**PROJECT REPORT**

**ON**

# MANAGE IT

**By**

**VAIBHAVI KHATRI (206330307069)**
**DHRUVESH PATEL (206330307089)**
**DHRUV THAKKAR (206330307099)**

**DEPARTMENT OF COMPUTER ENGINEERING**

L.J. POLYTECHNIC, AHMEDABAD

2022-2023

**DEPARTMENT OF COMPUTER ENGINEERING**

L.J. POLYTECHNIC, AHMEDABAD

2022-2023

# CERTIFICATE

**Date: ___/___/____**

This is to certify that **Ms.** ***Khatri Vaibhavi,*** **Mr.** ***Patel Dhruvesh*** **and Mr.** ***Thakkar Dhruv*** from LJ POLYTECHNIC having Enrollment No. ***206330307069, 206330307089 and 206330307099*** have completed project documentation and partial development on the problem definition of semester V during the academic year 2022-23 having Title ***Manage IT*** in a group consisting of 3 persons.

**Institute Guide** **Head of the Department**

# ACKNOWLEDGEMENT

With due sense of respect and gratitude, we would like to extend our heartiest thanks to all those people who provide us wonderful guidance, support and experience. We will fall short in our duties, if we forget those people, which not only carved this way for us, but were there whenever called for. They created a desire in us to pursue our goal and to attain new heights. We acknowledge the kind assistance and encouragement given by them by giving us valuable formation and clarifying our doubts. Also, the other faculty members and the faculties of our lab have been very kind to us in the competition of our project. These formal thanks are the only "TOKEN" that we can present to these wonderful people to appreciate their support and guidance given by them.

**VAIBHAVI KHATRI (206330307069)**
**DHRUVESH PATEL (206330307089)**
**DHRUV THAKKAR (206330307099)**

# Table of Contents

# Table INDEX

# Figure INDEX

# ABSTRACT

*Employees are the backbone of any company therefore their management plays a major role in deciding the success of an organization. Manage It makes it easy for the admin to keep track of all records. This software allows the administrator to edit employees, add new employees, transfer/promote/terminate employees. Each employee in the database is associated with a position that can be added and edited when need arises. Employees can be transferred between positions easily without having to retype back their information in the database A flexible and easy to use Manage It is the solution for small and medium sized companies provides modules for personnel information management thereby organization and companies can manage the crucial organization asset – people. The combination of these modules into one application assures the perfect platform for re-engineering and aligning Human Resource processes along with the organizational goals. The goal of this project is to design and develop an employee management system to fill existing gaps in the electronic management of employee*.

# CHAPTER-1

# INRTODUCTION

## 1.0 INTRODUCTION

In this world of growing technologies everything has been computerized. With large number of works opportunities, the Human workforce has increased. Thus, there is a need of a system which can handle the data of such many Employees. This project simplifies the task of maintaining records because of its user-friendly nature.

The objective of this project is to provide a comprehensive approach towards the management of employee information. This will be done by designing and implementing an HR management system that will bring up a major paradigm shift in the way that employee information is handled.

The objectives of this system include:

- Design of a web-based HR management system to fulfil requirements such as , leave management, issue management , attendance.
- Well-designed database to store employee information.
- A user friendly front-end for the user to interact with the system.

## 1.1. Need for the New system

Employees are the backbone of any company therefore their management plays a major role in deciding the success of an organization. Employee Management Software makes it easy for the employer to keep track of all records. This software allows the HR to edit employees, add new employees as well as evaluate an employee’s performance. Employees can be managed efficiently without having to retype back their information in the database. You can check to see if there are duplicate positions/employees in the database.

A flexible and easy to use Employee Management software solution for small and medium sized companies provides modules for personnel information management thereby organization and companies can manage the crucial organization asset – people. The combination of these modules into one application assures the perfect platform for re-engineering and aligning

Human Resource processes along with the organizational goals. This system brings about an effortless way of maintaining the details of employees working in any organization.

It is simple to understand and can be used by anyone who is not even familiar with simple employees’ system. It is user friendly and just asks the user to follow step by step operations by giving easy to follow options. It is fast and can perform many operations for a company.

## 1.2. Detailed Problem Definition

Manual handling of employee information poses several challenges. This is evident in procedures such as leave management where an employee is required to fill in a form which may take several weeks or months to be approved. The use of paperwork in handling some of these processes could lead to human error, papers may end up in the wrong hands and not forgetting the fact that this is time consuming. Several current systems lack employee self-service meaning employees are not able to access and manage their personal information directly without having to go through their HR departments or their managers. Another challenge is that multinational companies will have all the employee information stored at the headquarters of the company making it difficult to access the employee information from remote places when needed at short notice.

## 1.3. Viability of the System

This system is expected to be user friendly and will offer easy access to data as well as services such as online leave management, issue management, employee tracking, monitoring employee trainings.

The employee is expected to have direct interaction with this system through a password protected user account therefore proposed system is web based to enable accessibility from any location as long as internet connectivity is available. This direct interaction with the system will enable employee self-service.

Without an employee management system, it is a tedious job for the human resource department to keep track of every employee and even harder for a Admin for assigning department to the employee. The Employee management system will be developed to provide information of employees and many other facilities at the click of a button.

## 1.4. Presently Available Systems for the same

1. monday.com – all-in-one HR software

**[Figure 1: Monday.com]**

2. QuickBooks Time – Best for Employee Time Tracking

**[Figure 2: quickbooks.com]**

3. Gusto – Best HR Software for Small Businesses

**[Figure 3: gusto.com]**

## 1.5. Future Prospects

- Recruitment Process
- Document Repository
- Cloud base System
- Shift Management
- Scheduling Meetings
- Performance Appraisals
- Exit Interview and Termination
- Resume Tracking
- Salary Payment

# CHAPTER-2

# ANALYSIS

## 2.1 Requirement Analysis

### 2.1.1. Functional Requirements

**Authentication**

- Login- The user can login to the Employee management system with his/her username and password.
- Logout- The user can log out from the Employee management system.
- Login failure- If the user does not exist in the database or the user has not yet been authorized by the Employee management system admin.

**Authorization**

- 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.

**Process Data**

- Display- Users with defined roles can display the content of the database. To be more specific, an employee can only view his/her personal information. HR can see his/her personal information but also employee's information who are under his/her department. Admin and HR can display their personal information and all employees' information.
- Edit- A user with an employee role can edit his/her specific personal information. HR can only edit employees' personal information that is

under his/her coverage except user role type. Admin can edit all information related to all employees' including their user role type.

- Search- User with a HR role can search the content of the database for the employees who are under his/her coverage. HR and admin roles can search all the employees' information in the database.

- Update authentication- This feature can be used only by admin role type. Admin can update the role type of a specific user. For example, an employee got a promotion, and his role type will be changed from employee role id to HOD. Admin will be able to update this authentication mechanism.

**Leave Application/Approval**

- Leave application- The user can be able to fill in a leave application form in the appropriate fields.

- Leave approval- The HR can be able to approve leave applications based on the reasons stated, length of leave as well as available leaves.

- Leave days accrued- The user shall be able to check the number of leave days accrued.

**Report generation**

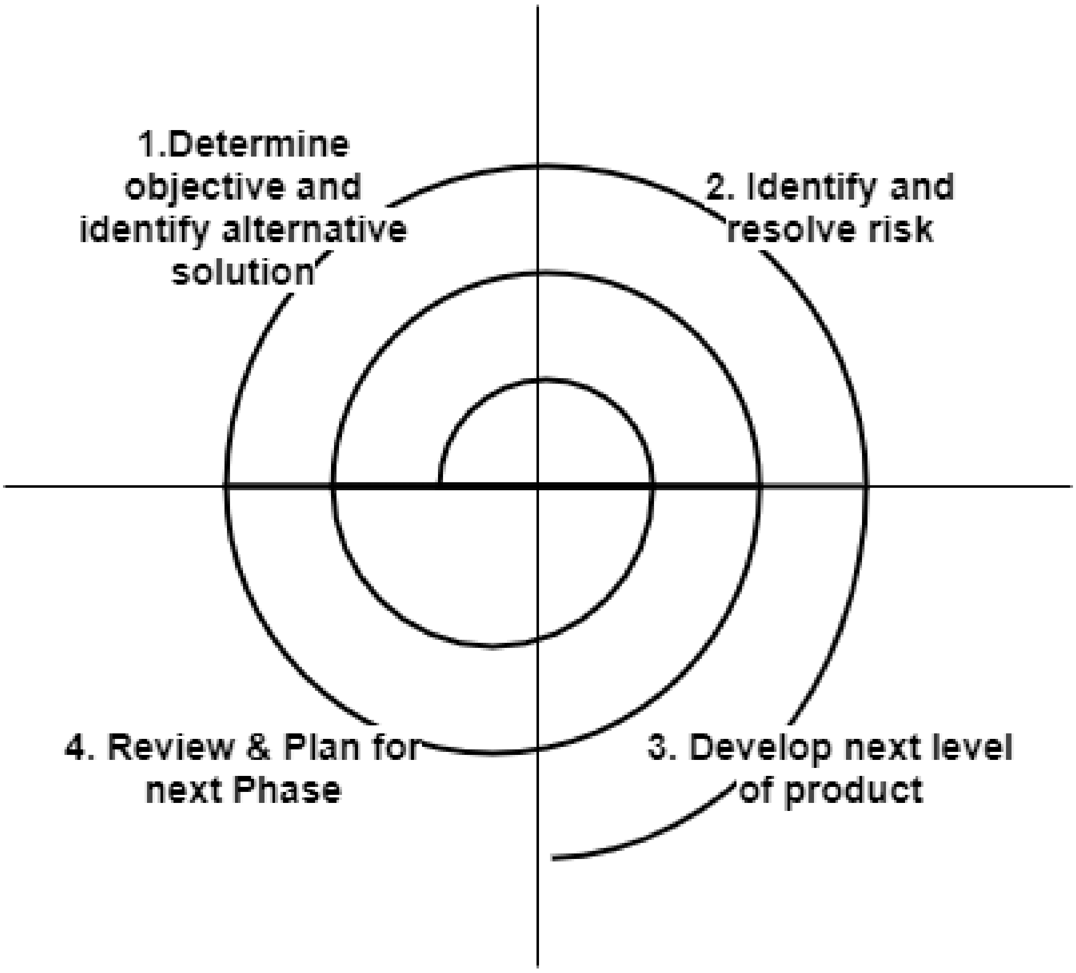
- Report generation- HR will be able to generate a report in pdf format for each employee based on the information in the database.

**Trainings**

- Training: The HOD shall create training and assign employees that are required to attend the training as well.

## 2.2. Project Model

Software Development lifecycle model


1.Determine objective and identify alternative solution
2. Identify and resolve risk
3. Develop next level of product
4. Review & Plan for next Phase

**[Figure 4: Spiral Model]**

The spiral model, initially proposed by Boehm, is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model. It implements the potential for rapid development of new versions of the software. Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.

**Each cycle in the spiral is divided into four parts:**

**Objective setting:** Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists.

**Risk Assessment and reduction:** The next phase in the cycle is to calculate these various alternatives based on the goals and constraints. The focus of evaluation in this stage is located on the risk perception for the project.

**Development and validation:** The next phase is to develop strategies that resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.

**Planning:** Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, if performance or user-interface risks are treated more essential than the program development risks, the next phase may be an evolutionary development that includes developing a more detailed prototype for solving the risks.

**When to use Spiral Model**

- When deliverance is required to be frequent.
- When the project is large
- When requirements are unclear and complex
- When changes may require at any time
- Large and high budget projects

## 2.3 Schedule Representation

Generalized project scheduling tools and technique can be applied with little modification to software projects.

Program evaluation and review techniques (PERT) and critical path method (CPM) are two project scheduling method that can be applied to software development. Both techniques are driven by information already developed in earlier project planning activities:

- Estimate of effort.
- A decomposition of the product function.
- The selection of appropriate process models and task sets.
- Decomposition of tasks.

**[Table 1 - Schedule Representation]**

| ACTIVITY | START DATE | FINISH DATE |
|---|---|---|
| Requirement Analysis | | |
| System Analysis | | |
| System Design | | |
| System Coding | | |
| Testing and Integration | | |

## 2.4 Feasibility Study

**Technically feasible:**

This project is very much technically feasible. This project is very much concerned with specifying equipment and the software will successfully satisfy almost all the admin’s requirements. The technical need for this system may vary considerably but might include:

- The facility to produce output in a specified time.
- Response time under certain conditions.
- Ability to process data at a particular speed.

Therefore, the basic input/output of data is identified. So, the project can easily be built up and it will also be technically feasible.

**Economical Feasibility:**

As our project eliminates the need of labour to manage the employee, the cost of hiring the labour and maintaining them is no longer necessary, only an operator is needed for the system use. The employee data is handled more precisely and consistently.

**Operational Feasibility:**

Due to the project consistency and efficiency of managing the data, it is helpful for the HR team to assign the appraisals fairly. This results in better employ spirits and improved work environment. This also helps the admin to keep track of employee data, their progress and contribution toward the growth of the organization.

# CHAPTER-3

# DESIGN

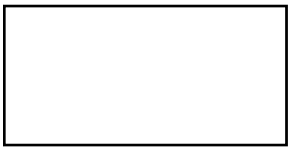
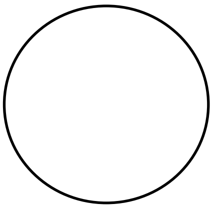
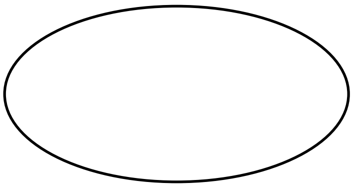
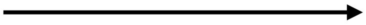
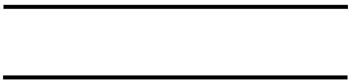
## 3.1 Data Flow Diagram

- DFD (data flow diagram) is also known as bubble chart or data flow graph.
- DFD's are very useful in understanding the system and can be effectively used during analysis. It shows flow of data through a system visually. The DFD is a hierarchical graphical model of a system the different processing activities or functions that the system performs and the data interchange among these functions.
- It views a system as a function that transforms the inputs into desired output.
- Each function is considered as a process that consumes some input data and produces some output data.
- Function model can be represented using DFD.
- DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system.
- The visual representation makes it a good communication tool between User and System designer.
- Structure of DFD allows starting from a broad overview and expand it to a hierarchy of detailed diagrams.

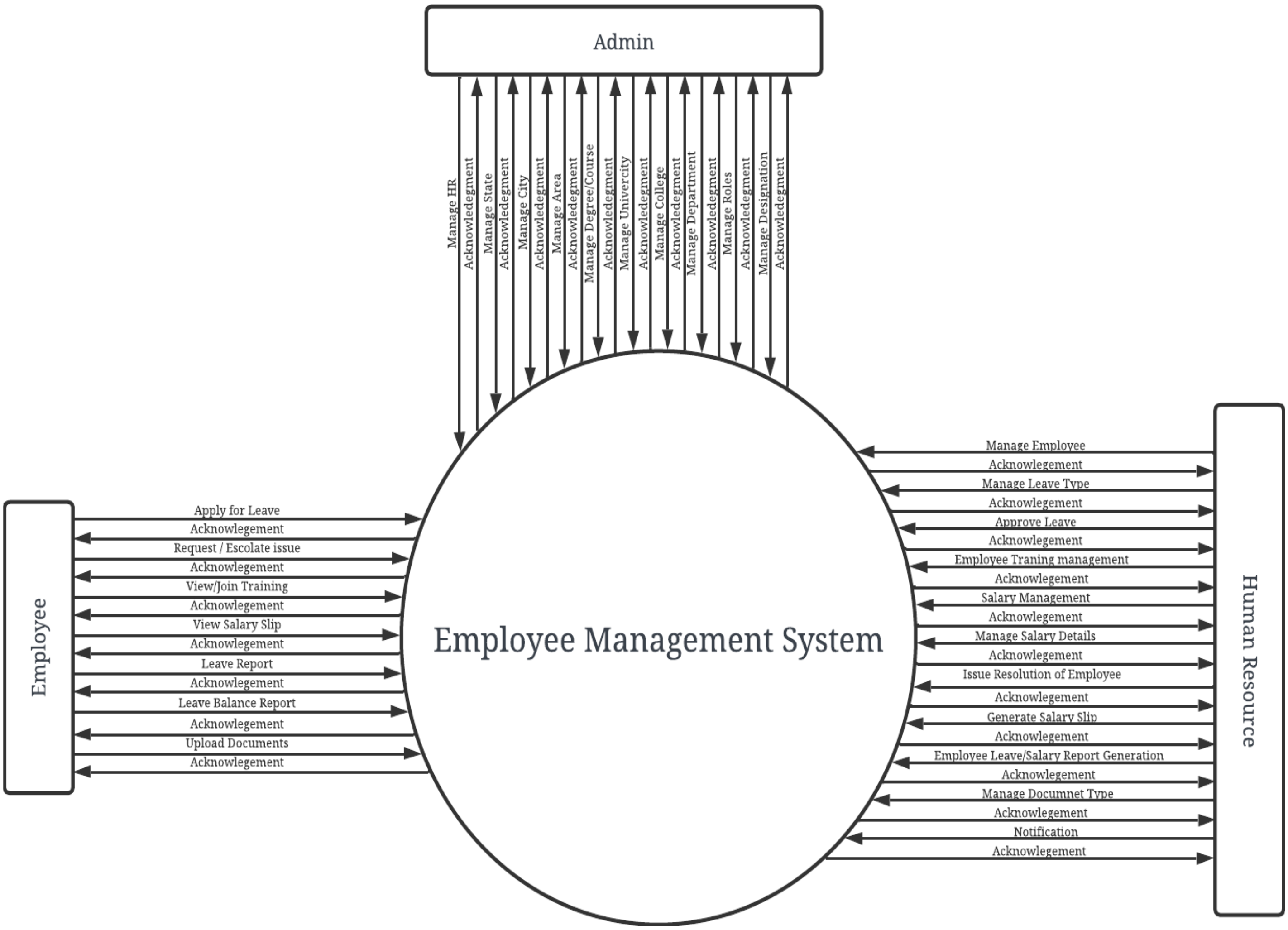
- DFD has often been used due to the following reasons:

  1. Logical information flow of the system.
  2. Determination of physical system construction requirements.
  3. Simplicity of notation.
  4. Establishment of manual and automated systems requirements.

**[Table 2: Data Flow Diagram Symbols]**

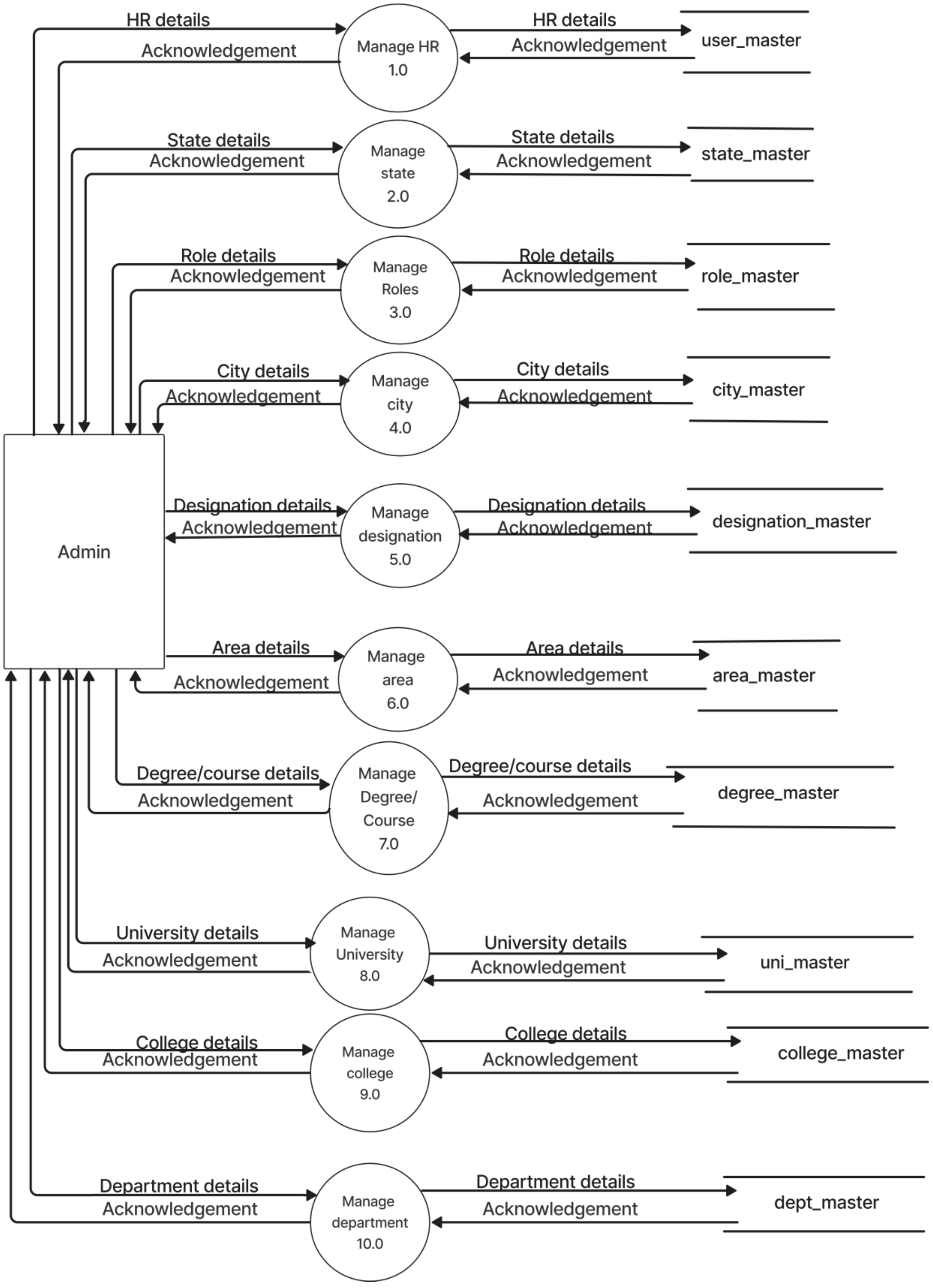
| SYMBOLS | DESCRIPTION |
|---|---|
|  | **Entity:** Entities are external to the system which interacts by inputting the data. |
|  | **System:** It shows the system name. |
|  | **Process:** It shows the part of the system that transforms into outputs. |
|  | **Data Flow:** It passes the data from one part to another. |
|  | **Data Store:** Data store is represented by two parallel lines. It is generally logical file or database. |

## Level 0: context



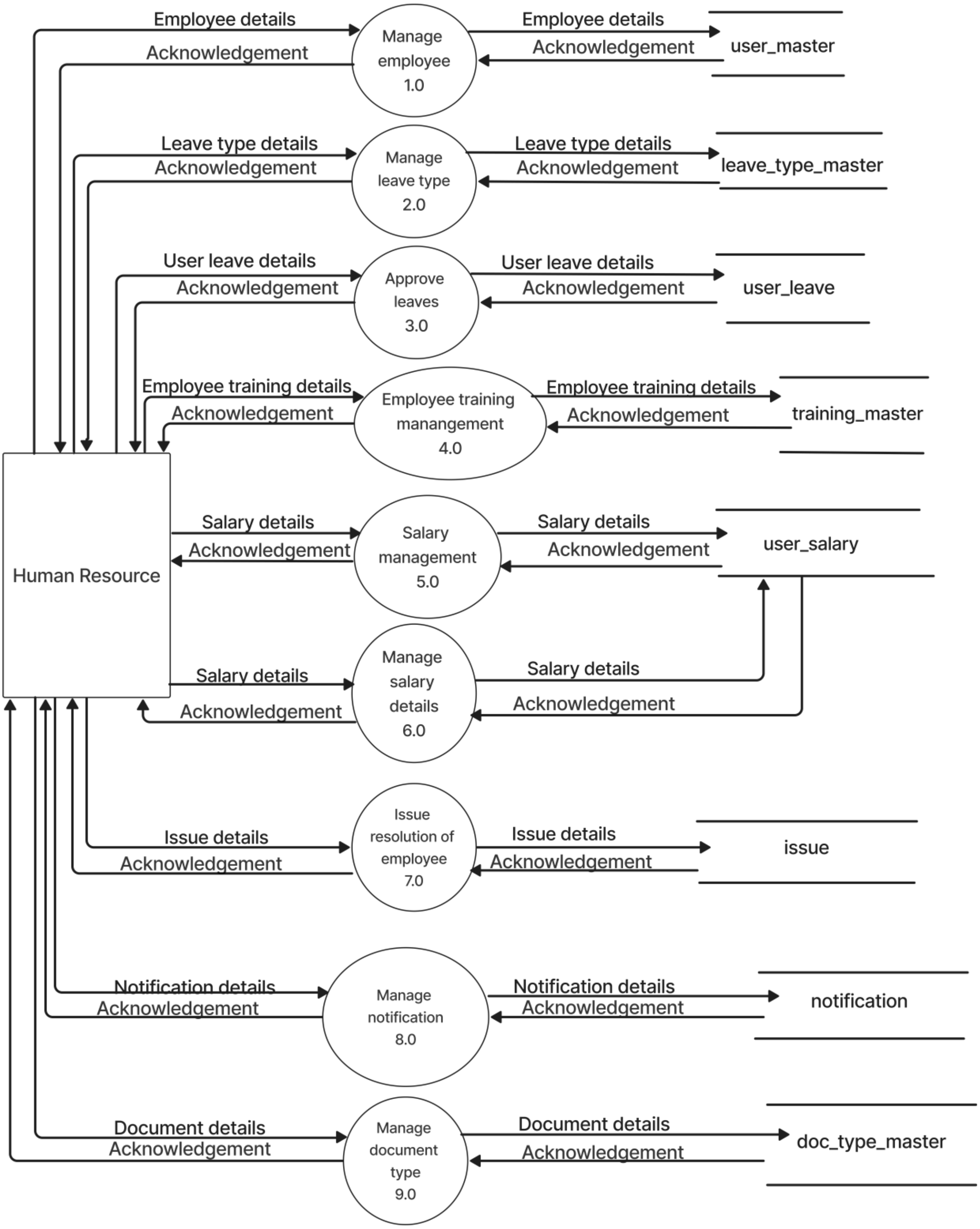
**[Figure 5: Context Level]**

## Level 1: Admin



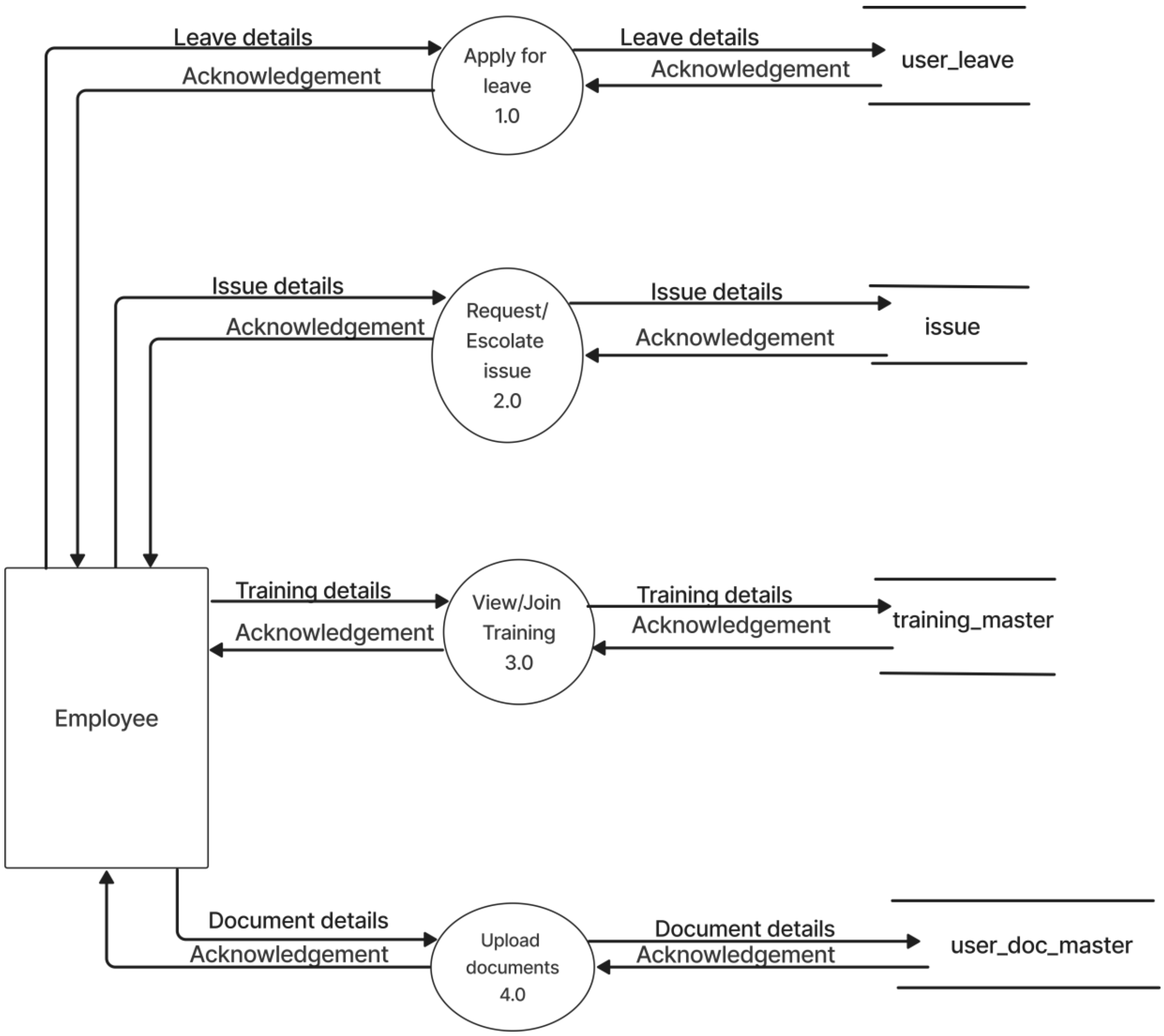
**[Figure 6: Admin Level 1]**

## Level 1: Human Resource



**[Figure 7: HR Level 1]**

## Level 1: Employee



**[Figure 8: Employee Level 1]**

## 3.2 ER-Diagram

In software engineering, an **entity–relationship model** (**ER model**) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database. The main components of ER models are entities (things) and the relationships that can exist among them.

Entity–relationship modeling was developed by Peter Chen and published in a 1976 paper. However, variants of the idea existed previously, and have been devised subsequently such as super type and subtype data entities and commonality relationships.

An entity may be defined as a thing capable of an independent existence that can be uniquely identified. An entity is an abstraction from the complexities of a domain. When we speak of an entity, we normally speak of some aspect of the real world that can be distinguished from other aspects of the real world. Paul Beynon-Davies (2004 Database Systems. Hound mills, Basingstoke, UK: Palgrave. An entity is a thing that exists either physically or logically.) An entity may be a physical object such as a house or a car (they exist physically), an event such as a house sale or a car service, or a concept such as a customer transaction or order (they exist logically—as a concept). Although the term entity is the one most used, following Chen we should really distinguish between an entity and an entity-type. An entity-type is a category. An entity, strictly speaking, is an instance of a given entity-type. There are usually many instances of an entity-type. Because the term entity-type is somewhat cumbersome, most people tend to use the term entity as a synonym for this term.

Entities can be thought of as nouns. Examples: a computer, an employee, a song, a mathematical theorem.

A relationship captures how entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. Examples: an owns relationship between a company and a computer, a supervise relationship between an employee and a department, performs relationship between an artist and a song, a proved relationship between a mathematician and a theorem.

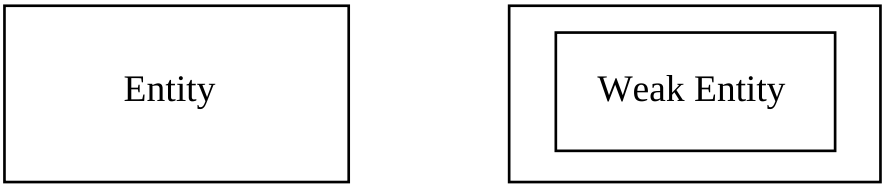
The model's linguistic aspect described above is utilized in the declarative database query language ERROL, which mimics language constructs. ERROL's semantics and implementation are based on reshaped relational algebra (RRA), a relational algebra that is adapted to the entity– relationship model and captures its linguistic aspect.

Entities and relationships can both have attributes. Examples: an employee entity might have a Social Security Number (SSN) attribute; the proved relationship may have a date attribute.

Every entity (unless it is a weak entity) must have a minimal set of uniquely identifying attributes, which is called the entity's primary key.

## ENTITIES:-

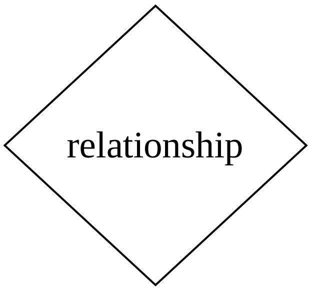
- **Entities** are objects or concepts that represent important data. They are typically nouns, e.g., customer, supervisor, location, or promotion.
- **Strong entities** exist independently from other entity types. They always possess one or more attributes that uniquely distinguish each occurrence of the entity.
- **Weak entities** depend on some other entity type. They don't possess unique attributes (also known as a primary key) and have no meaning in the diagram without depending on anotheriety. This other entity is known as the Owner.



**[Figure 9:Entities]**

## RELATIONSHIP:-

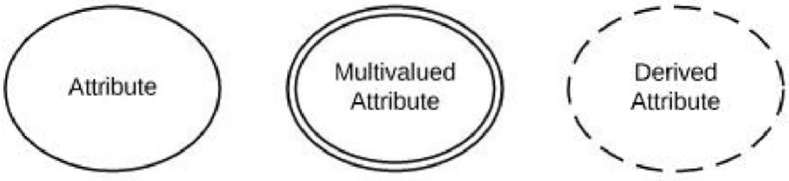
- **Relationship** are meaningful associations between or among entities. They are usually verbs e.g. assign, associate or track. A relationship provide useful information about not be discerned with just entity types.
- **Weak relationships** or identifying relationships that are connections that exists between a weak entity type and its ow.



**[Figure 10:Relationship]**

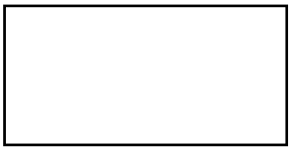
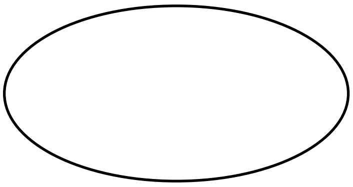
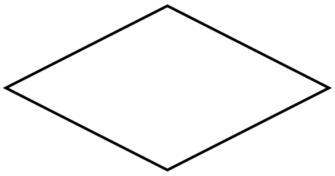
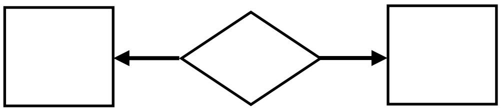
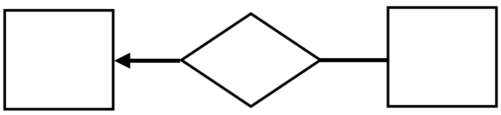
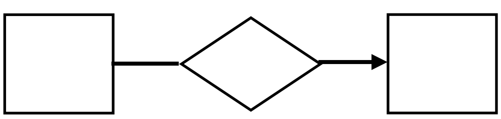
## ATTRIBUTES:-

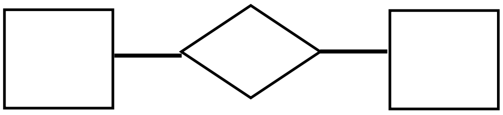
- **Attributes** are characteristics of either an entity, a many to many relationships, or a one-to-one relationship.
- **Multivalued Attributes** are those that can take on  more than one value.
- **Derived attributes** are attributing whose value can be calculated from related attribute values.



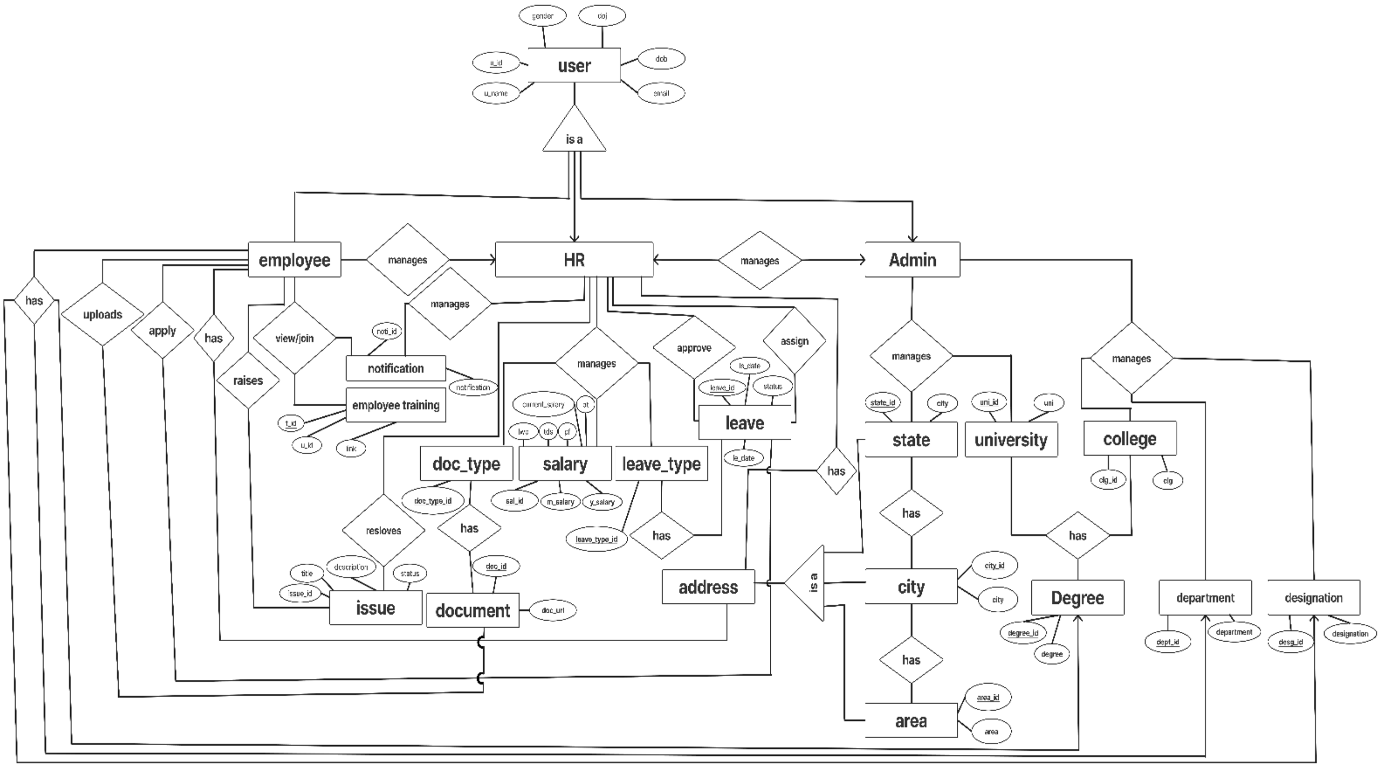
**[Figure 11 - Attributes]**

**[Table 3: ER-Diagram Symbols]**

| SYMBOLS | DESCRIPTION |
|---|---|
|  | **Entity:** Data object is real world entity or thing. It is represented by a rectangle shape. An entity is an object or concept about which you want to store information. |
|  | **Attributes:** An attribute is property of characteristic of an entity. It is represented by oval shape. |
|  | **Relationship:** Entity are connected each other via relations. Generally, relationships in binary because there are two entities are related to each other. |
|  | **Cardinality (One to One):** An instance of entity A can relate to one instances of entity B. |
|  | **Cardinality (One to Many):** An instance of entity A can relate to one or many instances of B but we can only relate one instance of A. |
|  | **Cardinality (Many to One):** One or more instances of entity A can relate to one instances of B. |

| | |
|---|---|
|  | **Cardinality (Many to Many):** One or more instances of entity A can relate to one more instance of entity B. |

## ER-Diagram:



[Figure 12: ER Diagram]

# CHAPTER-4

# SYSTEM MODLING

## 4.1 Database Design

1. **Table Name:** state_master

   **Primary Key:** state_id

**[Table 4: state_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | state_id | Int | Primary key | State id |
| 2 | state | Char (20) | Unique | State names |

2. **Table Name:** city_master

   **Primary Key:** city_id

   **Foreign Key:** state_id

**[Table 5: city_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | city_id | Int | Primary Key | City id |
| 2. | city | Char | Not Null | City name |
| 3. | state_id | Int | Foreign Key | State id |

3. **Table Name:** area_master

   **Primary Key:** area_id

   **Foreign Key:** city_id,state_id

**[Table 6: area_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | area_id | Int | Primary key | area id |
| 2 | area | varchar (20) | Not Null | area name |
| 3 | city_id | Int | Foreign Key | City id |
| 4 | state_id | Int | Foreign key | State id |

4. **Table Name:** doc_type_master

   **Primary Key:** doc_type_id

**[Table 7: doc_type_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | doc_type_id | Int | Primary key | Document id |
| 2 | type | Varchar | Not Null | Type of the document |

5. **Table Name:** degree_master

   **Primary Key:** degree_id

**[Table 8: degree_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | degree_id | Int | Primary key | degree id |
| 2 | degree | varchar (20) | Not Null | Degree names |

6. **Table Name:** uni_master

   **Primary Key:** uni_id

**[Table 9: uni_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | uni_id | Int | Primary key | uni id |
| 2 | uni | Char (20) | Unique | University names |

7. **Table Name:** college_master

   **Primary Key:** clg_id

**[Table 10: college_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | clg_id | Int | Primary key | College id |
| 2 | clg | Char (20) | Unique | College names |

8. **Table Name:** role_master

   **Primary Key:** role_id

**[Table 11: role_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | role_id | Int | Primary key | Role id |
| 2 | role | varchar (20) | Not Null | Role name |

9. **Table Name:** dept_master

**Primary Key:** dept_id

**[Table 12: dept_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | dept_id | Int | Primary key | Department id |
| 2 | dept | varchar (20) | Not Null | Department name |

10. **Table Name:** leave_type_master

**Primary Key:** leave_type_id

**[Table 13: leave_type_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | leave_type_id | Int | Primary key | Leave type id |
| 2 | leave_type | Varchar | Not Null | Type of the leave |
| 3 | short_form | Varchar | Not Null | Short form of leave Type |

11. **Table Name:** user_master

**Primary Key:** u_id

**Foreign Key:** desg_id, state_id, city_id, area_id, role_id, dept_id

**[Table 14: user_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | u_id | Int | Primary key | user id |
| 2 | u_name | char (20) | Not Null | username |
| 3 | u_email | Varchar () | Not Null | User email |
| 4 | u_pass | varchar () | Not Null | User password |
| 5 | desg_id | varchar () | Foreign Key | Designation id |
| 6 | u_gender | varchar () | Not Null | User Gender |
| 7 | state_id | Int | Foreign key | State id |
| 8 | city_id | Int | Foreign Key | City id |
| 9 | area_id | Int | Foreign key | area id |
| 10 | role_id | Int | Foreign key | Role id |
| 11 | contact | Int (10) | Not Null | User contact Number |
| 12 | is_active | Boolean () | Not Null | Is user active |
| 13 | doj | DateTime() | Not Null | User date of joining |
| 14 | dob | DateTime | Not Null | User birthdate |
| 15 | dept_id | Int | Foreign key | Department id |

**12.Table Name:** designation_master

**Primary Key:** desg_id

**[Table 15: designation_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | desg_id | Int | Primary Key | Designation id |
| 2 | designation | Char | Not Null | Designation |

**13.Table Name:** training_master

**Primary Key:** t_id

**Foreign Key:** u_id

**[Table 16: training_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | t_id | Int | Primary Key | Training id |
| 2 | u_id | Int | Foreign Key | User Id |
| 3 | t_url | Varchar | Not Null | Training session link |
| 4 | t_descr | Varchar | Not Null | Training description |
| 5 | t_date | DateTime() | Not Null | Training session date |
| 6 | t_time | DateTime() | Not Null | Training session time |

**14.Table Name:** user_doc_master

**Primary Key:** notif_id

**Foreign Key:** u_id

**[Table 17: user_doc_master]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | doc_id | Int | Primary Key | Document Id |
| 2 | u_id | Int | Foreign Key | User Id |
| 3 | doc_type_id | Int | Foreign key | Document Type id |
| 4 | doc_no | Int | Not Null | Document Number |
| 5 | Doc | Varchar | Not Null | Document’s URL |

**15.Table Name:** user_education

**Primary Key:** u_edu_id

**Foreign Key:** u_id, degree_id, uni_id, clg_id

**[Table 18: user_education]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | u_edu_id | Int | Primary key | User education id |
| 2 | u_id | Int | Foreign Key | User Id |
| 3 | degree_id | Int | Foreign key | Degree id |
| 4 | uni_id | Int | Foreign key | University id |
| 5 | clg_id | Int | Foreign key | College id |
| 6 | degree_certi_doc | varchar | Not Null | User education degree document |
| 7 | passout | Int | Not Null | College passout Year |
| 8 | grade | Char | Not Null | User Grades |

**16.Table Name:** user_professional_detail

**Primary Key:** prof_id

**Foreign Key:** u_id

**[Table 19: user_professional_detail]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | prof_id | int | Primary Key | Profession id |
| 2 | u_id | Int | Foreign Key | User Id |
| 3 | employer | Char | Not Null | User Last company name |
| 4 | prev_doj | DateTime | Not Null | Previous company date of joining |
| 5 | prev_dol | DateTime | Not Null | Previous company date of leave |
| 6 | salary | Int(5) | Not Null | Previous company salary |
| 7 | prev_designation | Char | Not Null | Previous company designation |

**17.Table Name:** user_leave

**Primary Key:** leave_id

**Foreign Key:** leave_type_id, u_id

**[Table 20: user_leave]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | leave_id | Int | Primary key | User leave id |
| 2 | leave_type_id | Int | Foreign key | Leave type id |
| 3 | u_id | Int | Foreign Key | User Id |
| 4 | ls_date | DateTime() | Not Null | User Leave start date |
| 5 | status | Boolean | Not Null | Approved status of leave |
| 6 | no_of_leave | Int | Not Null | Number of leave |
| 7 | le_date | DateTime() | Not Null | User leave ending date |
| 8 | la_date | DateTime() | Not Null | User leave approval date |
| 9 | leave_reason | Varchar | Not Null | User leave reason |
| 10 | rej_reason | Varchar | Not Null | User leave rejection reason |

**18.Table Name:** leave_balance

**Primary Key:** bal_id

**Foreign Key:** u_id, leave_type_id

**[Table 21: leave_balance]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | bal_id | Int | Primary key | Balance Id |
| 2 | u_id | Int | Foreign Key | User Id |
| 3 | leave_type_id | Int | Foreign key | Leave type id |
| 4 | lv_balance | Int | Not Null | Amount of leave left |

**19.Table Name:** user_salary

**Primary Key:** sal_id

**Foreign Key:** u_id

**[Table 22: user_salary]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | sal_id | Int | Primary key | Salary id |
| 2 | u_id | Int | Foreign Key | User Id |
| 3 | m_salary | Int | Not Null | Month salary |
| 4 | y_salary | Int | Not Null | Year salary |
| 5 | work_days | Int | Not Null | Number of working days |
| 6 | lv_days | Int | Not Null | Number of leave days |
| 7 | lwp | Int | Not Null | Lwp amount for employee |
| 8 | current_salary | Int | Not Null | current employee salary |
| 9 | tds | Int | Not Null | TDS amount for employee |
| 10 | pf | Int | Not Null | pf amount for employee |

| | | | | |
|---|---|---|---|---|
| 11 | pt | Int | Not Null | pt amount for employee |
| 12 | net_payable_salary | Int | Not Null | Employee Net Payable Salary |

**20.Table Name:** issue

**Primary Key:** issue_id

**Foreign Key:** u_id

**[Table 23: Issue]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | issue_id | Int | Primary Key | Issue id |
| 2 | u_id | Int | Foreign Key | User Id |
| 3 | issue_desc | Varchar | Not Null | Issue description |
| 4 | title | Char | Not Null | Issue Title |
| 5 | issue_date | DateTime | Not Null | Issue apply date |
| 6 | issue_status | Boolean | Not Null | Issue status |

**21.Table Name:** notification

**Primary Key:** notif_id

**Foreign Key:** u_id

**[Table 24: notification]**

| SR.NO | FIELD NAME | DATATYPE(SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | notif_id | Int | Primary Key | Notification Id |
| 2 | Notification | Varchar | Not Null | Notification content |
| 3 | u_id | Int | Foreign Key | User Id |

# CHAPTER-5

# TECHNICAL SPECIFICATION

## 5.1 Hardware Specification:

**5.1.1 Ram:** 4GB

**5.1.2 Hard drive Storage Needed:** 200GB

**5.1.3 Other Hardware Requirements:** None

## 5.2 Platform:

**5.2.1 Supported Operating System:** Windows XP and above LINUX and MacOS is compatible.

**5.2.2 Programmer Server:** Wamp Apache Server 3.1.9 64-bit and 2.2e.

**5.2.3 Framework:** Bootstrap v5.0

## 5.3 Programming Languages used:

**5.3.1 Mark-up Language:** HTML4 and HTML5.

**5.3.2 Programming Language:** PHP 5.3

## 5.4 Technical Specification:

**5.4.1 Front-End:** PHP 5.3v

**5.4.2 Back-End:** MySQL5.5.24

**5.4.3 IDE:** Visual Studio Code.

**5.4.4 UML Tools:** Figma.

**5.4.5 SRS Tools:** Microsoft Word 2021.

## 5.5 DESIGN LAYOUT

**[Figure 13: Admin sign in page]**

**[Figure 14: Admin Registration page]**