

CHAPTER – 1

INTRODUCTION

“Payroll Management” is a distributed application, developed to evaluate the performance of employees working in any organization. It maintains the information about a company, personal details of their employees, also the projects details assigned to particular developer. The application is actually a site of applications developed using Java.

It is simple to understand and can be used by anyone who is not even familiar with simple employee system. It is User friendly and just asks the user to follow step by step operations by giving him few options. It is fast and can perform many operations of accompany.

The Payroll Management System is designed to automate the existing manual system using computerized equipment and cutting-edge computer software, meeting client’s needs so that their valuable data and information can be stored for a longer period with easy access and manipulation.

1.1 OBJECTIVE

The payroll management system is a set of processes that helps you streamline salaries, bonuses, deductions, taxes, and other necessary aspects of the net pay of all the employees in your organization. You don't have to worry about handling, managing, and creating pay slips, salaries, and deductions of the employees. The tax deductions are also automated or handled by the outsourcing team.

To save Business Cost

- Reduce the spending on maintenance of software system and technology upgrades
- Control operating cost by keeping a low headcount of employees
- Reduces spending on technology upgrades

Other indirect time/cost savings

- Reduces errors from manual handling of data
- Reduces risks involved in security and confidentiality issues

Increase the level of work convenience in different staff-

- Real-time management reports
- Attain cost savings by reducing capital investments

Strengthen internal control and planning

- Reduce turnover, rehiring and retraining

Main objectives:

- To prepare the detailed salary record of all the employees in an organization .
- To generate pay- in-slip through the calculation of salary.
- Proper usage of manpower.
- To maintain allowances, deductions, loans, savings and arrears details for the

employees.

- To generate reports in user-defined format

1.2 LITERATURE SURVEY

Improved HR is another great advantage of outsourcing business processes. Several market studies have shown that HR outsourcing can lead to cost savings in the range of 20-40 percentage for customers. It also leads to improved HR. HR business process outsourcing (HR BPO) is a market that has experienced extensive growth over the past few years and is now set for even more rapid expansion. This is based on the fact that gradually more companies are looking at outsourcing transactions and processes to create a more strategically focus HR function.

Clearly, companies all over the world are getting motivated to outsource their HR processes and transactions to run their HR functions more efficiently, free the HR function to focus on strategic people practices that drive growth and add to the organizations long-term success and facilitate their professionals to add value their business. Companies today, require complete domestic and global HR delivery through systems and processes that can generate economics of scale to reduce or eliminate their need to make future investments in quickly outdated technology. Companies can save between 20-40 percentage of their HR costs, depending on their business priorities and the pace at which they want to move.

Indian BPO Segments

Business process customer interaction services are among the fastest outsourcing in India is organized in many segments. Back-office processing and and largest growing segments that contribute significantly to the Indian BPO market, the main activities are areas covered by the BPOs include customer, such as remote maintenance, help desk, and sales support; finance and administration, examples of which are data analysis, insurance claims, and HR and payment service including payroll, credit-card services, check processing, and employee leasing.

CHAPTER – 2

PREVIOUS AND RELATED WORK

One of the key business functions of any organisation is maintaining the payroll management system works in India. It directly contributes to the employee compensation like advances, attendance and employee leave management, bonuses, loans, and statutory compliance and tax deductions concerning the company's policies. Quite a set of legal obligations related to payroll are exercised on firms that are executing business affairs in India; this, in turn, complicates the programming logic required to control the tasks of payroll software.

The payroll management system is defined as a HR Laws that is mandatory in determining the salary of professionals working for the company. Earlier, loads of paperwork were used to defer the whole procedure for the company and the employees, particularly the staff employed in the finance and human resource divisions. Apart from the stress of managing and documenting huge numbers of physical files, there used to exist numerous chances of error posting. A company's overall productivity was affected due to deferred monthly payments that further lowered employee morale.

It the present decade, IT infrastructure has considerably improved, facilitating many firms to enjoy the privilege of using advanced payroll software. By incorporating these systems, the HR personnel have been relieved from tedious monthly routines. Salaries now get credited at the beginning of the month without fail; also, the company's finance team never misses the tax paying dates and thus can consistently preserve their image in the industry segment.

Payroll Management Companies – Their Significance in India

To successfully manage payroll management system in India, we recommend you recruit a payroll management firm. Professionals hired in those agencies are experienced and qualified sufficiently to execute all necessary exercises required to keep a check on the employee payroll services. They also offer transparency to the hirers regarding the payroll-related tasks mechanism. The experts take care of the entire set of formal proceedings starting from the post-payroll phase.

The entrusted agency is provided with a detailed document of the company norms jotted down by the human resource division. Employees also abide by these rules as long as they work for that organisation. In the next step, all crucial information is collected from the respective departments and conveyed to the payroll management company. This action demands interdepartmental coordination to complete the payroll process in time. The agency checks the

legitimacy of all databases fetched from various departmental records as a cautionary action. This step cannot be skipped; otherwise, the company might face huge losses if any of the elements related to the employee salary slip is miscalculated.

Let us now understand the post-payroll steps. Again, these hold considerable significance as, in this phase, reports are prepared that need to be presented in front of the governmental agencies. Many key activities such as EPF, TDS and ES come into consideration depending on the annual turnover of the company and its overall employee strength. Without paying attention to the factors above, it is not feasible to announce the net salary of the employees. Thus, each activity appears to be of equal significance. Lastly, the service-providing agency duly records all the statements of the invoices for the efficient functioning of the payroll management system.

CHAPTER – 3

PROPOSED SYSTEM

This being a web-based application is very easy to access from anywhere with internet. Data and records are managed in centralized database. Data is secure and easy to retrieve, store, and analyse, so chances of miscalculations and occurrence of error are very less. The services provided by the proposed payroll management system are payroll checks and reports, time sheet, and tax reports. It processes monthly payroll in shortest time frame and keeps employees updated on their statutory obligation. It has an easy to use interface which makes it easy to analyse different aspects included in Employee's salary.

The employee information handling and time management is the responsibility of the administrator. Admin issues a specific id to the employees and staffs with which employees are required to register into the system and submit their particulars. This system itself updates based on whatever is required or requested from the user-end. In existing traditional and manual system, the chances of error occurrence would be very high, but this being an online computerized application, gives users alert messages, helps, and warnings on whatever required or requested. The proposed software will solve all the problems they are facing now.

The software is designed such that it will generate the salary automatically every month in time. So there are not much worries. The software is also equipped with the ability to enter the attendance of each employee in the organisation, it helps them track each employee attendance, based on this we can generate the salary. The software is built to generate individual pay slip and summary of the payroll. It also has an option to generate the report for provident fund and ESI. So they can take the print out of provident fund and ESI to submit to the department.

3.1 PROCESS MODEL WITH JUSTIFICATION

SDLC :

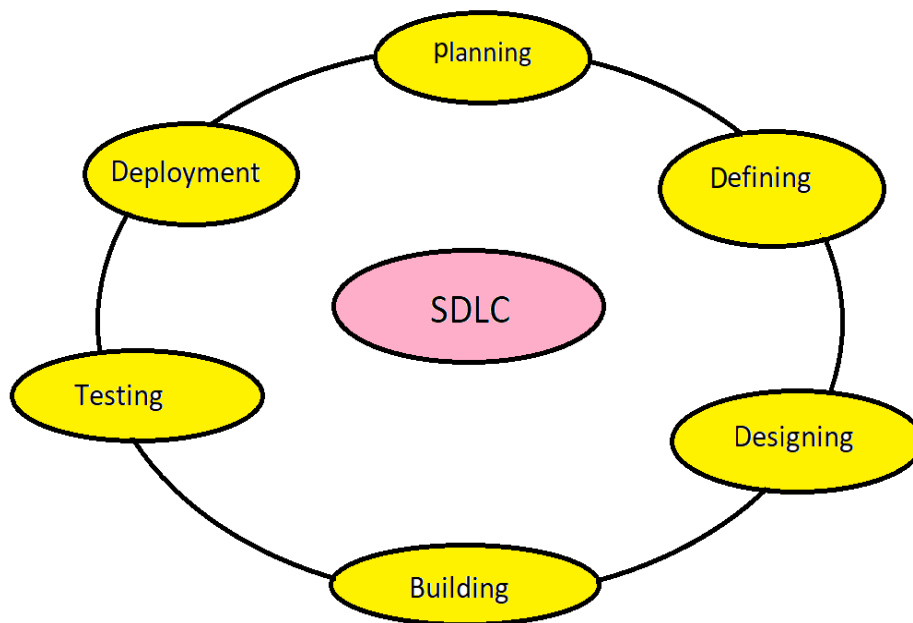
Software Development Life Cycle is a process used by the software industry to design, develop and test high quality software. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

SDLC is the acronym of Software Development Life Cycle. It is also called as Software Development Process. SDLC is a framework defining tasks performed at each step in the software development process.

What is SDLC?

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.



Stages of SDLC

A typical Software Development Life Cycle consists of the following stages

Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and thirdparty modules. The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment User acceptance testing.

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targetting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

Tkinter

- Tkinter is a Python interface to the Tk graphics library.
- Tk is a graphics library widely used and available everywhere
- Tkinter is included with Python as a library.

To use it:

```
– import * from Tkinter  
  
    (or)  
  
_ from Tkinter import *
```

What can it do?

- Tkinter gives you the ability to create Windows with widgets in them
- Definition: widget is a graphical component on the screen (button, text label, drop-down menu, scroll bar, picture, etc...)
- GUIs are built by arranging and combining different widgets on the screen.

TIKINTER WIDGETS

Message Canvas

Check button

LableFrame

Entry

Text

Button

MessageBOX

Panned Window

Frame

Label

ListBox

Scale

Spinbox

Menubutton

Toplevel

Menu

Scrollbar

Radiobutton

Button ():

The Button Widget is used to add various kinds of buttons to the python application.

Ex: W = Button (parent, options)

Different options Under Buttons:

There are many options under this Button Widget they are as follows

Active foreground

Bd

Bg

Command

Fg

Font

Height

Relief

State

Underline

Width

Wrap length

High light colour

Image

Justify

Message box:

Message Box Widget is used to display the message boxes in the python applications. This module is used to display a message using provides a number of functions.

Message box widget options

Bg

bd

Cursor	font
Fg	height
Image	justify
Relief	text
Textvariable	underline
Width	wraplength

Python Modules:

A python module can be defined as a python program file which contains a python code including python functions, class, or variables. In other words, we can say that our python code file saved with the extension (.py) is treated as the module. We may have a runnable code inside the python module.

Modules in Python provides us the flexibility to organize the code in a logical way.

Loading the module in our python code:

1. The import statement
2. The from-import statement

1.The import statement

The import statement is used to import all the functionality of one module into another. Here, we must notice that we can use the functionality of any python source file by importing that file as the module into another python source file. We can import multiple modules with a single import statement, but a module is loaded once regardless of the number of times, it has been imported into our file.

2.The from-import statement

Instead of importing the whole module into the namespace, python provides the flexibility to import only the specific attributes of a module. This can be done by using from? import statement.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION

3.2.1 OVERALL DISCRPTION

A Software Requirements specification (SRS) – a requirement specification for a software system is a complete description of the behavior of the system to be developed. It includes a set of use cases that describe all the interactions the user will have with the software in addition to use cases, the SRS also contain non-functional requirements. non-functional requirements are requirements which impose constrains on the design or implementation.

SRS is a structured collection of information that embodies the requirements of a system. A Software Requirements Specification (SRS) is a document that describes the nature of a project, software or application. In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application. This document is also known by the names SRS report, software document. A software document is primarily prepared for a project, software or any kind of application.

There are a set of guidelines to be followed while preparing the software requirement specification document. This includes the purpose, scope, functional and nonfunctional requirements, software and hardware requirements of the project. In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc.

3.2.2 EXTERNAL INTERFACE REQUIREMENTS

User Interface

The user interface of the system is user friendly python Graphical User Interface.

Hardware Interface

The interaction between user and Tkinter through python capabilities.

Software Interface

The software is python

Front end is Tkinter

Back end is python

Operating Environment

Windows11

Hardware Requirements

- Processor - 1GHz
- Memory (RAM) - 8GB
- Hard disk - 64GB

Software Requirements

- Operating system - Windows 11
- Programming language - python 3.1

CHAPTER – 4

SYSTEM DESIGN

The design involves the production of technical and visual prototypes. This stage has some non-technical aspects such as gathering of web content. For the serverside programming and other technical aspects of the design emphasis will be laid on such design concepts and principles as effective modularity, information hiding and stepwise elaboration.

4.1 SYSTEM ARCHITECTURE

Payroll Management System Project in Mongodb Node.js with Source Cod. This system is meant to supply the power to line up all the tasks of employee payment. At first, the user has got to undergo login system to urge access, then the user can add, list, update and take away the employee's record. This system deals with the financial aspects of employee's Salary, Deductions, allowances, Net pay.

The user can view the account of each and every employee's and update their payments, and the user can also manage deductions, modify overtime and salary rate. Each and every detail about employee's payment is displayed which includes: Name with deduction, overtime, bonus and net pay. This system makes easier to the user for managing payroll system as it is not time-consuming. This project is not difficult to operate and understood by the users.

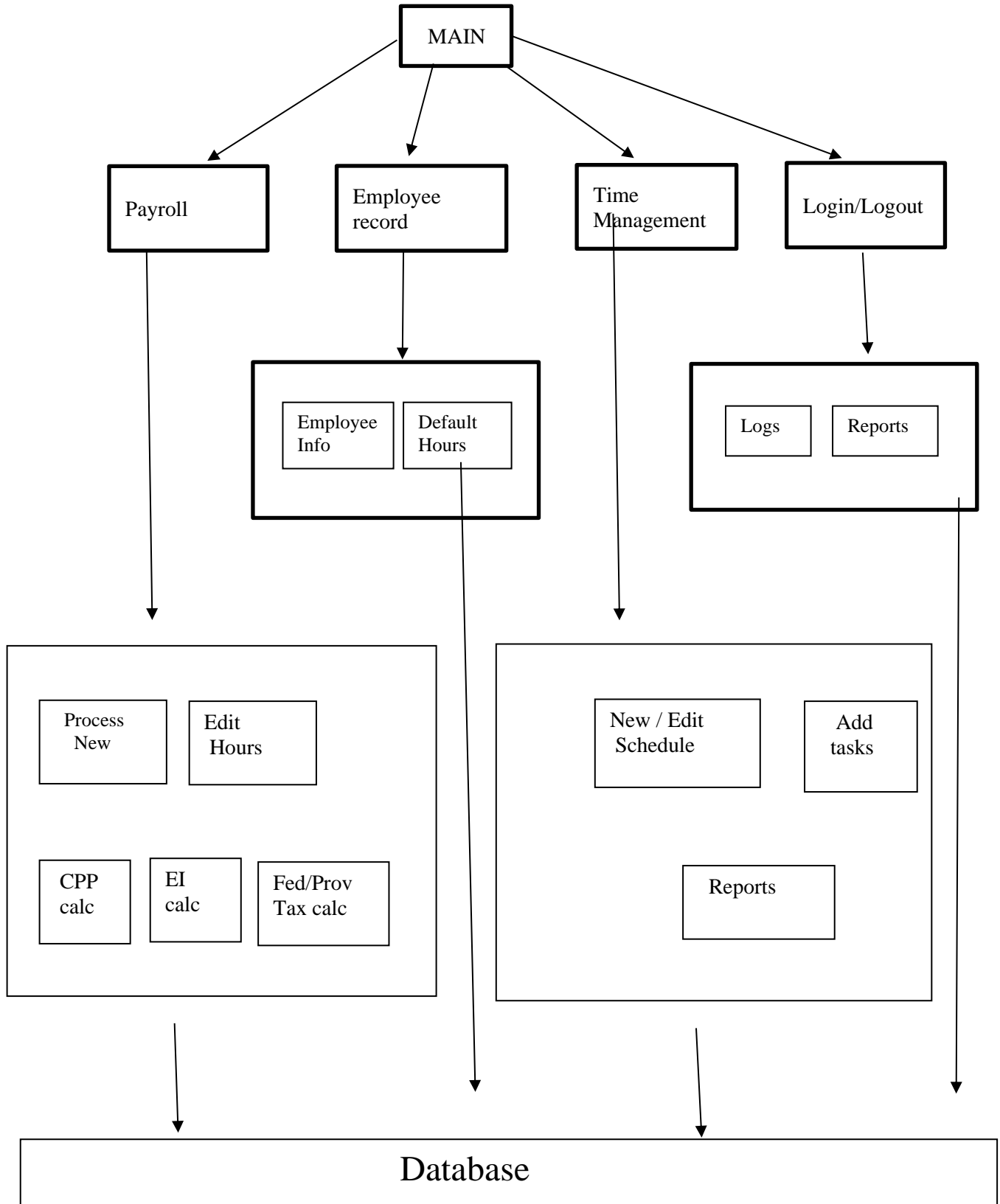


Figure 4.1: System Architecture

4.2 MODULES

Payroll management system is a three-modules project. These modules are briefly described below:

1.Login

This simple module allows administrator and employees to log in to the system with their unique user ids username and password.

2.Employee

Employees can log in to payroll system using their specific user ids. They can analyse salary and deductions such as tax, loan, etc for every month. With this module employees can interact with the system administrator, manage profiles, and generate pay slip report. The forms or sub-modules under employee module are:

- **New employee registration:** New employees are required to log in to the system by filling out this form.
- **Update employee:** This form allows employees to update their profile.
- **Apply leave:** This form can be used by employees to ask for leave. It contains information such as no. of days, reasons for applying leave, apply date, and leave date.
- **View salary report:** Employees can view their salary report along with deductions via this form.
- **Check leave:** The status of the leave asked for or applied is displayed by this form.
- **Work sheet:** Employees can enter work of a particular day.
- **Forgot password:** Employees can retrieve the password if they ever forget password to log in to the system.

4.3 UML DIAGRAMS

A component diagram in the (UML) Unified Modeling Language shows how parts are wired together to explain the parts of payroll management systems. They are used to show the structure of any kind of system.

The UML component diagram shows how a payroll management system will be made up of a set of deployable components, such as dynamic-link library (DLL) files, executable files, or web services. Using well-defined interfaces, these parts communicate with each other and keep their internal details hidden from each other and the outside world

❖ Class Diagram

Payroll management system class diagram describes the structure of a payroll management system classes, their attributes, operations, and the relationships among objects. The main classes of the payroll management system are payroll, salary, employee, appraisals, working points, payments.

Classes of Payroll Management System Class Diagram:

- **payroll class:** Manage all the operations of Payroll
- **Salary class:** Manage all the operations of Salary
- **Employee class:** Manage all the operations of Employee
- **Appraisals class:** Manage all the operations of Appraisals
- **Working Points class:** Manage all the operations of Working Points
- **Payments class:** Manage all the operations of Payments

Classes and their attributes of Payroll Management System Class

Diagram:

- **Payroll Attribute:** payroll _title, payroll _type, payroll _description
- **Salary Attributes:** salary _id, salary _employee-id,
salary _amount, salary _total, salary _type,
salary _description
- **Employee Attributes:** employee _id, employee _name,
employee _mobile, employee _email,
employee _username, employee _password,
employee _address
- **Appraisals Attributes:** appraisal _id, appraisal _employee,
appraisal _name, appraisal _type,
appraisal _description
- **Working Points Attributes:** point -id, point -title, point -type,
point -description
- **Payments Attributes:** payment -id, payment -customer -id,
payment -date, payment amount,
payment -description

Classes and their methods of payroll Management System Class Diagram:

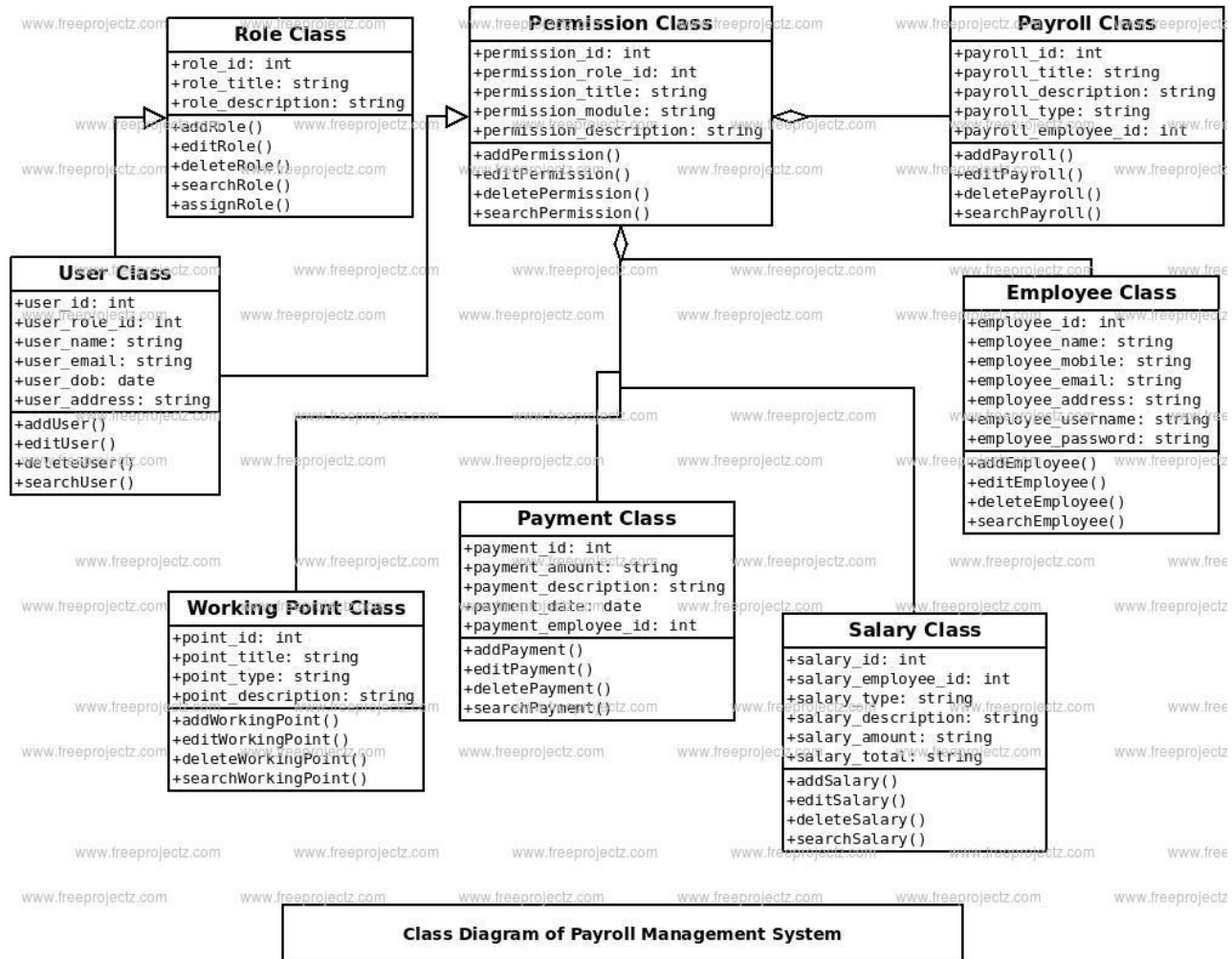


Figure 4.3.1: Class Diagram

EXPLANATION:

- **payroll Methods:** add Payroll (), edit Payroll (), delete Payroll (),
update Payroll (), save Payroll (), search Payroll ()
- **Salary Methods:** add Salary (), edit salary (), delete Salary (),
Update Salary (), save Salary (), search Salary ()
- **Employee Methods:** add Employee (), edit Employee (),
delete Employee (), update Employee (),
save Employee (), search Employee ()
- **Appraisals Methods:** add Appraisal (), edit Appraisals (),
delete Appraisals (), update Appraisals (),
save Appraisals (). Search Appraisals ()
- **Woking Points Methods:** add Woking Points (),
edit Woking Points (),
delete Woking Points (),
update Woking Points (),
save Woking Points (),
search Woking Points ()
- ❖ **Payments Methods:** add Payments (), edit Payments (),
delete Payments (), update Payments (),
save Payments (), search Payments ()

❖ Use Case Diagram

Use case diagrams are a sort of behaviour diagram that is defined and generated through use case analysis. Use case diagrams are utilized to graphically represent the operation of a system. Typically, they are used to map the functionality of a system and to illustrate how various actors interact with it.

Use case diagrams can be employed in a variety of ways to depict operation of a system. One technique is to utilize it to map out a system's requirements. This can assist in determining which features are required and how to implement them. It can also help keep track of how different actions will use the system and how they will interact with it.

Use case diagrams can also be used to illustrate how the system will be utilized. This can assist in determining which features are required and how to implement them. It can also help keep track of how different actors will use the system and how they will interact with it. Use case diagrams are an efficient way to map out the functionality of a system and illustrate how its many actors interact with it. They can be used to figure out what features are needed, how to build them, and how the system will be used.

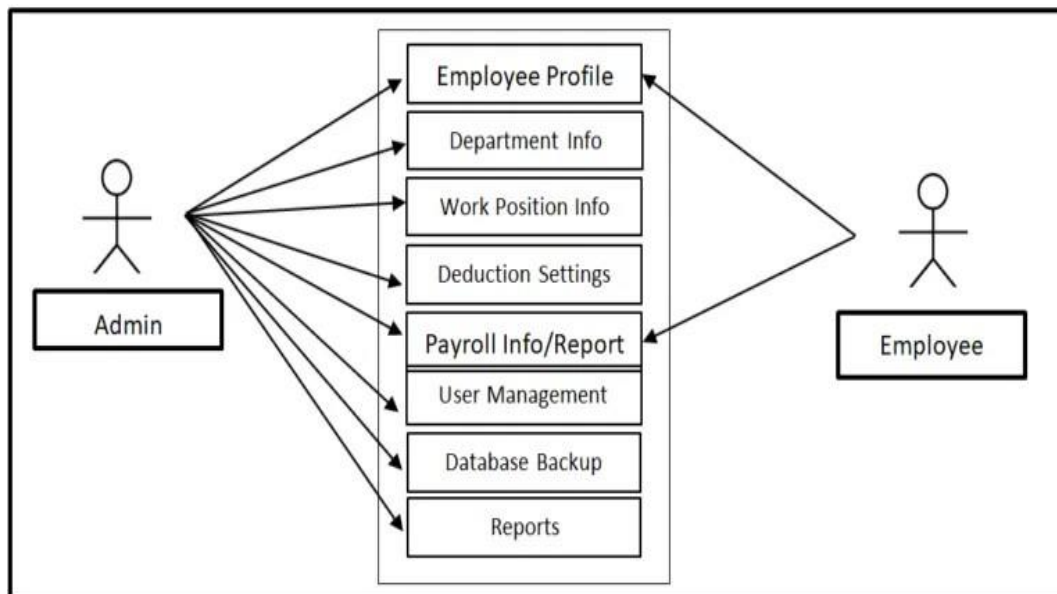


Figure 4.3.2: Use Case Diagram

❖ Sequence Diagram

Sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along their corresponding occurrence specification on the lifelines. To create a sequence diagram use case flow of events.

Payroll process sequence diagram it will perform three basic process function like personal Idetail Form () verification () store (). In their flow of the events show by the indicating arrow. Payroll sequence diagram we can get the clear view of the event process from the one object to the another. Traceability diagram it maintain employee details, sales details, report generator, employee details, sales details.

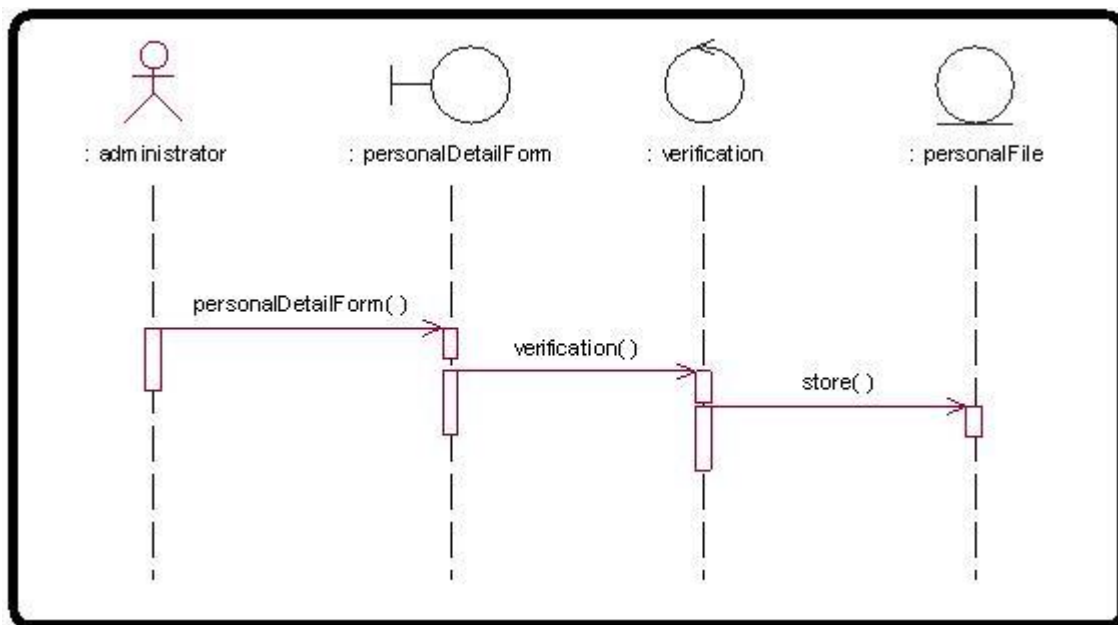


Figure 4.3.3: Sequence Diagram

❖ Collaboration Diagram

To auto generate a collaboration diagram for the basic flow of use case. By through pressing F5 from sequence diagram it will automatically generate the sequence diagram to collaboration diagram. These another view of the sequence diagram for the payroll process. it also have the same process and same function with compare to the payroll sequence diagram.

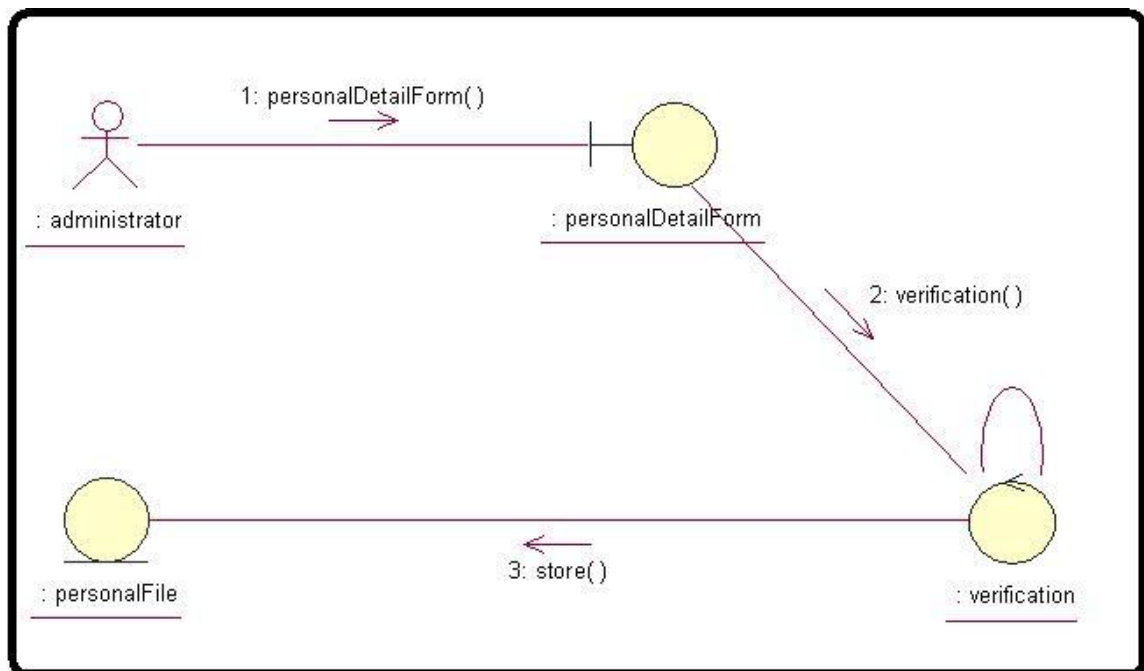


Figure 4.3.4: Collaboration Diagram

Result: Thus the collaboration diagram for “Maintaining Employee Details “ use case was auto generated. Rational rose case tool lap algorithm.

❖ Component Diagram

The component diagram for payroll management system is used to show how the system's parts work together to make the desired payroll operate correctly. This component diagram shows how the payroll components are organized and how they depend on each other. It gives a high-level look at the parts of a system. The potential components of a payroll management system can be part of software or hardware. They could be a database, a user interface, or something else that helps the payroll management system work.

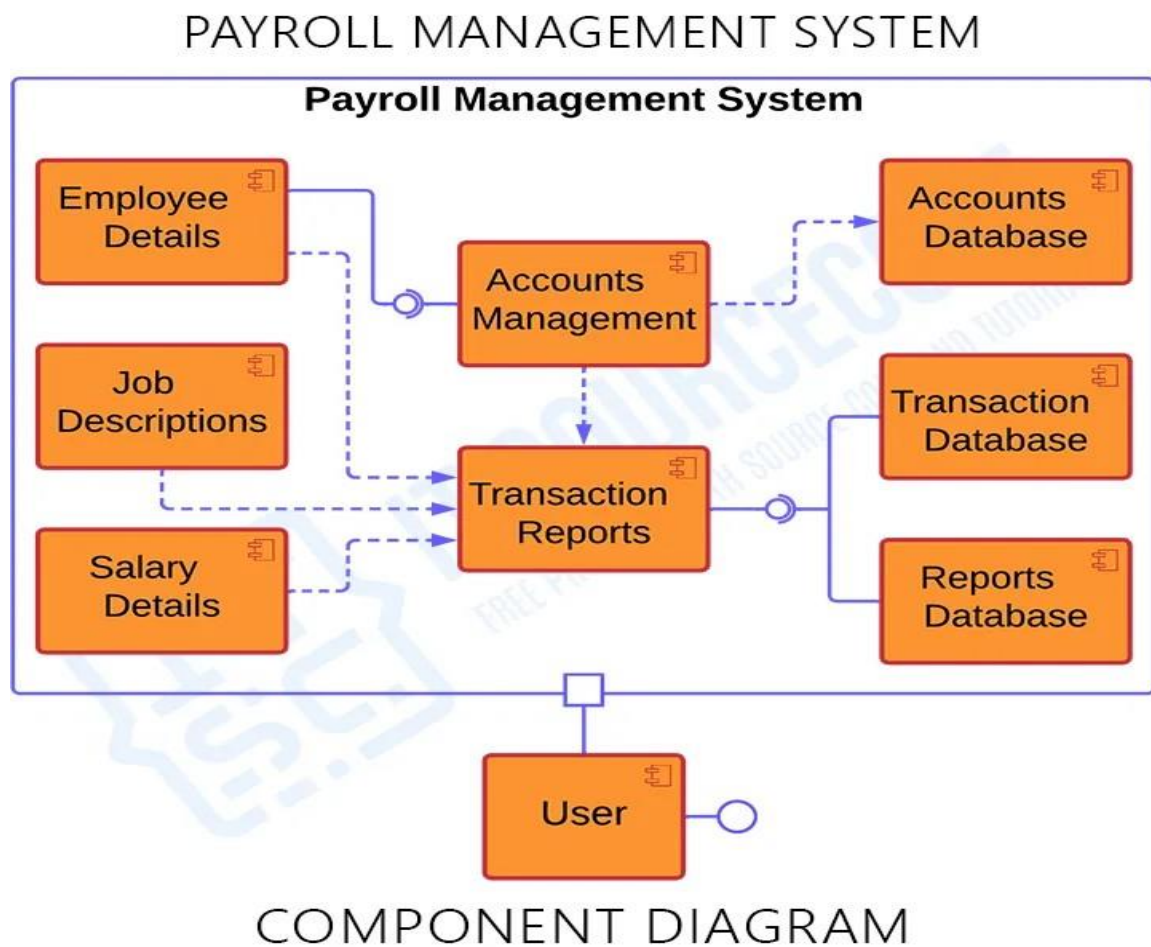


Figure 4.3.5: Component Diagram

The Payroll Management System UML component diagram explains the sketch of the required software and hardware components and the dependencies between them. These components are labeled to clarify their part in the system's operation. They were represented by symbols that explain their function and role in the overall payroll management system operation. The dependencies on each component are explained through the lines and arrows drawn in the diagram.

❖ Deployment Diagram

The designed deployment diagram for payroll system shows the components (nodes) included to carry out the process. Nodes are represented by boxes that are labeled as software or hardware that specifies the included components to carry out the payroll management process. The boxes will then be connected and labeled to declare the type of connection they have with the other components.

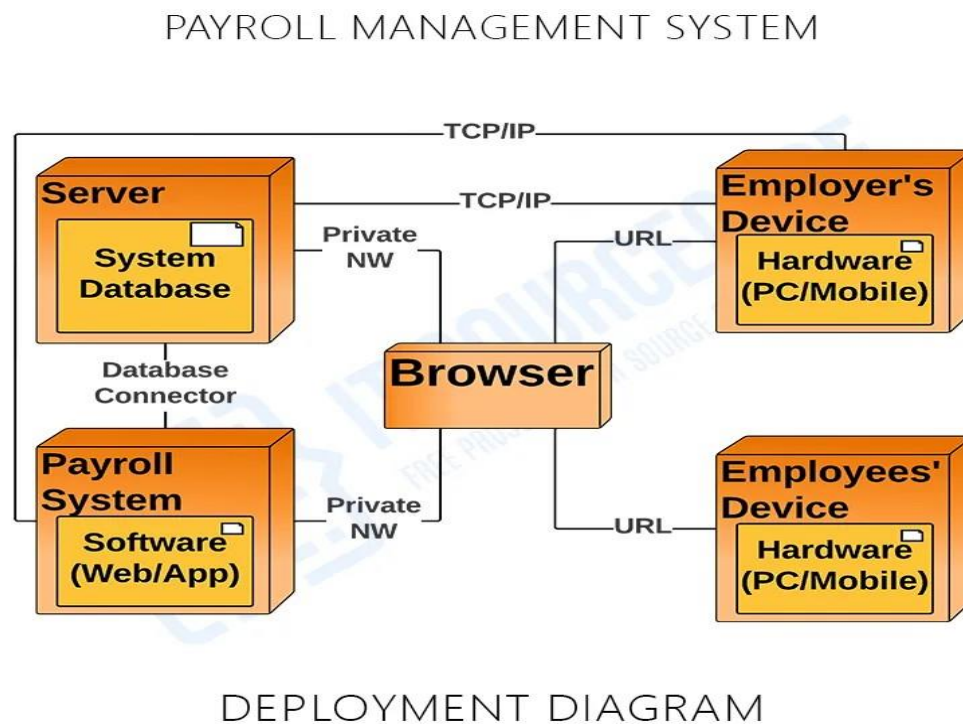


Figure 4.3.6: Deployment diagram

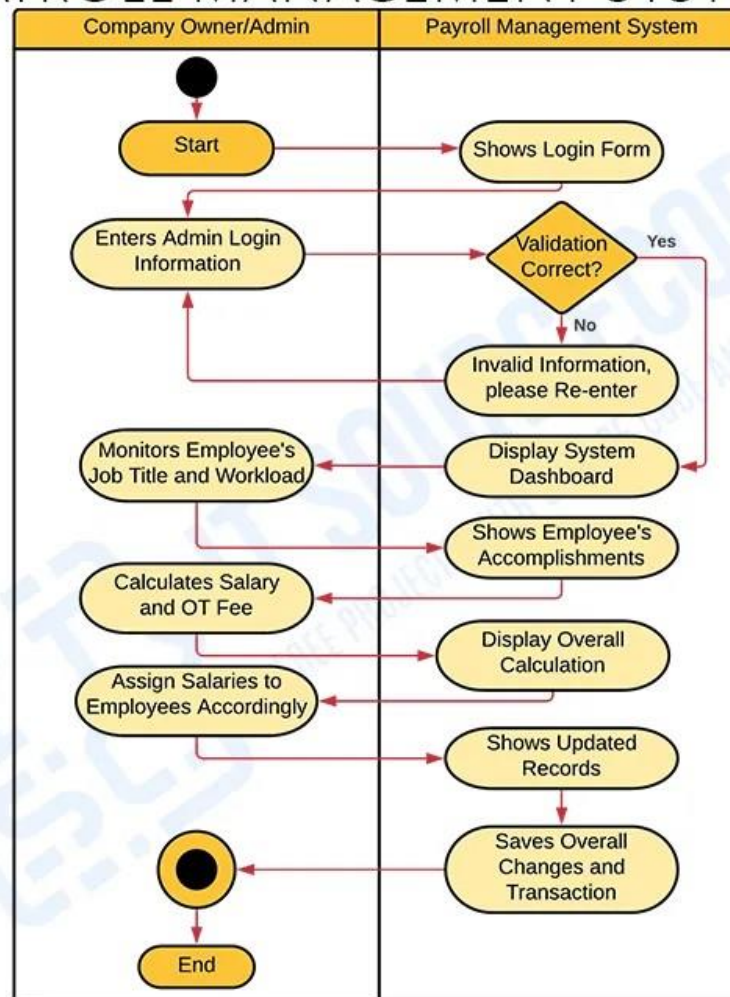
The deployment diagram shows the scenario when the system is deployed. It has 5 nodes represented with boxes and relationship connections. The nodes are the payroll management system, the browser, the employer's device, the employee's device, and the database (system server). The system server node contains a developed database that will hold the details of the system online.

For the connection, the system is connected to the server database using a private network which enables it to pass a connection to the devices and enable users to access the system and database. The employer and the employees then can communicate through the system.

❖ Activity Diagram

A Payroll Management System Activity Diagram is a picture that shows how the system works. It shows how the Payroll Management System interacts with the people who use it. The Payroll Management System UML Activity Diagram also gives the people working on the project good ideas and helps them through the whole business process of making software.

PAYROLL MANAGEMENT SYSTEM



ACTIVITY DIAGRAM

Figure 4.3.6: Activity Diagram

This diagram shows what happens when the project admin or owner uses it. It starts by validating the admins' information to make sure that the user is the main admin or the owner.

Then, the system will show the overall activity collaboration, which only the admin can see. Because of this, the activity diagrams for the employees and the admins were made separately.

You should be told that the diagram can be changed to make the system work the way you want it to. You can also make your own system function to meet the needs of all of your clients. And if you want, you can use all of the ideas here instead of making your own.

You could also combine this diagram with others to make your activity diagram more general. But you still need to be specific so that your readers and people who use the system can understand how the Payroll Management software development works.

4.4 DATA FLOW DIAGRAM

Payroll Management System Data flow diagram is often used as a preliminary step to create an overview of the Payroll without going into great detail, which can later be elaborated. It normally consists of overall application dataflow and processes of the Payroll process. It contains all of the userflow and their entities such as all the flow of Employee, Attendance, Leave, Salary, Task, Pay-Slip, Department. All of the below diagrams have been used for the visualization of data processing and structured design of the Payroll process and working flow.

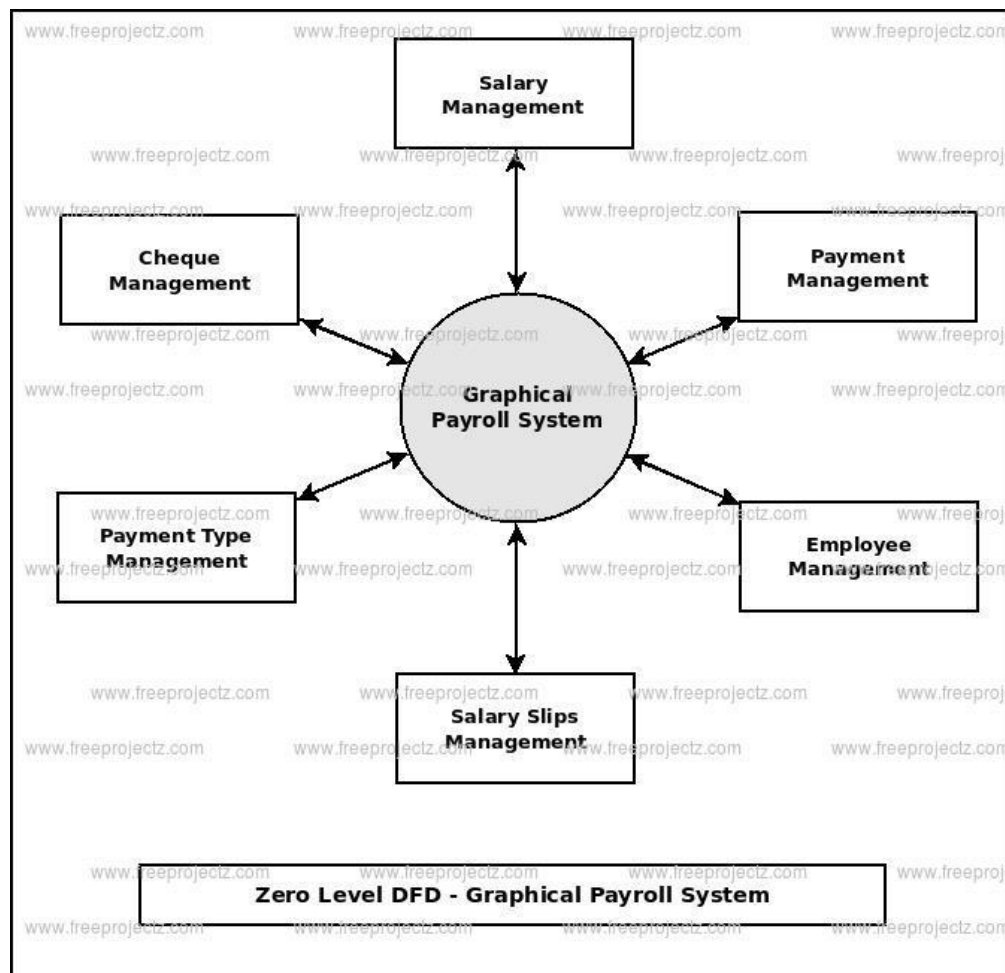
Zero Level Data Flow Diagram (0 Level DFD) Of Payroll Management System:

This is the Zero Level DFD of Payroll Management System, where we have elaborated the high level process of Payroll. It's a basic overview of the whole Payroll Management System or process being analyzed or modeled. It's designed to be an at-a-glance view of Task, Pay-Slip and Department showing the system as a single high-level process, with its relationship to external entities of Employee, Attendance and Leave. It should be easily understood by a wide audience,

including Employee, Leave and Task In zero level DFD of Payroll Management System, we have described the high level flow of the Payroll system.

High Level Entities and process flow of Payroll Management System:

- Managing all the Employee
- Managing all the Attendance
- Managing all the Leave
- Managing all the Salary
- Managing all the Task
- Managing all the Pay-Slip
- Managing all the Department



4.4.1 Zero Level Data Flow Diagram

First Level Data Flow Diagram (1st Level DFD) Of Payroll Management

System:

First Level DFD (1st Level) of Payroll Management System shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the Payroll Management System system as a whole. It also identifies internal data stores of Department, Pay-Slip, Task, Salary, Leave that must be present in order for the Payroll system to do its job, and shows the flow of data between the various parts of Employee, Leave, Pay-Slip, Department, Task of the system.

DFD Level 1 provides a more detailed breakout of pieces of the 1st level DFD. You will highlight the main functionalities of Payroll.

Main entities and output of First Level DFD (1st Level DFD):

- Processing Employee records and generate report of all Employee
- Processing Attendance records and generate report of all Attendance
- Processing Leave records and generate report of all Leave
- Processing Salary records and generate report of all Salary
- Processing Task records and generate report of all Task
- Processing Pay-Slip records and generate report of all Pay-Slip
- Processing Department records and generate report of all Department

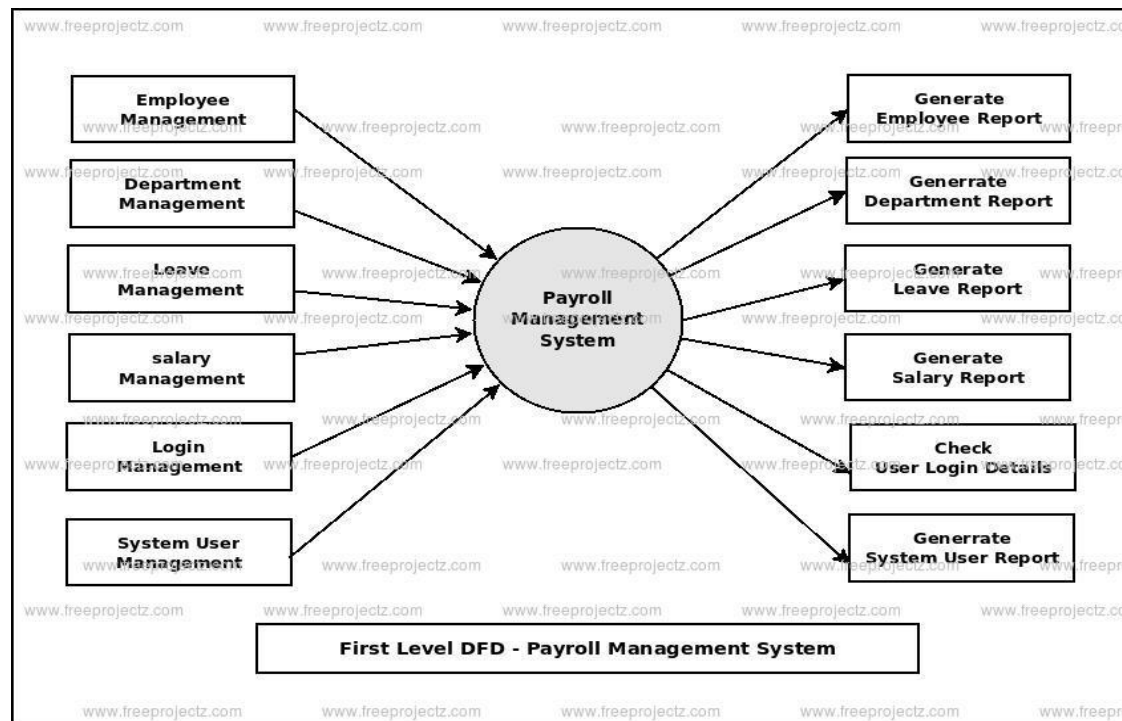


Figure 4.4.2: First Level Data Flow Diagram

Second Level Data Flow Diagram (2nd Level DFD) Of Payroll Management System:

DFD Level 2 then goes one step deeper into parts of Level 1 of Payroll. It may require more functionalities of Payroll to reach the necessary level of detail about the Payroll functioning.

First Level DFD (1st Level) of Payroll Management System shows how the system is divided into sub-systems (processes). The 2nd Level DFD contains more details of Department, Pay-Slip, Task, Salary, Leave, Attendance, Employee.

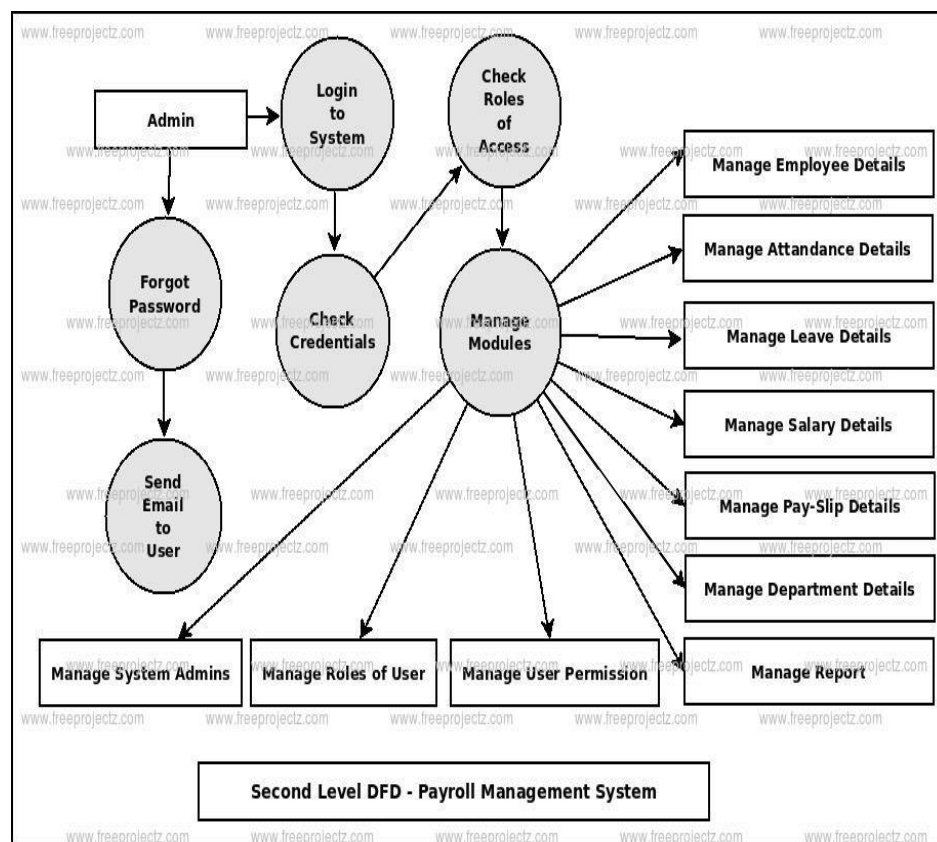


Figure 4.4.4 Second Level Data Flow Diagram

Low level functionalities of Payroll Management System:

- Admin logs in to the system and manage all the functionalities of Payroll Management System
- Admin can add, edit, delete and view the records of Employee, Leave, Task, Department
- Admin can manage all the details of Attendance, Salary, Pay-Slip
- Admin can also generate reports of Employee, Attendance, Leave, Salary, Task, Pay-Slip
- Admin can search the details of Attendance, Task, Pay-Slip
- Admin can apply different level of filters on report of Employee, Salary, Task
Admin can track the detailed information of attendance, leave, salary, task.

CHAPTER – 5

IMPLEMENTATION

5.1 PYTHON

Definition of python:

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

History of Python:

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991 as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989.

Why Python was created?

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

Why the name Python?

It wasn't named after a dangerous snake. Rossum was fan of comedy series from late seventies. The name “python” was adapted from the same series “Monty Python’s Flying Circus”.

FEATURES OF PYTHON:

- **A simple language which is easier to learn**

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.

- **Free and open-source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software written in it, you can even make changes to the Python's source code. Python has a large community constantly improving it in each iteration.

- **Portability**

You can move Python programs from one platform to another, and run it without any changes. It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

- **Extensible and Embeddable**

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code. This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

- **A high-level, interpreted language**

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on. Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

5.2 TKINTER

- Tkinter is a Python interface to the Tk graphics library.
- Tk is a graphics library widely used and available everywhere
- Tkinter is included with Python as a library.

To use it:

```
– import * from Tkinter  
  
    (or)  
  
_ from Tkinter import *
```

What can it do?

- Tkinter gives you the ability to create Windows with widgets in them
- Definition: widget is a graphical component on the screen (button, text label, drop-down menu, scroll bar, picture, etc...)
- GUIs are built by arranging and combining different widgets on the screen.

In this project we used some widgets from Python tkinter Widgets are

Message	Canavas
Button	Check button
MessageBOX	LableFrame
Entry Frame	Label
Panned Window Text	ListBox
Menubutton	Scale
Spinbox	Menu
Toplevel	Scrollbar

5.3 DATA BASE (SQLITE)

A “database” is a collection of related data which represents some aspect of the real world. A database system is designed to be built and populated with data for a certain task. “Database Management System” is a software for storing and retrieving users’ data while considering appropriate security measures. It consists of a group of programs which manipulate the database.

The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large users and other third-party software to store and retrieve data. DBMS allows users to create their own databases as per their requirement. The term “DBMS” includes the user of the database and other application programs. It provides an interface between the data and the software application.

Popular DBMS Software

Here, is the list of some popular DBMS system:

MySQL	Microsoft Access
Oracle	PostgreSQL
dBASE	FoxPro
SQLite	IBM DB2
LibreOffice Base	MariaDB
Microsoft SQL Server etc.	

Application of DBMS

Below are the popular database system applications:

Sector	Use of DBMS
Banking	For customer information, account activities, payments, deposits, loans, etc.
Airlines	For reservations and schedule information.
Universities	For student information, course registrations, colleges and grades.

Telecommunication

It helps to keep call records, monthly bills, maintaining balance, etc.

Finance

For storing information about stock, sales, and purchases of financial instruments like stocks and bonds.

Types of DBMS

The main Four Types of Database Management System are

- Hierarchical database
- Network database
- Relational database
- Object-Oriented database

Hierarchical DBMS

In a Hierarchical database, model data is organized in a tree-like structure. Data is Stored Hierarchically format. Data is represented using a parent-child relationship. In Hierarchical DBMS parent may have many children, but children have only one parent.

Network Model

The network database model allows each child to have multiple parents. It helps you to address the need to model more complex relationships like as the orders/parts many-to-many relationship. In this model, entities are organized in a graph which can be accessed through several paths.

Relational Model

Relational DBMS is the most widely used DBMS model because it is one of the easiest. This model is based on normalizing data in the rows and columns of the tables. Relational model stored in fixed structures and manipulated using SQL.

Object-Oriented Model

In Object-oriented Model data stored in the form of objects. The structure which is called classes which display data within it. It is one of the components of DBMS that defines a database as a collection of objects which stores both data members values and operations.

5.4 SOURCE CODE

```
From tkinter import *
from tkinter import ttk
import random
import time
import datetime
import sqlite3
import tkinter.messagebox as mb

##GLOBALFONTS#####
headlabelfont = ("Noto Sans CJK TC", 15, 'bold')
labelfont = ('Garamond', 14)
entryfont = ('Garamond', 12)

payroll = Tk()
payroll.geometry("1350x800")
payroll.resizable(1,1 )
payroll.title("Payroll Management Systems")
#####db#####
connector = sqlite3.connect('payroll.db')
cursor = connector.cursor()
connector.execute(
"CREATE TABLE IF NOT EXISTS payroll (Empid INTEGER , EmployeeName TEXT,
Reference TEXT, NetPay TEXT,Totalsalary TEXT, NINumber TEXT, NICode TEXT)"
)
print("created db")

def exit():
    payroll.destroy()

def reset():
    EmployeeName.set("")
```

```

Address.set("")
Reference.set("")
EmployerName.set("")
City.set("")
Basic.set("")
OverTime.set("")
GrossPay.set("")
NetPay.set("")
Tax.set("")
PostCode.set("")
Gender.set("")
PayDate.set("")
Pension.set("")
StudenLoan.set("")
NIPayment.set("")
Totalsalary.set("")
TaxPeriod.set("")
NINumber.set("")
NICode.set("")
TaxablePay.set("")
PensionablePay.set("")
OtherPaymentDue.set("")

```

```
def PayRef():
```

```

    PayDate.set(time.strftime("%d/%m/%Y"))
    refPay = random.randint(20000, 709467)
    refPaid = int( str(refPay))
    Reference.set(refPaid)

```

```

    NIPay = random.randint(20000, 559467)
    NIPaid = int(str(NIPay))
    NINumber.set(NIPaid)

```

```
def add_record():
```

```

    global empid,ename,ref,netpay,totalsalary,ninumber,nicode
    empid = Empid.get()
    ename = EmployeeName.get()
    ref = Reference.get()
    netpay = NetPay.get()
    totalsalry = Totalsalary.get()
    ninumber = NINumber.get()

```

```

nicode = NICode.get()

print(empid)
print(ninumber)
print(ename)
if not empid or not ename or not ref or not netpay or not totalsalary or not ninumber or not
nicode:
    mb.showerror('Error!', "Please fill all the missing fields!!")
else:
    try:
        connector.execute(
            'INSERT INTO payroll(Empid, EmployeeName, Reference, NetPay,
Totalsalary,NINumber,NICode) VALUES (?, ?, ?, ?, ?, ?, ?)', (empid, ename, ref, netpay,
totalsalry, ninumber , nicode)
        )
        connector.commit()
        mb.showinfo('Record added', f"Record of {ename} was successfully added")
        reset()
    except:
        mb.showerror('Wrong type', 'The type of the values entered is not accurate. Pls note that
the contact field can only contain numbers')

def PayPeriod():
    i = datetime.datetime.now()
    TaxPeriod.set(i.month)

    NCode = random.randint(1200, 3467)
    CodeNI = int(str(NCode))
    NICode.set(CodeNI)

def MonthlySalary():
    if Basic.get() == "":
        BS = 0
    else:
        try:
            BS = float(Basic.get())
        except ValueError:
            messagebox.showinfo("Error", "Wrong values!!! Use numbers.")
            Basic.set("")

    if City.get() == "":

```



```

    CW = 0
else:
    try:
        CW = float(City.get())
    except ValueError:
        messagebox.showinfo("Error", "Wrong values!!! Use numbers.")
        City.set("")

if OverTime.get() == "":
    OT = 0
else:
    try:
        OT = float(OverTime.get())
    except ValueError:
        messagebox.showinfo("Error", "Wrong values!!! Use numbers.")
        OverTime.set("")

MTax = ((BS + CW + OT) * 0.1)
TTax = int(MTax)
Tax.set(TTax)

M_StudenLoan = ((BS + CW + OT) * 0.1)
MM_StudenLoan = int(M_StudenLoan)
StudenLoan.set(MM_StudenLoan)

M_Pension = ((BS + CW + OT) * 0.1)
MM_Pension = int(M_Pension)
Pension.set(MM_Pension)

M_NIPayment = ((BS + CW + OT) * 0.1)
MM_NIPayment = int(M_NIPayment)
NIPayment.set(MM_NIPayment)

Deduct = MTax + M_Pension + M_StudenLoan + M_NIPayment
D_DEDUCT= int(Deduct)
Totalsalary.set(D_DEDUCT)

NetPayAfter = ((BS + CW + OT) - Deduct)
NetAfter = int(NetPayAfter)
NetPay.set(NetAfter)

```

```

Gross_Pay = int((BS + CW + OT))
GrossPay.set(Gross_Pay)

TaxablePay.set(TTax)
PensionablePay.set(MM_Pension)
OtherPaymentDue.set("0.00")
Empid=StringVar()
EmployeeName = StringVar()
Address = StringVar()
Reference = StringVar()
EmployerName = StringVar()
City = StringVar()
Basic = StringVar()
OverTime = StringVar()
GrossPay = StringVar()
NetPay = StringVar()
Tax = StringVar()
PostCode = StringVar()
Gender = StringVar()
PayDate = StringVar()
Pension = StringVar()
StudenLoan = StringVar()
NIPayment = StringVar()
Totalsalary = StringVar()
TaxPeriod = StringVar()
NINumber = StringVar()
NICode = StringVar()
TaxablePay = StringVar()
PensionablePay = StringVar()
OtherPaymentDue = StringVar()

textInput = StringVar()

Tops=Frame(payroll, width=1350, height=50, bd=12, relief="raise")
Tops.pack(side=TOP)

LF=Frame(payroll, width=700, height=650, bd=12, relief="raise")
LF.pack(side=LEFT)

RF=Frame(payroll, width=600, height=650, bd=12, relief="raise")

```

```
RF.pack(side=RIGHT)
```

```
#=====
```

```
lblTitle = Label(Tops, font=('arial', 50, 'bold'), text="Payroll Management Systems", fg="Steel  
blue", bd=10, anchor="w")  
lblTitle.grid(row=0, column=0)
```

```
#=====
```

```
InsideLF=Frame(LF, width=700, height=100, bd=8, relief="raise")  
InsideLF.pack(side=TOP)
```

```
InsideLFL=Frame(LF, width=325, height=400, bd=8, relief="raise")  
InsideLFL.pack(side=LEFT)
```

```
InsideLFR=Frame(LF, width=325, height=400, bd=8, relief="raise")  
InsideLFR.pack(side=RIGHT)
```

```
#=====
```

```
InsideRF=Frame(RF, width=600, height=200, bd=8, relief="raise")  
InsideRF.pack(side=TOP)
```

```
InsideRFL=Frame(RF, width=300, height=400, bd=8, relief="raise")  
InsideRFL.pack(side=LEFT)
```

```
InsideRFR=Frame(RF, width=300, height=400, bd=8, relief="raise")  
InsideRFR.pack(side=RIGHT)
```

```
#=====Left Side
```

```
lblEmployeeName = Label(InsideLF, font=('arial', 12, 'bold'), text="Empid", fg="Steel blue",  
bd=10, anchor="w")  
lblEmployeeName.grid(row=0, column=0)  
txtEmployeeName = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder  
blue", justify="left", textvariable = Empid)  
txtEmployeeName.grid(row=0, column=1)
```

```
lblEmployeeName = Label(InsideLF, font=('arial', 12, 'bold'), text="Employee Name", fg="Steel blue", bd=10, anchor="w")
lblEmployeeName.grid(row=1, column=0)
txtEmployeeName = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder blue", justify="left", textvariable = EmployeeName)
txtEmployeeName.grid(row=1, column=1)
```

```
lblAddress = Label(InsideLF, font=('arial', 12, 'bold'), text="Address", fg="Steel blue", bd=10, anchor="w")
lblAddress.grid(row=2, column=0)
txtAddress = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder blue", justify="left", textvariable = Address)
txtAddress.grid(row=2, column=1)
```

```
lblReference = Label(InsideLF, font=('arial', 12, 'bold'), text="Reference", fg="Steel blue", bd=10, anchor="w")
lblReference.grid(row=3, column=0)
txtReference = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder blue", justify="left", textvariable = Reference)
txtReference.grid(row=3, column=1)
```

#-----Left Left Side

```
lblCity = Label(InsideLFL, font=('arial', 12, 'bold'), text="City allowance", fg="Steel blue", bd=10, anchor="w")
lblCity.grid(row=0, column=0)
txtCity = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = City)
txtCity.grid(row=0, column=1)
```

```
lblBasic = Label(InsideLFL, font=('arial', 12, 'bold'), text="Basic Salary", fg="Steel blue", bd=10, anchor="w")
lblBasic.grid(row=1, column=0)
txtBasic = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = Basic)
txtBasic.grid(row=1, column=1)
```

```
lblOverTime = Label(InsideLFL, font=('arial', 12, 'bold'), text="Over Time", fg="Steel blue", bd=10, anchor="w")
```

```
lblOverTime.grid(row=2, column=0)
txtOverTime = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="right", textvariable = OverTime)
txtOverTime.grid(row=2, column=1)
```

```
lblGrossPay = Label(InsideLFL, font=('arial', 12, 'bold'), text="Gross Pay", fg="Steel blue",
bd=10, anchor="w")
lblGrossPay.grid(row=3, column=0)
lblGrossPay = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="right", textvariable = GrossPay)
lblGrossPay.grid(row=3, column=1)
```

```
lblNetPay = Label(InsideLFL, font=('arial', 12, 'bold'), text="Net Pay", fg="Steel blue", bd=10,
anchor="w")
lblNetPay.grid(row=4, column=0)
lblNetPay = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="right", textvariable = NetPay)
lblNetPay.grid(row=4, column=1)
```

#-----Left Right Side

```
lblTax = Label(InsideLFR, font=('arial', 12, 'bold'), text="Tax", fg="Steel blue", bd=10,
anchor="w")
lblTax.grid(row=0, column=0)
txtTax = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="right", textvariable = Tax)
txtTax.grid(row=0, column=1)
```

```
lblPension = Label(InsideLFR, font=('arial', 12, 'bold'), text="Pension", fg="Steel blue", bd=10,
anchor="w")
lblPension.grid(row=1, column=0)
txtPension = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="right", textvariable = Pension)
txtPension.grid(row=1, column=1)
```

```
lblStudenLoan = Label(InsideLFR, font=('arial', 12, 'bold'), text="Student Loan", fg="Steel blue",
bd=10, anchor="w")
lblStudenLoan.grid(row=2, column=0)
txtStudenLoan = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="right", textvariable = StudenLoan)
txtStudenLoan.grid(row=2, column=1)
```

```

lblNIPavment = Label(InsideLFR, font=('arial', 12, 'bold'), text="National insurance", fg="Steel blue", bd=10, anchor="w")
lblNIPavment.grid(row=3, column=0)
txtNIPavment = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = NIPayment)
txtNIPavment.grid(row=3, column=1)

```

```

lblDeducations = Label(InsideLFR, font=('arial', 12, 'bold'), text="Total Salary", fg="Steel blue", bd=10, anchor="w")
lblDeducations.grid(row=4, column=0)
txtDeducations = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = Totalsalary)
txtDeducations.grid(row=4, column=1)

```

#=====Right Side

```

lblPostCode = Label(InsideRF, font=('arial', 12, 'bold'), text="Post Code", fg="Steel blue", bd=10, anchor="w")
lblPostCode.grid(row=0, column=0)
txtPostCode = Entry(InsideRF, font=('arial', 12, 'bold'), bd=10, width=50, bg="powder blue", justify="right", textvariable = PostCode)
txtPostCode.grid(row=0, column=1)

```

```

lblGender = Label(InsideRF, font=('arial', 12, 'bold'), text="Gender", fg="Steel blue", bd=10, anchor="w")
lblGender.grid(row=1, column=0)
txtGender = Entry(InsideRF, font=('arial', 12, 'bold'), bd=10, width=50, bg="powder blue", justify="right", textvariable = Gender)
txtGender.grid(row=1, column=1)

```

#-----

```

lblPayDate = Label(InsideRFL, font=('arial', 12, 'bold'), text="Pay Date", fg="Steel blue", bd=10, anchor="w")
lblPayDate.grid(row=0, column=0)
txtPayDate = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = PayDate)
txtPayDate.grid(row=0, column=1)

```

```

lblTaxPeriod = Label(InsideRFL, font=('arial', 12, 'bold'), text="Tax Period", fg="Steel blue", bd=10, anchor="w")
lblTaxPeriod.grid(row=1, column=0)

```

```
txtTaxPeriod = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="left", textvariable = TaxPeriod)
txtTaxPeriod.grid(row=1, column=1)
```

```
lblNINumber = Label(InsideRFL, font=('arial', 12, 'bold'), text="NI Number", fg="Steel blue",
bd=10, anchor="w")
lblNINumber.grid(row=2, column=0)
txtNINumber = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="left", textvariable = NINumber)
txtNINumber.grid(row=2, column=1)
```

```
lblNICode = Label(InsideRFL, font=('arial', 12, 'bold'), text="NI Code", fg="Steel blue", bd=10,
anchor="w")
lblNICode.grid(row=3, column=0)
txtNICode = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="left", textvariable = NICode)
txtNICode.grid(row=3, column=1)
```

```
lblTaxablePay = Label(InsideRFL, font=('arial', 12, 'bold'), text="Taxable Pay ", fg="Steel blue",
bd=10, anchor="w")
lblTaxablePay .grid(row=4, column=0)
txtTaxablePay = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="left", textvariable = TaxablePay)
txtTaxablePay .grid(row=4, column=1)
```

```
lblPensionablePay = Label(InsideRFL, font=('arial', 12, 'bold'), text="Pensionable Pay", fg="Steel
blue", bd=10, anchor="w")
lblPensionablePay.grid(row=5, column=0)
txtPensionablePay = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder
blue", justify="left", textvariable = PensionablePay)
txtPensionablePay.grid(row=5, column=1)
```

```
lblOtherPaymentDue = Label(InsideRFL, font=('arial', 12, 'bold'), text="Other Payment Due",
fg="Steel blue", bd=10, anchor="w")
lblOtherPaymentDue.grid(row=6, column=0)
txtOtherPaymentDue = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder
blue", justify="left", textvariable = OtherPaymentDue)
txtOtherPaymentDue.grid(row=6, column=1)
```

```
#####
```

```

def show():
    Label(payload, text='Employee data base', font=headlabelfont, bg='DarkGreen',
fg='LightCyan').pack(side=TOP, fill=X)
    tree = ttk.Treeview(payload, height=100,
selectmode=BROWSE,columns=("Empid", "EmployeeName", "Reference", "NetPay",
"Totalsalary", "NINumber", "NICode"))
    X_scroller = Scrollbar(tree, orient=HORIZONTAL, command=tree.xview)
    Y_scroller = Scrollbar(tree, orient=VERTICAL, command=tree.yview)
    X_scroller.pack(side=BOTTOM, fill=X)
    Y_scroller.pack(side=RIGHT, fill=Y)
    tree.config(yscrollcommand=Y_scroller.set, xscrollcommand=X_scroller.set)
    tree.heading('Empid', text='Empid', anchor=CENTER)
    tree.heading('EmployeeName', text='EmployeeName', anchor=CENTER)
    tree.heading('Reference', text='Reference', anchor=CENTER)
    tree.heading('NetPay', text='NetPay', anchor=CENTER)
    tree.heading('Totalsalary', text='Totalsalary', anchor=CENTER)
    tree.heading('NINumber', text='NINumber', anchor=CENTER)
    tree.heading('NICode', text='NICode', anchor=CENTER)

    tree.column('#0', width=0, stretch=NO)
    tree.column('#1', width=60, stretch=NO)
    tree.column('#2', width=100, stretch=NO)
    tree.column('#3', width=150, stretch=NO)
    tree.column('#4', width=80, stretch=NO)
    tree.column('#5', width=70, stretch=NO)
    tree.column('#6', width=60, stretch=NO)
    tree.column('#7', width=70, stretch=NO)
    tree.place(y=30, relwidth=1, relheight=0.9, relx=0)

    curr = connector.execute('SELECT * FROM payroll')
    data = curr.fetchall()
    for records in data:
        tree.insert("", END, values=records)

#-----
btnWagePayment = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'),
width=14,

```



```

        text="Wage Paymant", bg="sky blue", command=MonthlySalary).grid(row=0,
column=0)

btnReset = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,
        text="Reset System", bg="sky blue", command=reset).grid(row=1, column=0)

btnPayRef = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,
        text="Pay Reference", bg="sky blue", command=PayRef).grid(row=2, column=0)

btnPayCode = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,
        text="Pay Code", bg="sky blue", command=PayPeriod).grid(row=3, column=0)

btnExit = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,
        text="Exit", bg="sky blue", command=exit).grid(row=4, column=0)
btnExit = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,
        text="save", bg="sky blue", command=add_record).grid(row=5, column=0)
btnExit = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,
        text="Show Database", bg="sky blue", command=show).grid(row=6, column=0)

payroll.mainloop()

```

5.5 EXAMPLES OF THE OUTPUTS

After completion of the source code we can run the source code , then we can get output is :

The screenshot shows the 'Payroll Management Systems' application window. The title bar reads 'Payroll Management Systems'. The main window has a title 'Payroll Management Systems' in a large blue font. Below the title, there are two main sections. The left section contains input fields for 'Empid', 'Employee Name', 'Address', and 'Reference'. Below these are two columns of fields: 'City allowance', 'Basic Salary', 'Over Time', 'Gross Pay', and 'Net Pay' on the left; and 'Tax', 'Pension', 'StudenLoan', 'National insurance', and 'Total saary' on the right. The right section contains input fields for 'Post Code' and 'Gender'. Below these are two columns of fields: 'Pay Date', 'Tax Period', 'NI Number', 'NI Code', 'Taxable Pay', 'Pensionable Pay', and 'Other Payment Due' on the left; and a vertical stack of buttons on the right: 'Wage Payment', 'Reset System', 'Pay Reference', 'Pay Code', 'Exit', 'save', and 'Show Database'.

After we can enter the employee details like Empid, Employee Name, Address, City allowance, Basic Salary, Over Time.

The screenshot shows the 'Payroll Management Systems' application window with the same layout as the previous one, but with some fields filled in. The 'Empid' field contains '12345', 'Employee Name' contains 'HENRY', 'Address' contains 'KHAMMAM', and 'City allowance' contains '1000'. The 'Basic Salary' field contains '20000', 'Over Time' contains '2000', and 'Gross Pay' is empty. The 'Net Pay' field is empty. The 'Tax' field is empty, 'Pension' is empty, 'StudenLoan' is empty, 'National insurance' is empty, and 'Total saary' is empty. The 'Post Code' field is empty, and the 'Gender' field is empty. The buttons on the right are 'Wage Payment', 'Reset System', 'Pay Reference', 'Pay Code', 'Exit', 'save', and 'Show Database'.

When we press **Wage Payment** button then we can get Gross Pay, Net Pay, Tax, Pension, StudentLoan, National insurance, Totalsalary, Taxable Pay, Pensionable Pay, Other Payment Due

Payroll Management Systems

Empid	12345
Employee Name	HENRY
Address	KHAMMAM
Reference	

City allowance	1000	Tax	2300
Basic Salary	20000	Pension	2300
Over Time	2000	StudenLoan	2300
Gross Pay	23000	National insurance	2300
Net Pay	13800	Total saary	9200

Post Code	
Gender	

Pay Date	
Tax Period	
NI Number	
NI Code	
Taxable Pay	2300
Pensionable Pay	2300
Other Payment Due	0.00

Wage Payment

Reset System

Pay Reference

Pay Code

Exit

save

Show Database

After we can press on **Pay Reference** then we can get Reference, Pay Date, NI Number

Payroll Management Systems

Empid	12345
Employee Name	HENRY
Address	KHAMMAM
Reference	440847

City allowance	1000	Tax	2300
Basic Salary	20000	Pension	2300
Over Time	2000	StudenLoan	2300
Gross Pay	23000	National insurance	2300
Net Pay	13800	Total saary	9200

Post Code	
Gender	

Pay Date	10/01/2023
Tax Period	
NI Number	352823
NI Code	
Taxable Pay	2300
Pensionable Pay	2300
Other Payment Due	0.00

Wage Payment

Reset System

Pay Reference

Pay Code

Exit

save

Show Database

When we can press **Pay Code** then we can get Tax period, NI Code

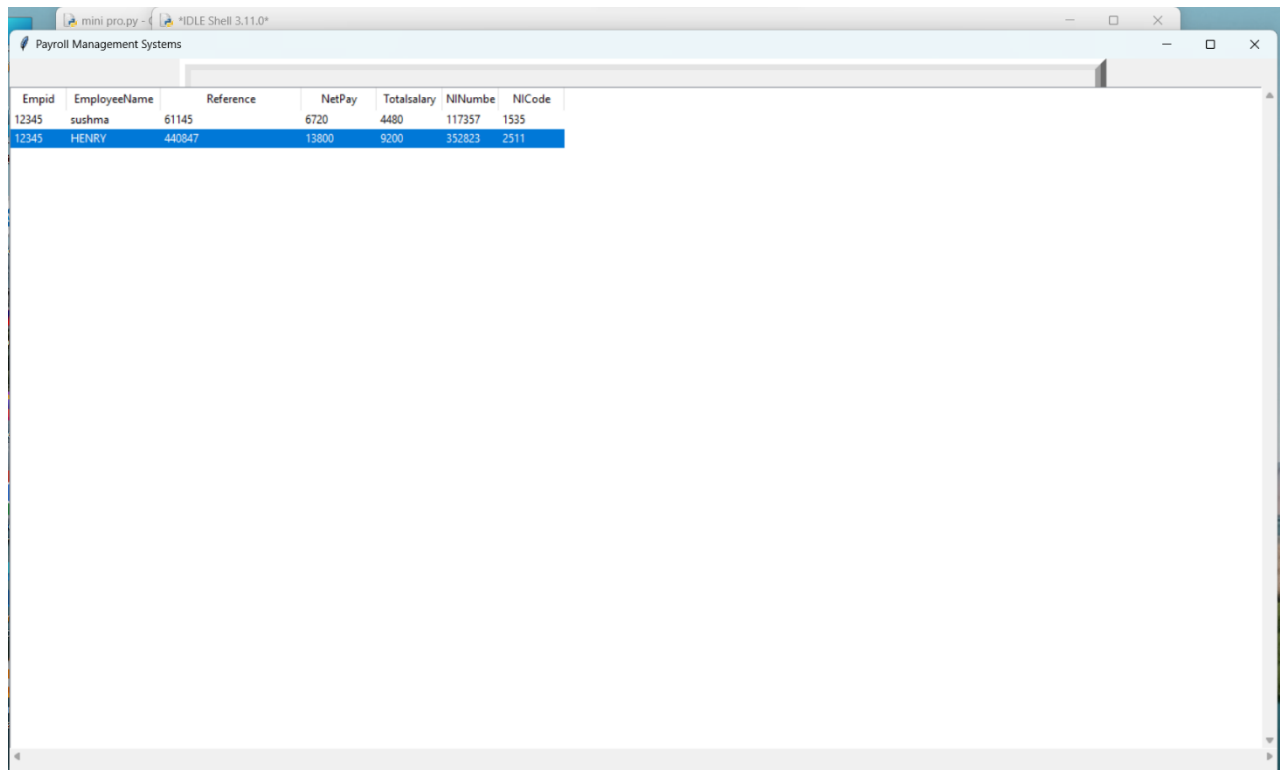
The screenshot shows the 'Payroll Management Systems' application window. The title bar reads 'Payroll Management Systems'. The main title 'Payroll Management Systems' is displayed in a large blue font. The interface is divided into several sections:

- Employee Information:** Fields for Empid (12345), Employee Name (HENRY), Address (KHAMMAM), and Reference (440847).
- Financial Data:** A table with two columns. The left column lists items: City allowance (1000), Basic Salary (20000), Over Time (2000), Gross Pay (23000), and Net Pay (13800). The right column lists items: Tax (2300), Pension (2300), StudenLoan (2300), National insurance (2300), and Total saary (9200).
- Payment Details:** Fields for Post Code, Gender, Pay Date (10/01/2023), Tax Period (1), NI Number (352823), NI Code (2511), Taxable Pay (2300), Pensionable Pay (2300), and Other Payment Due (0.00).
- Action Buttons:** A vertical stack of buttons on the right: Wage Payment, Reset System, Pay Reference, Pay Code, Exit, save, and Show Database.

After that we can press on **save** button the we can get like this

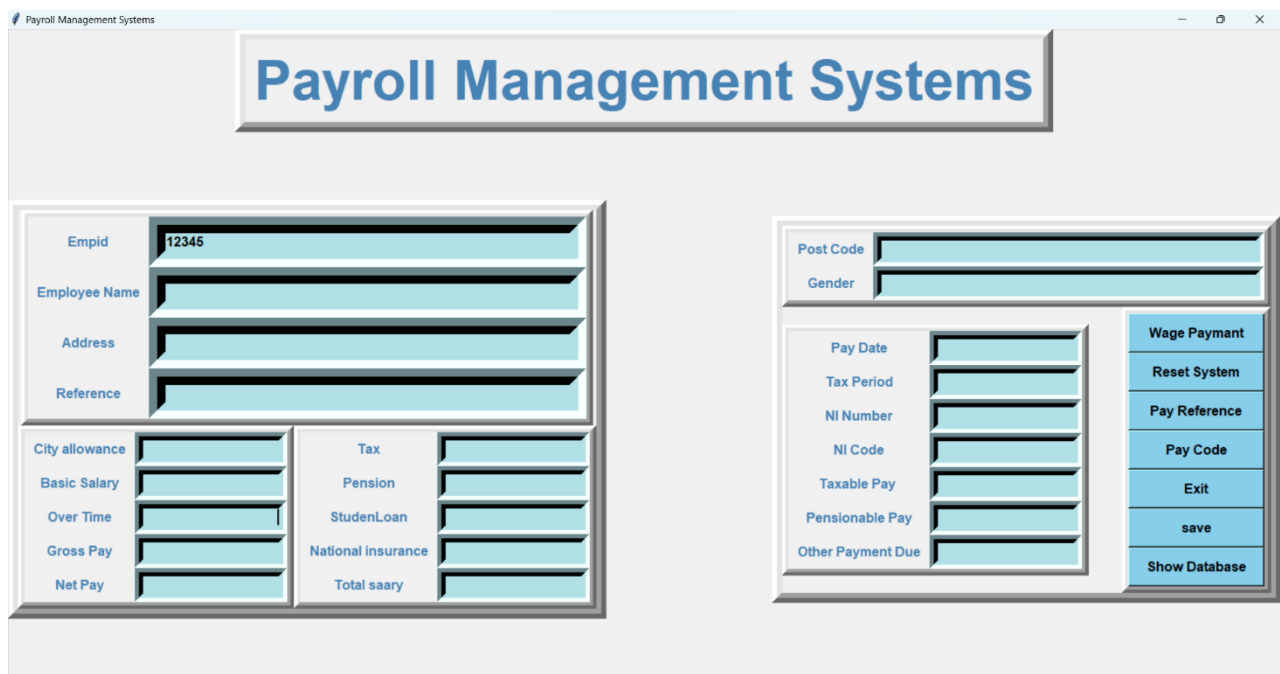
This screenshot shows the same 'Payroll Management Systems' application window as the previous one, but with a confirmation dialog box displayed in the center. The dialog box has a title bar 'Record added' and contains the message 'Record of HENRY was successfully added' with an 'OK' button. The background interface elements remain the same, including the employee information, financial data table, payment details, and action buttons.

Here we can press **Show Database** button then we get :



Empid	EmployeeName	Reference	NetPay	Totalsalary	NI Number	NI Code
12345	sushma	61145	6720	4480	117357	1535
12345	HENRY	440847	13800	9200	352823	2511

If case we can press the **Reset System** button then all details of employee can be earised and we can press the **Exit** then we can leave from the output.



Payroll Management Systems

Employee Details:

Empid:

Employee Name:

Address:

Reference:

Post Code:

Gender:

City allowance:

Basic Salary:

Over Time:

Gross Pay:

Net Pay:

Tax:

Pension:

StudenLoan:

National insurance:

Total saary:

Pay Date:

Tax Period:

NI Number:

NI Code:

Taxable Pay:

Pensionable Pay:

Other Payment Due:

Wage Payment:

Reset System:

Pay Reference:

Pay Code:

Exit:

save:

Show Database:

CHAPTER – 6

TESTING

Python is a powerful language that allows you to create complex applications and websites. It also provides a robust system for testing your code. In this article, we will learn about the different types of testing in python that you can use to easily identify and fix bugs in your code.

TYPES OF TESTING:

There are 4 types of testing available in Python –

1. Unit Testing
2. Feature Testing
3. Integration Testing
4. Performance Testing

1.UNIT TESTING:

In this case, we basically test only a logical unit of our code. It is used to test if the internal flow of methods & data is correct, and that edge cases are handled properly. This is the most granular form of testing in Python.

2.FEATURE TESTING:

In this case, we test the actual functionality of features. You can create a collection of unit tests for this purpose, or a single feature test also.

3.INTEGRATION TESTING:

Integration tests are used to test applications end to end. Even if new code is added to your application, the existing integration tests should work properly.

4.PERFORMANCE TESTING:

In this case, we are only checking the performance of a piece of code. Before we run performance tests, we should have ideally performed Unit & Feature Testing to ensure that it is working properly. Performance tests are basically calling the same function repeatedly over a given period of time to ensure that it does not crash the application. For this purpose, you can use Python libraries like time it.

CHAPTER - 7

CONCLUSION

“ Payroll Management System” software developed for a company has been designed to achieve maximum efficiency and reduce the time taken to handle the Payroll activity. It is designed to replace an existing manual record system thereby reducing time taken for calculations and for storing data. The system uses Asp Net as front end and Microsoft SQL as a backend for the database.

The system is strong enough to withstand regressive daily operations under conditions where the database is maintained and cleared over a certain time of span. The implementation of the system in the organization will considerably reduce data entry, time and also provide readily calculated reports.

CHAPTER – 8

FUTURE SCOPE

- Does not require paperwork.
- Only Human error while - entering the information.
- Not required much space.
- Automatically search and sort the information.
- Require less physical work and manpower.
- Editing is a lot easier.

CHAPTER – 9

REFERENCES

[1]. <http://www.ijetae.com/files/Volume4Issue4/IJE>

TAE 0414_140.

[2]. Schmitt, Bob (10 October 2014). & quot; Collection Management Systems",.Retrieved 15 November 2015.

[3]. Carpinone, Elana C. (May 2010), Museum Collection Management systems: One Size Does NOT Fit All. p. 26.Retrieved 5 December 2015.

[4]. [http://www.tangedco.gov.in/linkpdf/Agri For m.pdf](http://www.tangedco.gov.in/linkpdf/Agri%20Form.pdf): The pdf is a form to apply for agricultural loan. This gives us the idea about attributes required in database "Agricultural loans".

[5]. [http://ceoharyana.nic.in/?module=pages&pageid -42](http://ceoharyana.nic.in/?module=pages&pageid=42): The link tells us about the requirements for Voter ID Generation. This information can be used to maintain the database.

[6]. A Java API for the description of large complex networks under the object-oriented paradigm A. Ni-nol:y, C. Mu-noz-Carol, S. Reyes1 and M. Castillol1 SciCom Research Group. Escuela Superior de Inform_atica de Ciudad Real.Universidad de Castilla-La Mancha. Pasco de la Universidad 4. 13004 Ciudad Real, Spain.

[7]. Compiling and Optimizing Java 8 Programs for GPU Execution Kazuaki Ishizaki IBM Research- TokyoAkihiro Hayashi Rice University,Gita Koblents IBM Canada, Vivek Sarkar Rice University.