Contents

[Chapter 2: Analysis 2](#_Toc15333318)

[2.1 introduction 2](#_Toc15333319)

[2.1.1 Need for Analysis 2](#_Toc15333320)

[2.2 Analysis Methodology: 2](#_Toc15333321)

[2.2.1 what is hard system approach: 2](#_Toc15333322)

[2.2.3 Information gathering technique: 3](#_Toc15333323)

[1. interview: 3](#_Toc15333324)

[2. Questionnaire: 3](#_Toc15333325)

[2.3 Feasibility study: 4](#_Toc15333326)

[2.3.1 factors of feasibility study: 4](#_Toc15333327)

[2.4 SRS (Software Requirement Specification): 5](#_Toc15333328)

[2.4.1 Functional requirements: 5](#_Toc15333329)

[2.4.3 Non-functional requirements: 6](#_Toc15333330)

[2.4.4 MOSCOW prioritization: 7](#_Toc15333331)

[2.4.4 Hardware specification: 8](#_Toc15333332)

[2.4.5 software specification: 8](#_Toc15333333)

[2.5 use case diagram: 9](#_Toc15333334)

[2.6 NLA (Natural Language Analysis): 9](#_Toc15333335)

[2.6.1 Initial class diagram: 10](#_Toc15333336)

[2.7 system architecture: 10](#_Toc15333337)

# Chapter 2: Analysis

# 2.1 introduction

Analysis is the process of breaking a complex topic or substance into smaller parts in order to gain better understanding of it. In other word analysis is a systematic examination and evaluation of data or information, which is broken down into smaller parts to uncover their interrelationships, thus providing basis for problem solving and decision making.

## 2.1.1 Need for Analysis

Analysis is one of the important parts of software development process which help in determining the design an architecture of the project. Here in this phase we decide the software methodology that will be used while developing the software. Which will help us to figure out the specification, design of our software. One of the main aspects of analysis is to configure the requirement to meet the functional specification of the software. Among many software development analyses, I chose hard approach analysis methodology.

# 2.2 Analysis Methodology:

The project follows hard approach analysis methodology. Hard system Approach or HSA system analysis refers to taking a highly structured approach to the analysis of information system.

## 2.2.1 what is hard system approach:

This a system analysis methodology that takes a highly structured approach to the analysis of information system. this approach follows a logical sequence of steps and follows the rules, guidelines and standards. It is particularly appropriate to use when working on large/complex information system and it is also used for smaller-scale business information systems projects. Some of the examples of hard approach methodology are SSADM, prototyping, Joint Application and Design (JAD), Rapid Analysis and Design (RAD), Scrum, Agile etc. Here, I will use SSADM.

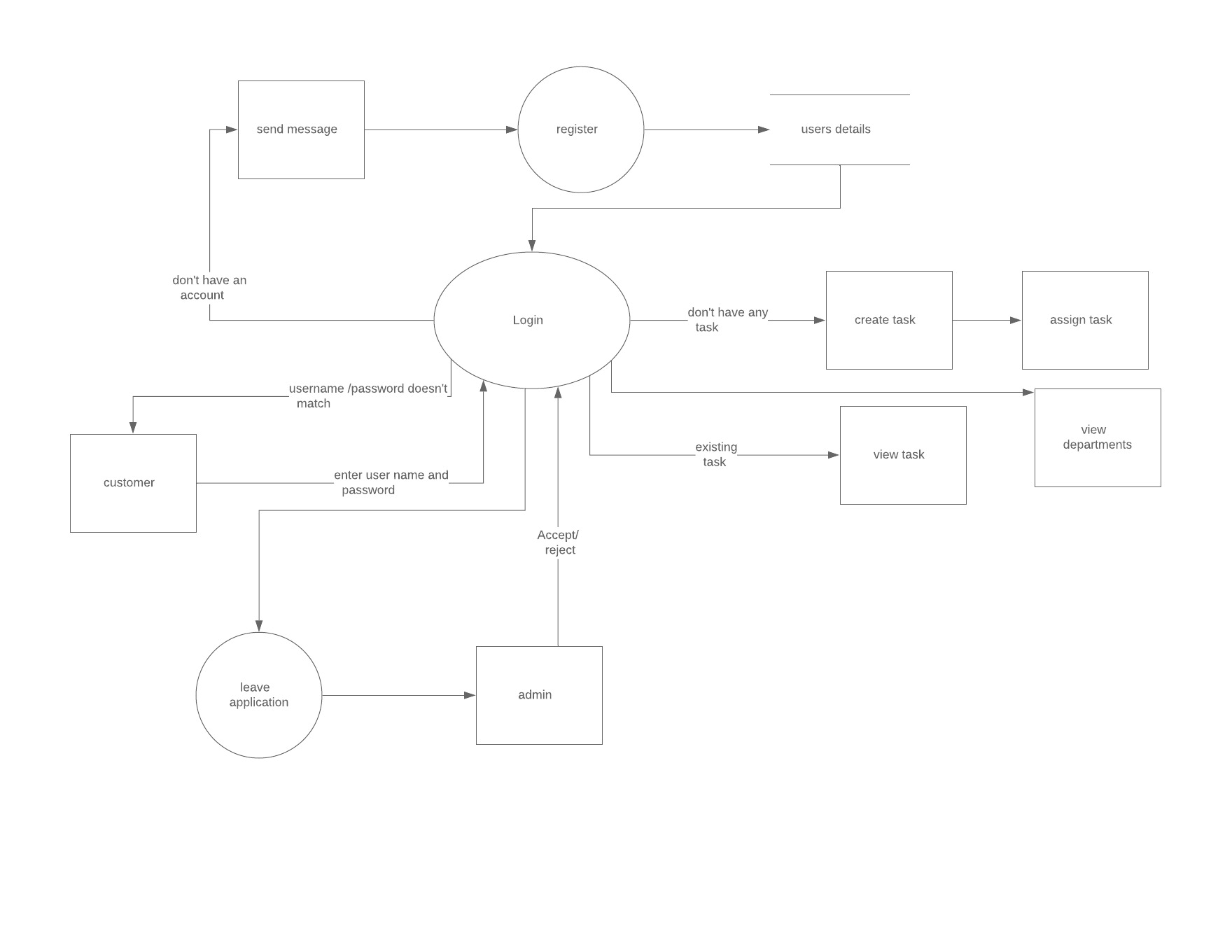
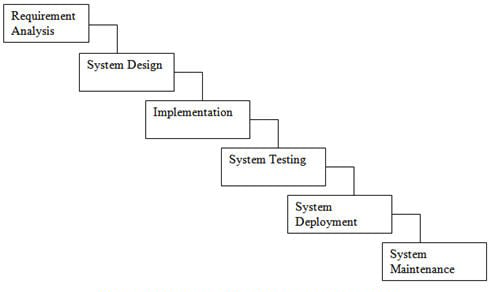


Figure 1 data flow diagram

## 2.2.2 steps in SSADM:



## 2.2.3 Information gathering technique:

For determining the requirement of a system, information must be gathered form the users. The information obtained will aid a well-defined, accurate and complete description of what kind of people, function and data are involved. There are many methods that can be used for gathering information but among those many methods we will use the following techniques:

### interview:

For a proper and accurate information gathering a formal face-to-face meeting is necessary. Here, an interviewer gets a chance to understand stakeholder’s needs. Hence, leading the product to be of better quality. Everything said in an interview should be clearly written which will be later used for product development.

### Questionnaire:

Questionnaire consist of a series of question which will be answered by the product users. Questionnaire can help us (analysts) to study attitudes, beliefs, characteristics and behavior of the stakeholders. Here, Attitude means what user wants in the system; behavior means how the users might use the software; characteristics means what users handle in the organization.

Here are some questions that might be asked in the questionnaire;

1. General profile:
2. Gender:
3. Age group:
4. Experience in work:
5. Approximately how many people are employed in your organization?
6. Nature of the employee for the system:
7. how familiar are you the employee manage system?
8. What are the techniques that are used in your organization for assessing performance/ information?
9. Does your organization use any other management system software?

# 2.3 Feasibility study:

Feasibility study is an analysis that takes all relevant factors of a project into account i.e. economical, technical, legal, social and scheduling factors. Which will determine the likelihood of completing the project successfully. Feasibility studies can be used to separate the pros and cons of undertaking a project before investing time and money into the project.

## 2.3.1 factors of feasibility study:

**I) economic feasibility:**

Also known as cost/benefit analysis, this factor is crucial in determining the likelihood of the completing the project successfully. if befits outweigh costs, then the decision is made to design and implement the system. It is always important to identify cost and benefit factors before undertaking any projects.

**II) legal feasibility:**

This factor determines if the proposed system conflicts with legal requirements, example: a data processing system must comply with the local data protection acts.

**III) schedule feasibility:**

Any software fails if it takes too long time to complete. Whenever scoping a project, the number of resources and initial schedule estimates should be referenced against existing projects. This project may help in many ways. For example, you can always hire more people if you realize that the project is not moving in the expected speed.

**IV) technical feasibility:**

Technical feasibility can be defined as the feasibility that is concerned with specifying equipment and software that will successfully satisfy the user requirement. For example, if you are designing a product does the technology exit to enable it to operate as you envision?

**V) Operational feasibility:**

Operational feasibility is mainly concerned with issues like whether the system will be used if it is developed and implemented. Operational feasibility is a measure of how will a proposed system solves the problem. For example, if the software after development and deployment isn’t as useful for problem solving as it intended it isn’t operationally feasible.

# 2.4 SRS (Software Requirement Specification):

Software Requirement Specification also known as requirements document is a detailed description of a software system to be developed with its functional and non-functional requirements. It is an agreement between customer and contractor. This may include the use cases of how the user is going to interact with software system. A good SRS defines how software system will interact with all internal modules, hardware, communication with other program and human interaction with wide range of real-life scenarios.

## 2.4.1 Functional requirements:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Title | Description | Rational | Dependency |
| FR01 | Registration | If new to the system the user can create a new account with this function | To create a new personal account. | N/A |
| FR02 | Login | The user can login to the system with their username and password. | To login to their account. | FR01 |
| FR03 | Logout | With this function a user can logout of their account after logging in. | To logout of their account. | FR02 |
| FR04 | User role check | After logging in the user role will be checked from the database and the user interface will be displayed according to their role. | To check their role in the organization. | FR02 |
| FR05 | Edit profile | After logging in the user can edit any personal information. And admin can edit all information related to all employees’ including their user role type. | To edit information on their account. | FR02 |
| FR06 | Search | HR and admin can search all the employees’ information in the database. Users can write a specific keyword in the search bar and press the available search button. | HR and admin can search employees’ information in the database. | FR02 |
| FR07 | Delete | Users or admin can delete any information or even their own account in the system with this function. | To delete information and account in the system. | FR02 |
| FR08 | Leave application | The user will be able to fill in leave application form in the appropriate fields. | Ask for the leave/day off. | FR02 |
| FR09 | Leave approval | The head of the department or admin will be able to approve leave application based on the reasons stated by the user. | Admin can approve leave asked by the user. | FR08 |
| FR10 | View | This function allows users to view list of information in the system. | View list of information from the database. | FR02 |
| FR11 | Create departments | This function helps to create a group for employee as a department. | Creates group/ department. | FR02 |
| FR12 | View department | The function will allow employees to see which department they are in. | Show user list of departments in an organization. | FR02 |
| FR13 | View task | Different employees have different task assigned. This function will them to see the list of tasks. | Show the list of tasks to the employees. | FR02 |
| FR14 | Assign task | Users can create task and also assign the task to others employee or themselves. | To assign task to other employee. | FR02 |
| FR15 | Edit user roles | Before registering an account, users are allowed to input their roles. This function will allow them to edit that role. | to edit user roles in an organization. | FR02, FR12 |

## 2.4.3 Non-functional requirements:

a) **performance**: here the system must have a good response time, throughput, utilization etc. these are the things that determine the performance of the system. in order to run the software smoothly a good performance of the system is very crucial.

b) **usability**: the software cannot be fully useable if It isn’t user friendly. This means the software must make users feel comfortable and be simple and easy to use.

c) **reliability**: users have to trust the system, even after using it for a long time. In other words, the users must be able to rely on the software for a long time.

d) **security**: the software must be able to provide high security along with providing the service as it is supposed to the users. Example a secure system can be gained by the authorization and authentication during logging in.

e) **recoverability**: when the system crashes or fails while using it, it must be able to recover as much data as possible that It missed. The data must have a backup options during interruptions.

f) **supportability**: the software/ website should be supported in all kind of devices or screen size of the users. i.e. the website must be responsive or supportable in all kind of screen resolution.

g) **availability**: the system should be available when ever the user wants to user it. The system must be able to run in every platform as well.

## 2.4.4 MOSCOW prioritization:

Here I have down the **MoSCoW prioritization** for all the functional requirement that I have listed above. The definition of MoSCoW gives more insight to the importance and priority of the requirements. Below I have written the full form and description of MoSCoW:

**Must Have**: must have means that the function is guaranteed to be in the software.

**Should Have**: this means that the function/ requirement is important but is not vital.

**Could Have**: this means that the requirement is decided to be in the system but isn’t too important even if it’s not delivered.

**Won’t Have**: this means the requirements which are not necessary and won’t be available or deliver to the user.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Requirement | MOSCOW | Rational | Remarks |
| M01 | Registration | Must have | To create a new personal account. |  |
| M02 | Login | Must have | To login to their account. |  |
| M03 | Logout | Must have | To logout of their account. |  |
| M04 | User role check | Could have | To check their role in the organization. |  |
| M05 | Edit profile | Could have | To edit information on their account. |  |
| M06 | Search | Must have | HR and admin can search employees’ information in the database. |  |
| M07 | Delete | Could have | To delete information and account in the system. |  |
| M08 | Leave application | Could have | Ask for the leave/day off. |  |
| M09 | Leave approval | Could have | Admin can approve leave asked by the user. |  |
| M10 | View employees’ name | Should have | View list of information from the database. |  |
| M11 | Create departments | Should have | Creates group/ department. |  |
| M12 | View department | Should have | Show user list of departments in an organization. |  |
| M13 | View task | Must have | Show the list of tasks to the employees. |  |
| M14 | Assign task | Must have | Let employee assign task to other. |  |
| M15 | Edit user roles | Should have | Let users to edit roles in an organization. |  |

## 2.4.4 Hardware specification:

Employee management system or EMS should be able to work on a computer with the following minimum hardware specifications:

OS: Windows XP/Vista/7/8 and Linux

CPU: Pentium III (700MHz) and above

Memory: 128 MB and above

Capacity: 4GB of hard drive

Others: Network interface card, mouse, keyboard, and monitor

## 2.4.5 software specification:

Since EMS application is a web-based application, internet connection must be established. The EMS software personal database model will support MySQL environment as DBMS.

# 2.5 use case diagram:

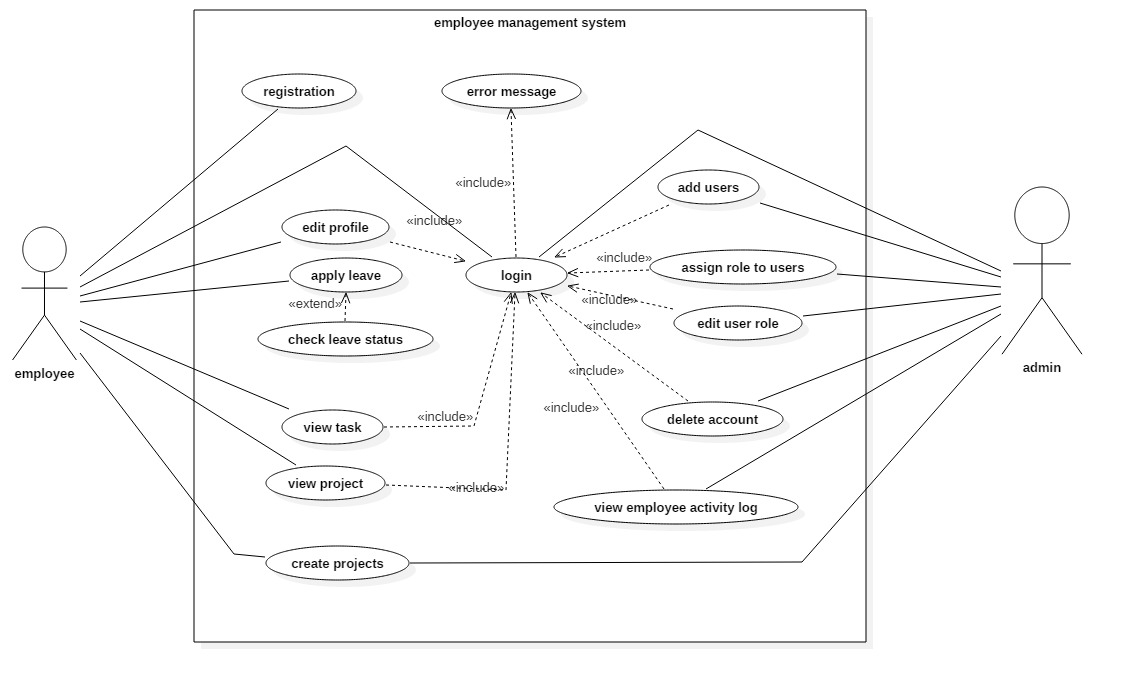


Figure 2 use case diagram

# 2.6 NLA (Natural Language Analysis):

In short NLA or Natural Language Analysis can be defined as the process of distinguishing adjectives, nouns and verbs from the description text where Noun represents classes, Adjectives represent the attributes and Verbs represents the Functions in the system. Below I have followed this process so as to create a proper class diagram from the given scenario:

**Scenario**:

A small company needs a system/software that can help them manage the employees there. The system will have users who can create their personal account. The employee from departments having an account can create projects along with the project members in it. The employee will also be able to create an application asking for a leave. The employee can have and leave notes for other employees too. The users will also have activity log (which includes attendance, leave, task done etc.).

|  |
| --- |
| **List of nouns (classes) from the scenario:** |
| Users | User will be able create a personal account. |
| Departments | User will be in different department. |
| Leave | Employee can ask for leave. |
| Projects | Employee will be assigned to project or create one. |
| Notes | Employee can leave note for one another. |
| Employee | Employee will have different roles. |
| Activity log | Every activity can be monitored. |
| Project members | Every project will have a different member. |
| Task | Employee will have task daily aside from projects. |

## 2.6.1 Initial class diagram:

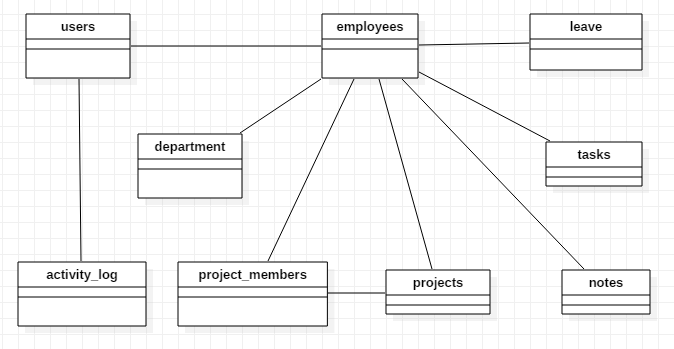


Figure 3 class diagram

# 2.7 system architecture:

To build a system a system architecture must be used so that to handle its object. In this project one among the many architecture has been used i.e. 3-tier architecture.

Three tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are develop and maintained as independent modules on separate platforms. Below is the diagram of three-tier architecture for better understanding:



Figure 4 system architecture

The three-tier architecture consists of three part:

* **Client tier**: this part of the architecture is visible to the end users. Here the users are allowed to input data and also shown data. Here, when the user inputs any data those data are sent to other tier of the architecture.
* **Application tier**: this part of the architecture is responsible for the manipulation of the data provided by the user in the client tier. This tier can also be represented as the logical part of the architecture.
* **Database tier**: the last tier of the architecture is responsible for the basic database operations for functional storage, selection, aggregation, processing and integrity of data.