CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Payroll Management System is designed to make the existing manual system automatic with the help of computerised equipment and full-edged computer software, fulfilling their requirements, so that their valuable data and information can be stored for a longer period with easy access and manipulation of the same. Payroll Management system aids in the streamlining and centralization of your company's wage payments. The software calculates salaries, tax deductions, incentives, and bonuses on its own, sorting out payment and deduction issues and digitally recording them in its database.

1.2 PROJECT SPECIFICATION

The proposed system is a website in which user can see their monthly salary. Also that the employee can view all the deductions and allowances incured on their salary.

The system includes following modules.

1. Allowances & Deduction Management:

The allowance and deduction management will help the admin in managing all the allowances and deductions from the salary of their Employees. These deductions were done payment for the employees' benefits and taxes. He has the overall control of the system. Admin can add or update allowances/deductions details, manage employee data etc.

2. **Employee Information Management**

Admin can register and manage employee datas.Admin can add Employees salaries. Employee management will help to list the employee data,update the employee data and add new employee.

3. .Department & Positions Management (Admin side)

Admin can add, edit, remove departments of the organization. Admin can also add positions to each of the registered departments.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minute's detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies, a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

2.2 EXISTING SYSTEM

The pre-existing system for this purpose is offline so doesn't require any device or internet connection. Previously all the information was written on the papers and the employees are hired to manage the paperwork of the company. All this work will be depended on the individual's working speed. All the employees have to synchronize the paperwork manually by communicating with each other.

This paperwork takes a lot of time and requires more space to store safely and all the additional resources required to ensure safety. In this system, there is no surety or reliability as humans are likely to make an error.

2.3 DRAWBACKS OF EXISTING SYSTEM

- Required a lot of paperwork.
- Everything is done on paper and these are highly prone to damages and require a good amount of security and space to store. Likely to have an error.
- Lack of storage space for the documents.
- If these files are less in number then it is not that big problem but if the number increases then searching for a particular document can give a personal nightmare as these are more prone to damages and even it gets the searching job done the time required to do the job is not ideal at all.
- Require more physical work and manpower.
- Editing is another level of problem in those paper documents the only option is to make new documents if any correction needs to be made that is not good at all for the environmental credentials.
- No backup of the information.
- Where all the drawbacks of the pre-existing system have been overcome in the proposed system. It definitely improves the speed by automating most of the tasks required to be done manually previously.

Automated systems are more reliable and efficient now a day for these types of work.

2.4 PROPOSED SYSTEM

- Does not require paperwork.
- Not required much space.
- Require less physical work and manpower.
- Editing is a lot easier.

.

2.5 ADVANTAGES OF PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources, and the system will work in almost all configurations. It has got following features:

- **Better security: -**

For data to remain secure measures must be taken to prevent unauthorized access. Security means that data are protected from various forms of destruction. The system security problem can be divided into four related issues: security, integrity, privacy and confidentiality. Username and password requirement to sign in ensures security. It will also provide data security as we are using the secured databases for maintaining the documents.

- **Ensure data accuracy: -**

The proposed system eliminates the manual errors while entering the details of the users during the registration.

- **Better service: -**

The product will avoid the burden of hard copy storage. We can also conserve the time and human resources for doing the same task. The data can be maintained for longer period with no loss of data.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features: -

3.1.1 Economical Feasibility

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

The proposed system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

The cost of project, DREAMS was divided according to the system used, its development cost and cost for hosting the project. According to all the calculations the project was developed in a low cost. As it is completely developed using open source software.

3.1.2 Technical Feasibility

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation are:

- Does the existing technology sufficient for the suggested one?
- Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project requires High Resolution Scanning device and utilizes Cryptographic techniques. Through the technology may become obsolete after some period of time, due to the fact that newer version of same software supports older versions, the system may still be used. So there are minimal constraints involved with this project. The system has been developed using PHP in front end and MySQL in server in back end, the project is technically feasible for development. The system has been developed using PHP in front end and MySQL in server in back end, the project is technically feasible for development. The System used was also of good performance of Processor Intel i3 core; RAM 4GB and, Hard disk 1TB

3.1.3 Behavioral Feasibility

The proposed system includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - Intel core i3

RAM - 4 GB

Hard disk - 1 TB

3.2.2 Software Specification

Front End - HTML, CSS

Backend - MYSQL

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, AJAX, PHP, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 PHP

PHP is a server side scripting language designed for web development but also used as a general purpose programming language. PHP is now installed on more than 244 million websites and 2.1 million web servers. Originally created by Rasmus Ledorf in 1995, the reference implementation of PHP is now produced by the PHP group. While PHP originally stood for personal Home page ,it now stands for PHP:HypertextPreprocessor, a recursive acronym.PHP code is interpreted by a web server with a PHP processor module which generates the resulting web page.PHP commands can be embedded directly into a HTML source document rather than calling an external file to process data. It has also evolved to include a command-line interface capability and can be used in standalone incompatible with the GNU General Public License (GPL) due to restrictions on the usage of the term PHP.PHP can be deployed on most web servers and also as a standalone shell on almost every operating system and platform, free of charge.

3.3.2 MySQL

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. The MySQL Web site provides the latest information about MySQL software.

- **MySQL is a database management system.**

  A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

- **MySQL databases are relational.**

  A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and "pointers" between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data. The SQL part of "MySQL" stands for "Structured Query Language". SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax. SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, "SQL92" refers to the standard released in 1992, "SQL: 1999" refers to the standard released in 1999, and "SQL: 2003" refers to the current version of the standard. We use the phrase "the SQL standard" to mean the current version of the SQL Standard at any time.

- **MySQL software is Open Source.**
  Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL or need to embed MySQL code into a commercial application, you can buy a commercially licensed version from us. See the MySQL Licensing Overview for more information.

- **The MySQL Database Server is very fast, reliable, scalable, and easy to use.**
  If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available.

- **MySQL Server works in client/server or embedded systems.**
  The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs). We also provide MySQL Server as an embedded multi-threaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

UML stands for **Unified Modeling Language**. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After

some standardization, UML has become an OMG standard. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete. UML includes the following nine diagrams.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- Statechart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation forthe modeling of real-world objects and systems.

System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service-oriented task. For example, use cases in a product sales environment would include item ordering, catalog updating, payment processing, and customer relations. A use case diagram contains four components.

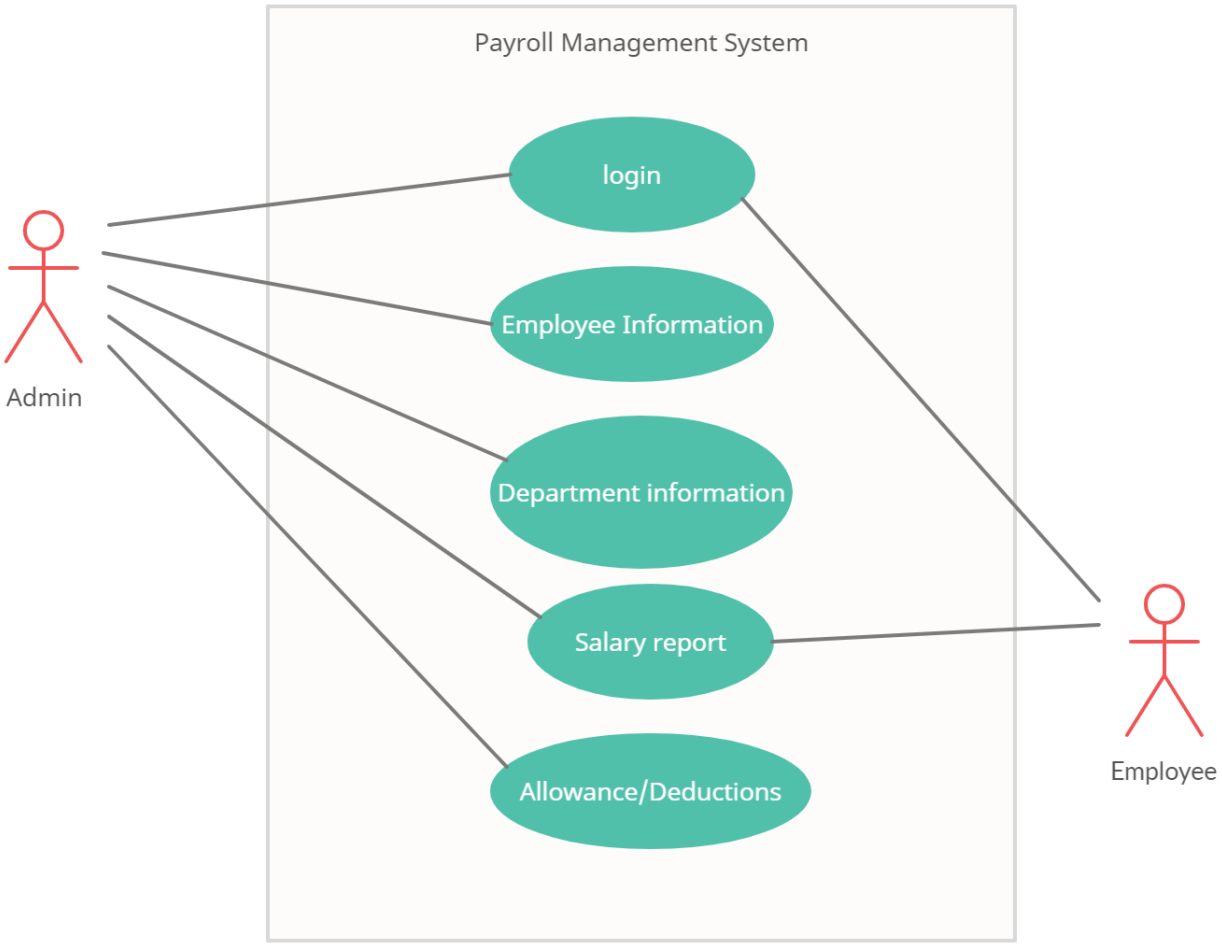
- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their

roles.

- The use cases, which are the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram

- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.


Payroll Management System
login
Employee Information
Department information
Salary report
Allowance/Deductions
Admin
Employee

4.2.2 SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

Sequence Diagram Notations –

i. **Actors –** An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.

ii. **Lifelines –** A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram

iii. **Messages –** Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

Messages can be broadly classified into the following categories:

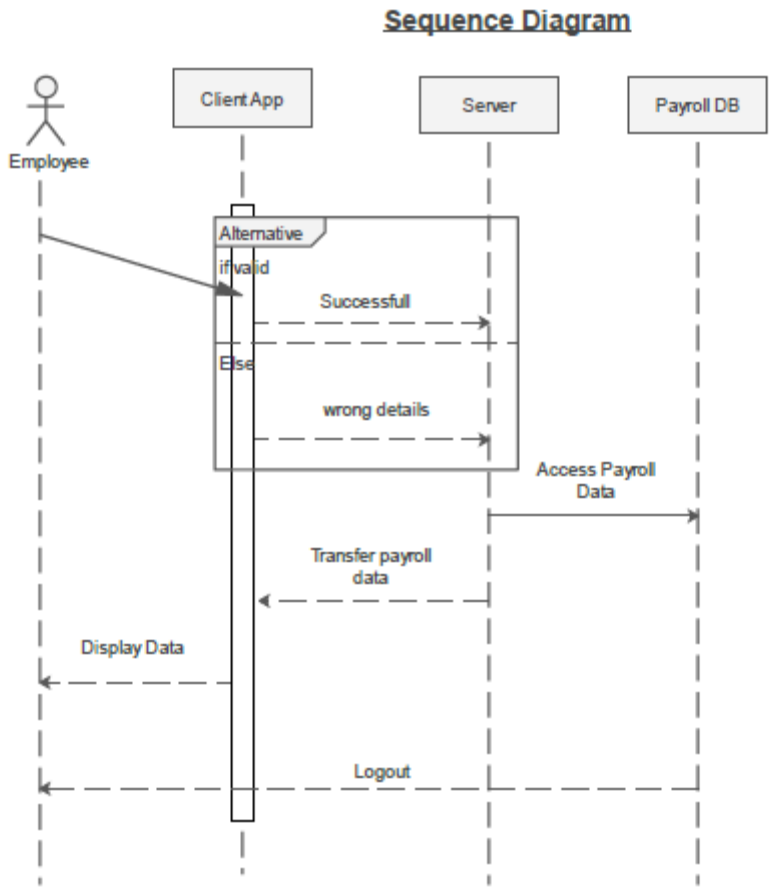
- Synchronous messages
- Asynchronous Messages
- Create message
- Delete Message
- Self-Message

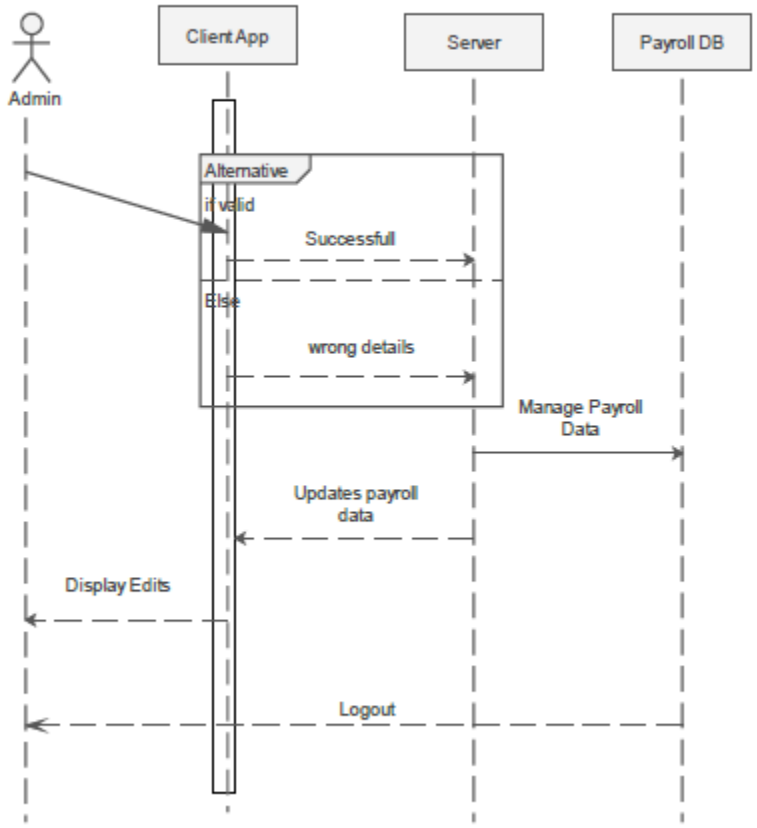
- Reply Message
- Found Message
- Lost Message

iv. **Guards –** To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

Uses of sequence diagrams –

- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualize how messages and tasks move between objects or components in a system.


Sequence Diagram
Employee
Client App
Server
Payroll DB
Alternative
if valid
Successfull
Else
wrong details
Access Payroll Data
Transfer payroll data
Display Data
Logout


Admin
Client App
Server
Payroll DB
Alternative
if valid
Successfull
Else
wrong details
Manage Payroll Data
Updates payroll data
Display Edits
Logout

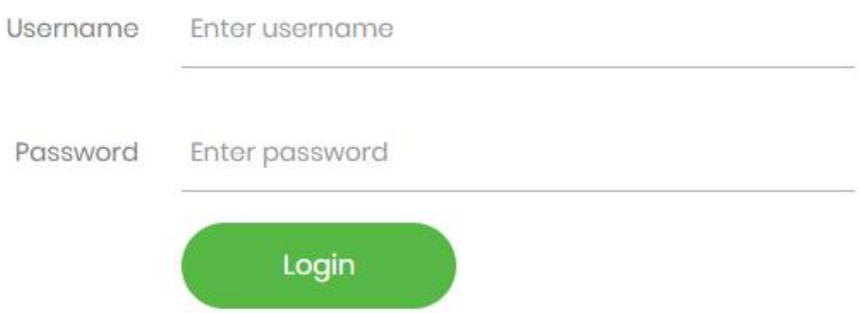
4.5 USER INTERFACE DESIGN

4.5.1-INPUT DESIGN

Form Name : User Login

4.5.2 OUTPUT DESIGN

User / Admin Login



DATABASE DESIGN

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

The database design is a two level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual DBMS.

In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design. The organization of the data in the database is aimed to achieve the following two major objectives.

- Data Integrity
- Data independence

4.6.1 Relational Database Management System (RDBMS)

A relational model represents the database as a collection of relations. Each relation resembles a table of values or file of records. In formal relational model terminology, a row is called a tuple, a column header is called an attribute and the table is called a relation. A relational database consists of a collection of tables, each of which is assigned a unique name. A row in a tale represents a set of related values.

Relations, Domains & Attributes

A table is a relation. The rows in a table are called tuples. A tuple is an ordered set of n elements. Columns are referred to as attributes. Relationships have been set between every table in the database. This ensures both Referential and Entity Relationship Integrity. A domain D is a set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in interpreting its values.

Every value in a relation is atomic, that is not decomposable.

Relationships

- Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
- Entity Integrity enforces that no Primary Key can have null values.
- Referential Integrity enforces that no Primary Key can have null values.
- Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other key are Super Key and Candidate Keys.

4.6.2 Normalization

Data are grouped together in the simplest way so that later changes can be made with minimum impact on data structures. Normalization is formal process of data structures in manners that eliminates redundancy and promotes integrity. Normalization is a technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion, and updating anomalies. Normal form in data modelling use two concepts, keys and relationships. A key uniquely identifies a row in a table. There are two types of keys, primary key and foreign key. A primary key is an element or a combination of elements in a table whose purpose is to identify records from the same table. A foreign key is a column in a table that uniquely identifies record from a different table. All the tables have been normalized up to the third normal form.

As the name implies, it denotes putting things in the normal form. The application developer via normalization tries to achieve a sensible organization of data into proper tables and columns and where names can be easily correlated to the data by the user. Normalization eliminates repeating groups at data and thereby avoids data redundancy which proves to be a great burden on the computer resources. These include:

- ✓ Normalize the data.
- ✓ Choose proper names for the tables and columns.
- ✓ Choose the proper name for the data.

First Normal Form

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form. This can be donor by moving data into separate tables where the data is of similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each non-atomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

Second Normal Form

According to Second Normal Form, for relations where primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and setup a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data that is only dependent on a part of the key. A relation is said to be in second normal form if and only if it satisfies all the first normal form conditions for the primary key and every non-primary key attributes of the relation is fully dependent on its primary key alone.

Third Normal Form

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In this we decompose and set up relation that includes the non-key attributes that functionally determines other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and more over the non key attributes of the relation should not be depend on other non-key attribute.

TABLE DESIGN

1.tbl_users

Primary_key: u_id

SI.No	Feild	Data_type	Description
1.	u_id	varchar(10)	Primary key of the user
2.	username	Varchar(50)	UserName of the user
3.	fullname	Varchar(50)	Name of the user
4.	email	Varchar(25)	Email of the user
5.	phone	Int(10)	Phone of the user
6.	password	Varchar(50)	Password of the user
7.	type	Int(1)	Determine type of user

2.tbl_departments

Primary_key :dept_id

SI.No	Feild	Data_type	Description
1.	dept_id	Varchar(10)	Primary key
2.	department	Varchar(20)	Name of the department

3.tbl_employee

Primary_key : emp_id
Foreign key : dept_id , pos_id

SI.No	Feild	Data_type	Description
1.	employee_id	Varchar(10)	Primary key
2.	ename	Varchar(20)	Name of the employee
3.	dept_id	Varchar(10)	Foreign key
4.	pos_id	Varchar(10)	Foreign key

4.tbl_position

Primary key : pos_id

SI.No	Feild	Data_type	Description
1.	pos_id	Varchar(10)	Primary key
2.	position	Varchar(25)	Position designation

5.tbl_allowances

Primary key : allo_id

SI.No	Feild	Data_type	Description
1.	allo_id	Varchar(10)	Primary key
2.	name	Varchar(10)	Allowance name
3.	amount	Int(10)	Allowance amount

6. tbl_deductions

Primary key : ded_id

SI.No	Feild	Data_type	Description
1.	ded_id	Varchar(10)	Primary key
2.	name	Varchar(10)	Deduction name
3.	amount	Int(10)	Deductionamount

7.tbl_salary

Primary key : allo_id
Foreign key : e_id, pos_id, dept_id, allo_id, ded_id

SI.No	Feild	Data_type	Description
1.	sal_id	Varchar(10)	Primary key
2.	e_id	Varchar(10)	Foreign key
3.	allo_id	Varchar(10)	Foreign key
4.	ded_id	Varchar(10)	Foreign key
5.	pos_id	Varchar(10)	Foreign key
6.	basicpay	Int(10)	Employee's basic pay
7.	salary	Int(10)	Employee's net salary

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified? Software testing is often used in association with the terms verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are:

Testing is a process of executing a program with the intent of finding an error.

- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If a testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrate that the software function appear to be working according to the specification, that performance requirement appear to have been met.

There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

Test for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

5.2 TEST PLAN

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

- Unit testing
- Integration Testing
- Data validation Testing
- Output Testing

5.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white-box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified. After coding each module is tested and run individually. All unnecessary code where removed and ensured that all modules are working, and gives the expected result.

5.2.2 Integration Testing

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop. After performing unit testing in the System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover differences in program structures were removed and a unique program structure was evolved.

5.2.3 Validation Testing or System Testing

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

5.2.4 Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points:

- Input Screen Designs,
- Output Screen Designs,

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion.

Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover.

The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.

6.2 IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we have to

ensure that the resistance does not build up, as one has to make sure that:

- The active user must be aware of the benefits of using the new system.
- Their confidence in the software is built up.
- Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won't take place.

6.2.1 User Training

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer based system, it is essential for the people who will be involved to be confident of theirrole in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

6.2.2 Training on the Application Software

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy

6.2.3 System Maintenance

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes".

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

Payroll Management System" software developed for a company has been designed toachieve maximum efficiency and reduce the time taken to handle the Payroll activity. It isdesigned to replace an existing manual record system thereby reducing time taken forcalculations and for storing data. The system uses HTML, css, javascript as front end and PHP as a backend for the database.The system is strong enough to withstand regressive daily operations underconditions where the database is maintained and cleared over a certain time of span. Theimplementation of the system in the organization will considerably reduce data entry, time and also provide readily calculated reports.

7.2 SCOPE

This Application works in Multiple PC's installed on multiple Computers but sharing same database by which users of different department can use it sitting at different locations simultaneouslyBut in future we can make the Application where the database will be hosted in order to manage the all departments which will be located in different places and by keeping domain of Application as Online.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, "*System Analysis and Design*", 2009.
- Roger S Pressman, "*Software Engineering*", 1994.
- PankajJalote, "So*ftware engineering*: a precise approach", 2006.
- James lee and Brent ware Addison, "Open source web development with LAMP", 2003
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

WEBSITES:

- www.w3schools.com
- www.jquery.com
- http://homepages.dcc.ufmg.br/~rodolfo/es-1-03/IEEE-Std-830-1998.pdf
- www.agilemodeling.com/artifacts/useCaseDiagram.html

CHAPTER 9

APPENDIX

9.1 Sample Code

Login.html

```
<!DOCTYPE html>
<html lang="en">

<head>
	<title>Login V15</title>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<!--
===============================================================================
=======================-->
	<link rel="icon" type="image/png" href="images/icons/favicon.ico" />
	<!--
===============================================================================
=======================-->
	<link rel="stylesheet" type="text/css" href="vendor/bootstrap/css/bootstrap.min.css">
	<!--
===============================================================================
=======================-->
	<link rel="stylesheet" type="text/css" href="fonts/font-awesome-4.7.0/css/font-awesome.min.css">
	<!--
===============================================================================
=======================-->
	<link rel="stylesheet" type="text/css" href="fonts/Linearicons-Free-v1.0.0/icon-font.min.css">
	<!--
===============================================================================
=======================-->
	<link rel="stylesheet" type="text/css" href="vendor/animate/animate.css">
	<!--
===============================================================================
=======================-->
	<link rel="stylesheet" type="text/css" href="vendor/css-hamburgers/hamburgers.min.css">
	<!--
===============================================================================
=======================-->
	<link rel="stylesheet" type="text/css" href="vendor/animsition/css/animsition.min.css">
	<!--
===============================================================================
=======================-->
	<link rel="stylesheet" type="text/css" href="vendor/select2/select2.min.css">
	<!--
===============================================================================
=======================-->
	<link rel="stylesheet" type="text/css" href="vendor/daterangepicker/daterangepicker.css">
	<!--
===============================================================================
=======================-->
	<link rel="stylesheet" type="text/css" href="css/util.css">
	<link rel="stylesheet" type="text/css" href="css/main.css">
```

```
    <!--
===============================================================================
======================-->
</head>

<body>

    <div class="limiter">
        <div class="container-login100">
            <div class="wrap-login100">
                <div class="login100-form-title" style="background-image: url(images/bg-01.jpg);">
                    <span class="login100-form-title-1">
                                                    Sign In
                                            </span>
                </div> <center style="color: red;
                margin-top:50px;"><?php
                        if(isset($_GET['show']))
                        {
                            $val=$_GET['show'];
                            echo $val;
                        }
                        ?></center>

                <form class="login100-form validate-form" action="index_action.php" method="post">
                    <div class="wrap-input100 validate-input m-b-26" data-validate="Username is required">
                        <span class="label-input100">Username</span>
                        <input class="input100" type="text" name="username" placeholder="Enter username">
                        <span class="focus-input100"></span>
                    </div>

                    <div class="wrap-input100 validate-input m-b-18" data-validate="Password is required">
                        <span class="label-input100">Password</span>
                        <input class="input100" type="password" name="password" placeholder="Enter
password">
                        <span class="focus-input100"></span>
                    </div>

                     <div class="flex-sb-m w-full p-b-30"  >
                         <style >

                                .flex-sb-m w-full p-b-30 {
                                    display: inline-block;
                                    color: aliceblue;
                                }

                             </style>
                        <div class="contact100-form-checkbox">

                                                                </label>
                        </div>



                    <div class="container-login100-form-btn">
                        <button class="login100-form-btn">
```

```
                                                    Login
                                                </button>
                    <div style="display: inline-block;">


                    </div>
                </form>
            </div>
        </div>
    </div>

    <!--
===============================================================================
=======================-->
    <script src="vendor/jquery/jquery-3.2.1.min.js"></script>
    <!--
===============================================================================
=======================-->
    <script src="vendor/animsition/js/animsition.min.js"></script>
    <!--
===============================================================================
=======================-->
    <script src="vendor/bootstrap/js/popper.js"></script>
    <script src="vendor/bootstrap/js/bootstrap.min.js"></script>
    <!--
===============================================================================
=======================-->
    <script src="vendor/select2/select2.min.js"></script>
    <!--
===============================================================================
=======================-->
    <script src="vendor/daterangepicker/moment.min.js"></script>
    <script src="vendor/daterangepicker/daterangepicker.js"></script>
    <!--
===============================================================================
=======================-->
    <script src="vendor/countdowntime/countdowntime.js"></script>
    <!--
===============================================================================
=======================-->
    <script src="js/main.js"></script>

</body>

</html>
```

Login.php

```
<?php
var_dump($_POST);
$conn= new mysqli('localhost','root','','payroll_db')or die("Could not connect to
mysql".mysqli_error($con));
$username=$_POST['username'];
$password=$_POST['password'];
$query="SELECT * from `users` where username='$username' and password='$password'";
$exe=mysqli_query($conn,$query);
if(mysqli_num_rows($exe))
{
   echo"Login SUCESSFULL!!!!";
   $row=mysqli_fetch_array($exe);
   $type=$row['type'];
   echo $type;
   if($type==0)
   {
      header('Location:employee/index.php');
   }
   else if($type==1)
   {
      header('Location:admin/index.php');
   }
}
else
{
   header('Location:index.php?show=invalid username/password');
}


?>
```

Addallowances.php

```
<?php include('db_connect.php');?>

<div class="container-fluid">

   <div class="col-lg-12">
      <div class="row">
         <!-- FORM Panel -->
         <div class="col-md-4">
         <form action="" id="manage-allowances" method="POST">
            <div class="card">
               <div class="card-header">
                    Allowances Form
               </div>
               <div class="card-body">
                     <input type="hidden" name="id">
                     <div class="form-group">
                        <label class="control-label">Allowance</label>
                        <textarea name="allowance" id="" cols="30" rows="2" class="form-control"
```

```
required></textarea>
                    </div>
                    <div class="form-group">
                       <label class="control-label">Description</label>
                       <textarea name="description" id="" cols="30" rows="2" class="form-control"
required></textarea>
                    </div>



                </div>

                <div class="card-footer">
                   <div class="row">
                      <div class="col-md-12">
                         <button class="btn btn-sm btn-primary col-sm-3 offset-md-3"name="submit">
Save</button>
                         <button class="btn btn-sm btn-default col-sm-3" type="button"
onclick="_reset()"> Cancel</button>
                      </div>
                   </div>
                </div>
             </div>
          </form>
          </div>
          <?php
          include 'db_connect.php';
          if(isset($_POST['submit']))
             {
             $id=$_POST['id'];
             $allowance=$_POST['allowance'];
             $description=$_POST['description'];
             $query="INSERT INTO `allowances`(`id`, `allowance`, `description`) VALUES
('$id','$allowance','$description')";
             $exe=mysqli_query($conn,$query);
             }
       ?>
          <!-- FORM Panel -->

          <!-- Table Panel -->
          <div class="col-md-8">
             <div class="card">
                <div class="card-body">
                   <table class="table table-bordered table-hover">
                      <thead>
                         <tr>
                            <th class="text-center">#</th>
                            <th class="text-center">Allowance Information</th>
                            <th class="text-center">Action</th>
                         </tr>
                      </thead>
                      <tbody>
                         <?php
                         $i = 1;
                         $allowances = $conn->query("SELECT * FROM allowances order by id asc");
```

```
                        while($row=$allowances->fetch_assoc()):
                        ?>
                        <tr>
                            <td class="text-center"><?php echo $i++ ?></td>

                            <td class="">
                                <p>Name: <b><?php echo $row['allowance'] ?></b></p>
                                <p class="truncate"><small>Description: <b><?php echo $row['description']
?></b></small></p>
                            </td>
                            <td class="text-center">
                                <a href="index.php?page=edit_allowance&&sid=<?php echo $row['id']
?>"<button class="btn btn-sm btn-primary edit_allowances" type="button" data-id="<?php echo
$row['id'] ?>" data-allowance="<?php echo $row['allowance'] ?>" data-description="<?php echo
$row['description'] ?>" >Edit</button></a>
                                <a href="delete_allowance.php?sid=<?php echo $row['id']?>"<button
class="btn btn-sm btn-danger delete_allowances" type="button" data-id="<?php echo $row['id']
?>">Delete</button></a>
                            </td>
                        </tr>
                        <?php endwhile; ?>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    <!-- Table Panel -->
    </div>
  </div>

</div>
<style>

  td{
    vertical-align: middle !important;
  }
  td p{
    margin: unset
  }
  img{
    max-width:100px;
    max-height:150px;
  }
</style>
```

Adddeductions.php

```
<?php include('db_connect.php');?>

<div class="container-fluid">

  <div class="col-lg-12">
    <div class="row">
      <!-- FORM Panel -->
```

```
        <div class="col-md-4">
        <form action="" id="manage-deductions" method="POST">
          <div class="card">
            <div class="card-header">
                Deductions Form
            </div>
            <div class="card-body">
                <input type="hidden" name="id">
                <div class="form-group">
                  <label class="control-label">Deduction Name</label>
                  <textarea name="deduction" id="" cols="30" rows="2" class="form-control"
required></textarea>
                </div>
                <div class="form-group">
                  <label class="control-label">Description</label>
                  <textarea name="description" id="" cols="30" rows="2" class="form-control"
required></textarea>
                </div>



            </div>

            <div class="card-footer">
              <div class="row">

                <div class="col-md-12">
                  <button class="btn btn-sm btn-primary col-sm-3 offset-md-3"name="submit">
Save</button>
                  <button class="btn btn-sm btn-default col-sm-3" type="button"
onclick="_reset()"> Cancel</button>
                </div>
              </div>
            </div>
          </div>
        </form>
      </div>
      <?php
        include 'db_connect.php';
        if(isset($_POST['submit']))
          {
          $id=$_POST['id'];
          $deduction=$_POST['deduction'];
          $description=$_POST['description'];
          $query="INSERT INTO `deductions`(`id`, `deduction`, `description`) VALUES
('$id','$deduction','$description')";
          $exe=mysqli_query($conn,$query);
          }
      ?>
        <!-- FORM Panel -->

        <!-- Table Panel -->
        <div class="col-md-8">
          <div class="card">
            <div class="card-body">
```

```
                    <table class="table table-bordered table-hover">
                        <thead>
                            <tr>
                                <th class="text-center">#</th>
                                <th class="text-center">Deduction Information</th>
                                <th class="text-center">Action</th>
                            </tr>
                        </thead>
                        <tbody>
                            <?php
                            $i = 1;
                            $deductions = $conn->query("SELECT * FROM deductions order by id asc");
                            while($row=$deductions->fetch_assoc()):
                            ?>
                            <tr>
                                <td class="text-center"><?php echo $i++ ?></td>

                                <td class="">
                                    <p>Name: <b><?php echo $row['deduction'] ?></b></p>
                                    <p class="truncate"><small>Description: <b><?php echo $row['description']
?></b></small></p>
                                </td>
                                <td class="text-center">
                                    <a href="index.php?page=edit_deduction&&sid=<?php echo $row['id'];
?>"><button class="btn btn-sm btn-primary edit_deductions" type="button" data-id="<?php echo
$row['id'] ?>" data-deduction="<?php echo $row['deduction'] ?>" data-description="<?php echo
$row['description'] ?>" >Edit</button></a>
                                    <a href="delete_deduction.php?sid=<?php echo $row['id']; ?>"><button
class="btn btn-sm btn-danger delete_deductions" type="button" data-id="<?php echo $row['id']
?>">Delete</button></a>
                                </td>
                            </tr>
                            <?php endwhile; ?>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
        <!-- Table Panel -->
    </div>
</div>
<style>

    td{
        vertical-align: middle !important;
    }
    td p{
        margin: unset
    }
    img{
        max-width:100px;
        max-height:150px;
    }
</style>
```

Adddepartments.php

```
<?php include('db_connect.php');?>

<div class="container-fluid">

   <div class="col-lg-12">
      <div class="row">
         <!-- FORM Panel -->
         <div class="col-md-4">
         <form action="" method="POST" id="manage-department">
            <div class="card">
               <div class="card-header">
                    Department Form
               </div>
               <div class="card-body">
                     <input type="hidden" name="id">
                     <div class="form-group">
                        <label class="control-label">Name</label>
                        <textarea name="name" id="" cols="30" rows="2" class="form-
control"></textarea>
                     </div>



               </div>

               <div class="card-footer">
                  <div class="row">
                     <div class="col-md-12">
                        <button name="submit" class="btn btn-sm btn-primary col-sm-3 offset-md-3">
Save</button>
                        <button class="btn btn-sm btn-default col-sm-3" type="button"
onclick="_reset()"> Cancel</button>
                     </div>
                  </div>
               </div>
            </div>
         </form>
         </div>
         <?php
         include 'db_connect.php';
         if(isset($_POST['submit']))
            {
            $id=$_POST['id'];
            $name=$_POST['name'];
            $query="INSERT INTO `department`(`id`, `name`) VALUES ('$id','$name')";
            $exe=mysqli_query($conn,$query);
            }
      ?>
         <!-- FORM Panel -->

         <!-- Table Panel -->
         <div class="col-md-8">
            <div class="card">
```

```
                <div class="card-body">
                  <table class="table table-bordered table-hover">
                    <thead>
                      <tr>
                        <th class="text-center">#</th>
                        <th class="text-center">Department</th>
                        <th class="text-center">Action</th>
                      </tr>
                    </thead>
                    <tbody>
                      <?php
                      $i = 1;
                      $department = $conn->query("SELECT * FROM department order by id asc");
                      while($row=$department->fetch_assoc()):
                      ?>
                      <tr>
                        <td class="text-center"><?php echo $i++ ?></td>

                        <td class="">
                            <p> <b><?php echo $row['name'] ?></b></p>
                        </td>
                        <td class="text-center">
                          <a href="index.php?page=edit_department&&dept=<?php echo
$row['id'];?>" <button class="btn btn-sm btn-primary edit_department" type="button" data-id="<?php
echo $row['id'] ?>" data-name="<?php echo $row['name'] ?>"  >Edit</button></a>
                          <a href="delete_department.php?dept=<?php echo $row['id'];?>"<button
class="btn btn-sm btn-danger delete_department" type="button" data-id="<?php echo $row['id']
?>">Delete</button></a>
                        </td>
                      </tr>
                      <?php endwhile; ?>
                    </tbody>
                  </table>
                </div>
              </div>
            </div>
            <!-- Table Panel -->
          </div>
        </div>

      </div>
      <style>

        td{
          vertical-align: middle !important;
        }
        td p{
          margin: unset
        }
        img{
          max-width:100px;
          max-height:150px;
        }
      </style>
      <!-- <script>
```

```
function _reset(){
    $('[name="id"]').val('');
    $('#manage-department').get(0).reset();
}

$('#manage-department').submit(function(e){
    e.preventDefault()
    start_load()
    $.ajax({
        url:'ajax.php?action=save_department',
        data: new FormData($(this)[0]),
        cache: false,
        contentType: false,
        processData: false,
        method: 'POST',
        type: 'POST',
        success:function(resp){
            if(resp==1){
                alert_toast("Data successfully added",'success')
                setTimeout(function(){
                    location.reload()
                },1500)

            }
            else if(resp==2){
                alert_toast("Data successfully updated",'success')
                setTimeout(function(){
                    location.reload()
                },1500)

            }
        }
    })
})
$('.edit_department').click(function(){
    start_load()
    var cat = $('#manage-department')
    cat.get(0).reset()
    cat.find("[name='id']").val($(this).attr('data-id'))
    cat.find("[name='name']").val($(this).attr('data-name'))
    end_load()
})
$('.delete_department').click(function(){
    _conf("Are you sure to delete this department?","delete_department",[$(this).attr('data-id')])
})
function displayImg(input,_this) {
if (input.files && input.files[0]) {
    var reader = new FileReader();
    reader.onload = function (e) {
        $('#cimg').attr('src', e.target.result);
    }

    reader.readAsDataURL(input.files[0]);
}
}
```

```
    function delete_department($id){
        start_load()
        $.ajax({
            url:'ajax.php?action=delete_department',
            method:'POST',
            data:{id:$id},
            success:function(resp){
                if(resp==1){
                    alert_toast("Data successfully deleted",'success')
                    setTimeout(function(){
                        location.reload()
                    },1500)

                }
            }
        })
    }
</script> -->
```

Addpositions.php

```
<?php include('db_connect.php');?>

<div class="container-fluid">

    <div class="col-lg-12">
        <div class="row">
            <!-- FORM Panel -->
            <div class="col-md-4">
            <form action="" id="manage-position">
                <div class="card">
                    <div class="card-header">
                        Position Form
                    </div>
                    <div class="card-body">
                            <input type="hidden" name="id">
                            <div class="form-group">
                                <label class="control-label">Department</label>
                                <select class="custom-select browser-default select2" name="department_id">
                                    <option value=""></option>
                                <?php
                                $dept = $conn->query("SELECT * from department order by name asc");
                                while($row=$dept->fetch_assoc()):
                                ?>
                                    <option value="<?php echo $row['id'] ?>"><?php echo $row['name']
?></option>
                                <?php endwhile; ?>
                                </select>
                            </div>
                            <div class="form-group">
                                <label class="control-label">Name</label>
                                <textarea name="name" id="" cols="30" rows="2" class="form-
control"></textarea>
                            </div>
```

```
            </div>

            <div class="card-footer">
              <div class="row">
                <div class="col-md-12">
                  <button class="btn btn-sm btn-primary col-sm-3 offset-md-3"> Save</button>
                  <button class="btn btn-sm btn-default col-sm-3" type="button"
onclick="_reset()"> Cancel</button>
                </div>
              </div>
            </div>
          </div>
        </form>
        </div>
        <!-- FORM Panel -->

        <!-- Table Panel -->
        <div class="col-md-8">
          <div class="card">
            <div class="card-body">
              <table class="table table-bordered table-hover">
                <thead>
                  <tr>
                    <th class="text-center">#</th>
                    <th class="text-center">Position</th>
                    <th class="text-center">Department</th>
                    <th class="text-center">Action</th>
                  </tr>
                </thead>
                <tbody>
                  <?php
                  $i = 1;
                  $position = mysqli_query($conn,"SELECT * FROM position join department on
department.id=position.department_id order by position.id asc");
                    while($row=mysqli_fetch_array($position)){
                    ?>
                    <tr>
                      <td class="text-center"><?php echo $i++ ?></td>

                      <td class="">
                          <p> <b><?php echo $row[2] ?></b></p>
                      </td>
                      <td class="text-center">
                          <p> <b><?php echo $row[4] ?></b></p>
                      </td>
                      <td class="text-center">
                        <button class="btn btn-sm btn-primary edit_position" type="button" data-
id="<?php echo $row['id'] ?>" data-name="<?php echo $row['name'] ?>" data-department_id="<?php
echo $row['department_id'] ?>"  >Edit</button>
                        <button class="btn btn-sm btn-danger delete_position" type="button" data-
id="<?php echo $row['id'] ?>">Delete</button>
                      </td>
                    </tr>
                    <?php } ?>
```

```
                    </tbody>
                  </table>
                </div>
              </div>
            </div>
            <!-- Table Panel -->
          </div>
        </div>

</div>
<style>

    td{
        vertical-align: middle !important;
    }
    td p{
        margin: unset
    }
    img{
        max-width:100px;
        max-height:150px;
    }
</style>
```

Addemployees.php

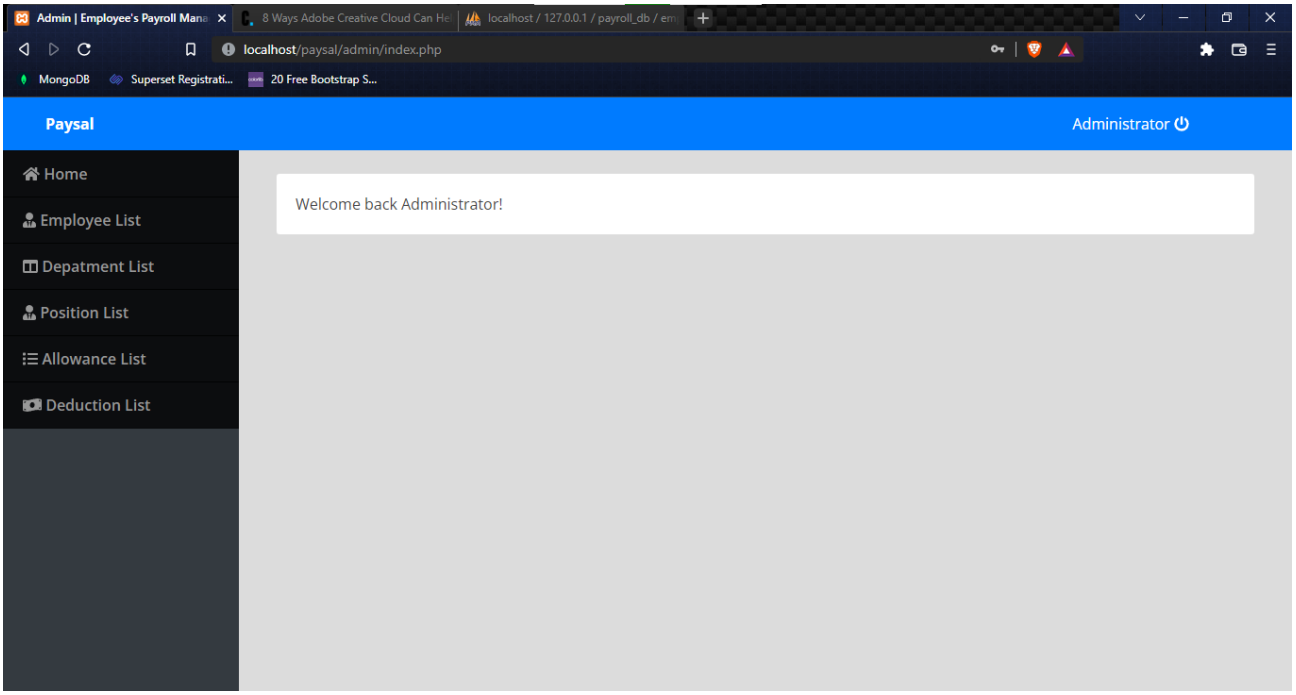
```
<?php include('db_connect.php') ?>
        <div class="container-fluid " >
            <div class="col-lg-12">

                <br />
                <br />
                <div class="card">
                    <div class="card-header">
                        <span><b>Employee List <a href="index.php?page=users">User List</a></b></span>
                        <button class="btn btn-primary btn-sm btn-block col-md-3 float-right" type="button"
id="new_emp_btn"><span class="fa fa-plus"></span> Add Employee</button>
                    </div>
                    <div class="card-body">
                        <table class="table" class="table table-bordered table-striped">
                            <thead>
                                <tr>
                                    <th>Employee No</th>
                                    <th>Full Name</th>
                                    <th>Username</th>
                                    <th>Department</th>
                                    <th>Position</th>
                                    <th>Action</th>
                                </tr>
                            </thead>
                            <tbody>
                                <?php
                                    $d_arr[0] = "Unset";
```

```
                        $p_arr[0] = "Unset";
                        $dept = $conn->query("SELECT * from department order by name asc");
                          while($row=$dept->fetch_assoc()):
                            $d_arr[$row['id']] = $row['name'];
                          endwhile;
                          $pos = $conn->query("SELECT * from position order by name asc");
                          while($row=$pos->fetch_assoc()):
                            $p_arr[$row['id']] = $row['name'];
                           endwhile;
                        $employee_qry=$conn->query("SELECT * FROM employee") or
die(mysqli_error());
                        while($row=$employee_qry->fetch_array()){
                      ?>
                      <tr>
                        <td><?php echo $row['employee_no']?></td>
                        <td><?php echo $row['name']?></td>
                        <td><?php echo $row['username']?></td>
                        <td><?php echo $d_arr[$row['department_id']]?></td>
                        <td><?php echo $p_arr[$row['position_id']]?></td>
                        <td>
                          <center>
                           <button class="btn btn-sm btn-outline-primary view_employee" data-
id="<?php echo $row['id']?>" type="button"><i class="fa fa-eye"></i></button>
                           <button class="btn btn-sm btn-outline-primary edit_employee" data-
id="<?php echo $row['id']?>" type="button"><i class="fa fa-edit"></i></button>
                          <button class="btn btn-sm btn-outline-danger remove_employee" data-
id="<?php echo $row['id']?>" type="button"><i class="fa fa-trash"></i></button>
                          </center>
                        </td>
                      </tr>
                      <?php
                        }
                      ?>
                    </tbody>
                  </table>
                </div>
              </div>
            </div>
          </div>


    <script type="text/javascript">
      $(document).ready(function(){
        $('#table').DataTable();
      });
    </script>
    <script type="text/javascript">
      $(document).ready(function(){



        $('.edit_employee').click(function(){
          var $id=$(this).attr('data-id');
```

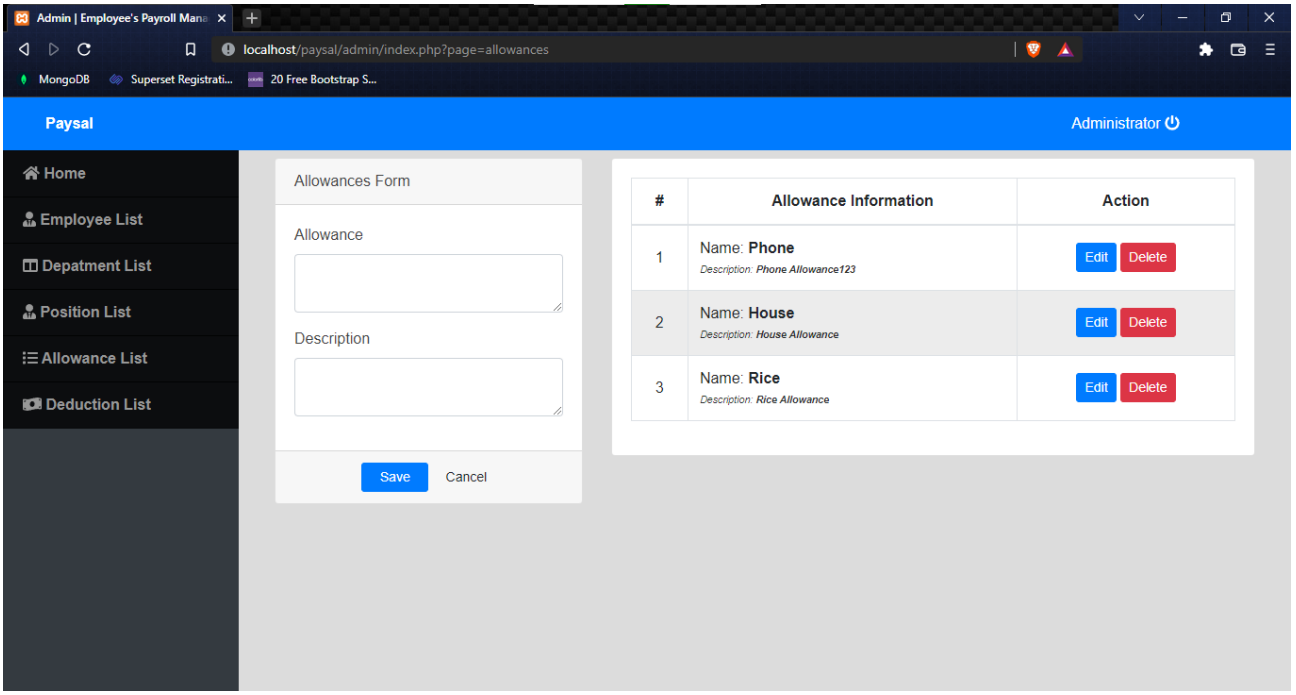
```
            uni_modal("Edit Employee","manage_employee.php?id="+$id)

        });
        $('.view_employee').click(function(){
            var $id=$(this).attr('data-id');
            uni_modal("Employee Details","view_employee.php?id="+$id,"mid-large")

        });
        $('#new_emp_btn').click(function(){
            uni_modal("New Employee","manage_employee.php")
        })
        $('.remove_employee').click(function(){
            _conf("Are you sure to delete this employee?","remove_employee",[$(this).attr('data-id')])
        })
    });
    function remove_employee(id){
        start_load()
        $.ajax({
            url:'ajax.php?action=delete_employee',
            method:"POST",
            data:{id:id},
            error:err=>console.log(err),
            success:function(resp){
                    if(resp == 1){
                        alert_toast("Employee's data successfully deleted","success");
                            setTimeout(function(){
                            location.reload();

                        },1000)
                    }
                }
        })
    }
</script> -->
```

9.2 Screen Shots

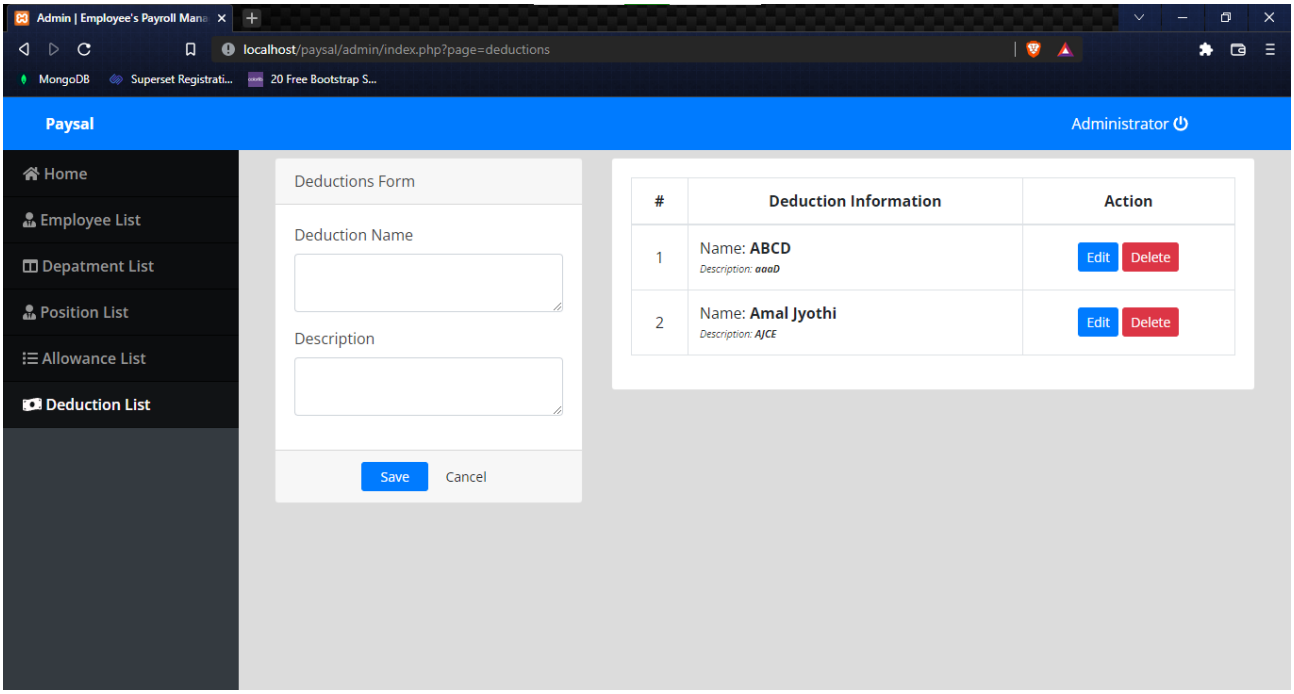
Home page



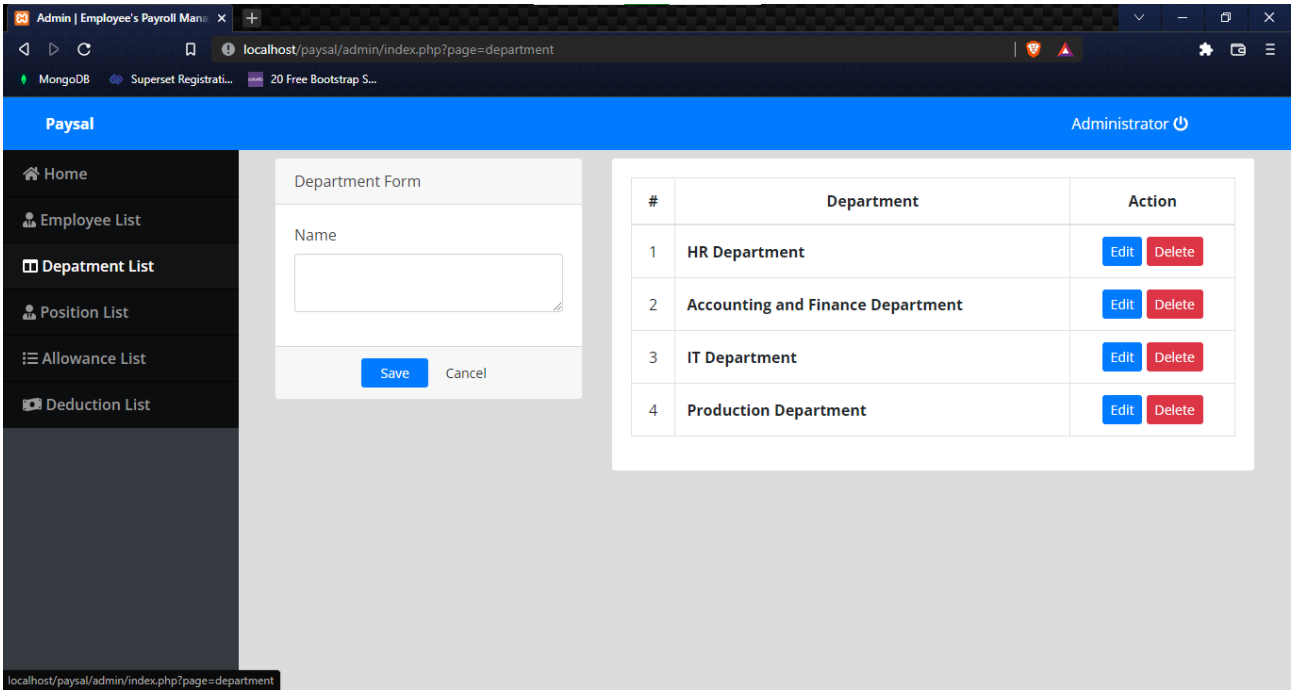
Add Allowances page



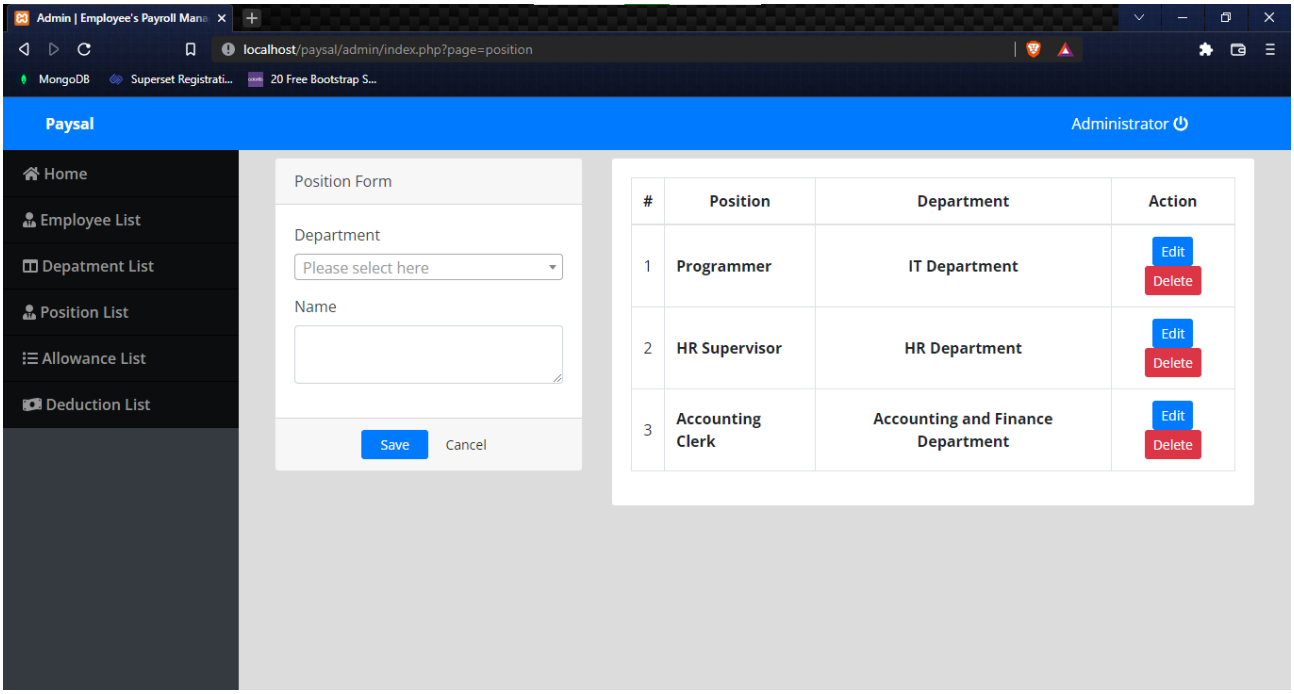
Add Deductions page



Add Departments



Add positions



Manage employee page