UCS 1617 – MINI PROJECT

ONLINE RECRUITMENT SYSTEM

TEAM MEMBERS:-

Prathyush S – 185001112
Praveen Kumar V – 185001113
Ramaprabha R – 185001123

PROBLEM STATEMENT:

Recruitment system is a process of selecting potential candidates for a vacant position and hiring the candidates who will fulfill the requirement of the organization .It is an integrated platform which eases the process of Recruitment. It is a robust system which enables HR of the company to do processes like posting new jobs, finding efficient candidates and hiring them. The job seekers can find the company which can match their interest and apply for them. The organization can shortlist the applications as per their requirements. The HR department selects the candidates based on their resume and the relevant skill set and hires them. This system reduces the time consumed in hiring process for both job seekers and organizations.

SOFTWARE REQUIREMENTS SPECIFICATION:

INTRODUCTION:-

This Project is to help jobseeker to look for job. To help company HR team to look for candidate for vacant positions in their respective company.

Now a days there are many graduates who are seeking for jobs but they may not know the new trends of companies and their hiring. Most companies require employees but they also can't find the appropriate employee. Even though the candidate has good knowledge, they are unable to get the job they deserve. To recruit qualified candidate for the proper job, we have designed this online Recruitment System.

PURPOSE:-

The purpose of Software Requirements Specification (SRS) document is to describe the external behavior of the Online Recruitment System. Requirements Specification defines and describes the operations, interfaces, performance, and quality assurance requirements of the Online Recruitment System. The document also describes the nonfunctional requirements such as the user interfaces.

It also describes the design constraints and other factors necessary to provide a complete and comprehensive description of the requirements for the software. The Software Requirements Specification (SRS) captures the software requirements for the system.

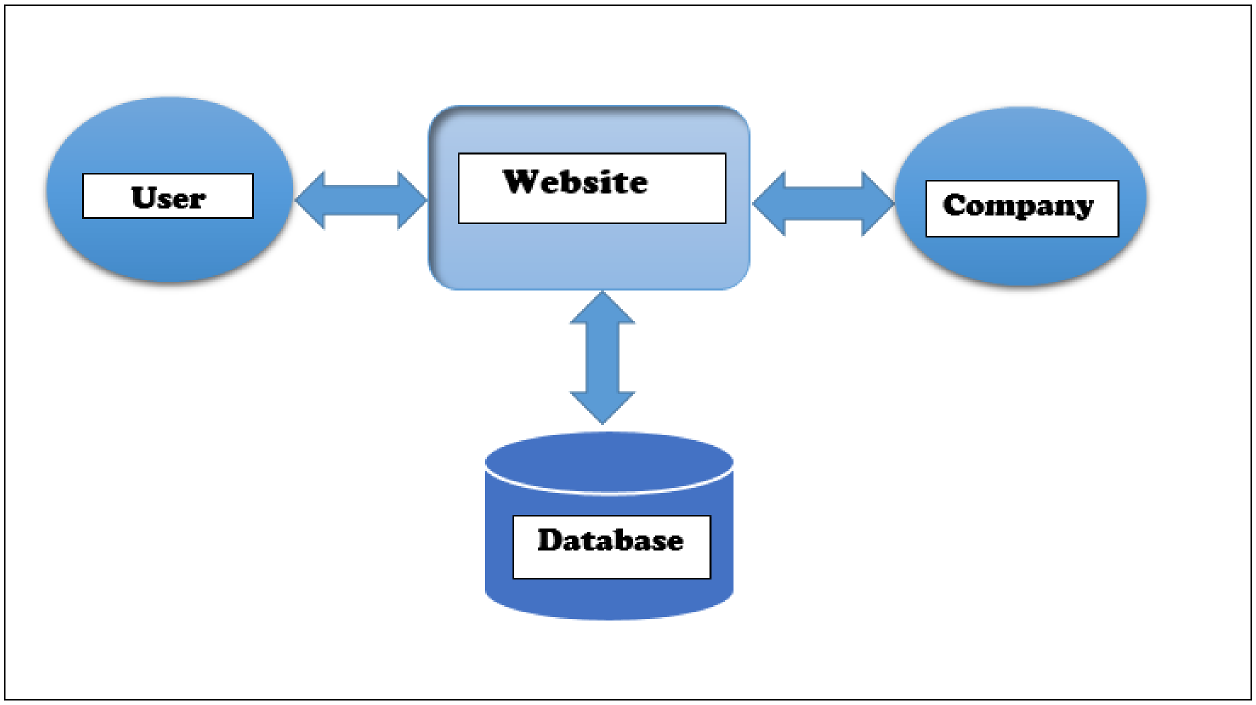
SCOPE:-

- ✓ To provide the information about the Job seekers to the Job providers/Employers. Company can get the information about the proper candidate according to its requirement.
- ✓ According to the requirement of the Companies, provide the information of Job seekers to the companies.
- ✓ Inform the Job seekers for the vacancies in the Company/Interview session to the Companies.
- ✓ To provide the information to the Job seekers about the new trends in the Industrial Area.
- ✓ This way of Recruitment System saves more time for both the parties.

OVERALL DESCRIPTION:-

PRODUCT PERSPECTIVE:-

- ✓ User login (Job seekers) & HR login (Job Providers)
- ✓ User should be able to see the number of Companies according to their Skills and Place.
- ✓ Companies should update available positions (Vacancies) for the User.
- ✓ User should be able to apply and upload his resume and have a sample test.
- ✓ Companies should be able to view their test scores and resume, select an appropriate Employee.
- ✓ Help Menu



PRODUCT FUNCTIONS:-

- ✓ Candidate should finish his/her degree and should have proper certificate while applying for job and the candidate should satisfy the eligibility criteria.
- ✓ The candidate must be able to search for the jobs based on skills and Location.
- ✓ The HR has the full permission to select or reject the candidate for interview.
- ✓ If the candidate is selected for the particular job he will receive the offer letter.
- ✓ HR and Candidate plays a major role in the recruitment process.
- ✓ The necessary details of the candidate have to be submitted to the HR. Verification has been done by the HR.
- ✓ HR must be able to view all possible applied candidates for the job.
- ✓ Selected candidate should be call to Interview.
- ✓ HR intimates the selected candidate. Finally Candidates accept or reject his offer letter.

USER CHARACTERISTICS:-

- ✓ The users of the system are Candidate (or) Job seekers and HR (or) Job providers.
- ✓ The candidates are allowed to login with username and password signed up by them and according to their skills and location available they apply for the job.
- ✓ The HR has a login to check the candidates who have applied and filter them based on the job vacancies.

CONSTRAINTS:-

- ✓ The website is accessed both by Job seeker and provider with separate login ID and password with known only by the individual.
- ✓ Candidates details ie., the username and details about his skills are visible to the HR.
- ✓ Details about the company and location without the HR name is shown to the Candidate.
- ✓ If candidate is selected for the particular job means all details about them will be added to the company database and the modification can also be done.

ASSUMPTIONS AND DEPENDENCIES:-

- ✓ The candidate and HR should have sufficient knowledge of computers.
- ✓ Both the candidate and HR should have Internet connection and Internet server capabilities.
- ✓ The candidates and HR must know the English language, as the user interface will be provided in English
- ✓ The HR can only access the database and the candidate cannot.

SPECIFIC REQUIREMENTS:-

This includes all the functional requirements:-

FUNCTIONALITY:-

1. **Logon Capabilities**

The system shall provide both the candidates and HR with logon capabilities.

2. **Mobile Devices**

The System is also supported on mobile devices such as cell phones.

3. **Alerts**

The system can alert the candidate or the HR in case of any problems to check the Help Menu that is provided.

USABILITY

- ✓ The system shall allow the users to access the website from the Internet using derivative technologies. The system uses a web browser as an interface.
- ✓ Since all users are familiar with the general usage of browsers, no specific training is required.
- ✓ The system is user friendly and self-explanatory.

RELIABILITY

- ✓ The system has to be very reliable due to the importance of data and the damages incorrect or incomplete data can do.

1. **Availability**

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

2. **Mean Time Between Failures (MTBF)**
The system will be developed in such a way that it may fail once in a year for maintenance purpose.

3. **Mean Time to Repair (MTTR)**
Even if the system fails, the system will be recovered back up within an hour or less.

4. **Accuracy**
The accuracy of the system is limited by the accuracy of the speed at which the Candidates or HR use the system.

5. **Access Reliability**
The system shall provide 100% access reliability.

PERFORMANCE

1. **Response Time**
The Information page must get downloaded within a minute, if proper Internet connection is present. The information is refreshed every two minutes. The access time for a mobile device should be less than a minute compared to the computer. The system shall respond to the member in not less than two seconds from the time of the request. The system shall be allowed to take more time when doing large processing jobs such as updating your details to the database.

2. **HR Response**
The system shall take as less time as possible to provide service to the Candidate after viewing the following Details.

3. **Throughput**
The number of details getting updated in the database is directly dependent on the number of users ie., Candidate or employees of the Company (HR) and Help Desk.

4. **Capacity**
The system is capable of handling 1000 users at a time.

5. **Resource Utilization**
   The resources are modified according the user requirements and also according to the available Jobs based on their skill and Location requested by the users.

DESIGN CONSTRAINTS

1. **Software Language Used**
   The languages that shall be used for coding the Recruitment System are SQL, CSS, Python, HTML, JavaScript.

2. **Development Tools**
   Will make use of the available Python Development Tool kits like Pycharm and Visual Studio Code. Also we will make use of Oracle, MySQL for database connectivity.

3. **Class Libraries**
   Will make use of the existing Javascript libraries for animation, bootstrap libraries for front end, python libraries for back end and database connectivity.

INTERFACES

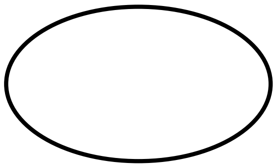
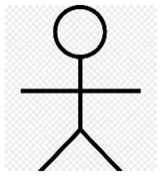
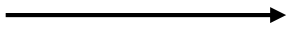
1. **User Interfaces**
   Will make use of the existing Web Browsers such as Microsoft Internet Explorer, Google chrome or Firefox.

2. **Hardware Interfaces**
   The hardware interfaces Computer Laptops and Mobile phones are used for Internet Connectivity.

3. **Software Interfaces**
   SQL, Heroku for hosting web app.

4. **Communications Interfaces**
   The Recruitment System will be connected to the World Wide Web.

USE CASE DIAGRAM

Name	Diagram	Description
Use cases		Horizontally shaped ovals that represent the different uses that a user might have.
Actors		Stick figures that represent the people actually employing the use cases.
Associations		A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
System boundary boxes		A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system.

Identification of Actors:

- ✓ Candidate
- ✓ HR
- ✓ Company Owner
- ✓ Help Desk

Identification of Scenarios:

Main Success Scenario:

- ✓ Register
- ✓ Authenticate User
- ✓ Create Company
- ✓ Delete Company
- ✓ Posting a Job
- ✓ Search Job
- ✓ Apply for the Job
- ✓ Check applied candidates and their details
- ✓ Upload Resume
- ✓ Create Test Questions
- ✓ Take the Test
- ✓ Check test marks
- ✓ Select a Candidate
- ✓ Confirm offered position
- ✓ Check Company Details
- ✓ Raise a complaint
- ✓ Resolve a complaint

Failure Scenario/Alternate Flow:

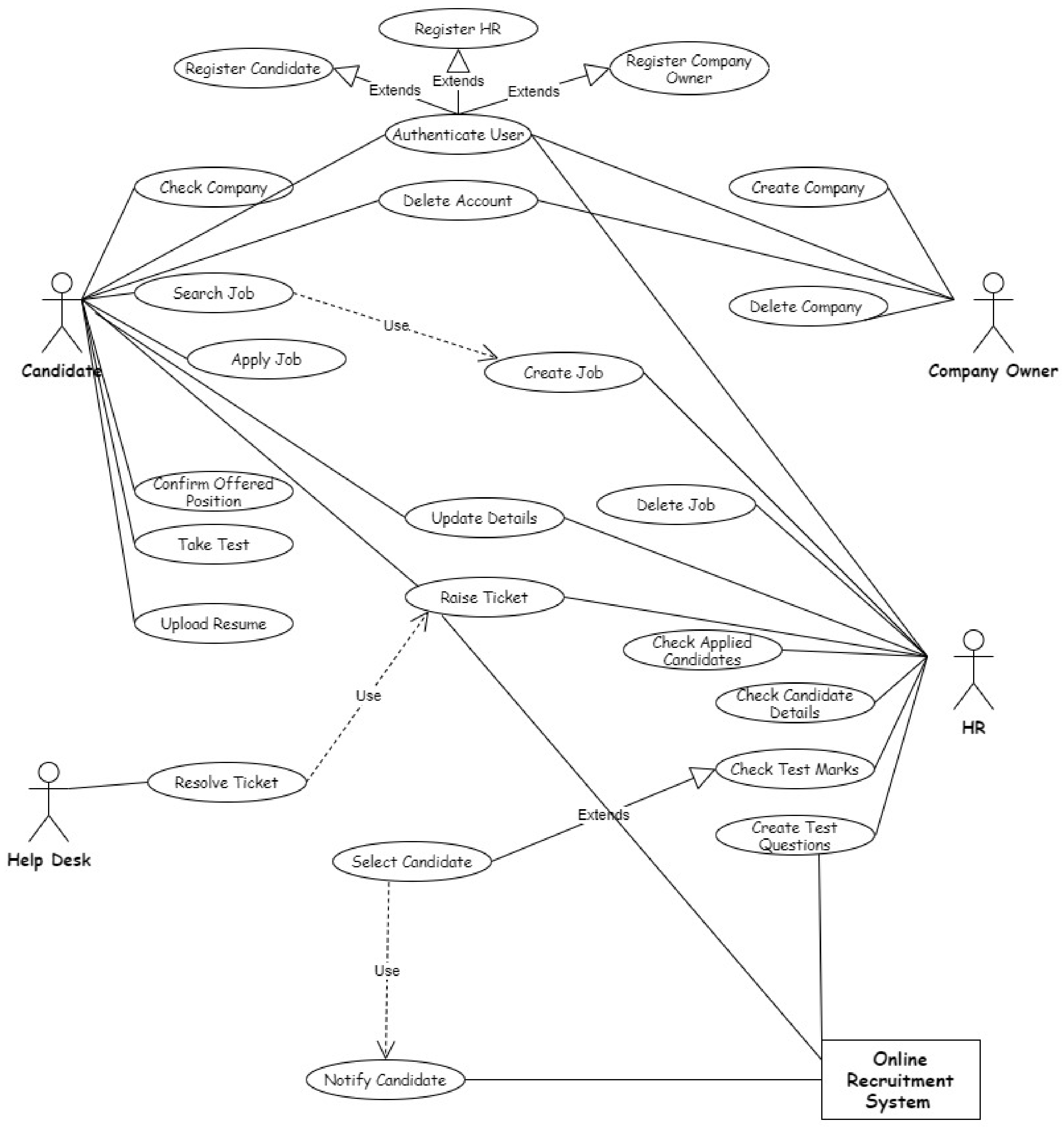
- ✓ Invalid Details
- ✓ Invalid User credentials
- ✓ System Failure
- ✓ Job Vacancy Not found
- ✓ Invalid Job Details
- ✓ Function failure

Subfunction:- (Selection Process)

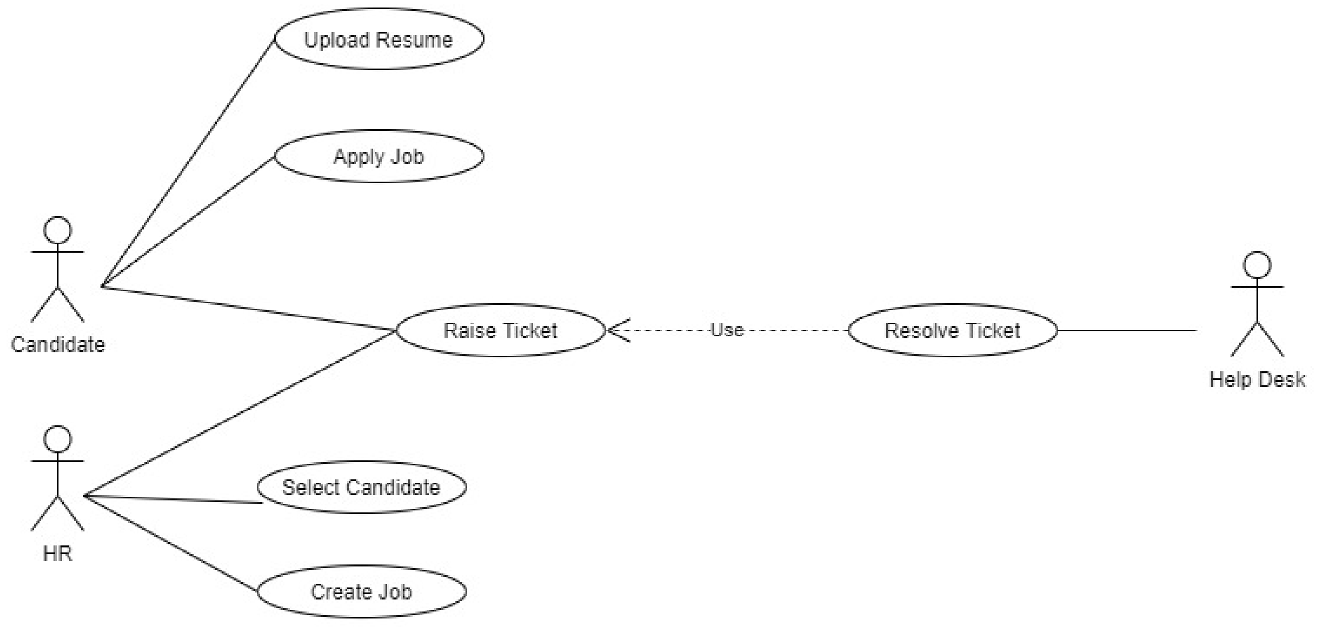
- ✓ Checking Applied candidates
- ✓ Verifying Candidate Details
- ✓ Filtering Candidates
- ✓ Select required Candidates

USE CASE DIAGRAM:-

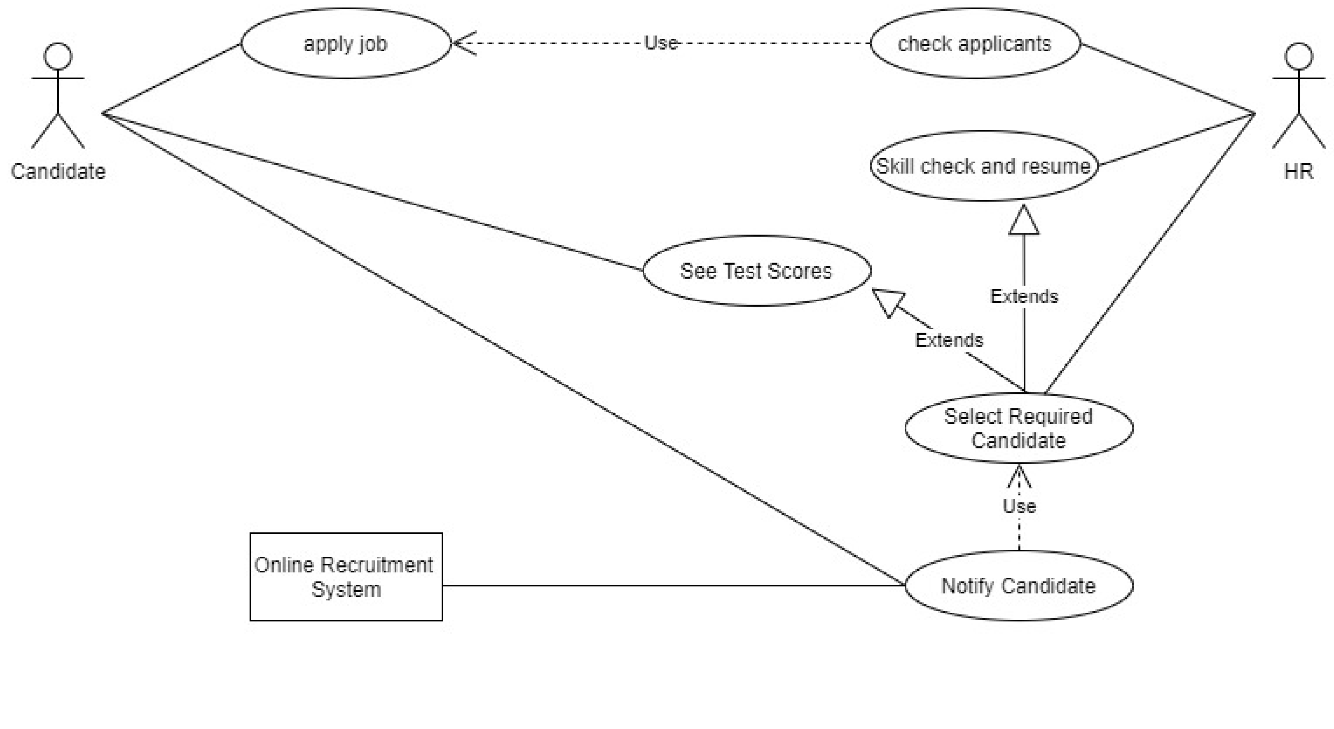
MAIN CASE SCENARIO:-



FAILURE SCENARIO:-



SUBFUNCTION (SELECTION PROCESS):-



Fully dressed Usecase Description :

a. **Main success scenario:**

- ✓ Candidate or HR or Company owner can signup with new credentials
- ✓ Previously signed up users can log in using his own credentials
- ✓ Company Owners can enroll valid companies into the system with appropriate details.
- ✓ Company Owners can withdraw his company from the Recruitment System.
- ✓ HR people from a specific Company can post a new job with number of vacancies, skill sets and Location.
- ✓ Candidate can search for specific job using filters such as Company name, Location and Skill set.
- ✓ If the Candidate finds an ideal job, he can enroll for the Recruitment Process.
- ✓ HR can check the Candidates applied for the Jobs.
- ✓ Candidates can update their details and also upload their new Resume whenever they want.
- ✓ HR creates the test questions with answers.
- ✓ Candidate take the test given by the Company.
- ✓ HR in the company examines the test marks and selects a Candidate
- ✓ Confirmation is sent to the Candidate by the HR which the candidate can accept or reject.
- ✓ Candidate can check the Company Details

b. **Failure Scenario:-**

- ✓ System notifies the user, records the error and enters a clean state. User can register again
- ✓ If the Log In details are given wrong, the system requests for the correct details again. If the User forgets the password he can reset the password using security question.
- ✓ Restarts the system, logs in, and requests recovery of prior state.

- ✓ If the Job short listed by the user has been filled before applying. The system should exit from the page and return to the home Page.
- ✓ If the posted details are wrong the HR can change them and it will be reflected to all users.
- ✓ If any of the Users are not able to do the success case functions he should be able to raise a complaint to the help desk.
- ✓ The complaints received by the help desk must be resolved by the team and reported back to the user

c. **<u>Subfunction (Selection Process):-</u>**

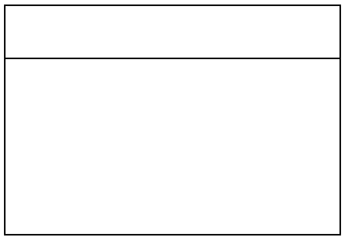
- ✓ HR checks the candidates applied for the job.
- ✓ HR verifies the Candidate's Personal details, educational details, work experience, project details, relevant Skill sets and their Resume.
- ✓ HR can filter out the Candidates based on their test marks and/or his details.
- ✓ If the job provider is satisfied with the candidate's performance he can select the candidate.

Documentation:

This use case is illustrative and exhaustive and provides a very good visual description and detail of the system. The use case brings forward the various functionalities of the system along with the alternative flows and stresses upon the important sub function of the system also. Summarizing this use case diagram gives a lucid and vivid description of the system and its functionalities.

UML DOMAIN MODEL AND CLASS DIAGRAM

UML notations for Domain model diagram:

Name	Diagram	Description
Domain object with attributes		This notation is used for describing a class or domain object along with its attribute.
Association	1 uses n	This notation is used for describing the relation between different classes along with cardinality ratio.

UML Notations for Class diagram:

- ✓ Used to show the name of the class.
- ✓ Used to show the attributes in the class.
- ✓ Used to describe the operations performed by the class.
- ✓ We can describe any additional components in this part (Optional One)

Object Notation: The Object is represented as the same as the class. As the object is an actual implementation of the class, known as the instance of the class. Hence, it has the same usage as the class.

Interface: The UML provides several ways to show interface implementation, providing an interface to clients, and interface dependency (a required interface). In the UML, interface implementation is formally called interface realization. It's useful to indicate “Class X requires (uses) interface Y” without drawing a line pointing to interface Y.

Qualified Association: A qualified association has a qualifier, used to select an object (or objects) from a larger set of related objects, based upon the qualifier key.

Association lines: Association is a relationship between classifiers which is used to show that instances of classifiers could be either linked to each other or combined logically or physically into some aggregation.

Aggregation: Aggregation is a vague kind of association in UML that loosely suggests whole-part relationships. It normally possess the ‘Has-a’ relationship.

Composition: Composition, also known as composite aggregation, is a strong kind of whole-part aggregation and is useful to show in some models. A "owns" B = Composition: B has no meaning or purpose in the system without A.

Singleton class: There is only one instance of a class instantiated—never two. In other words, it is a “singleton” instance. In a UML diagram, such a class can be marked with a ‘1’ in the upper right corner of the name compartment.

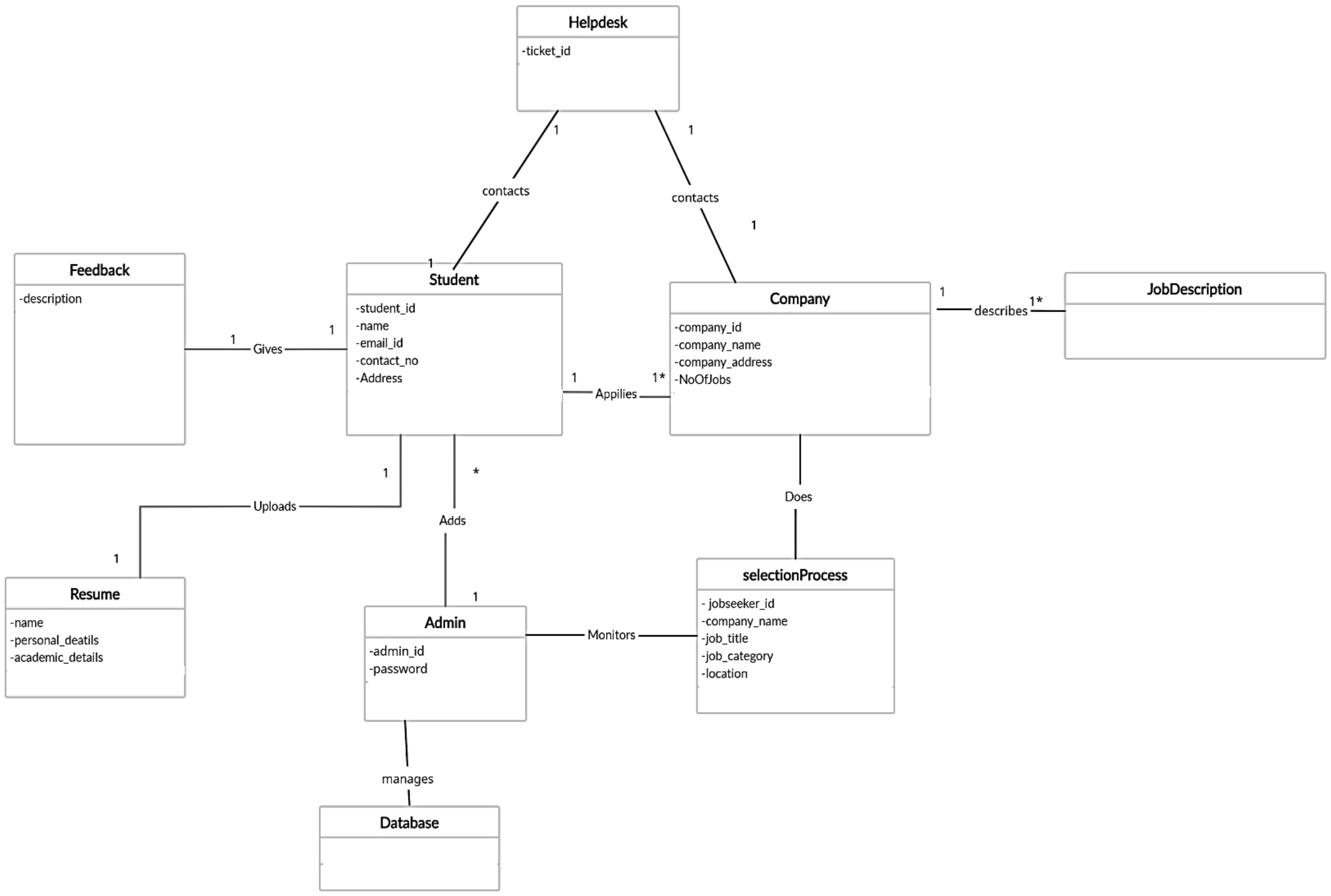
Constraints: Constraints may be used on most UML diagrams, but are especially common on class diagrams. A UML constraint is a restriction or condition on a UML element. It is visualized in text between braces

Keywords: A UML keyword is a textual adornment to categorize a model element. For example, the keyword to categorize that a classifier box is an interface is «interface». The «actor» keyword was used to replace the human stick-figure actor icon with a class box to model computer-system or robotic actors.

Abstract class: An abstract class is a class that is declared abstract —it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be sub classed.

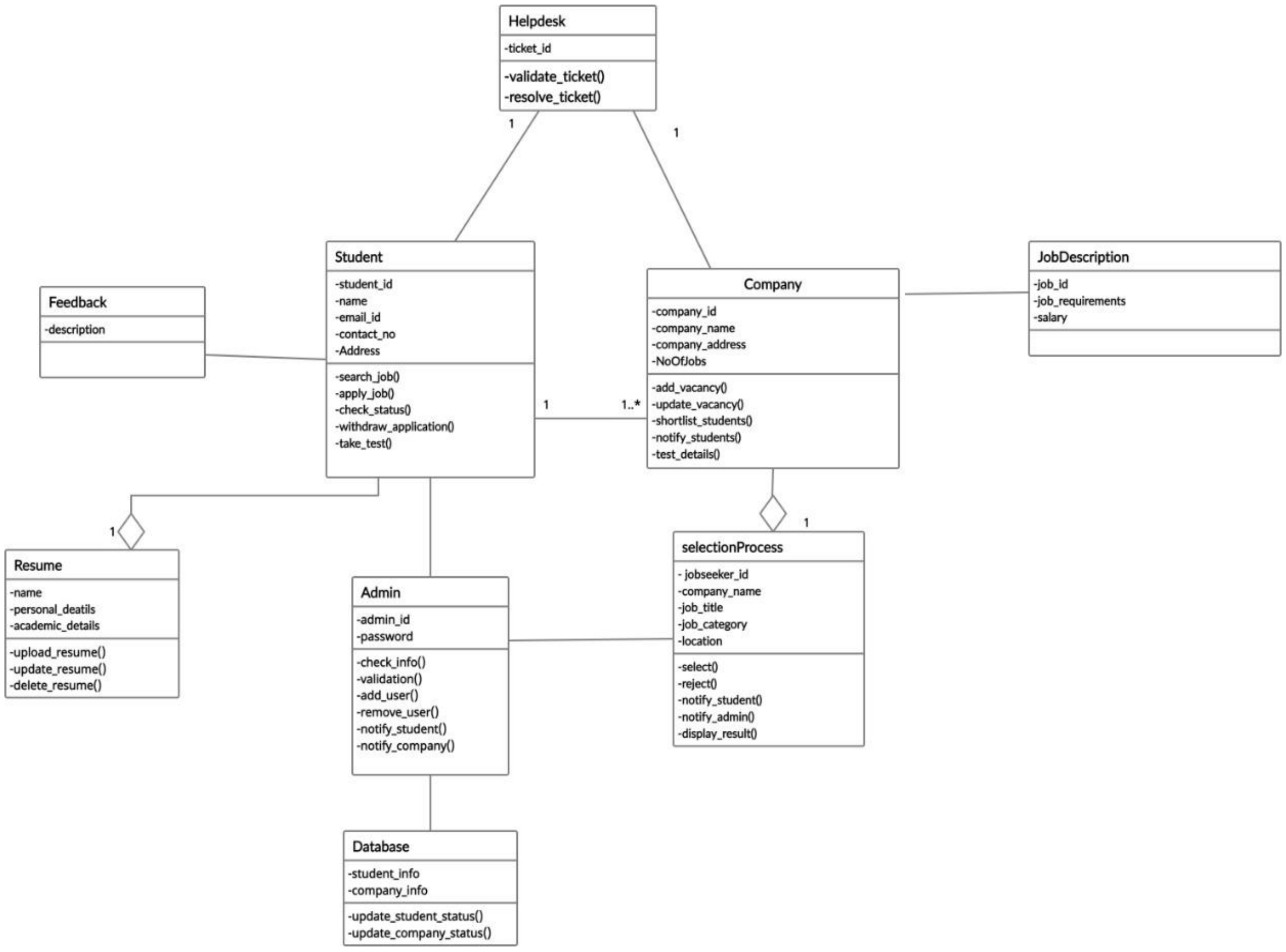
Notes: The use of note symbols is very common in UML, but this symbol can be specially used in class diagrams to represent the note or comment, constraint and method.

User Defined Compartments: In addition to common predefined compartments class compartments such as name, attributes, and operations, user-defined compartments can be added to a class box.



DOMAIN MODEL DIAGRAM

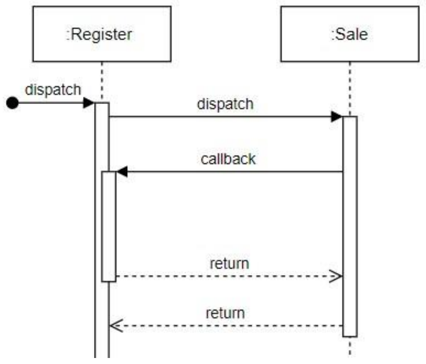
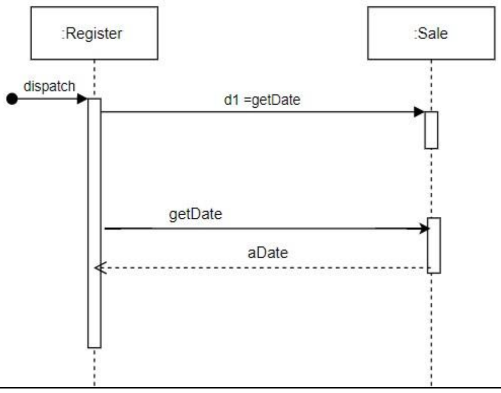
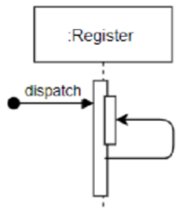
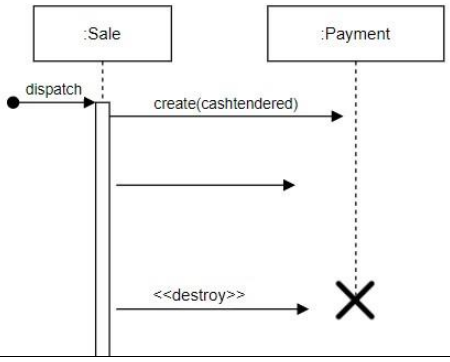
From the domain model diagram, we could identify the different attributes associated with each class.

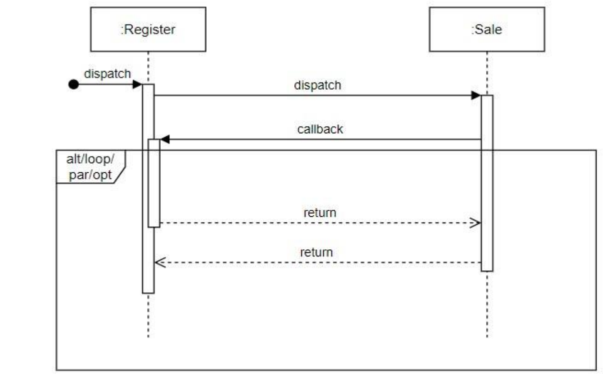
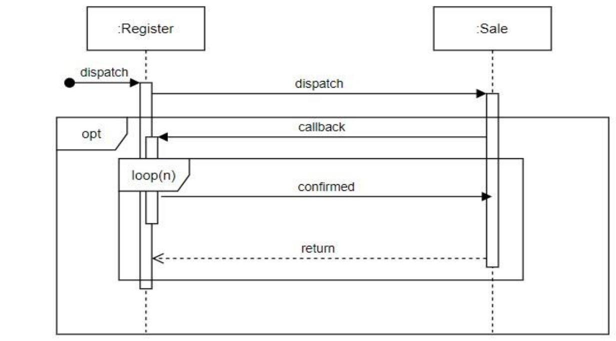
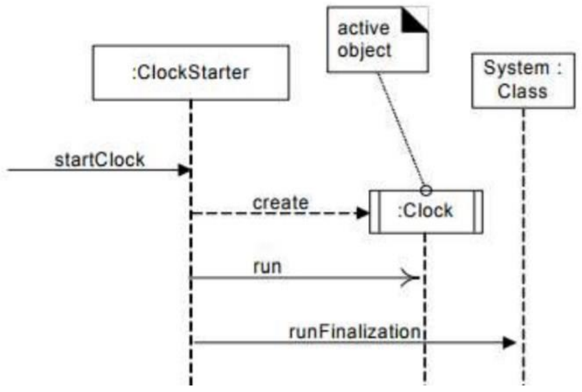


CLASS DIAGRAM

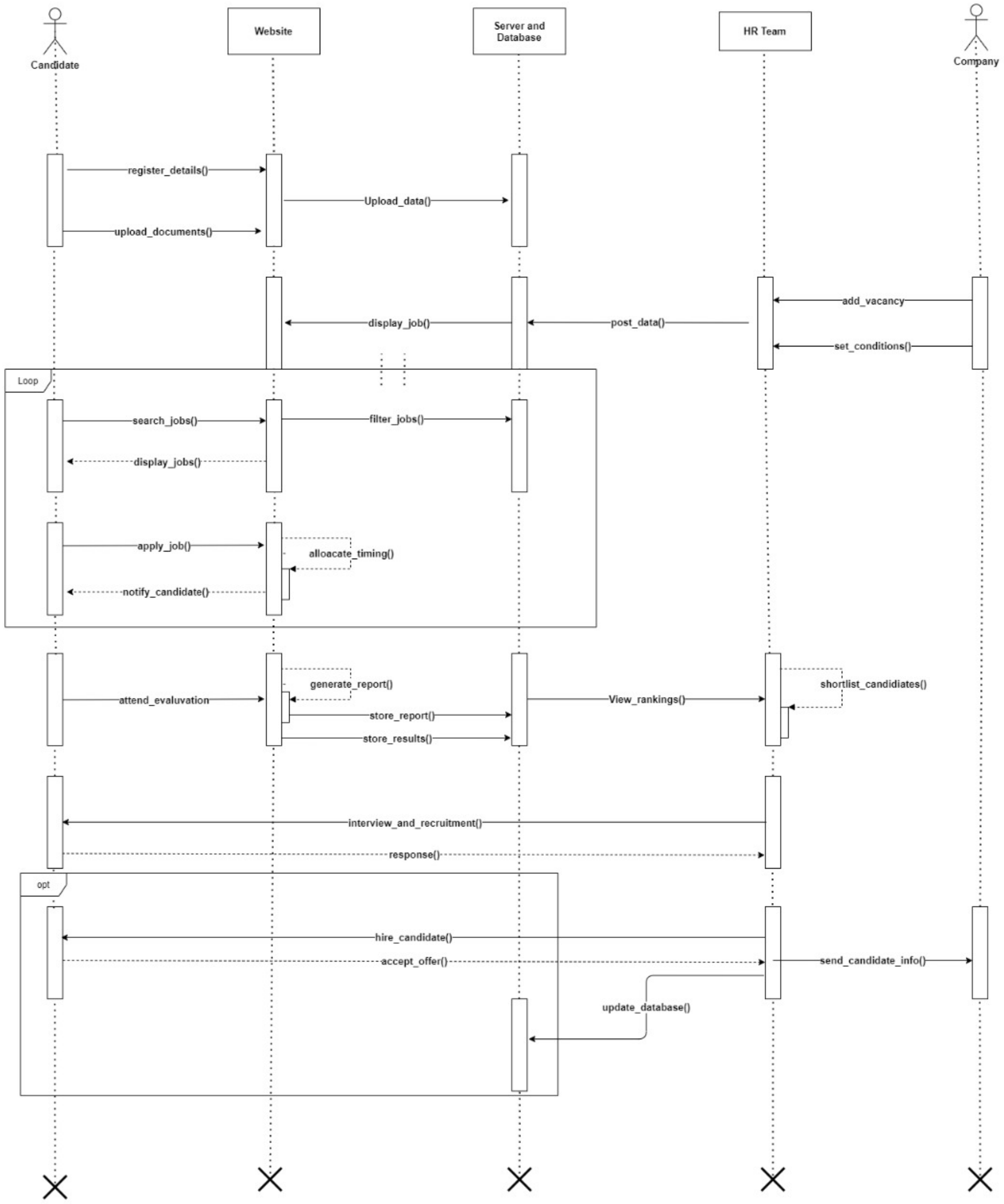
From the class diagram we could identify the different methods used in these classes. The multiplicity relations between the classes were drawn.

SEQUENCE DIAGRAM

Symbol	Description
 :Register :Sale dispatch dispatch callback return return	Lifeline boxes include a vertical line extending
 :Register :Sale dispatch d1 =getDate getDate aDate	Using a reply (or return) message line at the end of an activation
	A message being sent from an object to itself by using a nested activation bar
 :Sale :Payment dispatch create(cashtendered) <<destroy>>	Lifeline notation provides a way to express the destruction

 :Register :Sale dispatch dispatch callback alt/loop/ par/opt return return	Frame operation
 :Register :Sale dispatch dispatch opt callback loop(n) confirmed return	Nested Frames
 active object :ClockStarter System : Class startClock create :Clock run runFinalization	Asynchronous and Synchronous Calls a

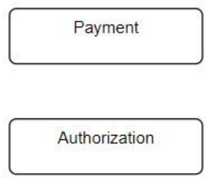
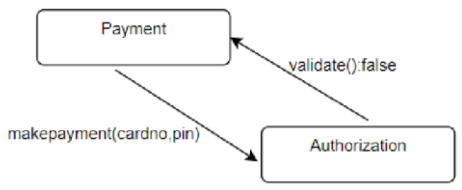
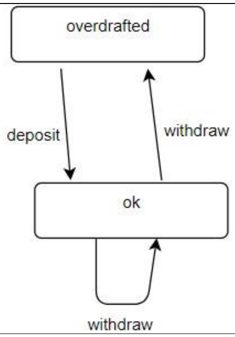
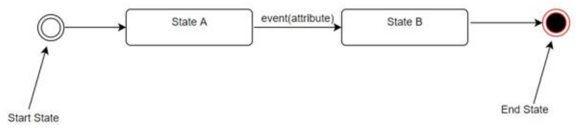
From the above sequence diagram we can see how the messages are exchanged between the actors and the objects. The sub-functions and failure scenarios depict the flow of control with various functions and are destroyed after their time period.

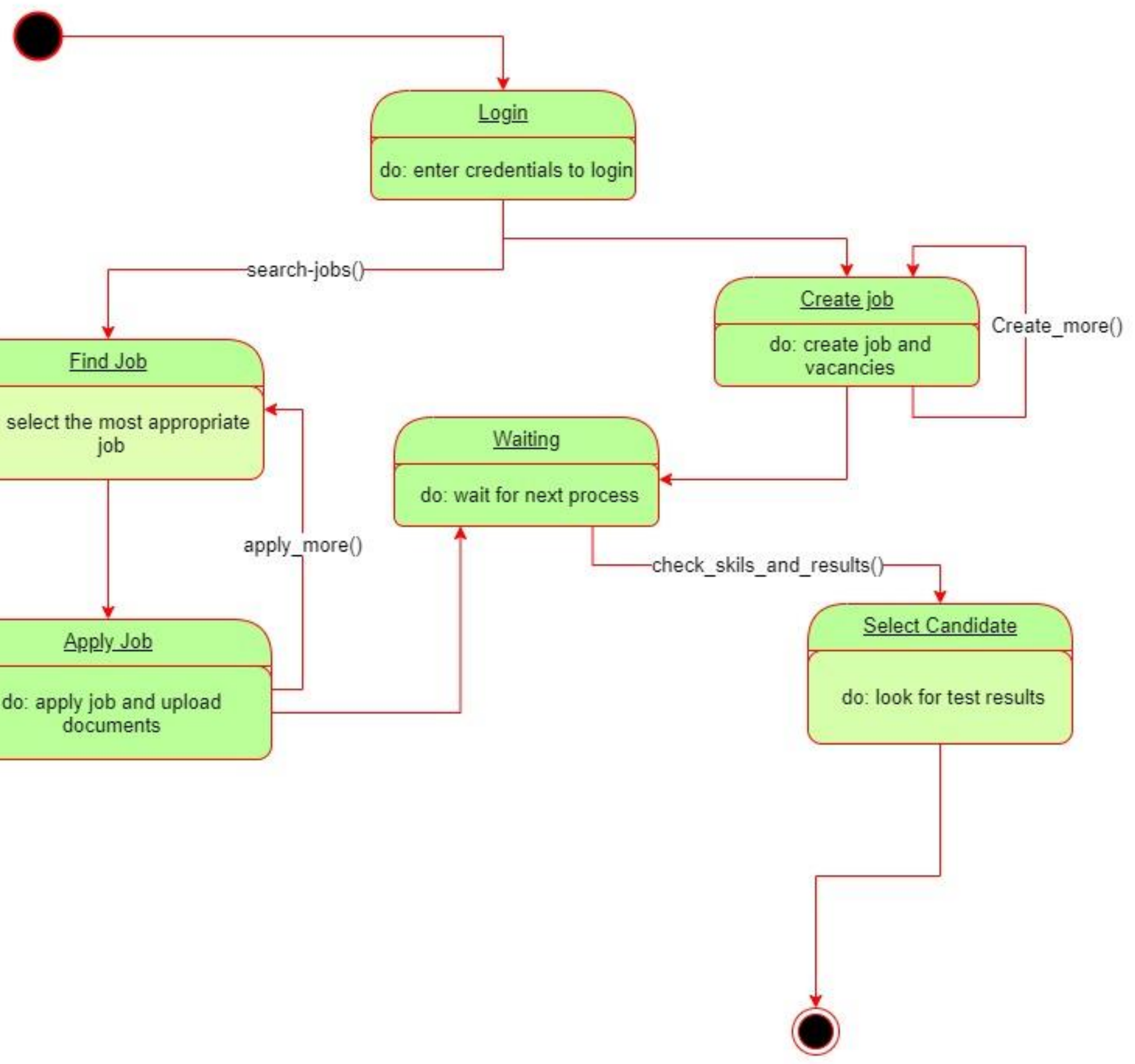


SEQUENCE DIAGRAM

STATE MACHINE DIAGRAM

A state machine diagram is a behavior that specifies the sequence of states that an object goes through in a lifetime, in response to events and also its responses those events.

Symbol	Description
	Condition/Situation at a specific moment in time
 Payment validate():false makepayment(cardno,pin) Authorization	Relationship indicating a state change
	Self-Transition
 State A event(attribute) State B Start State End State	Start State & End State

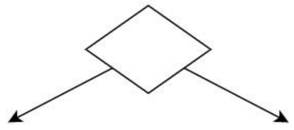
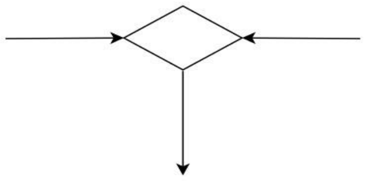
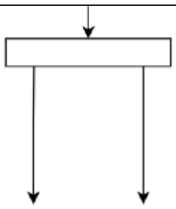
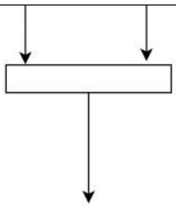
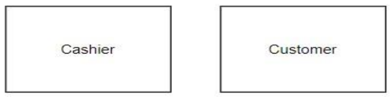

Login
do: enter credentials to login
search-jobs()
Create job
do: create job and vacancies
Create_more()
Find Job
do: select the most appropriate job
Waiting
do: wait for next process
apply_more()
check_skils_and_results()
Apply Job
do: apply job and upload documents
Select Candidate
do: look for test results

STATE MACHINE DIAGRAM – MAIN SUCCESS SCENARIO

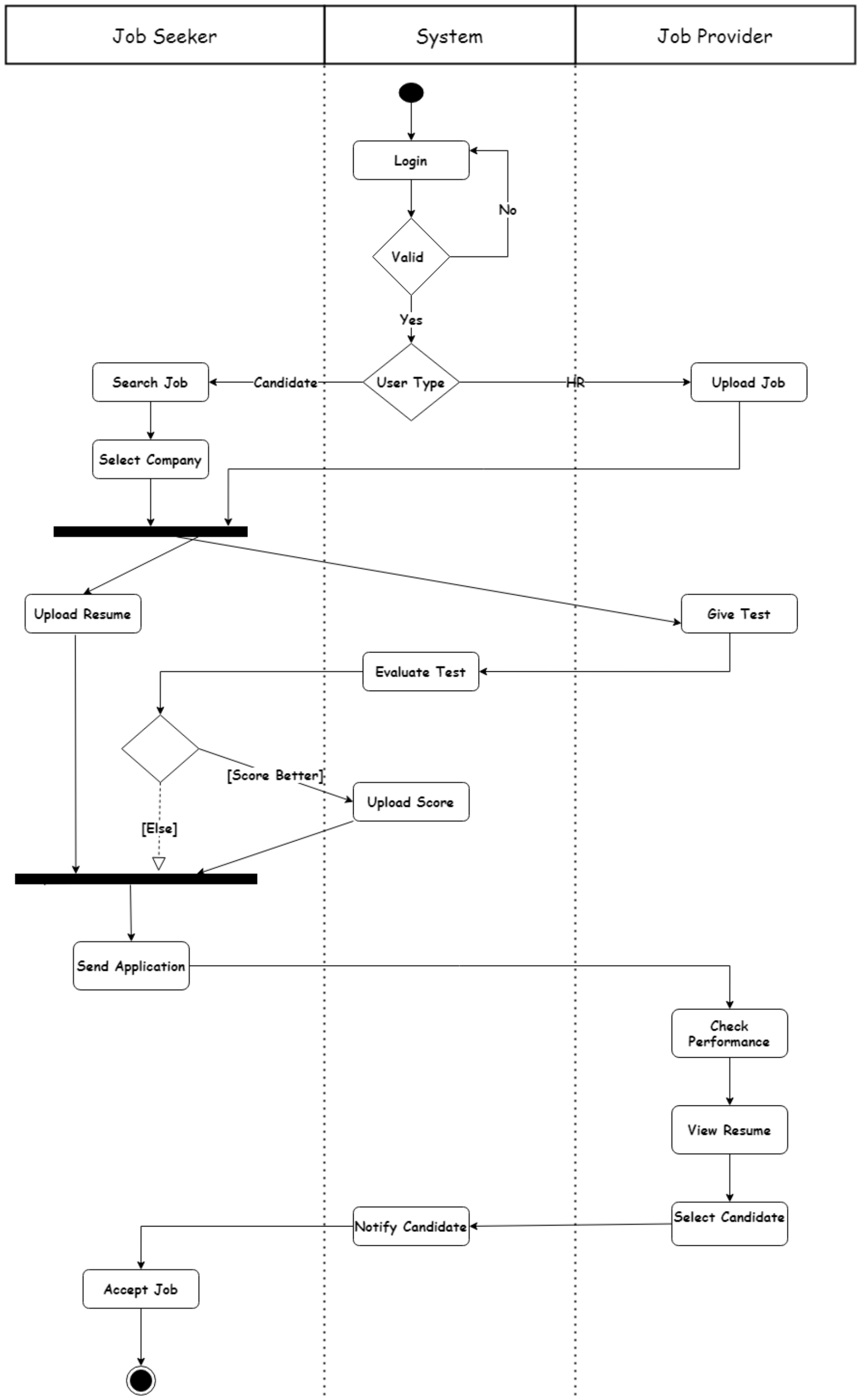
From the above state machine diagram we could see the behavior of an object from a single class. The first action is Login. The transition between the various states (login, find job) due to various events (search jobs) are shown. The create job object is in self-transition. The transition ends with selection of candidate by company. Thus, the above state diagram illustrates the sequence of states, that the objects of the recruitment system goes through and its response to events.

ACTIVITY DIAGRAM

Activity diagrams are the object-oriented equivalent of flow charts and data-flow diagrams from structured development

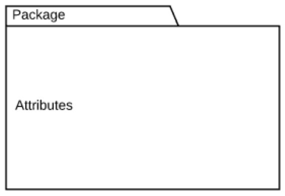
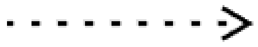
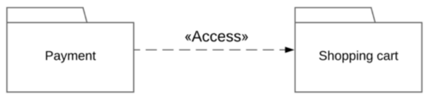
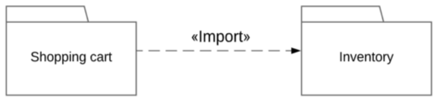
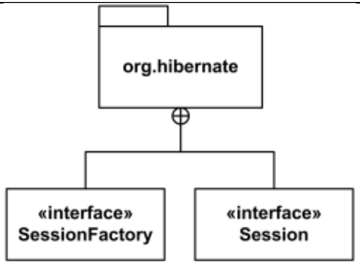
Symbol	Description
	To represent a test condition to ensure that the control
	To bring back together different decision paths
	To split set of concurrent flows of activities
	To bring back behavior into a set of concurrent flows of activities
	To group activities performed by the same actor on an activity diagram

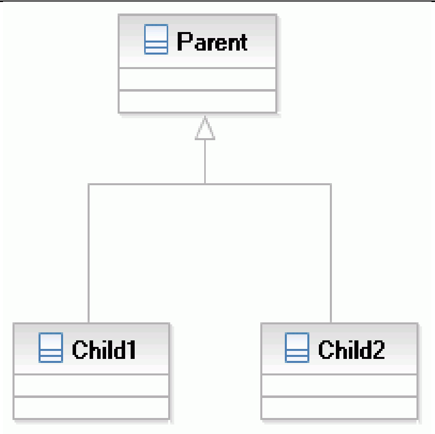
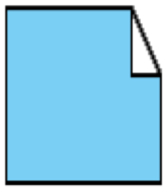
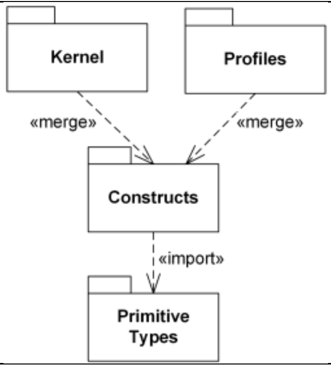
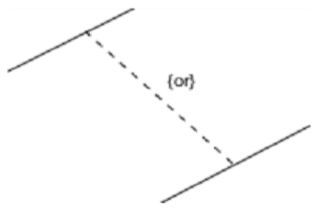
From the activity diagram below, we could see the flow of activity through a series of actions. The process starts with user registration. The user details are validated by the administrator and registered. The candidate searches for a job which is followed by applying for it and attending the assessment process. The employer uploads vacancies and recruits candidates based on reports (generated by system) and interviews. The process is terminated when a candidate gets a job. Thus, the above activity diagram illustrates the workflow behavior of the recruitment system.

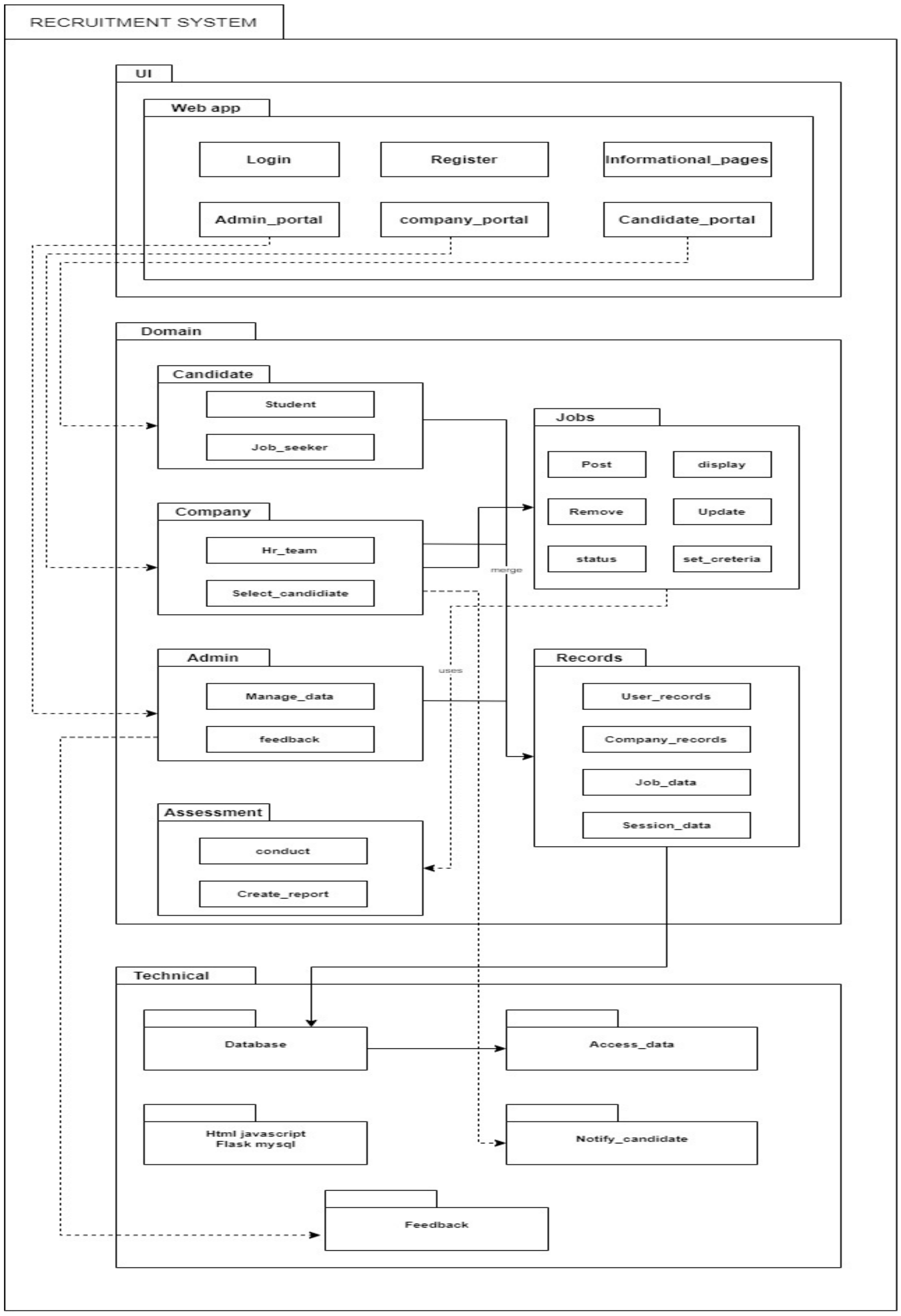


ACTIVITY DIAGRAM - MAIN SUCCESS SCENARIO

PACKAGE DIAGRAM:-

Notation	Symbol	Description
Package		Groups common elements based on data, behavior, or user interaction
Dependency		Depicts the relationship between one element (package, named element, etc) and another
Access		Indicates that one package requires assistance from the functions of another package.
Import		Indicates that functionality has been imported from one package to another.
Subsystem	 org.hibernate «interface» SessionFactory «interface» Session	A plus sign (+) within a circle is drawn at the end attached to the namespace (package). This notation for packages is semantically equivalent to composition

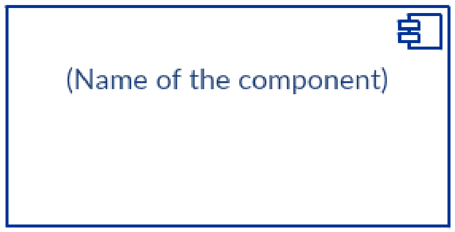
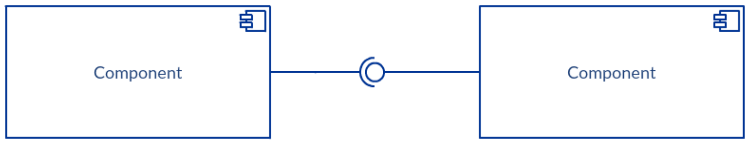
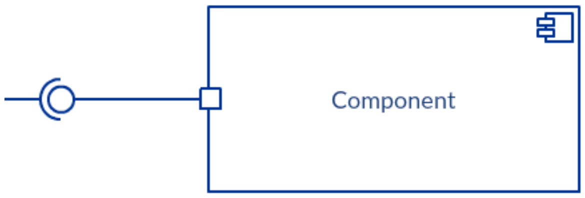
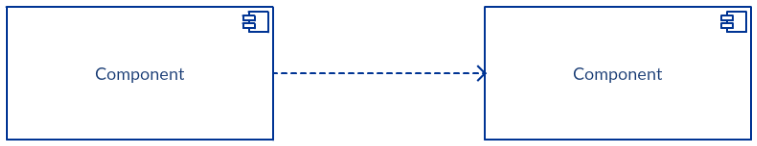
Generalization		Generalization relationships are used in class, component, deployment, and use-case diagrams to indicate that the child receives all of the attributes, operations, and relationships that are defined in the parent.
Note		Note is a modeling construct for adding textual information - such as a comment, constraint definition, or method body - to **UML** diagrams. **Notes** are depicted as a rectangle with the top-right corner folded over
Merge	 Kernel Profiles «merge» «merge» Constructs «import» Primitive Types	**Merge** is a directed relationship between two packages that indicates that content of one package is extended by the contents of another package.
Constraint		**Constraint** is a element which represents some condition, restriction or assertion related to some element (that owns the **constraint**) or several elements. **Constraint** is usually specified by a Boolean expression which must evaluate to a true or false.

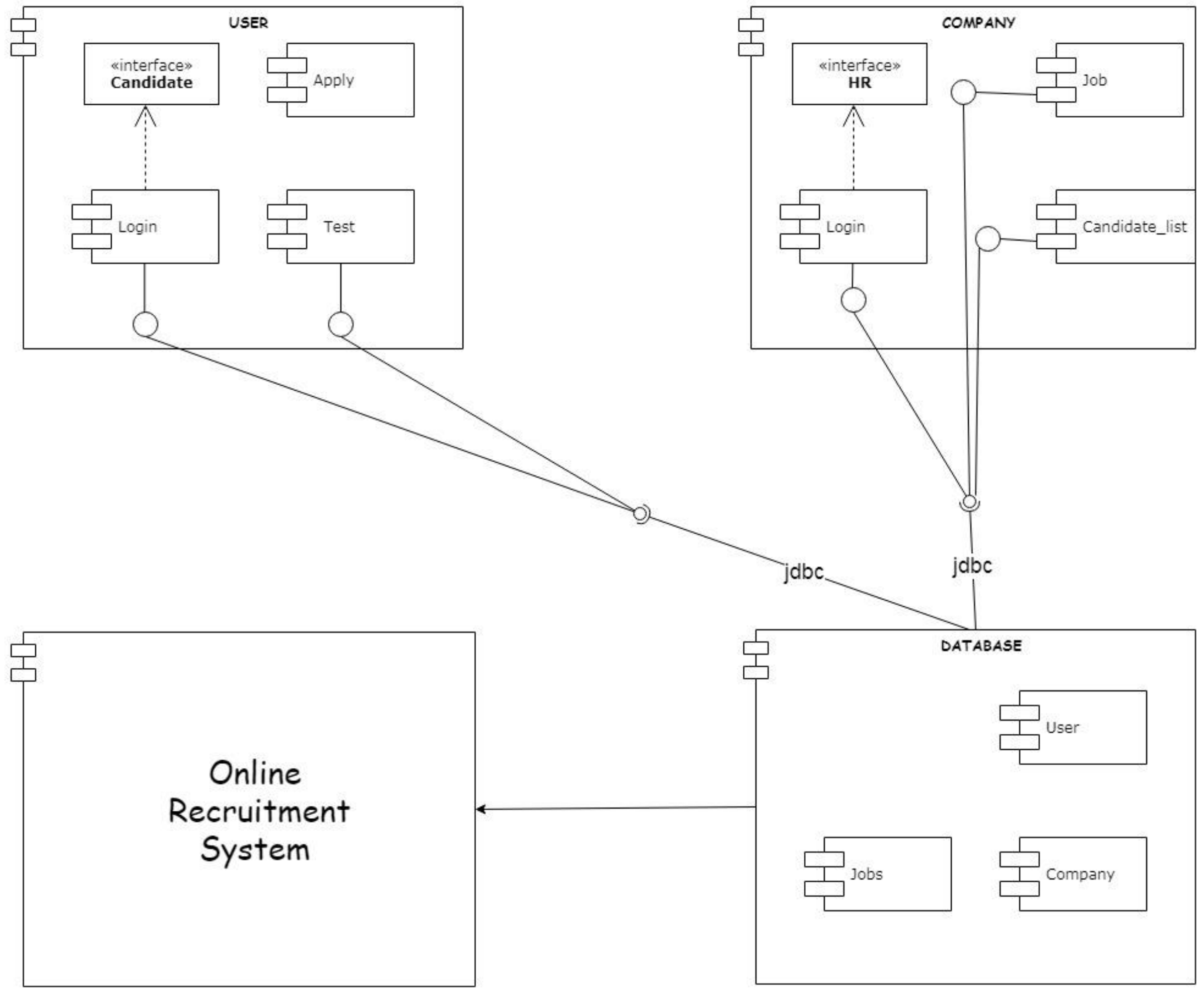


PACKAGE DIAGRAM

From the above package diagram we could see the dependencies between the various packages that make up a model. The multi-layered model consists of 3 layers (UI, domain, technical) which contains the various packages (candidate, admin, records) and the dependencies (merge, import) between them. They show the structure and the dependencies between sub-systems or modules.

COMPONENT DIAGRAM:-

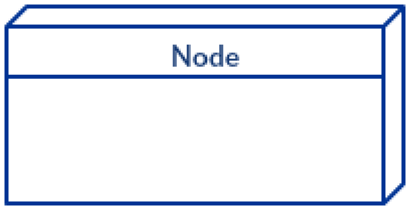
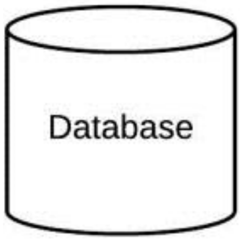
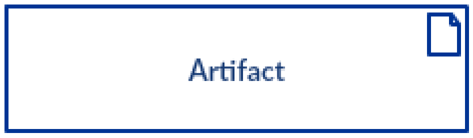
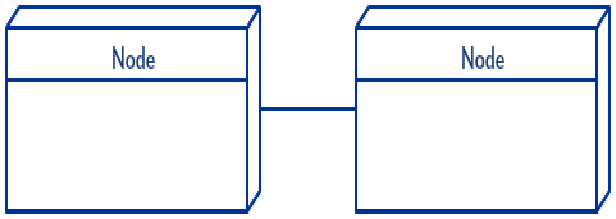
Symbol	Description
	Rectangle with the component stereotype (the text <<component>>). The component stereotype is usually used above the component name to avoid confusing the shape with a class icon.
	The assembly connector allows linking the component's required interface (represented with a semi-circle and a solid line) with the provided interface (represented with a circle and solid line) of another component.
	Port (represented by the small square at the end of a required interface or provided interface) is used when the component delegates the interfaces to an internal class.
	Dependency arrow to show the relationship between two components.

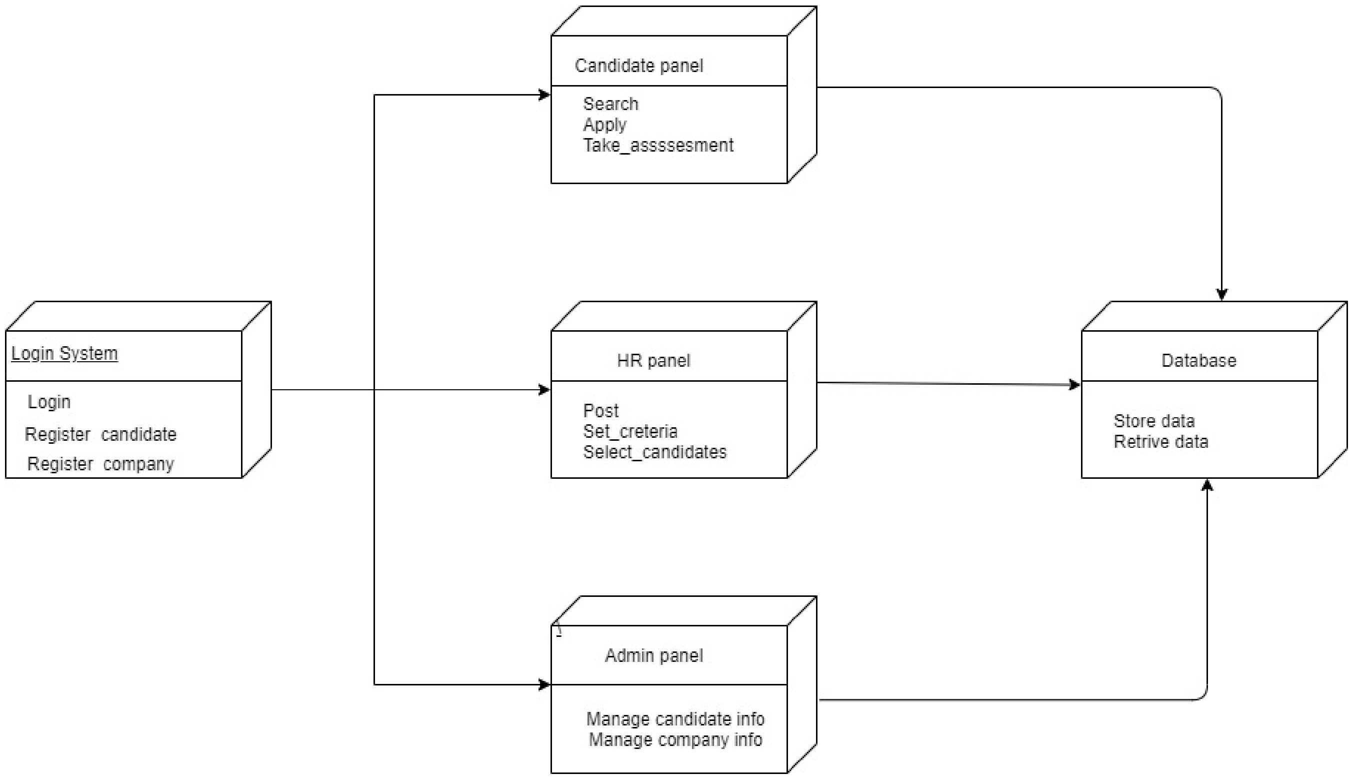


COMPONENT DIAGRAM

From the above component diagram we could construct executable by using forward and reverse engineering. The different high level reusable parts of a system are represented in a component diagram. The different components like User, Database, Company and various interfaces like candidate, HR represents the implementation perspective of the system.

DEPLOYMENT DIAGRAM:

Symbol	Description
	There are two types of nodes : **Device nodes** are computing resources with processing capabilities & the ability to execute programs that include PCs, laptops, and Mobile phones. An **execution environment node,** or EEN, is any computer system that resides within a device node. It could be an operating system, a JVM, or another servlet container.
	**Databases** represent any data stored by the deployed system.
	**Artifacts** are concrete elements that are caused by a development process. Examples are libraries, archives, configuration files, executable files etc.
	**Communication path** is a straight line that represents communication between two device nodes.



DEPLOYMENT DIAGRAM

From the above deployment diagram we could show the structure of the runtime system and capture the hardware that will be used to implement the system and the links between the different items of hardware. The different nodes like Login System, HR panel, Admin panel captures the topology of the system hardware.