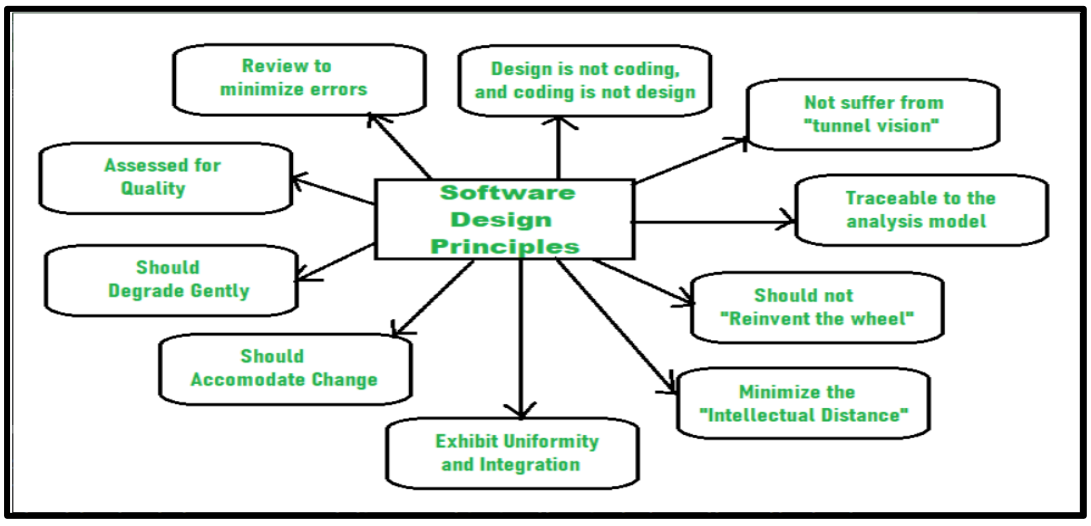
Practical-4

Aim: List at least 10 software design principles & online/offline tools for the software development process and Draw the UML diagram (Diagram Must 1. Use Case, 2. Activity, 3. State, 4. Sequence, 5. Data Flow (DFD) 6. Class) for JoBHunt.

- Software Design is also a process to plan or convert the software requirements into a step that are needed to be carried out to develop a software system. There are several principles that are used to organize and arrange the structural components of Software design. Software Designs in which these principles are applied affect the content and the working process of the software from the beginning.

❖ Software Design Principles:

1. Should not suffer from “Tunnel Vision”
2. Traceable to analysis model
3. Should not “Reinvent the wheel”
4. Minimize intellectual distance
5. Exhibit uniformity and integration
6. Accommodate change
7. Degrade gently
8. Assessed or quality
9. Review to discover errors
10. Design is not coding and coding is not design.



1. **Should not suffer from "Tunnel Vision" –**

- While designing the process, it should not suffer from "tunnel vision" which means that is should not only focus on completing or achieving the aim but on other effects also.

2. **Traceable to analysis model –**

- The design process should be traceable to the analysis model which means it should satisfy all the requirements that software requires to develop a high-quality product.

3. **Should not "Reinvent The Wheel" –**

- The design process should not reinvent the wheel that means it should not waste time or effort in creating things that already exist. Due to this, the overall development will get increased.

4. **Minimize Intellectual distance –**

- The design process should reduce the gap between real-world problems and software solutions for that problem meaning it should simply minimize intellectual distance.

5. **Exhibit uniformity and integration –**

- The design should display uniformity which means it should be uniform throughout the process without any change. Integration means it should mix or combine all parts of software i.e. subsystems into one system.

6. **Accommodate change –**

- The software should be designed in such a way that it accommodates the change implying that the software should adjust to the change that is required to be done as per the user's need.

7. **Degrade gently –**

- The software should be designed in such a way that it degrades gracefully which means it should work properly even if an error occurs during the execution.

8. **Assessed or quality –**

- The design should be assessed or evaluated for the quality meaning that during the evaluation, the quality of the design needs to be checked and focused on.

9. **Review to discover errors –**

- The design should be reviewed which means that the overall evaluation should be done to check if there is any error present or if it can be minimized.

10. **Design is not coding and coding is not design –**

- Design means describing the logic of the program to solve any problem and coding is a type of language that is used for the implementation of a design.

❖ Tools for software development process:

Sr No.	Tools Name
1.	Asana
2.	Clickup
3.	Monday
4.	Wrike
5.	jira

1. Asana:

- Asana is a management tool for small teams. It tracks tasks assigned to team members and the PM can be sure to know who does what. Asana can integrate with Google Drive, Chrome, Dropbox, GitHub, and other third-party services.

Fig- 4.2 Asana [2]

Applications:

- When you want simple management tool and dependency reviewing is not on your .
- When you want good security and custom goal setting.

2.Click UP:

- The ClickUp platform allows users to utilize more than 100 features which include: to-do lists, assign comments, resolve comments, have recurring tasks, sync with Google Calendar, sorting, customize assignees, collaboration detection, image mockups, assign multiple people to tasks.

Fig-4.3 ClickUp [3]

Applications:

- When you need scalability, customizability and attractive UI at affordable price.
- When you want to have a time-bound and also want to ensure growth of the humanresource by ensuring that they are skilled up enough to match the changing requirements.
- When you want to collaborate your workflow with other applications.

3.Monday:

- Monday.com is one of the best agile project management tools available in the market. It offers a clean interface with color-coded features. This easy-to-use software is suitable for big or small teams. It also offers a user-friendly mobile app and highly customizable workflows.

Fig-4.4 monday.com [4]

- Monday.com is a simple but intuitive tool that enables people to manage work, meet deadlines, and build a culture of transparency. The solution is best for streamlining discussions, to see who is working on what, and keeping everyone in the know.

Application

- Team Management.
- Work Planning.
- Project Planning.
- Marketing.
- Agile Project. Management.
- Creative Agencies.
- Bug Tracking.
- Sales Management.

4.Wrike:

- Wrike is a scrum project management tool with a multi-pane UI and aids in team collaboration and project management. It provides fully customizable team sprint dashboards that enable team members to view new, in-progress, and completed projects in one single place.

Fig-4.5 Wrike [5]

Applications:

- It can be used for internal teams communication at large enterprises.
- When the company is very big and cost is not a constrain.

5. Jira:

- Jira is one of the popular Scrum Project management tools that offers

thousands of functionalities such as Scrum boards, customizable backlogs, reporting options, to name a few. It is one of the best scrum project management tools for IT and software development companies and large businessorganizations.

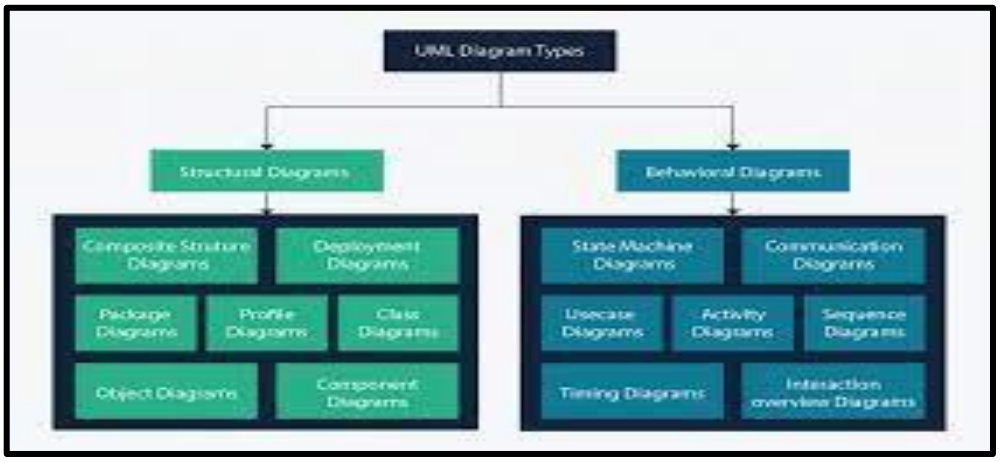
Fig- 4.6 Jira [6]

Application

- Requirements and Test casemanagement
- In Agile Methodolog
- Project Management
- Software Development
- Product Management
- Task Management
- Bug Tracking

❖ Unified Modeling Language(UML):

- Unified Modeling Language (UML) is a general purpose modelling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering. UML is not a programming language, it is rather a visual language.



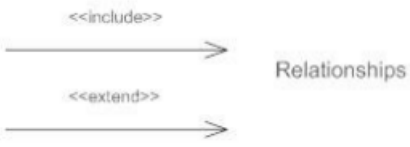
- **Use Case Diagram:** It represents the functionality of a system by utilizing actors and use cases. It encapsulates the functional requirement of a system and its association with actors. It portrays the use case view of a system.

- **System:** Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.

- **Use Case**: Draw use cases using ovals. Label the ovals with verbs that represent the system's functions.

- **Actors:** Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.

- **Relationships:** Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.



How to draw a Use Case diagram?

It is essential to analyse the whole system before starting with drawing a use case diagram, and then the system's functionalities are found. And once every single functionality is identified, they are then transformed into the use cases to be used in the use case diagram.

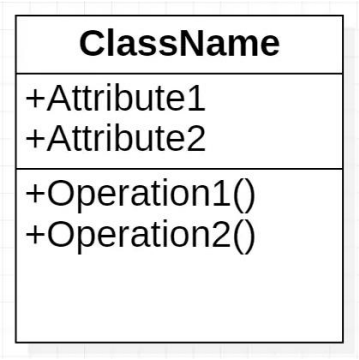
After that, we will enlist the actors that will interact with the system. The actors are the person or a thing that invokes the functionality of a system. It may be a system or a private entity, such that it requires an entity to be pertinent to the functionalities of the system to which it is going to interact.

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

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

1. A pertinent and meaningful name should be assigned to the actor or a use case of a system.
2. The communication of an actor with a use case must be defined in an understandable way.
3. Specified notations to be used as and when required.
4. The most significant interactions should be represented among the multiple no of interactions between the use case and acto

❖ **Class Diagram:** Class diagrams are one of the most widely used diagrams. It is the backbone of all the object-oriented software systems. It depicts the static structure of the system. It displays the system's class, attributes, and methods. It is helpful in recognizing the relation between different objects as well as classes.



❖ **How to draw a Class Diagram?**

➢ The class diagram is used most widely to construct software applications. It not only represents a static view of the system but also all the major aspects of an application. A collection of class diagrams as a whole represents a system.

- Some key points that are needed to keep in mind while drawing a class diagram are given below:

1. To describe a complete aspect of the system, it is suggested to give a meaningful name to the class diagram.
2. The objects and their relationships should be acknowledged in advance.
3. The attributes and methods (responsibilities) of each class must be known.
4. A minimum number of desired properties should be specified as more number of the unwanted property will lead to a complex diagram.
5. Notes can be used as and when required by the developer to describe the aspects of a diagram.
6. The diagrams should be redrawn and reworked as many times to make it correct before producing its final version.

UML Diagram-JOBHUNT

1. Use case Diagram

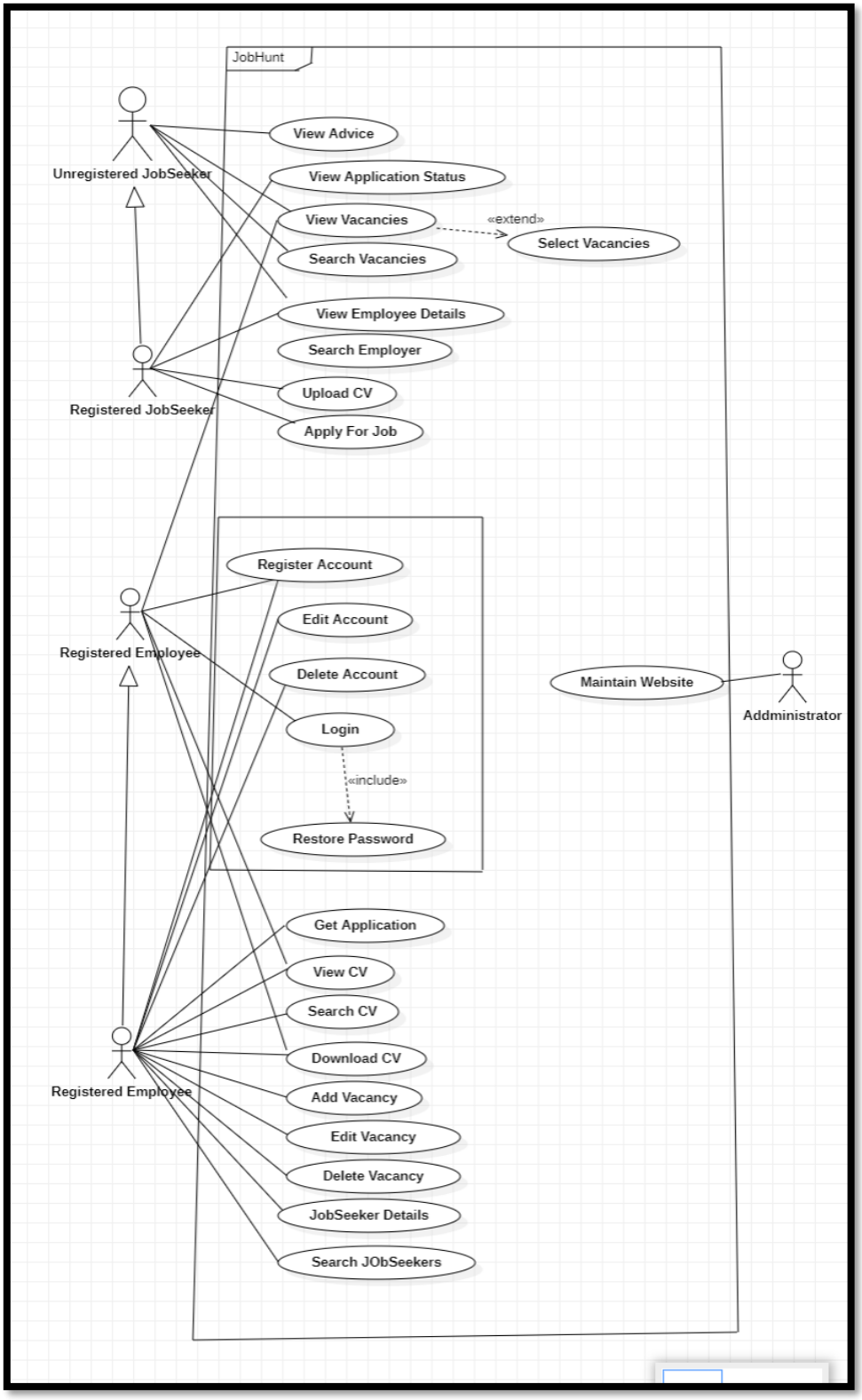


Fig:1-Use Case Diagram

2. Activity Diagram

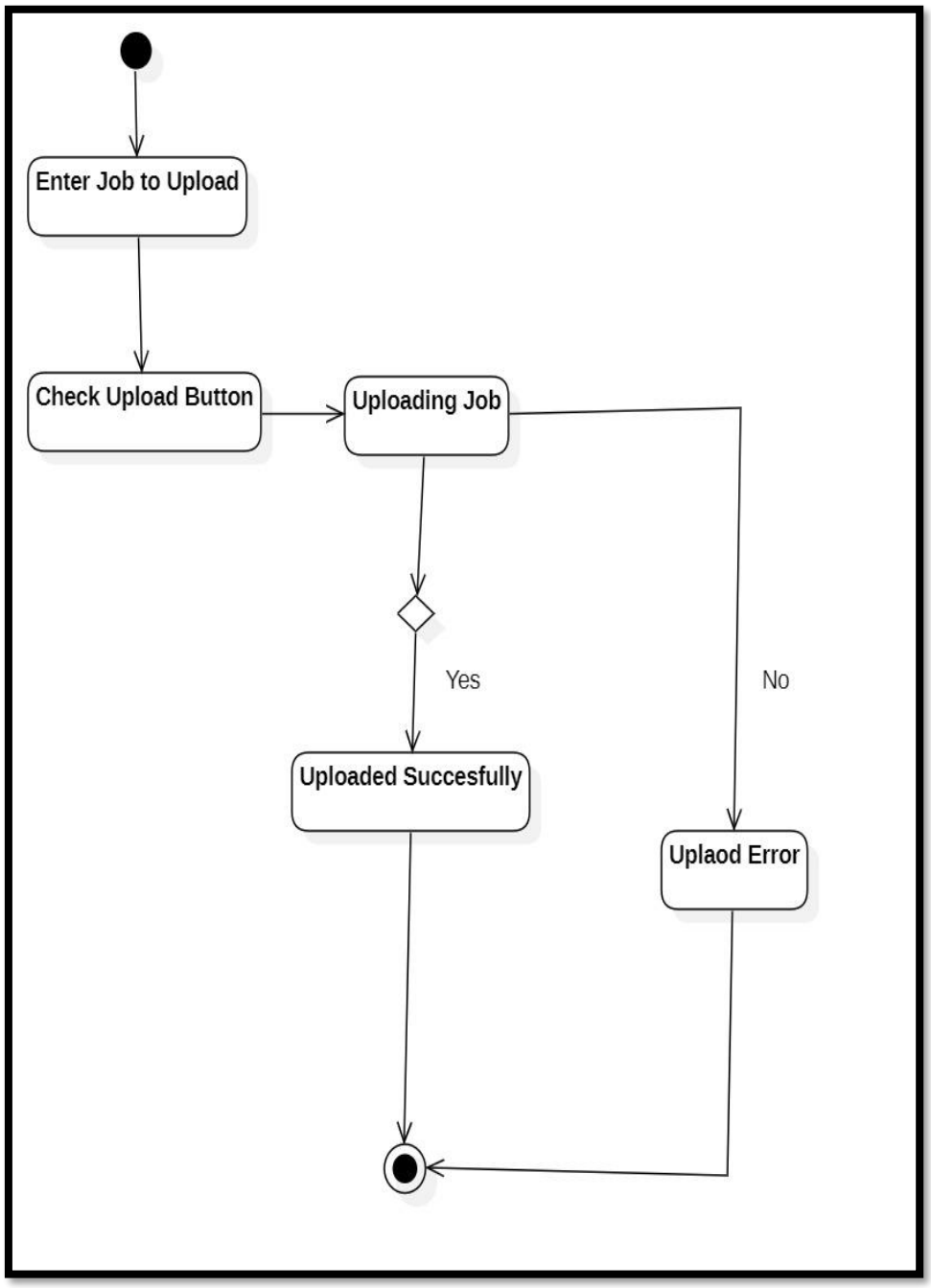


Fig:2-Upload JoB Activity Diagram

3. State Diagram

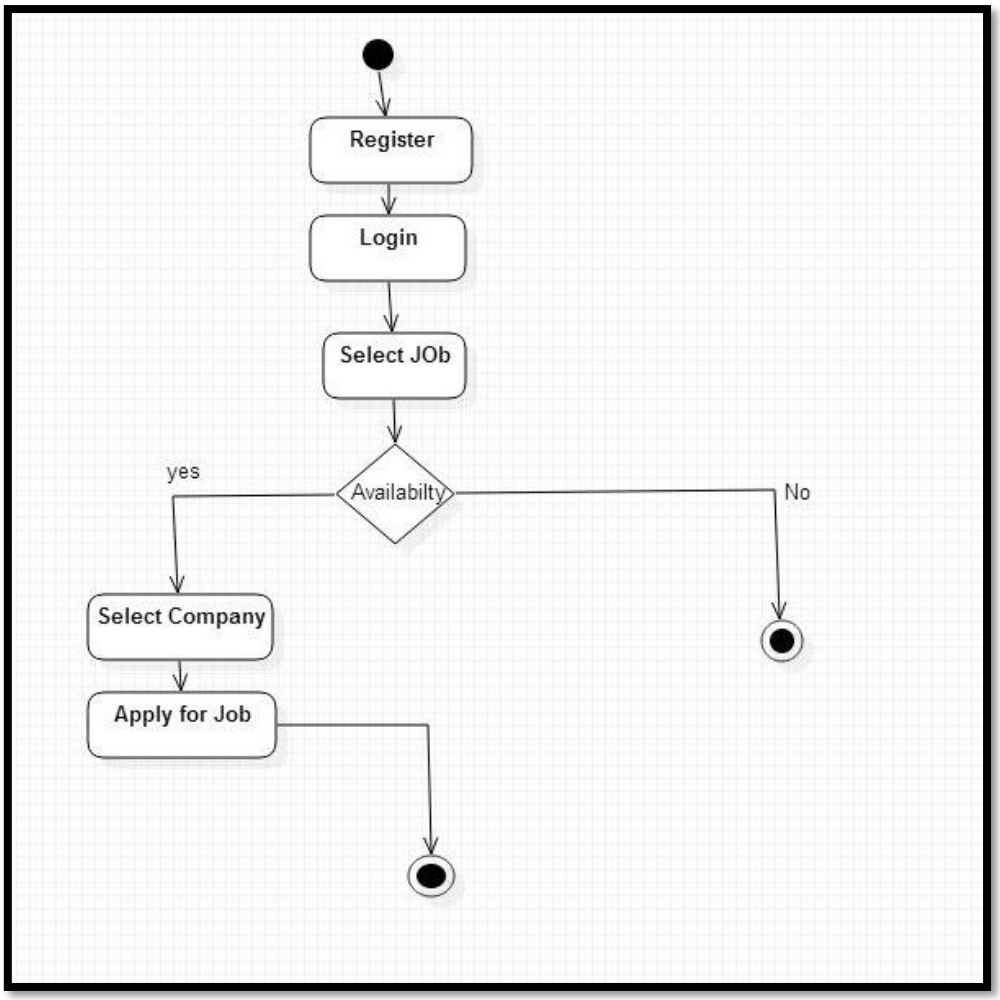


Fig:3-State Diagram

4, Class Diagram

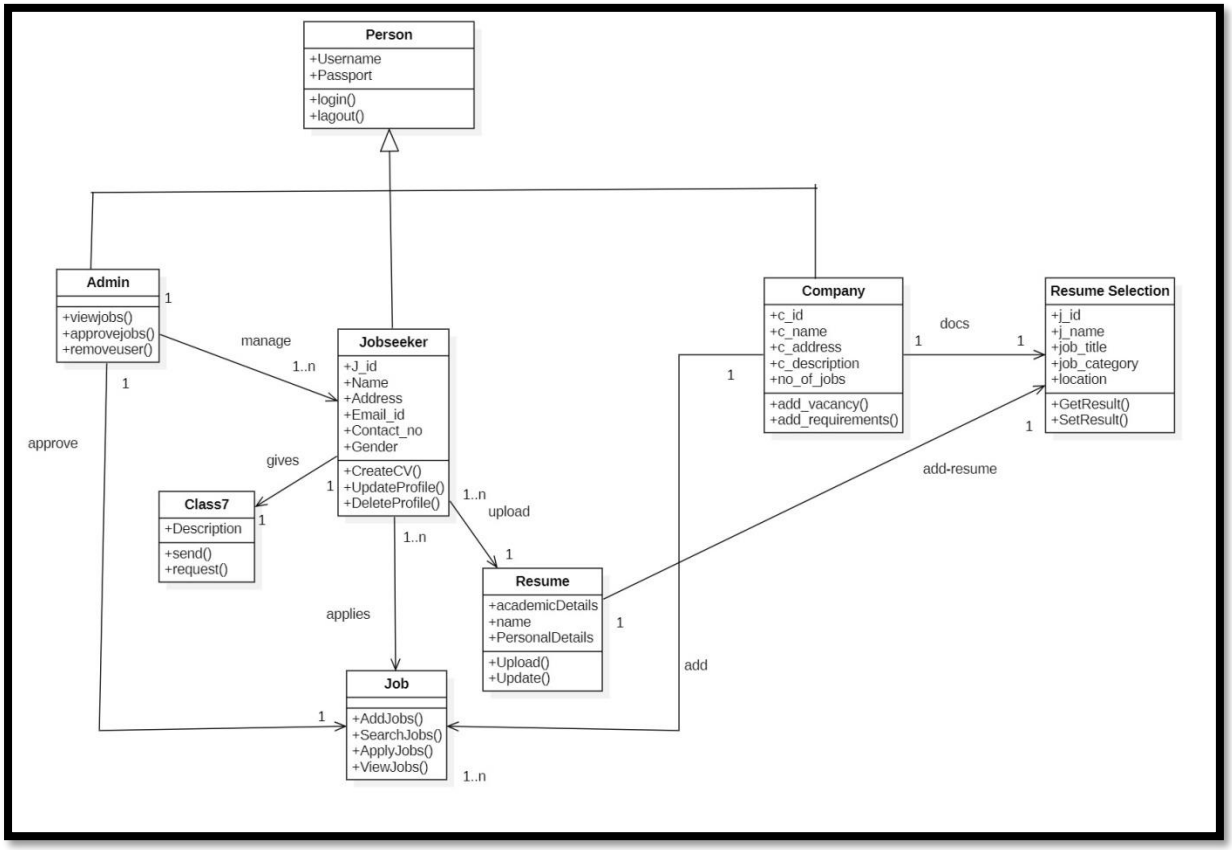


Fig:4-Class Diagram

❖ **References:**

[2] Asana:

https://static.businessworld.in/article/article_extra_large_image/1654670822_UF2nXH_kissflow.j pg

[3] ClickUp:

https://gdm-catalog-fmapi-prod.imgix.net/ProductLogo/581edf50-f567-4e9e-b561-8ba9f88ba1dd.png?auto=format&size=50

[4] Monday:

https://blog.wearedrew.co/hubfs/logomonday.com_.png

[5] Wrike:

https://i.pcmag.com/imagery/reviews/00Qb2h7l6Z4VyNqVm1FMz6a-23..v1608143900.png

[6] Jira:

https://wac-cdn.atlassian.com/dam/jcr:9e1841b9-2557-4eb2-ab47-d92428580b02/Jira%20Software@2x-blue.pn

Remark:_____.

Mark:_____.

Signature:_____.