# **Chapter 3: Design**

## **3.1 Introduction of Design**

Design phase is a major phase in software development life cycle where **architecture** is established. It breaks down the requirements and leads us from **problem domain** to **solution domain**. In this phase, required level of effort and amount of resources can be easily estimated with application workflows. It represents the **‘how’** phase; where components, interfaces, behaviors, etc. are planned to implement for the development of best application.

It explains in detail about the required specifications, features and operations as a functional requirements that helps in determining needs for the proposed application or system. This phase helps in considering essential components as hardware and software structure with their processing and procedures for accomplishment of the systems objective. It includes **structural** model, **behavioral** model, **database** model, **architectural** model and **UI design** which will be designed using various drawing tools as **Star** **UML, Balsamiq** and **Visual Paradigm**.

## **3.2 Modeling Software**

Modeling software is a modeling language that builds simulations or other models which expresses information in a structured way. It is an abstract model of a system that clary defines the systems perspective. It can be developed from existing system as well as new system which helps in implementation phase in future. It helps the analyst to understand the models and functionalities of system that are used for communication with clients.

## **3.3 Structural Model**

Structural model is both **static** and **dynamic model** that shows a structure of a system in one hand and the executing time period in another hand. It represents the system’s components and relationships which are created from discussion and designing the architecture of system or application. Likewise, I have chosen Class Diagram and Context Diagram among all structural model.

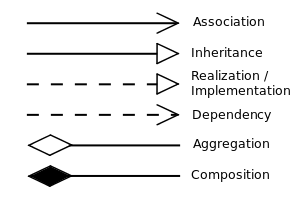
**3.3.1 Class Diagram**

Class diagram is a dependency among classes in **Unified Modeling Language (UML)** from illustrated relationships and source code. These are used while developing an object-oriented system model to show the classes and **association** between them. It is useful in Object-oriented programming as OOP modeling paradigms. The classes are arranged in groups in a class diagram for sharing characteristics which resembles flowchart that contains methods, attributes and operations. It is also a **blueprint** of an object-oriented modeling.

**Reason behind choosing Class Diagram as a Structural Diagram**

Class diagrams provides insight to the system structure and gives a sense of orientation. Class diagrams are simple, fast and easy to read. It is a foundation point for creating any system. It helps in visualizing the paths between classes which takes various forms of generalization. It is a major part during development of system that illustrates the complete structure of the system. Class diagram displays **attributes, operations** and their logical and statistical **inter-relationships**.

**Notations in class diagram**

****

The notations defines the relationships between classes which are of many types such as association, inheritance, realization, dependency, aggregation and composition. Among them, the notations applied on this methodology are shown in the table below:

|  |  |  |
| --- | --- | --- |
| Name | Notation | Description |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Final Class Diagram and Description**

**3.3.2 Context Diagram**

The context diagram is a single high-level process that goes under system consideration showing relationship having external entities as organizational groups, external data stores and many more. It supplies and receives information to and from system boundary which is also called **Context-Level Data-Flow Diagram** or **Level 0-Data Flow Diagram.** It is created for system analysis and design that helps to communicate interactions and data-flow between various business processes.

**Reason behind choosing Context Diagram as a Structural Diagram**

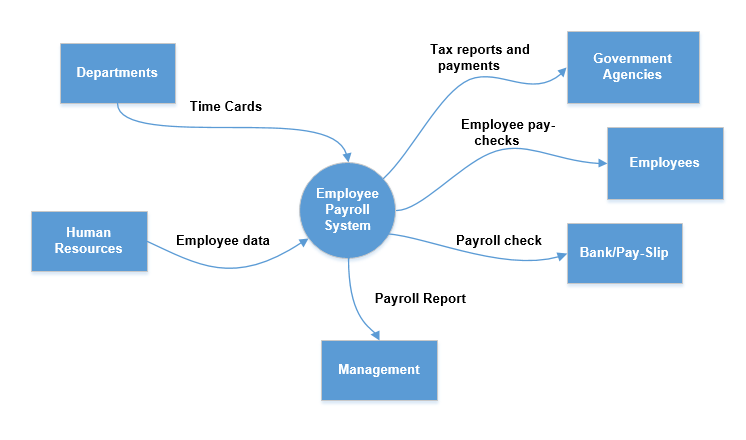
There are many benefits of context diagram that supports during implementation of the system where scopes and boundaries of system are clearly shown. Most important, it is easy to draw and expand with limited notations. For context diagram, there is no need of technical knowledge and understanding.

**Notations in context diagram**

The notations applied in context diagram are represented below with its description.

|  |  |  |
| --- | --- | --- |
| Name | Notation | Description |
| External Entity |  | It is a data destination represented as external data source. |
| Process |  | It is a process to indicate the scope of the system and actual composition of the system. (how it works.) |
| Data flow |  | It shows a direction of flow of data represented as straight or curved line with arrow head. |

**Context Diagram and Description**

****

The figure displays overall scenario of the employee payroll system **EPS** that depicts the input and output on the basis of their particular roles. Here, the department figures out the individual employee working basis per month or week or a day from different departments. Human resources department stores the employee’s details and the individual salary will be figured out through the management with overall allowance and deductions. Finally, with tax payments the total salary will be provided to the employee through either pay-slip or directly bank deposit. The management also provides the detailed reports of the employees.

## **3.4 Behavioral Model**

Behavioral model is an internal dynamic aspects of the system supporting business processes describing internal logic without specifying implementation process. The detailed design is fully specified for design and implementation phases. In this model, UML (Unified Modeling Language) diagrams such as sequence diagrams, communication diagrams and behavioral state are used as well as **CRUDE (Create, Read, Update, Delete, Execute)** operations are used.

**3.4.1 Activity Diagram**

Activity diagram is an UML diagram that describes dynamic aspects of application or system. As it represents flow of activities, it describes an operation of the system and also known as a flowchart.

**Reason behind choosing Activity Diagram as a Behavioral Diagram**

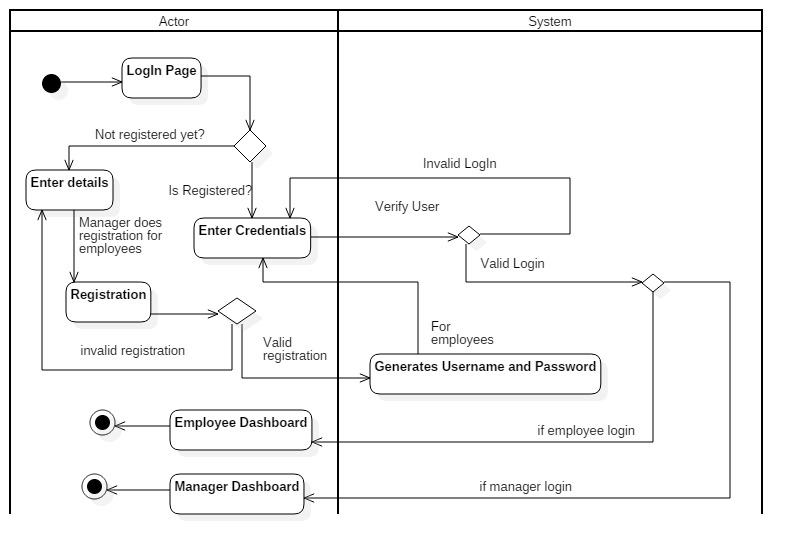
Activity diagram are easily comprehensive for both stakeholders and analyst and it is easy to understand the work-flow of the system. The diagram is very user-friendly and also regarded as important tool for analysts. It allows for multiple conditions and actors using swimlanes within a work-flow.

**Notations in activity diagram**

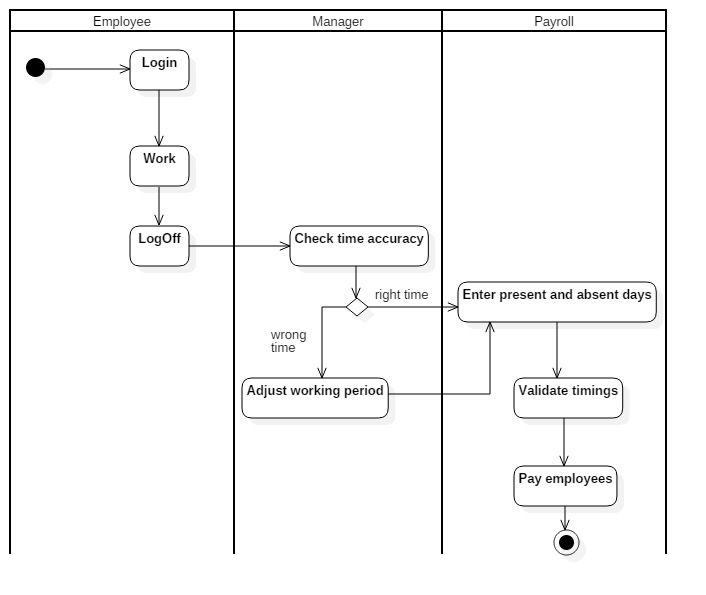
The notations that can be applied for activity diagram are as represented in tabular form below:

|  |  |  |
| --- | --- | --- |
| Name | Notation | Description |
| Activity |  | It specifies and organizes the subordinate behaviors reflecting the control and data flow of process. |
| Action |  | It is a basic process occurring within a system having particular functions. |
| Control Flow |  | It connects nodes modeling a active transition. |
| Initial Node |  | It defines the start flow while activity gets invoked. |
| Final Node |  | It indicates the completion of activity and diagram gets aborted. |
| Decision Node |  | It indicates the condition with different choices. |
| Fork Node |  | It spits a single flow into more than one flow showing different activities. |
| Join Node |  | It connects different activities to get together in a single activity. |
| Swimlane and Partition |  | These are such activities which are performed by same actor. |
| Send |  | The action represents sending signal or signal. |
| Receive |  | This action represents receiving or accepting the signal or request. |
| Region |  | It is an interruptible activity region. |

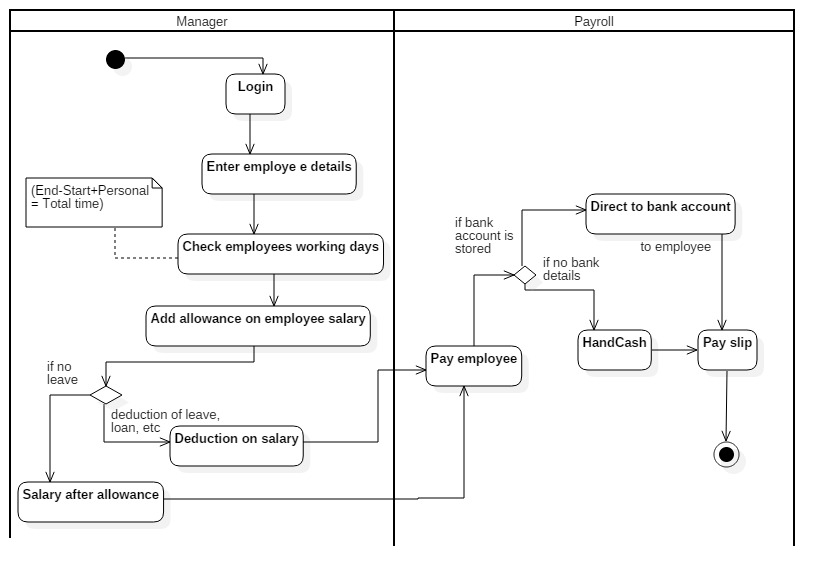
**Activity Diagram and Description**

****

In this activity diagram, the login and registration process is described in brief where manager registers first with detailed and valid information and then using credentials he/she can go to respective dashboard. (**Admin dashboard**). Similarly, employee’s details can only be registered by manager where username and password is provided for individual employees. The further employee login can be done through same username and password and get to their respective employee dashboard. The password can be updated after successful login.

****

In this activity diagram, there are three swimlanes or partitions as employee, manager and payroll where different operations or functionalities are performed in particular partition. Actions, activity and decisions are applied in this activity diagram with evaluating accurate time period of employees’ working period where over times and leaves are evaluated. Finally, on the basis of timings calculations employees’ salary are paid with pay-slip.

****

In this activity diagram, there are two swimlanes or partitions as manager and payroll where manager sets the employee salary and departments during adding details. The assigned salary gets increased from bonus, medical or many other things. Similarly, the salary gets deducted from work leave or loan taken from organization. Finally, the salary gets calculated and pay-check is provided to the employee either hand cash or directly deposited on bank account. This completes employee payroll system.

**3.4.2 Sequence Diagram**

Sequence diagram are the interaction diagrams that briefly describes about the operations performed in the system that are time focused. It exchanges the messages over time which are also known as event diagrams. It is a good way of visualizing and validating runtime scenarios that helps in modeling new system.

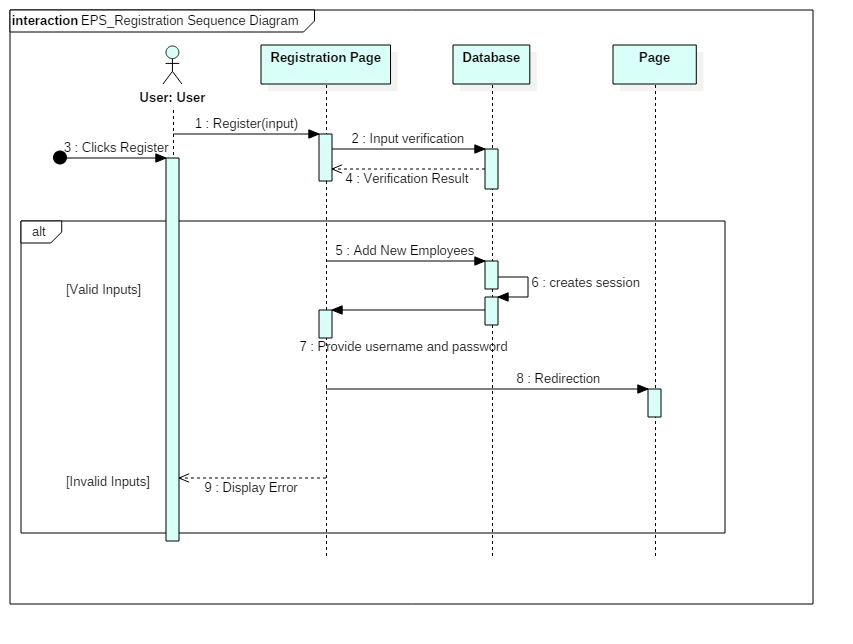
**Reason behind choosing Sequence Diagram as a Behavioral Diagram**

As a behavioral diagram, I have chosen sequence diagram due to its high-level interaction between objects within a collaboration realizing use case, operations and paths or flows. Moreover, sequence diagram are easier to read and understand that allows reverse engineering. The UML specification is more sequence diagram centric and are excellent for documentation purpose. The diagram is arranged in an time sequence where messages are exchanged between objects.

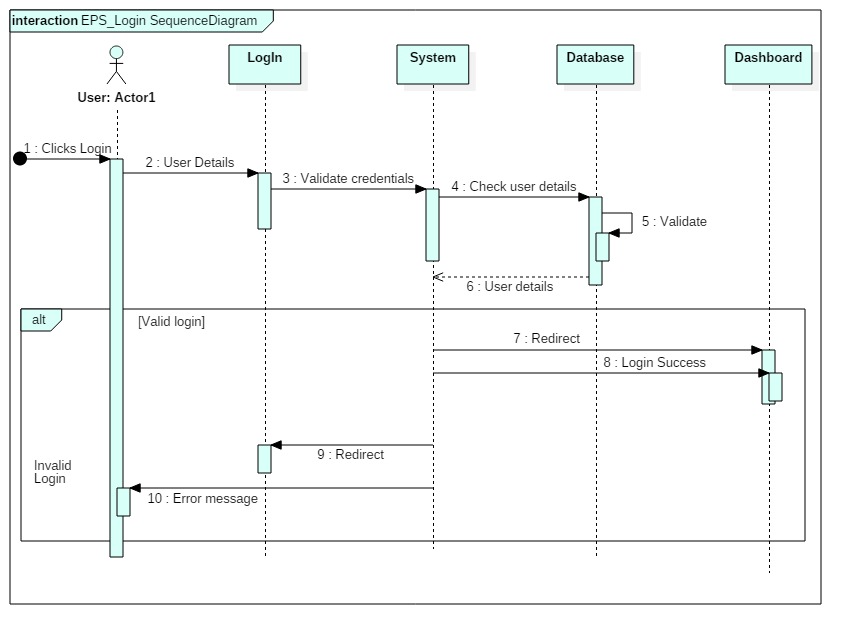
**Notations in sequence diagram**

|  |  |  |
| --- | --- | --- |
| Name | Notation | Description |
| Actor |  | Actor plays a role of interacting with the subject by exchanging data or signals. It represents human users, external hardware and many more. |
| Lifeline |  | It represents interaction within a individual participants. |
| Activations |  | It is like a lifeline with thin rectangle representing time period for an operation. |
| Call Message |  | It represents a communication between lifelines of interaction showing an invocation of operation. |
| Return message |  | It represents pass of the information back to the caller of a corresponded former message. |
| Self-Message |  | It represents the invocation of message of same timeline. |
| Recursive Message |  | It represents the invocation of message of same lifeline targeting to the activation on top where message was invoked from. |
| Create Message |  | It is a kind of message that represents instantiation of targeted lifeline. |
| Destroy message |  | It represents the request destroying the lifecycle of target lifeline. |
| Duration message |  | It represents the distance between two time instants for a message invocation. |
| Note |  | It is also a comment that contains information which is useful to a modeler. |
| Sequence Fragments |  | It makes easier to create and maintain accurate sequence diagrams which are represented as a box (**combined fragment**). |

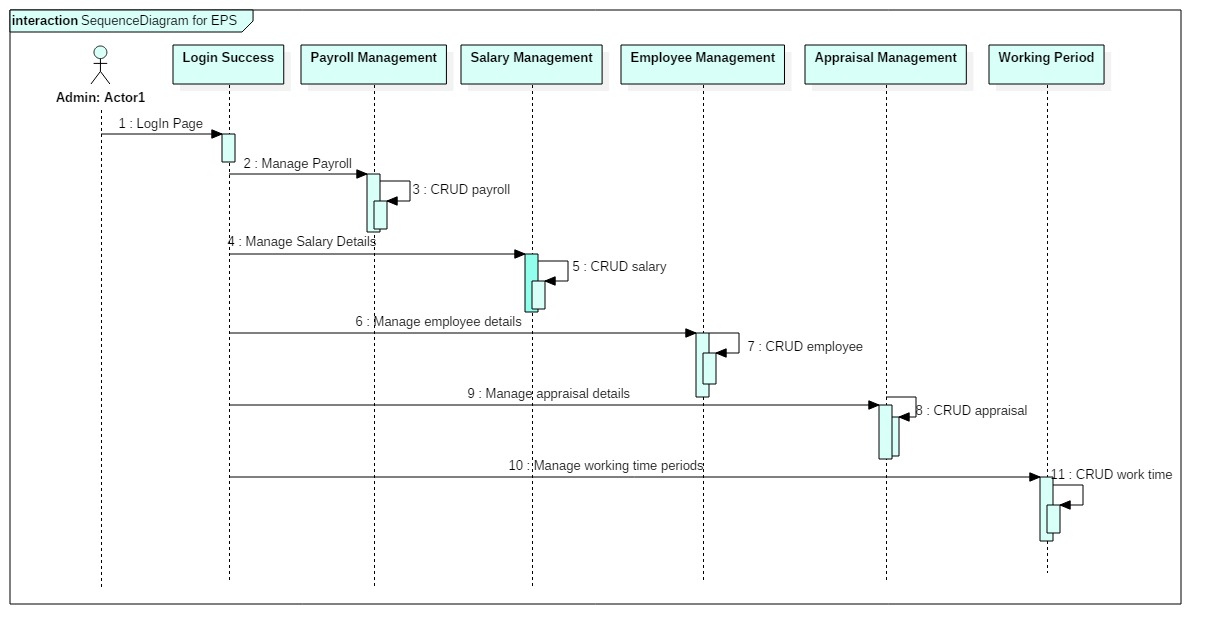
**Sequence diagram and descriptions**

****

In this sequence diagram, the registration process of employee payroll system is represented where valid inputs are only allowed and invalid displays error. During registration, valid details needs to be stored and with final registration success the system provides username and password to the user. Furthermore, the username and password are applied during login the system.

****

In this sequence diagram, the login system of the EPS is described clearly where user enters their respective username and password which will be validated to data stored on database and after correct validation the respective dashboard gets open. In case of incorrect validations, the system shows an error message. This diagram contains actors, lifelines, messages and self-message.

****

In this above sequence diagram, the complete employee payroll system is displayed where actors, lifeline and messages are clearly described which are easy to read. This diagram shows the payroll system where employees, salary, payroll, appraisals and working period are figured out with having CRUD operations. This completes the payroll system.

## **3.4 Database Modeling**

A database modeling is the modeling of a database diagram that represents a **logical structure** of database along with their relationships. The structure on the database determines the way of storing and accessing data.

**3.4.1 Data-Dictionary**

Data-Dictionary is also called data repository or system catalog which is an integral part of database that holds information about the database which is **metadata.** It is important part of DBMS **(Database Management System)** which contains actual database descriptions used by DBMS. The data dictionary is used to retain and handle database and control entrance which consist contents like column name, data type, length, constraints, nullable and many more.

**Table 1: Employee**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type | Length | Constraints | Nullable |
| EmployeeID | Integer | 10 | Primary Key [PK] | No |
| FullName | Varchar | 50 | - | Yes |
| DOB | Date | - | - | Yes |
| Gender | Varchar | 20 | - | Yes |
| Email | Varchar | 50 | Unique Key | No |
| ContactNo | Varchar | 50 | - | Yes |
| Address | Varchar | 50 | - | Yes |
| Postal Code | Varchar | 50 | - | Yes |
| DesignationId | Integer | 10 | Foreign Key [FK]  Designation : DesignationId | No |
| RoleId | Integer | 10 | Foreign Key [FK]  Roles : RoleId | No |
| PaymentId | Integer | 10 | Foreign Key [FK]  Payment : PaymentId | No |
| PayRollSlipID | Integer | 10 | Foreign Key [FK]  PayRollSlip : PayrollID | No |

**Table 2: Salary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type | Length | Constraints | Nullable |
| SalaryId | Integer | 10 | Primary Key [PK] | No |
| SalaryType | Varchar | 20 | - | Yes |
| Salary | Float | 10 | - | Yes |

**Table 3: Designation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type | Length | Constraints | Nullable |
| DesignationId | Integer | 10 | Primary Key[PK] | No |
| Designation | Varchar | 50 | - | Yes |
| Department | Varchar | 50 | - | Yes |
| JobTitle | Varchar | 50 | - | Yes |
| DateOfHire | Date | - | - | Yes |
| SalaryId | Integer | 10 | Foreign Key [FK]  Salary : SalaryId | No |

**Table 4: Roles**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type | Length | Constraints | Nullable |
| RoleId | Integer | 10 | Primary Key [PK] | No |
| Role | Varchar | 50 | - | Yes |

**Table 5: PayRollSlip**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type | Length | Constraints | Nullable |
| PayrollId | Integer | 10 | Primary Key [PK] | No |
| PayrollType | Varchar | 50 | - | Yes |

**Table 6: Leave**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type | Length | Constraints | Nullable |
| LeaveId | Integer | 10 | Primary Key[PK] | No |
| StartDate | Date | - | - | Yes |
| EndDate | Date | = | - | Yes |
| Days | Integer | 10 | - | Yes |
| Reason | Varchar | 80 | - | Yes |
| EmployeeID | Integer | 10 | Foreign Key [FK]  Employee : EmployeeID | No |

**Table 7: Payment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type | Length | Constraints | Nullable |
| PaymentId | Integer | 10 | Primary Key[PK] | No |
| PaymentDate | Date | - | - | Yes |
| PaymentAmount | Float | 10 | - | Yes |
| Allowance | Float | 10 | - | Yes |
| SalarySalaryId | Integer | 10 | Foreign Key [FK]  Salary : SalaryId | No |
| LeaveId | Integer | 10 | Foreign Key [FK]  Leave : LeaveId | No |
| PayRollSlipPayrollID | Integer | 10 | Foreign Key [FK]  PayRollSlip : PayrollID | No |

The above listed tables are the data-dictionary of Employee Payroll System which represents all entities with their column names, data types, lengths, constraints and nullable.

**3.4.2 ER Diagram [Entity Relationship Diagram]**

ER diagram, which is also known as entity relationship diagram or ERD is a structural diagram that are used for designing database of the system. ER diagram consists of various symbols and connectors within system scope and inter-relationship among entities.

**Reason behind choosing Entity Relationship Diagram as Database modeling**

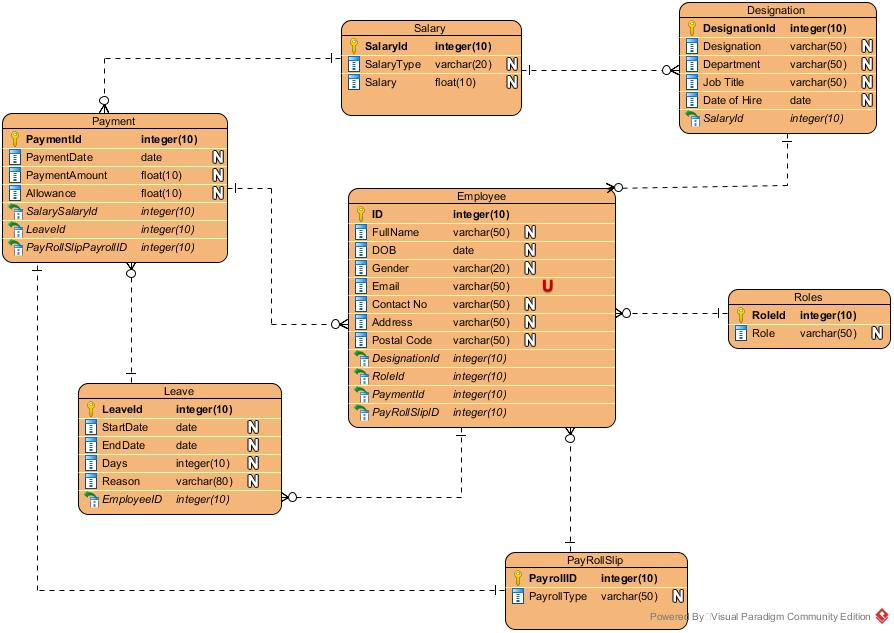
There are many benefits of ER diagram which supports for implementation phase. It offers visual presentation of the system and results in effective flow of communication and information. It is very easy to understand by individuals as the diagram is made in a simple manner. Hence, ER diagram makes the flow of data very efficient and increase the flexibility of the system.

**Notations in Entity Relationship Diagram**

An ER diagram consists of attributes, entities and relationships which are as listed in a tabular format.

|  |  |  |
| --- | --- | --- |
| Name | Notation | Description |
| Entity |  | ERD entity is a definable concept within a system. |
| Entity Attributes |  | It is also known as column which is a property or characteristic of entity that holds it. |
| Primary Key |  | It is also known as PK which is an entity attribute that uniquely defines a record in a database table. |
| Foreign Key |  | It is also known as FK which is a reference to a primary key in table. |
| One-to-One Cardinality |  | It is used to split an entity in two to provide information concisely and make it more understandable. |
| One-to-Many Cardinality |  | It refers to the relationship between two entities X and Y in which an instance of X may be linked to many instances of Y, but an instance of Y is linked to only one instance of X. |

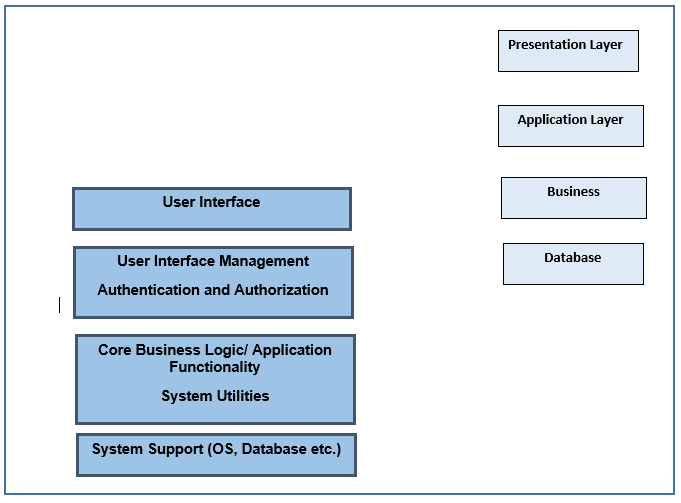
**Entity Relationship Diagram and Description**

****

The above ER diagram depicts the complete system in an easy way which can be understand by each and every individual. The diagram shows its seven entities in employee payroll system with respective attributes with primary key and foreign key constraints. They are linked by relationship as one-to-many cardinality relationship. The diagram is drawn in **Visual Paradigm** which is itself an easy UML tool.

## **3.5 Working Architecture of Project**

It is also known as architecture modeling where I have decided to go through layered architecture that supports incremental development of system. The layer gets developed continuously and the services are made available to the users timely. As the layer gets added or changed, the additional features are added step wise where localize machine depends upon inner layer. This tends to provide multi-platform for users of an application system. The machine-dependent which is inner layer are tend to be re-implemented to take account of facilities of the database or an operating system.

****

The above figure shows a layered architecture with four different layers where lowest supports database and operating system. Another layer represents application layer having components with functionalities. The next layer represents user interface that helps in providing user authentication and authorization and rest one providing user interface facilities.

The employee payroll system is concerned with following architecture as:

* The user interface represents the windows form.
* The user interface management authentication and authorization layer represents the login form that authenticates and authorizes potential user.
* The core business logic layer provides functionalities such as data retrieval. Generating reports, etc.
* The database layer represents the database software in which we create database for our system.

## **3.6 User Interface Design**

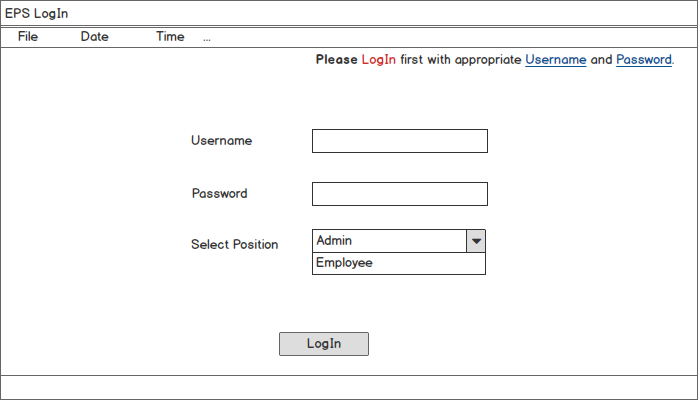
User Interface Design is also known as UI Design which is the process of creating **interfaces** in software that focuses mainly upon the styles and positions of images or formats. It is basically a visual elements of a particular product or a system that shows the actual presentation and interactivity of the system.

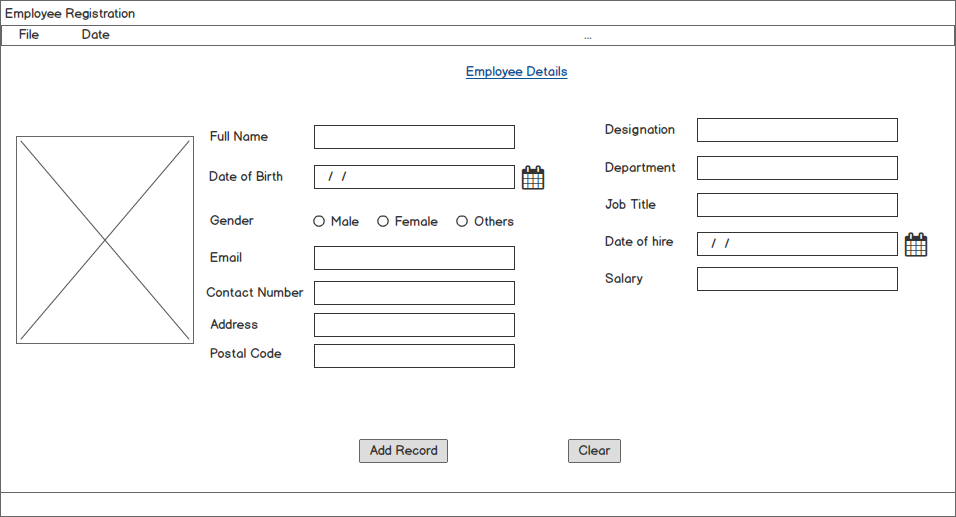
Similarly, I have chosen **Balsamiq** for prototyping purpose or for creating user-interface design.

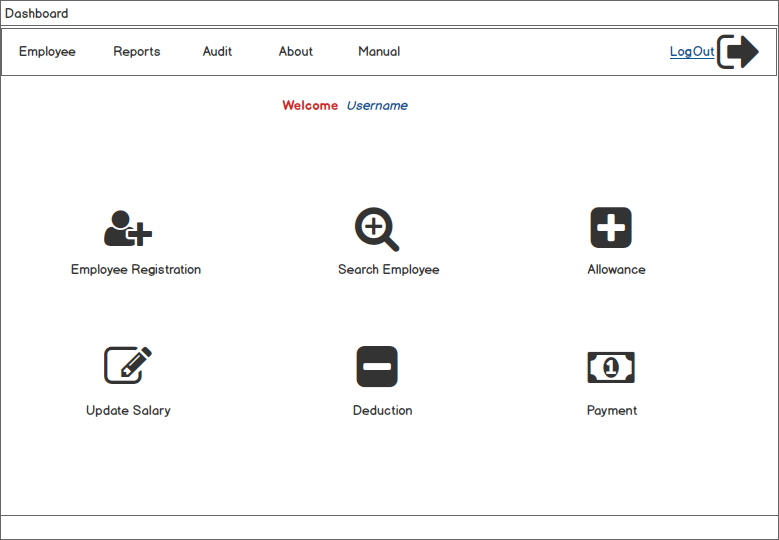
**Reason behind choosing User Interface Design during Design Phase**

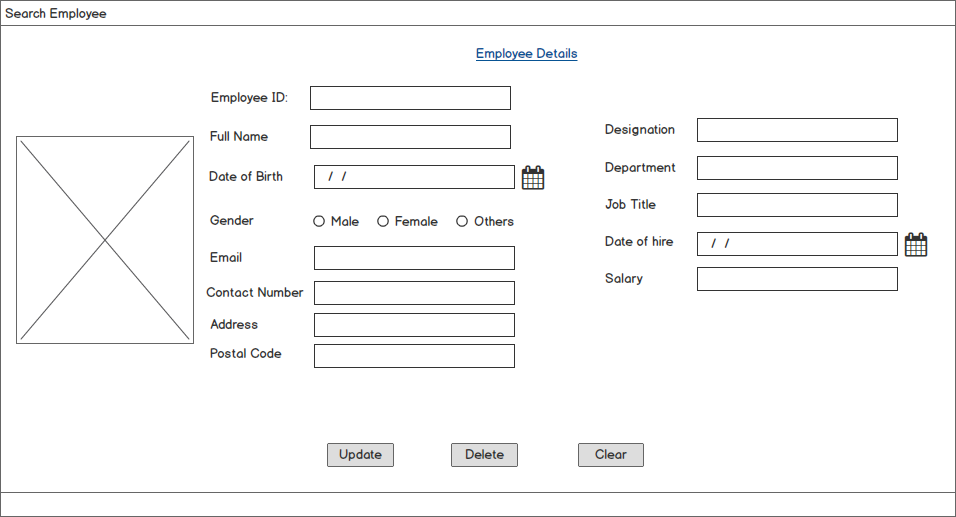
Basically, UI design is overall about experience that combines technical and creative aspects of system. It helps in enhancing sales and optimizing the user experience that helps in reducing initial development costs and further iterations. During User Interface design phase, there will be more interaction with customers or users that builds good communication and better understanding about the system.

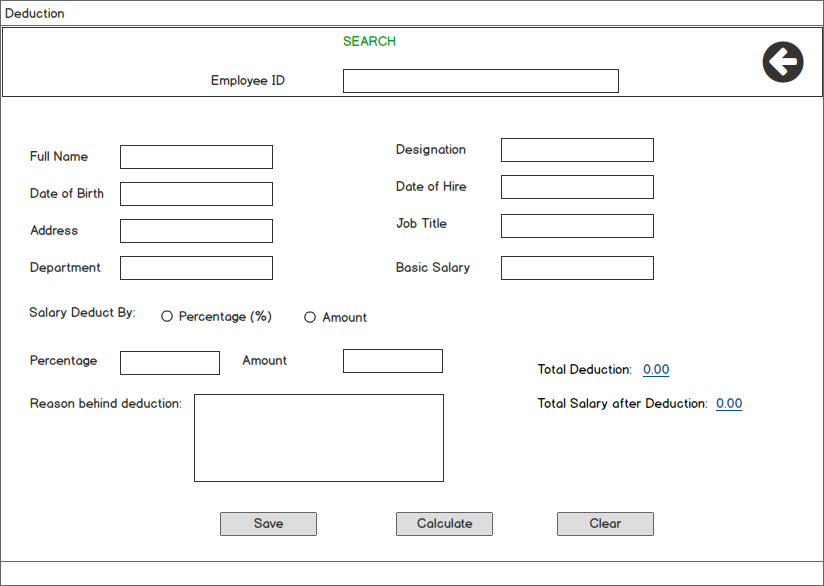
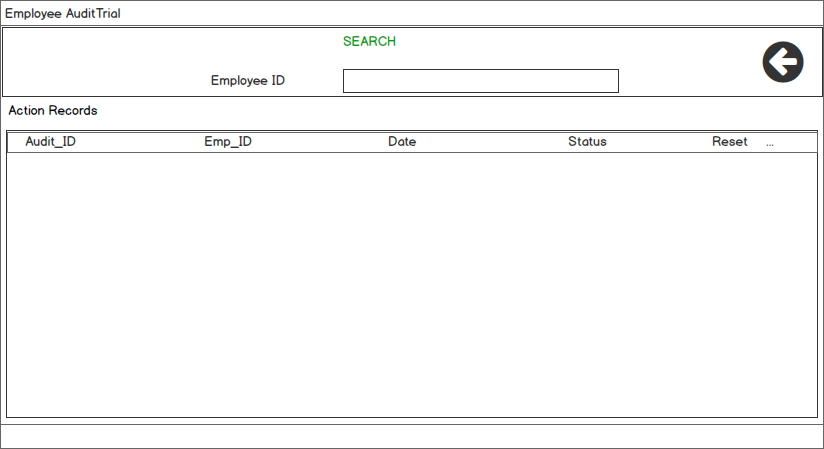
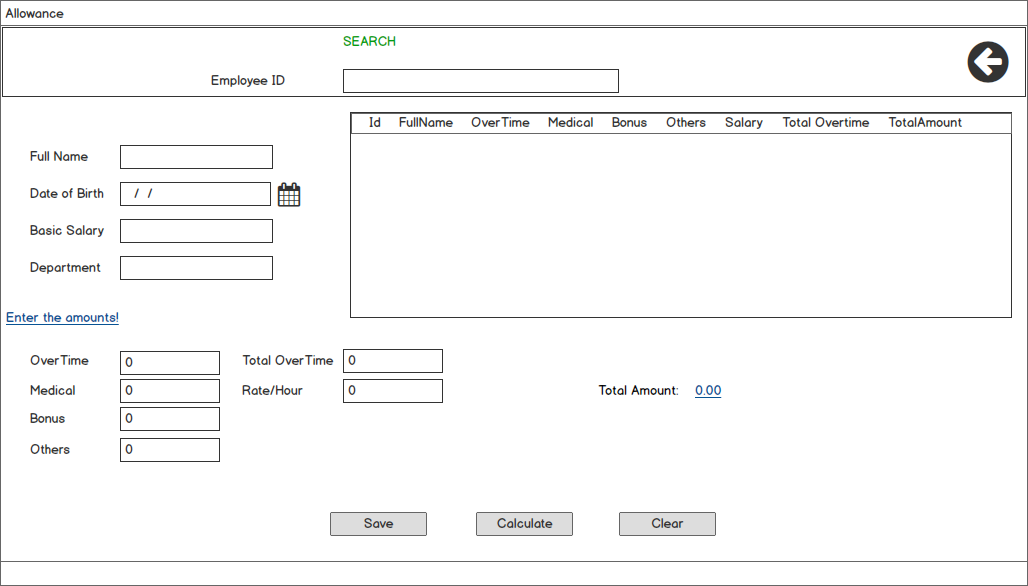
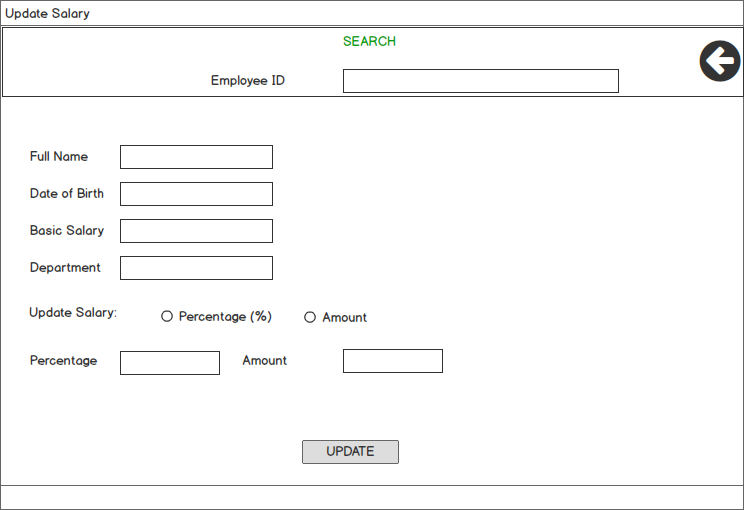
**Prototypes done on Balsamiq**

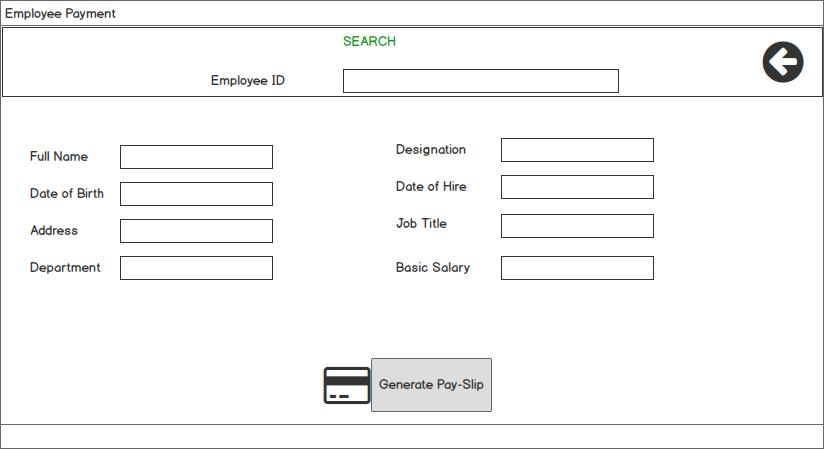
****

****

****

****

****

****