#confirm_modal').modal('show')
 }
window.alert_toast= function($msg = 'TEST',$bg = 'success'){
     $('#alert_toast').removeClass('bg-success')
     $('#alert_toast').removeClass('bg-danger')
     $('#alert_toast').removeClass('bg-info')
     $('#alert_toast').removeClass('bg-warning')
if($bg == 'success')
     $('#alert_toast').addClass('bg-success')

if($bg == 'danger')
```

```
      $('#alert_toast').addClass('bg-danger') if($bg
== 'info')
      $('#alert_toast').addClass('bg-info') if($bg
== 'warning')
      $('#alert_toast').addClass('bg-warning')
    $('#alert_toast .toast-body').html($msg)
    $('#alert_toast').toast({delay:3000}).toast('show');

}

  $(document).ready(function(){
    $('#preloader').fadeOut('fast', function() {
        $(this).remove();
      })
  })

  $('.datetimepicker').datetimepicker({
format:'Y/m/d H:i', startDate: '+3d'
  })

  $('.select2').select2({
placeholder:"Please select here",
width: "100%"
  })

</script>
</html>
```




**Login.php :**

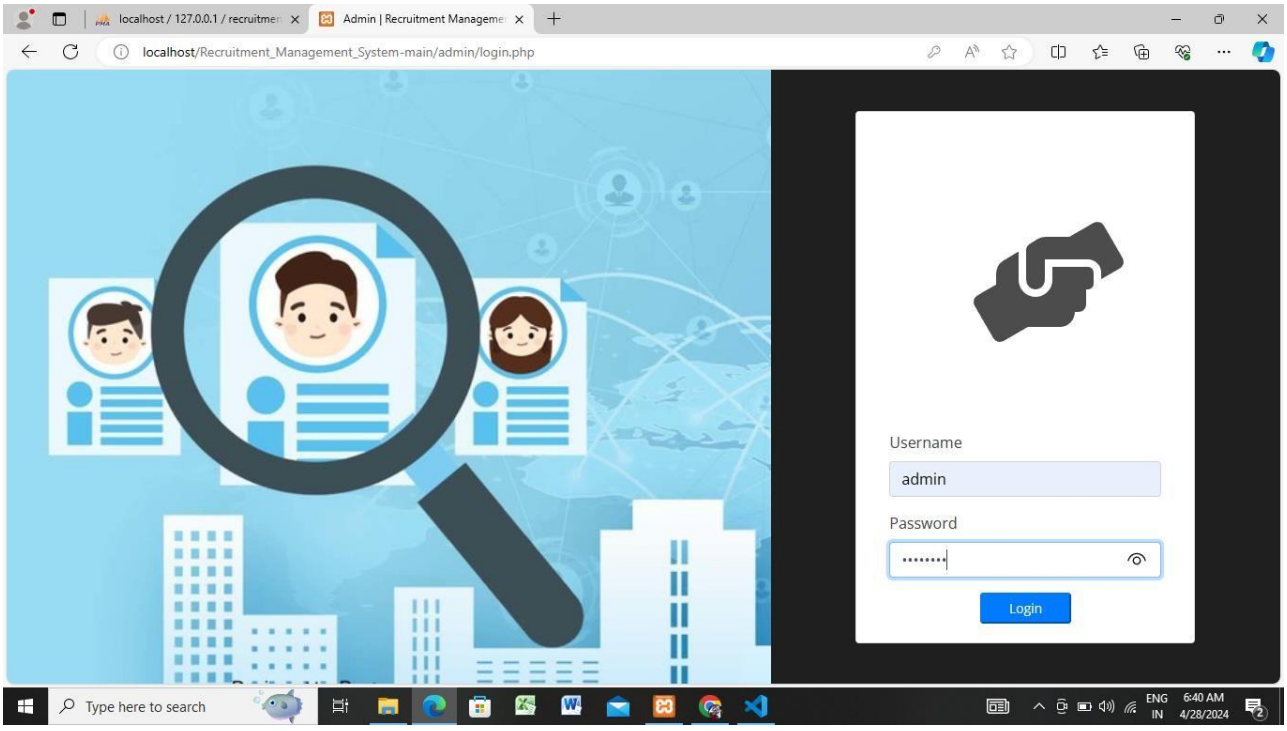
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta content="width=device-width, initial-scale=1.0" name="viewport">
<title>Admin | Recruitment system</title>



<?php include('./header.php'); ?>
<?php include('./db_connect.php'); ?>
<?php
session_start();
if(isset($_SESSION['login_id'])) header("location:index.php?page=home");
?>
</head>
<style>
 body{
```

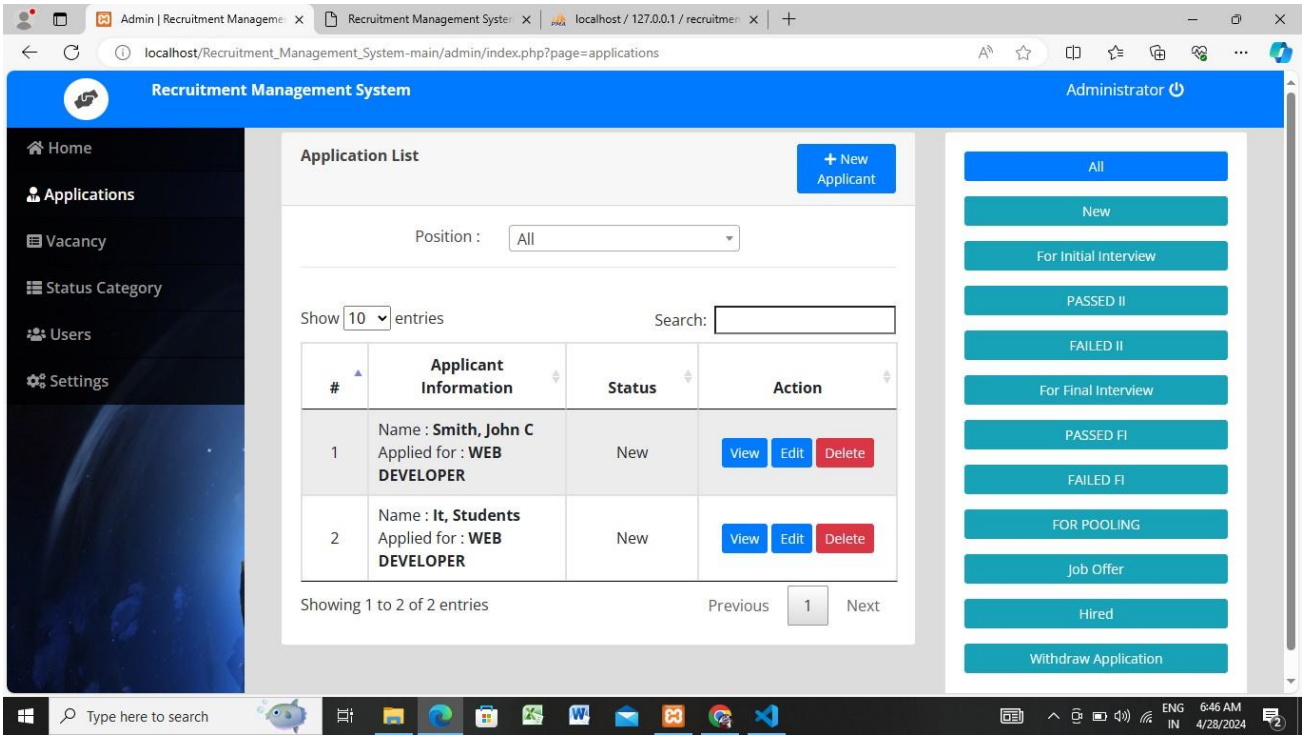
```
        width: 100%;
 height: calc(100%);
    /*background: #007bff;*/
 }  main#main{
   width:100%;   height:
calc(100%);
   background:white;
 }
 #login-right{     position:
absolute;          right:0;
   width:40%;
  height:          calc(100%);
background:white;    display:
flex;   align-items: center;
 }
 #login-left{
   position: absolute;
   left:0;
   width:60%;
  height:                calc(100%);
background:#59b6ec61;   display:
flex;   align-items: center;
   background: url(assets/img/recruitment-cover.jpg);
background-repeat: no-repeat;  background-size: cover;  }
 #login-right .card{
        margin: auto; z-index:
        1
 }  .logo { margin: auto;
font-size: 8rem;
background: white;
```

```
padding: .5em 0.7em;
border-radius: 50% 50%;
color: #000000b3; z-
index: 10; } div#login-
right::before { content:
""; position: absolute;
top: 0; left: 0;
width: calc(100%); height:
calc(100%); background:
#000000e0;
}

</style>
<body>
<main id="main" class=" bg-dark">
<div id="login-left">
</div>
<div id="login-right">
<div class="card col-md-8">
<div class="card-body">
<div class="logo">
<span class="fafa-hands-helping"></span> </div>
<form id="login-form" >
<div class="form-group">
<label for="username" class="control-label">Username</label>
<input type="text" id="username" name="username" class="form-control">
</div>
<div class="form-group">
<label for="password" class="control-label">Password</label>
<input type="password" id="password" name="password" class="form-control">
```

```
</div>
<center><button class="btn-smbtn-block btn-wave col-md-4 btnprimary">Login</button></center>
</form>
</div>
</div>
</div>
</main>
<a href="#" class="back-to-top"><i class="icofont-simple-up"></i></a>
</body>
<script>
 $('#login-form').submit(function(e){
        e.preventDefault()
  $('#login-form button[type="button"]').attr('disabled',true).html('Logging in...');
if($(this).find('.alert-danger').length > 0 )
        $(this).find('.alert-danger').remove();
        $.ajax({
        url:'ajax.php?action=login',
   method:'POST',
        data:$(this).serialize(),
   error:err=>{
   console.log(err)
        $('#login-form button[type="button"]').removeAttr('disabled').html('Login');
        },
        success:function(resp){
                if(resp == 1){
                        location.href ='index.php?page=home';
    }else if(resp == 2){      location.href ='voting.php';
                }else{
                        $('#login-form').prepend('<div class="alert alert-
danger">Username or password is incorrect.</div>')
```

```
                                    $('#login-form
button[type="button"]').removeAttr('disabled').html('Login');
                                }}
            })})
</script>
</html>
```

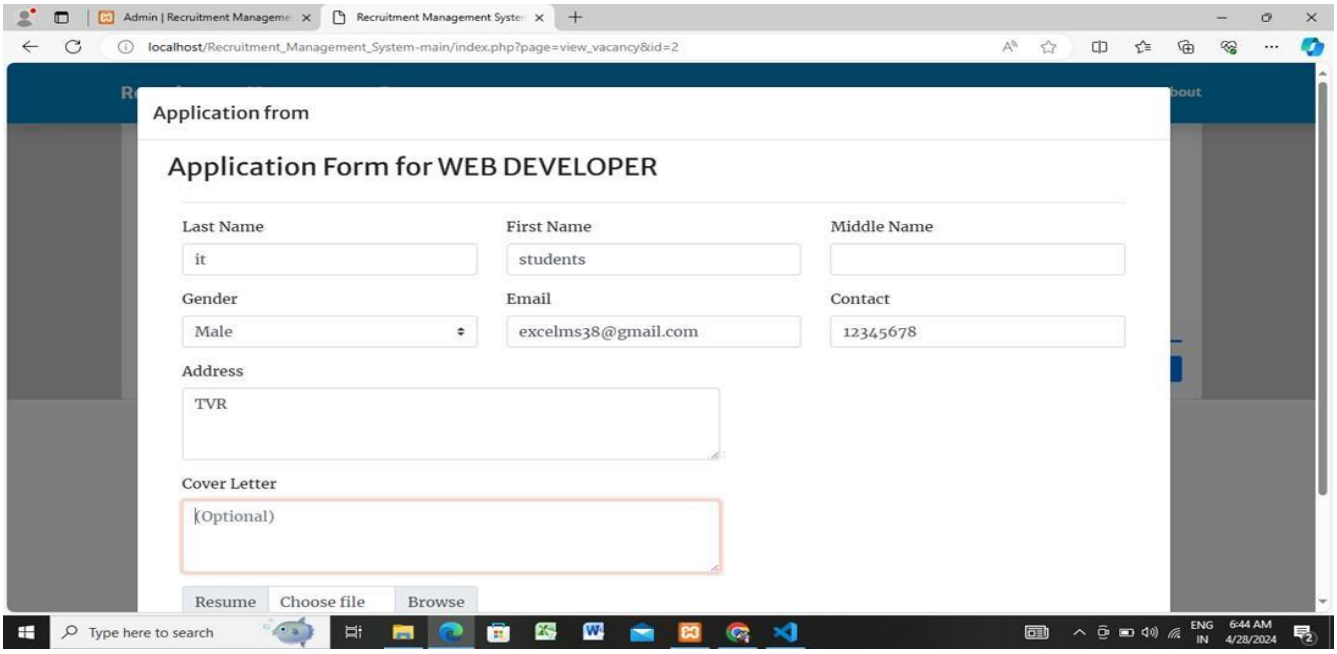





**Application.php :**

```
<?php
include 'db_connect.php';
$application = $conn->query("SELECT  a.*,v.position FROM application a inner join vacancy v on v.id = a.position_id where a.id=".$_GET['id'])->fetch_array(); foreach($application as $k => $v){
 $$k = $v;
}
      $fname = explode('_',$resume_path); unset($fname[0]);
      $fname = implode("",$fname);
?>
<div class="container-fluid">
 <div class="col-md-12">
            <p><b>Applied for :</b><?php echo $position ?></p>
   <p><b>Name :</b><?php echo ucwords($lastname.', '.$firstname.' '.$middlename) ?></p>
            <p><b>Gender :</b><?php echo $gender ?></p>
            <p><b>Address :</b><?php echo $address ?></p>
            <p><b>Email :</b><?php echo $email ?></p>
```

```
<p><b>Cover Letter :</p>
<hr>
<?php echo !empty($cover_letter) ? str_replace("\n","<br>",html_entity_decode($cover_letter)) : 'None'; ?>
<hr>
<p><b>Resume</p><a href="download.php?id=<?php echo $_GET['id'] ?>" target="_blank"><?php echo $fname ?></a>
</div></div>
```



| | |
|---|---|
| | **Test the software system for all the scenarios identified as per the usecase diagram** |

**Aim :**

To test the software system for all the scenarios identified as per the use case diagram.

**Testing :**

**Black box testing:**

Black box testing, also known as basic testing or specification-based testing, is a software testing method that focuses on an application's functionality without looking at its internal structure or workings.

❖ **Decision table testing :**

Decision table testing is a black box test design technique that uses decision tables to design test cases. A decision table is a visual representation of the actions to perform based on given conditions or inputs.

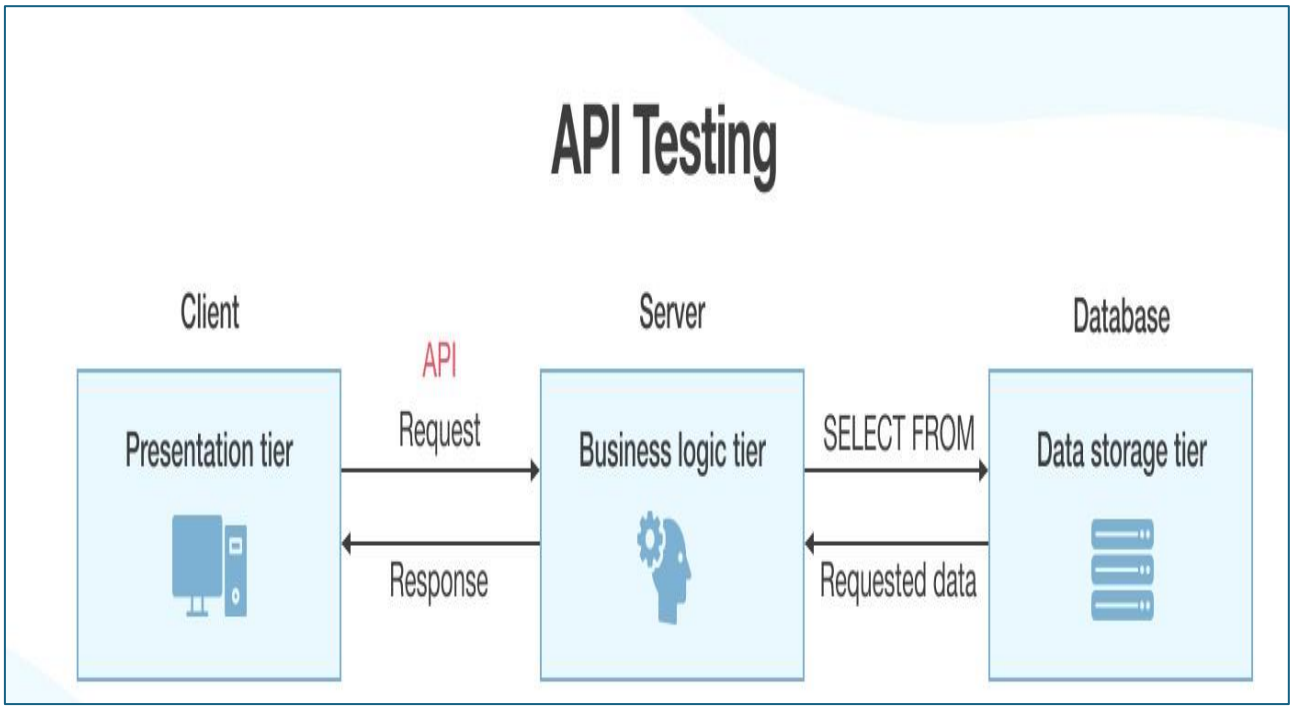
| | | | | |
|---|---|---|---|---|
| Email(condition1) | T | T | F | F |
| Password(condition2) | T | T | F | F |
| Expected (action) | Account page | Incorrect password | Incorrect email | Incorrect email |

**Grey box Testing :**

Grey box testing is a software testing method that blends black box and white box testing. Testers have some internal knowledge (like database structure or APIs used) to design better test cases, but not the full code picture. This helps find defects in areas where functionality meets internal structure.

**API testing** :

API testing is software testing that aims to validate the APIs. API testing aims to check APIs' reliability, performance, security, and functionality.


API Testing
Client
Server
Database
API
Presentation tier
Request
Business logic tier
SELECT FROM
Data storage tier
Response
Requested data

| | |
|---|---|
| | **Improve the reusability and maintainability of the software system by applying appropriate design patterns test it for various scenarios** |

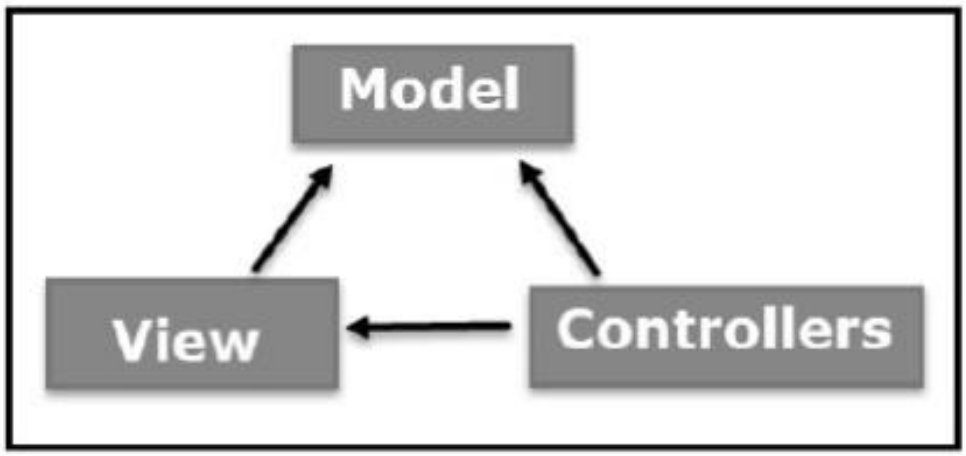
**Aim :**

To Improve the reusability and maintainability of the software system by applying appropriate design patterns and test it for various scenarios.

**Design Pattern- MVC**

MVC-description

The MVC framework separates an application into three main components: models, views, and controllers. Models are used to store data and business logic; views display data from a model on the screen; and controllers manage user interactions with these two components.




**Model**

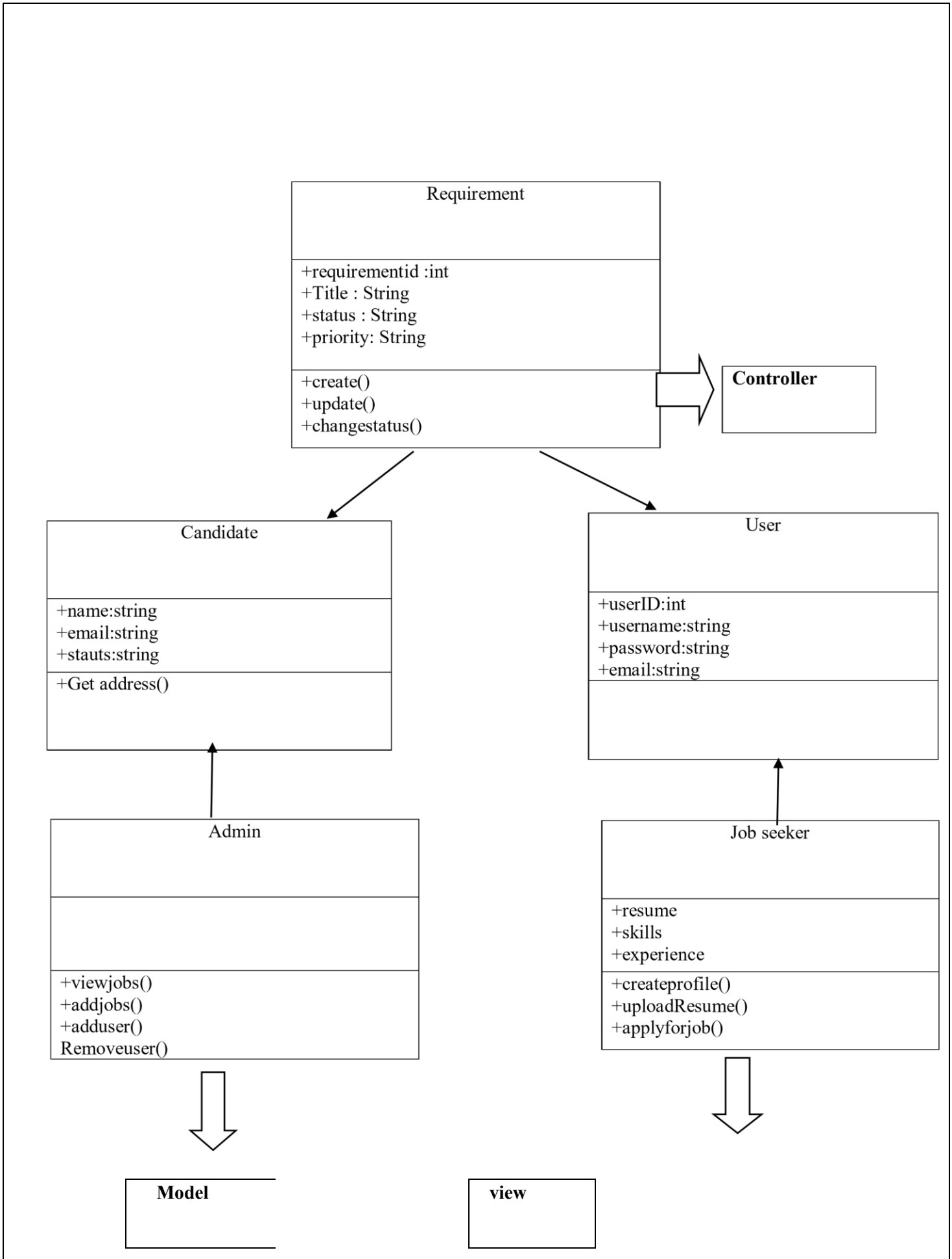
The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

**View**

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

**Controller**

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.


Requirement
+requirementid :int
+Title : String
+status : String
+priority: String
+create()
+update()
+changestatus()
Controller
Candidate
+name:string
+email:string
+stauts:string
+Get address()
User
+userID:int
+username:string
+password:string
+email:string
Admin
+viewjobs()
+addjobs()
+adduser()
Removeuser()
Job seeker
+resume
+skills
+experience
+createprofile()
+uploadResume()
+applyforjob()
Model
view

| | Data Flow Diagram |
|---|---|

**Aim :**

To identify Data flow diagram and develop a DFD modelwith Star UML.

**Data flow diagram :**

- A graphical tool, useful for communicating with users, managers, and other personnel .
- Used to perform structured analysis to determine logical requirements .
- Useful for analyzing existing as well as proposed systems.
- Focus on the movement of data between external entities and processes, and between processes and data stores .
- A relatively simple technique to learn and use.

## Procedure:

**Step : 1**
**Open StarUML 5.0**: Launch the StarUML application on your computer.

**Step : 2**
**Create a New Project**: Start a new project by clicking on "File" > "New" > "Project" or by using the shortcut Ctrl + N.

**Step : 3**
**Add a Data Flow Diagram**: Once the project is created, go to "Model" > "Add Diagram" > "Data Flow Diagram".

**Step : 4**
**Name the Diagram**: Give your diagram a name, like " Recruitment System DFD".

**Step : 5**
**Identify External Entities**: Identify the external entities involved in the recruitment process. These could include job applicants, recruiters, and the HR department.

**Step : 6**
**Add Processes**: Identify the main processes involved in the recruitment process, such as "Receive Application", "Screen Candidates", "Schedule Interviews", "Conduct Interviews", and "Offer Job".

**Step : 7**
**Add Data Stores**: Identify where data is stored during the process. This could include databases for storing applicant information, resumes, and interview feedback.

**Step : 8**
**Draw Data Flows**: Connect the processes, external entities, and data stores using data flows. Data flows represent the flow of information between different elements of the system.

**Step : 9**
**Define Data Flow Names**: Label the data flows with meaningful names to describe the information being passed between elements.

**Step : 10**

**Save Your Work:** Once you're satisfied with the diagram, save your work by clicking on "File" > "Save" or using the shortcut Ctrl + S.

**DFD for Recruitment System :**