

EMPLOYEE MANAGEMENT SYSTEM

A

MINOR PROJECT REPORT

*Submitted in partial fulfillment of the requirements
for the award of the degree of*

MASTER OF COMPUTER APPLICATION

In

COMPUTER APPLICATION

Submitted to



RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA,

BHOPAL (M.P.)

Submitted by

KESHAV

0115CA231053

Under the Guidance of

PROF. DEV NAGAR



DEPARTMENT OF MASTER OF COMPUTER APPICATION

NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY BHOPAL

DECEMBER-2024

NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY BHOPAL

DEPARTMENT OF MASTER OF COMPUTER APPLICATION

DECLARATION

I hereby declare that the Project entitled **“EMPLOYEE MANAGEMENT SYSTEM”** is our own work conducted under the supervision of **MR. DEV NAGAR, Asst. Prof. Department of Master of Computer Application at NRI Institute of Information Science & Technology, Bhopal.**

We further declare that to the best of our knowledge this report does not contain any part of work that has been submitted for the award of any degree either in this institute or in other institute without proper citation.

KESHAV

0115CA231053

NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY BHOPAL

DEPARTMENT OF MASTER OF COMPUTER APPLICATION

CERTIFICATE

This is to certify that the work embodied in this project entitled “**EMPLOYEE MANAGEMENT SYSTEM**” being submitted by **KESHAV (0115CA231053)** award of the degree of the **Master of Computer Application (MCA)** to **Rajiv Gandhi Proudhyogiki Vishwavidyalaya, Bhopal (M.P.)** is a record of Bonafide piece of work, carried out by them under our supervision and guidance in the **Department of Master of Computer Application, NRI Institute of Information Science and Technology, Bhopal (M.P.)**.

Guided By

PROF. DEV NAGAR

Assistant Professor

Master of Computer Application

NIIST, Bhopal

Approved By

PROF. MANISH SINGHAL

H O D MCA

Master Of Computer Application

NIIST, Bhopal

Director/Principal

NIIST, Bhopal

NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY BHOPAL

DEPARTMENT OF MASTER OF COMPUTER APPLICATION

ACKNOWLEDGEMENT

I would like to express our special thanks of gratitude to our teacher **Prof. Dev Nagar** who gave us the golden opportunity to do this wonderful project on the topic “**online employee employee system**”, which also helped us in doing a lot of Research and we came to know about so many new things we are really thankful to his.

Secondly, we would also like to thank our parents and also to all those people who directly or indirectly support us during our project.

Thanks again to all who helped us.

KESHAV
0115CA231053

NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY BHOPAL

DEPARTMENT OF MASTER OF COMPUTER APPLICATION

ABSTRACT

This report includes a development presentation of an information system for managing the staff data within a small company or organization. The system as such as it has been developed is called Employee Management System. It consists of functionally related GUI (application program) and database. The choice of the programming tools is individual and particular.

Programming languages, paradigms and practices don't stand still very long. It often seems that the methods and techniques we applied yesterday are out of date today of course this rapid rate of change is also one of the things that keep programming existing. There is always something new on the horizon. One characteristic that is constant in software industry today is the "change". Change is one of the most critical aspects of s/w development and management. New tools and new approaches are announced almost every day. The impact of these developments is often very extensive. Most important among them is maintaining ability, reusability, portability, security, and integrity and user friendliness.

To build today's complex software, we need to wound construction techniques and program structures that are easy to comprehend, implement and modify in wide variety of situations.

TABLE OF CONTENTS

(Note: Table of contents must include the following with page numbers.)

Page No.	
Declaration	2
Certificate	3
Acknowledgement	4
Abstract	5

LIST OF TABLE

S.No.	Contents	Page No.
CHAPTER 1	Introduction	1
	Objective	3
	Justification of study	3
	Scope of Study	3
	Literature Review	4
	Employee Management System	5
	Methodology	8
	Description of Methodology	9
CHAPTER 2	Analysis, Design and Development	10
	Introduction	10
	System Study	10
	System Analysis	10
	Requirements Specifications	11
	User Requirement	11
	Functional and Non Functional Requirements	11
	System Requirement	12
	Hardware Requirements	12
	System Design	13
	Logical Model	13
	System Architecture	15
	Entity Relationship (E-R) Diagram	16
	Attributes	16
	Process	16
	Actions	16
CHAPTER 3	Coding	18
	Index.html code	24
	Login page	29

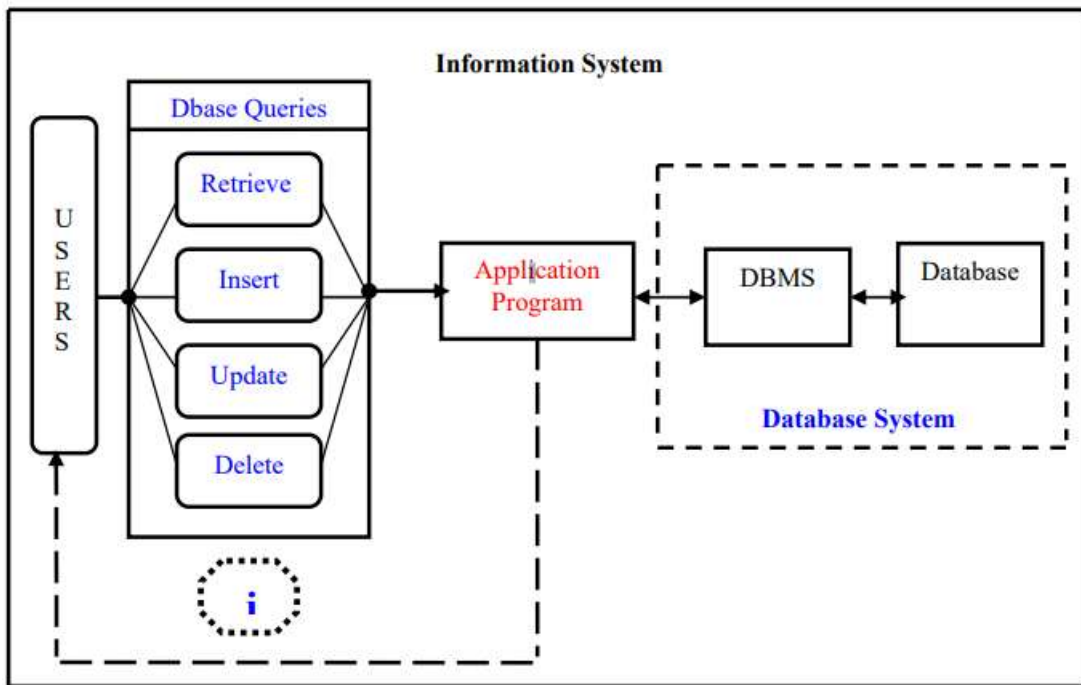
	Employee.html code	33
	User html code	40
	Screenshot image	47
CHAPTER 4	Market Potential & Competitive Advantages	52
	Likely Benefits	52
	Future scope	52

CHAPTER 1

INTRODUCTION

This chapter gives a brief theoretical preview upon the database information systems and goes through the essence of the problem that should be resolved.

Background Most of the contemporary Information systems are based on the Database technology as a collection of logically related data, and DBMS as a software system allowing the users to define, create, maintain and control access to the database. The process of constructing such kind of systems is not so simple. It involves a mutual development of application program and database. The application program is actually the bridge between the users and the database, where the data is stored. Thus, the well-developed application program and database are very important for the reliability, flexibility and functionality of the system. The so defined systems differentiate to each other and their development comprises a great variety of tasks to be resolved and implemented.



Information system suggests a computer technology to be used in order to provide information to users in an organization (for instance), as for the purposes of data transformation into useful information; computer hardware and software are designed and used [2]. A particular case is the Human Resources Information System development. This kind of systems are responsible for storing data of the staff within an organization and generating reports upon request.

Such kind of system could be integrated with other Information systems or modules: Accounting Information System (AIS) – designed to transform financial data into information, or Management Information System (MIS) that provides decision-oriented information to managers, and so on... “Organizations depend on Information Systems in order to stay competitive. Productivity, which is crucial to staying competitive, can be increased through better Information Systems.” .

Problem Statement

This report’s documentation goes through the whole process of both application program and database development. It also comprises the development tools have been utilized for these purposes.

Problem Discussion

This system should consist of an application program, on one hand, and a database (repository of data) on the other. The program should perform the basic operations upon the database as retrieving, inserting, updating and deleting data. Any additional functionality is a goal of a further module development. It is a kind of strategy to start the development from designing and constructing the database, as this structure will determine the further structure of the application program. The logical database model (tables, their content and the relationships between them) should respond to the given task and cover the basic requirements. The Interface of the program should be user-friendly, and the program should be as easy for use as it is possible. Both controls and forms should logically and functionally be related within the program and fully respond to the structure of the database. Another problem is establishing the connections with the database, every time, when a query is needed to be performed upon it. Exception-handling should also be taken into an account during the system’s development due to eventual exceptions that may occur employee that contains a login screen. The employee has to register himself before logging in to the system. After logging in, the employee can select a employee and can view the employee details. The employee has the option of selecting a employee from the list of employees and can view the employee’s details. The employee can request for an employee on his/her preferred day/time. The employee can view the location of the employee on map. In addition, the employee can contact to the employee and the employee by making a call or may send an email to the employee.

Objective

- 1 .To Create Web based online Employee Management system.
- 2 .Track and find out the availability of employee and manage all employee related information.
- 3 .Establishment of paperless environment.

Justification of study

We implement this system for better user experience. This system is very easy to access. Also for establish real time communication, using modern and updated technology. So, user can see the update without reload or refresh. This system will compatible with user device such as pc, laptop, tab & smart phone. So user can easily access the system anytime anywhere. This system is very simple & user friendly so, any user can use this system easily.

Scope of Study

Scope of the project is very broad in terms of other online employee portal. Few of them are:

1. Optimizes utilization of medical resources at the medical center
2. There is huge collection of employee information among their records.

LITERATURE REVIEW

Report Overview

The next chapter and its subsections will turn the attention to the method for resolving the problem, the programming environments used for developing the system and the implementation of the operations performed upon the database.

Waiting time simply means a period of time which one must wait in order for a specific action to occur, after that action is requested or mandated (Fernandes et al., 1994). Employees' waiting time has been defined as "the length of time from when the employee entered the outpatient clinic to the time the employee actually received his or her prescription" (Jamaiah, 2003). It is defined as the total time from registration until consultation with a employee. There were two waiting times, the first is time taken to see a physician and the second is time to obtain medicine (Suriani, 2003). This paper deals with the waiting time to see physicians. Long waiting times are a serious problem for employees using urban health centers in developing countries (Bachmann, 1998). A block employee system was introduced and evaluated in a large South African health center. Waiting times of all employees were measured over one-week period before and after the implementation of employees. Focus groups and individual interviews were conducted with staff and employees. After introducing employees, employees with acute and chronic illnesses and having employees had significantly shorter waits time than similar employees without employees (Mahomed, 1998). Employees had no benefits for employees not seeing employees or collecting repeat medication. There was, however, an overall increase in employees' waiting times after introducing the system, mainly due to one typical day in the follow-up study. Focus groups and interviews revealed that staff were skeptical at baseline but at follow-up were positive about the system. Employees were enthusiastic about the employee system at all stages. The study shows that block employees can reduce employees' waiting times for acute employees, but may not be suitable for all employees. Staff and employees had different views, which converged with experience of the new system (Mahomed, 1998).

Problem's Solution

This chapter involves some subsections that concern the basic scheme of resolving the given task and comprise both the methods and tools of its development as well.

Employee system

According to Dexter (1999), managing employee system is a computer application used to manage and reduce the employee waiting time in the health care center. Some worker care centers do not use any employee system. So it has a longer average employees' waiting time than the health care center that adopts the employee employee system. While employees can wait for more than one hour to be attended to by a physician in a health care center, they also can feel that they are being disregarded and treated unfairly. So when employees are given the time of employee in a health care centre, they can evaluate the quality of service in the centre (Dexter, 1999). Hence, developing employees' employee process for health care centers necessitates the use of a sophisticated queuing model that captures much of the real system's features (saving time, reducing idle time, etc). Therefore the employee schedule represents the real situation in the health care centre faced by employee employee schedulers (Rohleder, 2002). On the other hand, the standard practice for scheduling and processing employee employees are based on the nature of treatments of the employees and that better approaches more sensitive to employee needs are desirable (Klassen, 2002).

Online Booking System

An online system is also known as a web based system. A web is made up of page that is commonly known as web page or web site, and a web site is a computer program that runs a web server that provides access to a group of related web pages (Alex, 2000). A system is a set of independent components working together to achieve a common objective. Therefore a web based system is a system that is accessible over the internet by a user in order to achieve a particular task for a given purpose. The Internet is a system that is use to connect computers and computer networks. It helps to link millions of computer networks all over the world and it allows the users to get information stored on other computers from a long distance (James, 1999). According to Chua (2010) the public demand for better healthcare system and the alarming number of missed employees have forced the healthcare sector to recognize how they deliver care services. With the advance of IT technology today and seen healthcare system as a critical system, employee booking system lies at the intersection of delivering efficient, dependable and timely access to health services. The conventional way of employee booking is via fax, phone or email. But with the growing internet penetration, healthcare industry is moving towards the use of an online employee booking system.

web-based employee system is used in Taiwan; everyone is required to enroll in the national health insurance program. When one needs health service, he shows his health insurance card to employees in an employee to start with. There are several ways of making an employee. A person can either go to the employee directly for consultation day by day or make an employee from home through phone call or email if his condition is not emergent (Gruca, 2004). The Internet provides a wide range of technologies that enable employees to communicate with their employees. Recently, as the prevalence of Internet increasing, many employees initiated the website employee system. Electronic employee-provider communication promises to improve efficiency and it is limited due to the nature of the employee scheduling (Rohleder, 2002). Since employees are scheduled in the future, the exact model of call arrivals will only have limited impact on measures related to the time between the call and the employee time. For this reason, the challenge for making employee system is designing a suitable system based on the health care procedure environment (Klassen, 2002). Hence, the employee provider in the health care center can schedule a employee into an appropriate time slot on a given day. Klassen (2004) developed another method for managing employees' employee using multiple schedule employee in multiple period environments. Employees can call for any employee time but if the period time is full, they should replace the employee to another time. Moreover, various combinations for multi employee and double booking are measured and recommended for different operational use depending on the health care environment because the varying employee request has little effect on employee system performance, especially maintaining acceptable performance, except when the system has the overloaded option (Rohleder, 2004). Many studies about employees' employee have found that there are rules or policies for scheduling employee system such as no scheduling for more than 20 or 30 clients and the best schedule is to place two employees in the first employee and spread the rest consistently over a period based on average service times (Klassen, 2004). On the other hand, a employee can call for an employee without knowledge of the type of employee and employee queue number and the employee is not aware whether the employee is variable or not. Sometimes the exact duration for each employee can be known but at other times this is unknown (Rohleder, 2004). Another system developed by Mustafa, (2004) allows a registered employee, having user name and password, to access and explore the list of physicians alphabetically and select a physician whose email contact and profile are also provided. A employee can also view the physician working calendar to find out his/her working and non working day to make an employee. When the employee selects, view calendar the employee can then choose any valid day in any month to make an employee (Mustafa, 2004).

After that, the employee will receive an e-mail from the system to confirm the employee time or to inform the employee that the selected time is already taken by another employee or blocked by the physician. In general, the employee employee system provides all the choices and the capabilities to the employees, such as selecting a physician, selecting the time of employee, and allows them to

access the health care system day or night and schedule their own employees using the Internet without spending time holding for a nurse or having lengthy phone calls. Wijewickrama and Takakuwa (2005) opine that the health care operating time (due time) is from 8:30 am to 5:30 pm during the week days. Throughout this period, four types of employees arrive to have a consultation employee in the health care center-appointed employees, same day employee employees (walk-ins), employees who come for a medical test and new employees (Wijewickrama, 2005). Employees who have employees are given priority over those who walkin for consultation. Consequently, these latter employees have to wait a long time in the waiting room to meet a employee even if the consultation time only last few minutes (Takakuwa, 2005). Port-Sales et al. (2005) developed another system. The main concept of the system is contacting, screening and scheduling employee with the health care center initially by an expert nurse and the employee initiating contacting with the health care center using the telephone.

METHODOLOGY

3.0 Methodology

1. Waterfall Model

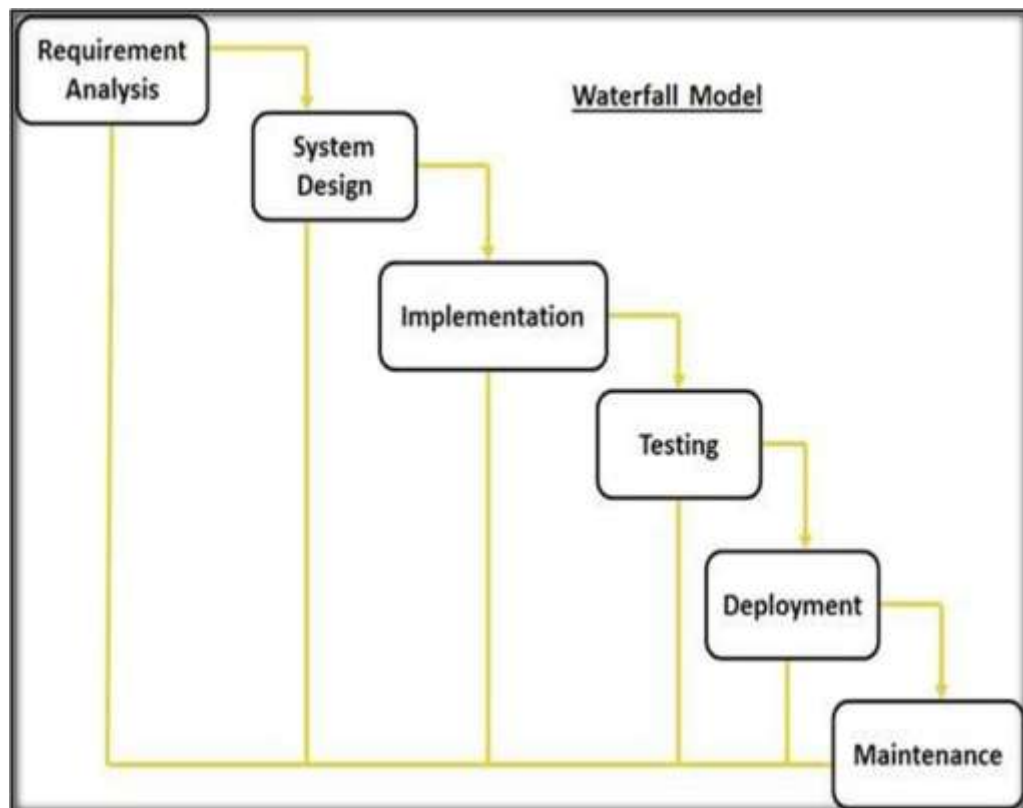


Fig. 3.0 Methodology

3.2 Justification of Methodology

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- 1) Requirements are very well documented, clear and fixed.
- 2) Product definition is stable.
- 3) Technology is understood and is not dynamic.
- 4) The project is short.
- 5) Simple and easy to understand and use
- 6) Easy to manage due to the rigidity of the model, each phase has specific deliverables and a review process.
- 7) Phases are processed and completed one at a time.
- 8) Easy to arrange tasks.

1.3 Description of Methodology

The sequential phases in Waterfall model are:

- 1) Requirement Gathering and analysis: All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- 2) System Design: The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- 3) Implementation: With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- 4) Integration and Testing: All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

CHAPTER 2

ANALYSIS, DESIGN AND DEVELOPMENT

Introduction

This chapter gives a brief theoretical preview upon the database information systems and goes through the essence of the problem that should be resolved.

System Study

The study was carried out at Employee, Employees and Employee the main purposed of the study was to find out how the process of recording employee's data is carried out. The system that is currently being used in Employee, Employee and Employee is entirely manuals. When a employee requests all the information is recorded manually from the employee then the system are very lazy and more hesitation from the real information, employee availability and proper time maintenance of the employee employee system.

System Analysis

During the system study phase, requirements of Online Employee System (ODAS) were categorized into user requirements, system and hardware requirements information was hard. Per the statistics carried 90% of the users were not contented with the system reason that is was not secure in terms of security and storage as it was prone to damages like loss of important information, worn out papers. The speed of recording and retrieval Employees information was average yet 10% were some ok with the system reason that the paper work can used for future reference. The users recommended that the proposed system should be user friendly, multipurpose enough to handle a number of users at a go, could generate feedback when request is submitted and a use of passwords which could deny access to unauthorized users of system which ensured security.

Context diagrams, Data flow diagrams and Entity Relationship Diagram (ERD) where used in the analysis and design of the system.

Requirements Specifications

After analyzing the data collected, we formulated a number of requirements namely user requirement, system hardware software attribute. These were grouped as user, functional, non- functional and systems requirements.

User Requirement

During data collection, we investigated and found out how the current system operates, not only that but also tried out which problems are faced and how best they can be settled. The users described some of the basic requirements of the system this includes Search for Employees, Register Employee, Update record, Employee information record, view employee availability record and view all types of reports.

Functional and Non Functional Requirements

The following is the desired functionality of the new system.

Accept of submissions in form of raw employees; perform analysis of financial to authenticate the users of the system.

And non functional requirement include the following

The system must verify the validate all user input ant user must be notified in case of errors detected in the database, the system should allow room for expansion.

System Requirement

This section describes the hardware components and software requirements needed for effective and efficient running of the system

Hardware Requirements

SL	Hardware	Minimum System Requirement
01	Processor	2.4 GHz Processor speed
02	Memory	2 GB RA
03	Disk Space	500 GB
SL	Software	Minimum System Requirement
01	Operating System	Windows Server 2008, Windows 7, 10, 11
02	Database Management System	MySQL 8.0.0G
03	Runtime Environment	STS-4 IDE
04	Server	Tomcat-09

The table above shows software requirements recommended to enable the system to run as required for using Online Employee Employee System (ODAS).

System Design

After interpretation of the data, tables were drawn and process of data determined to guide the researcher of the implementation stage of the project. The tools, which were employed during this methodology stage, were mainly tables, Data Flow Diagrams and Entity Relationship Diagrams. The design ensures that only allows authorized users to access the systems information.

Database Analyzing, design and implementation

The database for the system should include information of company's staff, respectively of its employees. The data is subdivided into the following groups:

Database Model

Employees' Basic Details	Working History	Time_Information
Employee_ID_Number Personal_ID_Number First_Name Middle_Name Last_Name Day_of_Birth Month_of_Birth Year_of_Birth Cellular_Phone Home_Phone City Address Postal_Code Qualification Current_Experience Start_Date_Day Start_Date_Month Start_Date_Year End_Date_Day End_Date_Month End_Date_Year Type_of_Employee Gender Marital_Status	Employee_ID_Number Company_Name Employer_Name Company_Employer_Address Company_Employer_Cellular_Phone Company_Employer_Office_Phone Previous_Qualification Previous_Experience p_Start_Date_Day p_Start_Date_Month p_Start_Date_Year p_End_Date_Day p_End_Date_Month p_End_Date_Year	Employee_ID_Number Wroked_Hours Off_Hours Days_off Over_Time Extra_Days w_From_Date_Day w_From_Date_Month w_From_Date_Year w_To_Date_Day w_To_Date_Month w_To_Date_Year

I have constructed a database that consists of six data tables. There will be one main table (parent table) and five child tables, related to each other. Patently, for this purpose the necessary primary and foreign keys should be defined into the responding tables. The so defined structure above is made up in conformity with the user's needs and demands. Each employee of the staff is intended to have several records, responding to his Working History, Contact Person Information, Salary Information, Time Information and Holiday Information, and only one record containing his basic information within the company – his personal details as: date of birth, gender, marital status, address and phone details, and his current working record. An employee is supposed to have not only one record of his Working history, or his Contact Person Information.....For instance, if we take a look to the Time Information data table – an employee may have several records in case he has some experience within the current company. It is absolutely the same with the Salary Information, Contact Person Information and Holiday Information data tables.

The relationships between the data tables are shown in Figure 4-Appendix A. In Figure 4 we can distinguish six tables that the database consists of. All of the relationships are of type: “one-to-many”. (For more details about the data tables, see Appendix A:

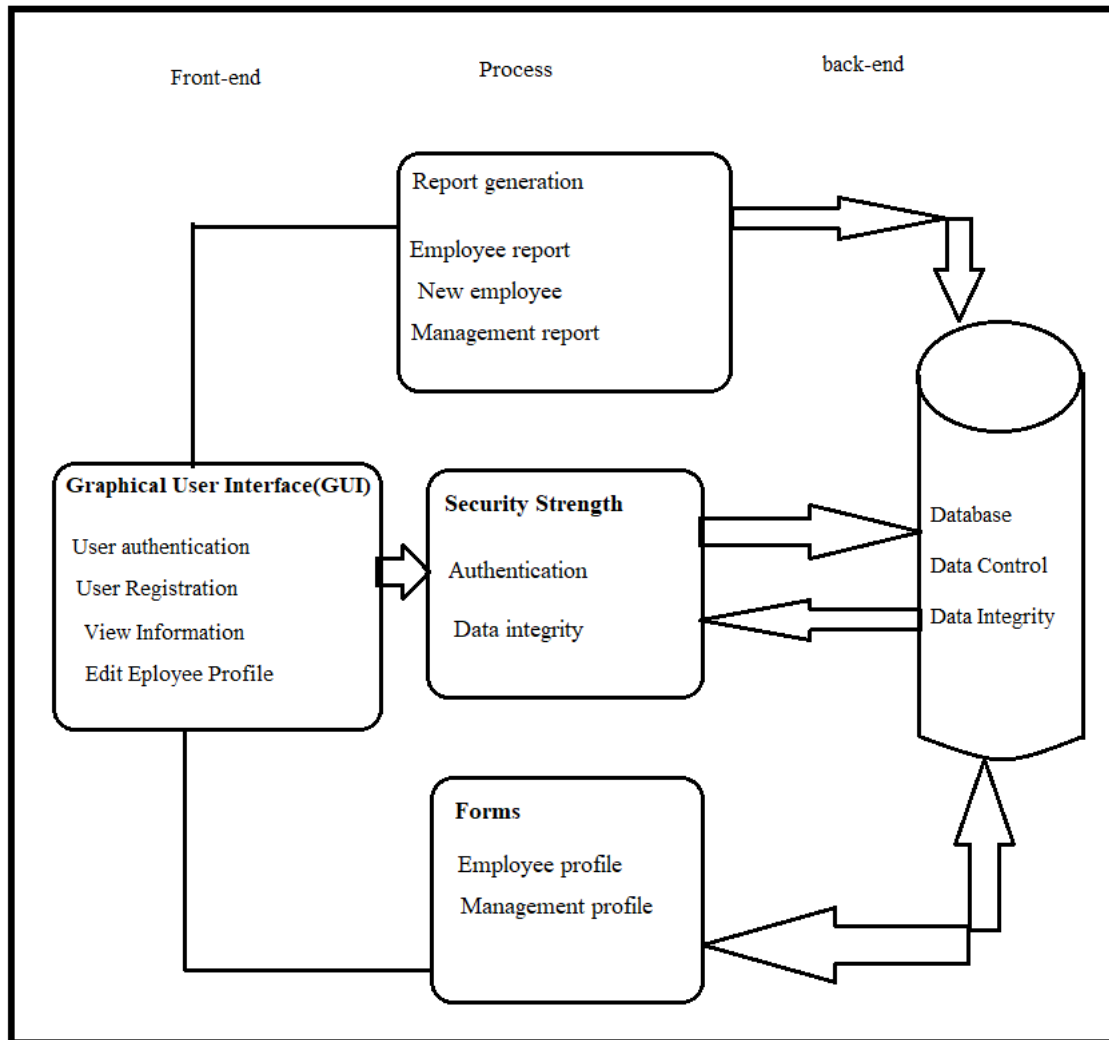
The primary key fields could be set to Auto-number data type as Access creates these values in an ascending order to ensure that they are unique within a table. Some of the fields should be “adjusted” to accept null-values. It is quite important to be done as it is tightly related to the input fields of the application program. I decided to perform it in the following way: those fields that are compulsory to be filled by the user I have set not to accept any null-values of data and on the other hand, those ones, that can be left blank, are set to accept null-values.

Retrieving data from the database

Retrieving data from a database is less or more tightly related to dealing with the SELECT query that should be applied to the database in order to extract the desirable result, which one should satisfy certain conditions. This SQL query has the following structure: SLECT FROM WHERE [(condition_1), (condition_2),(condition_n)]. Into the WHERE-statement, the following logical and arithmetical operators are included as well: [AND, OR, <=, >, >=, =]. The data from the database is retrieved in three different ways:

System Architecture

This gives a high level view of the new system with the main components of the system and the service they provide and how they communicate. The system is implemented using a three-tier architecture that comprises of our interface, process management and DBMS as illustrated bellow.



Entity Relationship (E-R) Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases. An entity relationship diagram is a means of visualizing how the information a system produces is related.

Entity

Which are represented by rectangle. An entity is an object or concept that has its existence in the real world. It includes all those things about which data is collected. A weak entity is an entity that must be defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.

Attributes

Which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.

An Entity Set

It is a set of entities of the same type that share the same properties, or attributes.

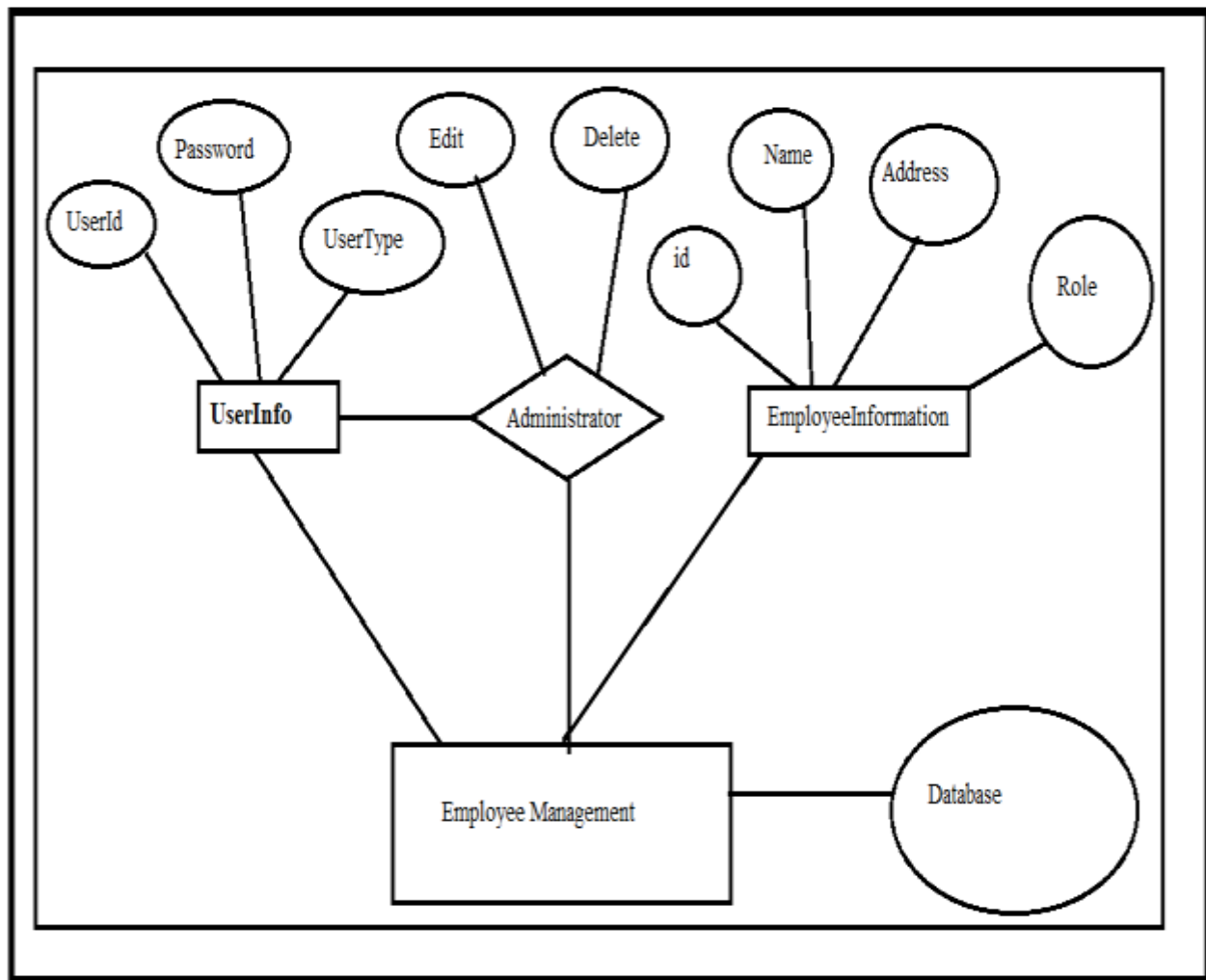
Process

A process shows a transformation or manipulation of data flows within the system.

ACTIONS:

Which are represented by diamond shapes, show how two entities share information in the database.

Data Flow Diagram



Code

index.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

<head>
<link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
    rel="stylesheet">
<link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/font-awesome@4.7.0/css/font-awesome.css">
<link rel="shortcut icon"
    href="https://cdn-icons-png.flaticon.com/512/470/470199.png">
<meta charset="UTF-8">
<title>EMS App</title>
</head>

<style>
body {
    overflow-x: hidden;
    scroll-behavior: smooth;
}

nav {
    background-color: navy !important;
}

.navbar-brand {
    color: whitesmoke !important;
    font-size: 30px !important;
    font-weight: bold;
}

.card {
    box-shadow: rgba(14, 30, 37, 0.12) 1px 2px 2px 0px,
                rgba(14, 30, 37, 0.32) 1px 2px 2px 0px;
}

.card-body {
    text-align: center;
}
```

```

.table-responsive {
    box-shadow: rgba(60, 64, 67, 0.3) 0px 1px 2px 0px,
                rgba(60, 64, 67, 0.15) 0px 2px 6px 2px;
}
th {
    text-align: center;
    font-size: 18px !important;
}

td {
    text-align: center;
    font-size: 15px;
    font-weight: 500;
}

label {
    font-weight: 500;
}
</style>

<body>

    <nav class="navbar navbar-expand-sm navbar-light mb-5">
        <div class="container">
            <a class="navbar-brand text-bold" href="#"><i
                class="fa fa-align-center" aria-
hidden="true">EMS</i></a>
            </div>
        </nav>

        <main>

<div class="container p-4">
    <div class="head_section">
        <div class="row row-cols-1 row-cols-md-4 g-3">
            <div class="col">
                <a href="#exampleModalToggle1" data-bs-
toggle="modal "
                    role="button" style="text-
decoration: none; color: white;">
                    <div class="card h-100 bq-
success"

```

```

</h5>
</div>
    </div>
    </a>
    </div>
<div class="col">
<a href="#exampleModalToggle2" data-bs-toggle="modal"
                                role="button" style="text-
decoration: none; color: white;">
<div class="card h-100 bq-primary">
<div class="card-body">
<h5 class="text-light">
<i class="fa fa-area-chart" aria-hidden="true"></i>Update
                                                                    Employee
                                </h5>
                                </div>
                                </div>
</a>
</div>
<div class="col">
<a href="#exampleModalToggle3" data-bs-toggle="modal"
role="button" style="text-decoration: none; color: white;">

<div class="card h-100 bq-danger">
    <div class="card-body">
        <h5 class="text-light">
            <i class="fa fa-trash"></i>Delete Employee
                </h5>
                </div>
            </div>
            </a>
            </div>
            <div class="col">
<a href="#deleteAllModal4" data-bs-toggle="modal" role="button"
style="text-decoration: none; color: white;">
                                <div class="card
h-100 bq-dark">
                                    <div class="card-body">
                                        <h5 class="text-light">
<i class="fa fa-warning"></i>Delete All
                                            </h5> </div>
                                            </div>
                                        </a>
                                        </div>
                                        </div>
                                        </div>
<br>
<div class="items_table mt-5 mb-4">
<div class="table-responsive p-2">

```

```
<h4 class="text-center p-2 class=" p-1" mt-2" style="font-family: 'Times New Roman', Times, serif; font-weight: bold;">Employee Management System</h4>
```

```
<table class="table table-bordered table-hover mt-5">
```

```
    </thead>
```

```
    <tbody>
```

```
    <tr th:each="employee, index : ${employees}">
```

```
    <td th:text="${index.index + 1}"></td>
```

```
    <td th:text="${employee.id}"></td>
```

```
    <td th:text="${employee.employeeName}"></td>
```

```
    <td th:text="${employee.employeeEmail}"></td>
```

```
    <td th:text="${employee.employeePhone}"></td>
```

```
    <td th:text="${employee.employeeGender}"></td>
```

```
    <td th:text="${employee.employeeSalary}"></td>
```

```
    <td th:text="${employee.employeeRole+ ' Developer'}"></td>
```

```
</tr>
```

```
    </tbody>
```

```
    </table>
```

```
    </div>
```

```
    </div>
```

```
</div>
```

```
</main>
```

```
    <!-- model for create-->
```

```
<div class="modal fade" id="exampleModalToggle1" aria-hidden="true"
aria-labelledby="exampleModalToggleLabel" tabindex="-1">
```

```
<div class="modal-dialog modal-dialog-centered">
```

```
    <div class="modal-content">
```

```
        <div class="modal-body">
```

```
        <div th:if="${success}" class="alert alert-success" role="alert">
```

```
        <p th:text="${success}"></p>
```

```
        </div>
```

```
<form class="p-2" th:action="@{/create}" th:object="${employee}"
method="post">
```

```
    <center>
```

```
    <h4 style="font-family: 'Times New Roman', Times, serif;">Add
Employee</h4>
```

```
    </center>
```

```
        <label class="p-1" for="employeeEmail">Email</label> <input
type="text" th:field="*{employeeEmail}" class="form-control"
placeholder="email address" required>
```

```
    </div>
```

```
<div class="row p-2">
```

```
        <label class="p-1" for="employeePhone">Phone</label> <input
type="tel" th:field="*{employeePhone}" class="form-control"
placeholder="phone number" required>
```

```

</div>
<div class="row p-2">
    <label class="p-1">Gender</label> <select
th:field="*{employeeGender}" class="form-select" required>
    <option value="" selected>select option</option>
    <option value="Male">Male</option>
    <option value="Female">Female</option>
</select>
</div>
    <div class="row p-2">
        <label class="p-1" for="employeeSalary">Salary</label> <input
        type="number" th:field="*{employeeSalary}" class="form-control"
placeholder="salary" required>
    </div>
    <div class="row p-2">
<label class="p-1" for="employeeRole">Employee Role</label> <select
th:field="*{employeeRole}" class="form-select" required>
    <option value="" selected>select option</option>
    <option value="Java">Java Developer</option>
    <option value="Python">Python Developer</option>
    <option value="Web">Web Developer</option>
    <option value="Android">Android Developer</option>
    <option value="UI">UI Developer</option>

</select>
</div>
<button type="submit" class="btn btn-success mt-3 mb-2">Add
    Employee</button>
</form></div>
</div>
</div>

    <!-- model for update-->
<div class="modal fade" id="exampleModalToggle2" aria-hidden="true"
    aria-labelledby="exampleModalToggleLabel" tabindex="-1">
    <div class="modal-dialog modal-dialog-centered">
        <div class="modal-content">
            <div class="modal-body">
                <form class="p-2" th:action="@{/update}" th:object="{employee}"
method="post">
    <!-- Check if errorMessage is present in the model and
display it -->
                <div th:if="{errorMessage}" class="alert alert-danger role="alert">
<p th:text="{errorMessage}"></p>
                </div>
            <center>
<div class="row p-1">
    <label class="p-1" for="id">Employee ID</label> <input

```

```

type="text" th:field="*{id}" class="form-control"
    placeholder="employee id" required>
        </div>
    <div class="row p-1">
        <label class="p-1" for="employeeName">Employee Name</label> <input
type="text" th:field="*{employeeName}" class="form-control"
    placeholder="employee name" required>
        </div>
    <div class="row p-1">
        <label class="p-1" for="employeeEmail">Email</label>

<input type="text" th:field="*{employeeEmail}" class="form-
control" placeholder="email address" required>
        </div>
    <div class="row p-1">
        <label class="p-1" for="employeePhone">Phone</label> <input
type="tel" th:field="*{employeePhone}" class="form-control"
    placeholder="phone number" required>
        </div>

    <div class="row p-1">
<label class="p-1">Gender</label> <select
th:field="*{employeeGender}" class="form-select" required>
    <option value="" selected>select option</option>
        <option value="Male">Male</option>
        <option value="Female">Female</option>
    </select>
    </div>
    <div class="row p-1">
        <label class="p-1" for="employeeSalary">Salary</label> <input
type="number" th:field="*{employeeSalary}" class="form-control"
placeholder="salary" required>
        </div>
        <div class="row p-1">
<label class="p-1" for="employeeRole">Employee Role</label> <select
th:field="*{employeeRole}" class="form-select" required>
    <option value="" selected>select option</option>
        <option value="Java">Java Developer</option>
        <option value="Python">Python Developer</option>
        <option value="Web">Web Developer</option>
        <option value="Android">Android Developer</option>
        <option value="UI">UI Developer</option>
    </select>
    <button type="submit" class="btn btn-primary mt-3 mb-2">Update
Employee</button>
        </form>
    </div>

```

```

<div class="row p-2">
    <label class="p-1" for="id">Employee ID</label> <input
type="text" th:field="*{id}" class="form-control "
placeholder="employee id" required>
    </div>
<button type="submit" class="btn btn-danger mt-3 mb-2">Delete
    Employee</button>
    </form>
</div>
</div>

<!-- Delete All employees-->
<div class="modal fade" id="deleteAllModal4" aria-hidden="true"
    aria-labelledby="deleteAllModalLabe4" tabindex="-1">
    <div class="modal-dialog modal-dialog-centered">
        <div class="modal-content">

            <div class="modal-body">
                <form class="p-2" th:action="@{/remove/all}"
                    th:object="${confirmationForm}"
method="post">
<center><h4 style="font-family: 'Times New Roman', Times, serif;">Delete
    All Employees</h4></center>

            <div class="row p-2">
                <label class="p-3 text-warning" for="confirmation">Type
                    'Yes' For Confirmation</label> <input
type="text" th:field="*{confirmation}" class="form-control "
placeholder="confirmation" required>
                </div>
                <button type="submit" class="btn btn-dark mt-3 mb-2">Delete
                    All Employees</button>
            </form>
        </div>
    </div>
</div>
</div>

<!-- bootstrap js -->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bund
le.min.js"></script>
</body>

</html>

```

New_employee.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Employee Management System</title>
<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
      integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
      crossorigin="anonymous">
</head>
<body>
  <div class="container">
    <h1>Employee Management System</h1>
    <hr>
    <h2>Save Employee</h2>

    <form action="#" th:action="@{/saveEmployee}" th:object="${employee}"
          method="POST">
      <input type="text" th:field="*{firstName}"
            placeholder="Employee First Name" class="form-control mb-4 col-
4">

      <input type="text" th:field="*{lastName}"

placeholder="Employee Last Name" class="form-control mb-4 col-4">

      <input type="text" th:field="*{email}"
            placeholder="Employee Email" class="form-control mb-4 col-4">

      <button type="submit" class="btn btn-info col-2"> Save
Employee</button>
    </form>

    <hr>
    <a th:href="@{/}"> Back to Employee List</a>
  </div>
</body>
</html>
```

addEmployee.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Employee Management System</title>
```



```

<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
</head>
<body>
  <div class="container">
    <h1>Employee Management System</h1>
    <hr>
    <h2>Update Employee</h2>

    <form action="#" th:action="@{/saveEmployee}" th:object="${employee}"
      method="POST">
      <!-- Add hidden form field to handle update -->
      <input type="hidden" th:field="*{id}" />

      <input type="text" th:field="*{firstName}" class="form-control mb-4 col-
4">

      <input type="text" th:field="*{lastName}" class="form-control mb-
4 col-4">

      <input type="text" th:field="*{email}" class="form-control mb-4 col-4">

      <button type="submit" class="btn btn-info col-2"> Update Employee</button>
    </form>
    <hr>
    <a th:href="@{/}"> Back to Employee List</a>
  </div>
</body>
</html>

```

registration.html

```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Registration</title>
<link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
      integrity="sha384-
BVYiSiFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
      crossorigin="anonymous">
</head>
<body>
  <!-- create navigation bar ( header) -->
  <nav class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">

```

```

<div class="navbar-header">
<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
target="#navbar" aria-expanded="false" aria-controls="navbar">
<span class="sr-only">Toggle navigation</span> <span class="icon-bar"></span> <span
class="icon-bar"></span> <span class="icon-bar"></span></button>
<a class="navbar-brand" href="#" th:href="@{/}">Employee Management System</a>

</div> </div>
</nav><br><br>
<!-- Create HTML registration form -->
<div class="container">
<div class="row">
<div class="col-md-6 col-md-offset-3">

<!-- success message -->
<div th:if="${param.success}">
<div class="alert alert-info">You've successfully registered
to our awesome app!</div>
</div>

<h1>Registration</h1>

<form th:action="@{/registration}" method="post"
th:object="${user}">

<div class="form-group">
<label class="control-label" for="firstName"> First
Name </label>
<input id="firstName" class="form-control"
th:field="*{firstName}" required autofocus="autofocus" />
</div>

<div class="form-group">
<label class="control-label" for="lastName"> Last
Name </label> <input
id="lastName" class="form-control"
th:field="*{lastName}"
required autofocus="autofocus" />
</div>

<div class="form-group">
<label class="control-label" for="email"> Email </label>
<input id="email" class="form-control" th:field="*{email}" required
autofocus="autofocus" />
</div>
<div class="form-group">
<label class="control-label" for="password"> Password </label>
<input id="password" class="form-control" type="password"
th:field="*{password}" required autofocus="autofocus" />

```

login.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Registration and Login App</title>

<link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
      integrity="sha384-
BVYiSiFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
      crossorigin="anonymous">

</head><body>
  <!-- create navigation bar ( header) -->
  <nav class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed"
          data-toggle="collapse" data-target="#navbar" aria-
expanded="false"    aria-controls="navbar">
<span class="sr-only">Toggle navigation</span> <span
                                class="icon-bar"></span> <span class="icon-bar"></span>
<span
                                class="icon-bar"></span>
                                </button>
        <a class="navbar-brand" href="#" th:href="@{/}">Employee
Management System</a>
      </div>      </div>
    </nav>
    <br> <br>
    <div class = "container">
      <div class = "row">
        <div class = "col-md-6 col-md-offset-3">
          <h1> Sign-in </h1>
          <form th:action="@{/login}" method="post">
            <!-- error message -->
            <div th:if="${param.error}">
              <div class="alert alert-danger">Invalid username or
                password.</div>
            </div>
```

```

        <!-- logout message -->
        <div th:if="{param.logout}">
            <div class="alert alert-info">You have been logged
out.</div>

        </div>

        <div class="form-group">
            <label for="username">Username </label> :
            <input type="text" class="form-control" id="username"
name="username"
placeholder="Enter Email ID" autofocus="autofocus">
        </div>

        <div class="form-group">
<label for="password">Password</label>: <input type="password"      id="password"
name="password" class="form-control"
placeholder="Enter Password" /></div>

<div class="form-group">
    <div class="row">
        <div class="col-sm-6 col-sm-offset-3">
            <input type="submit" name="login-submit" id="login-submit"
                class="form-control btn btn-primary" value="Log In" />
            </div> </div> </div> </form>

        <div class="form-group"><span>New user? <a href="/"
th:href="@{/registration}">Register here</a></span>
        </div> </div></div></div></body>

</html>

```

pom.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.2</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>

```

```

<artifactId>thymeleaf</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>thymeleaf</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>

    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>

    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>

    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <scope>runtime</scope>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter test</artifactId>
    <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>

```

```

<groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>

    </plugin>
</plugins>
</build>
</project>

```

Java Code:

EmployeeManagementSystemProjectApplication.java

```

package in.Keshav;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class EmployeeManagementSystemProjectApplication {

    public static void main(String[] args) {

        SpringApplication.run(EmployeeManagementSystemProjectApplication.class, args);
    }

}

```

ServletInitializer.java

```

package in.Keshav;

import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
public class ServletInitializer extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return

        application.sources(EmployeeManagementSystemProjectApplication.class);
    }

}

```

EmployeeController.java

```

package in.Keshav.controller;

import java.util.List;

```

```

import java.util.Optional;
import java.util.Random;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import in.Keshav.entity.ConfirmationForm;
import in.Keshav.entity.Employee;
import in.Keshav.repository.EmployeeRepo;

```

```

@Controller
public class EmployeeController {
    //https://www.javaguides.net/2021/07/spring-boot-tutorial-build-employee-management-
    project.html

```

```

    @Autowired
    private EmployeeRepo employeeRepo;
    // display the html page
    @GetMapping("/")

    public String getIndex(Model model) {
        List<Employee> employeeList = employeeRepo.findAll();
        model.addAttribute("employees", employeeList);
        model.addAttribute("employee", new Employee());
        model.addAttribute("confirmationForm", new ConfirmationForm());
        return "index";
    }

```

```

    // Insert employee data

```

```

    @PostMapping("/create")

```

```

    public String newEmployee(Employee employee, Model model) {
        model.addAttribute("employee", new Employee());

        // creating dynamic Employee ID
        String empId = "EMP";
        Random random = new Random();
        long randomNumber = 1000 + random.nextInt(9000);
        empId = empId + randomNumber;
        employee.setId(empId);

        // save the employee
        employeeRepo.save(employee);

        return "redirect:/";
    }

```

```

// update the existing employee
@PostMapping("/update")
public String updateEmployee(@ModelAttribute Employee employee, Model
model) {
    model.addAttribute("employee", new Employee());
    Optional<Employee> existingEmployee =
employeeRepo.findById(employee.getId());
    // checking employee exist or not
    if (existingEmployee.isPresent()) {
        employeeRepo.save(employee);
    } else {
        model.addAttribute("errorMessage", "Employee with ID " +
employee.getId() + " not found.");
    }
    return "redirect:/";
}

// delete an employee by id
@PostMapping("/remove")

public String removeEmployee(Employee employee, Model model) {
    model.addAttribute("employee", new Employee());
    Optional<Employee> existingEmployee =
employeeRepo.findById(employee.getId());
    if (existingEmployee.isPresent()) {
        employeeRepo.deleteById(employee.getId());
    }
    return "redirect:/";
}

// delete all employees data by confirmation
@PostMapping("/remove/all")
public String removeAll(@ModelAttribute ConfirmationForm confirmationForm,
Model model) {
    String confirmation = confirmationForm.getConfirmation();
    if ("Yes".equalsIgnoreCase(confirmation)) {
        employeeRepo.deleteAll();
    } else {
        return "redirect:/";
    }
    return "redirect:/";
}
}

```


ConfirmationForm.java

```
package in.Keshav.entity;
public class ConfirmationForm {

    private String confirmation;

    public ConfirmationForm() {
        super();
    }

    public ConfirmationForm(String confirmation) {
        super();
        this.confirmation = confirmation;
    }

    public String getConfirmation() {
        return confirmation;
    }

    public void setConfirmation(String confirmation) {
        this.confirmation = confirmation;
    }

    @Override
    public String toString() {return "ConfirmationForm [confirmation=" +
confirmation + "]}";

    }

}
```

Employee.java

```
package in.Keshav.entity;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table (name="employee_system")
public class Employee {
    @Id
    private String id;
    private String employeeName;
    private String employeeEmail;
    private Long employeePhone;
```

```

private String employeeGender;
private String employeeSalary;
private String employeeRole;

    public Employee() {
        super();
    }

    public Employee(String id, String employeeName, String employeeEmail, Long
employeePhone, String employeeGender,
        String employeeSalary, String employeeRole) {
        super();
        this.id = id;
        this.employeeName = employeeName;
        this.employeeEmail = employeeEmail;
        this.employeePhone = employeePhone;
        this.employeeGender = employeeGender;
        this.employeeSalary = employeeSalary;
        this.employeeRole = employeeRole;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getEmployeeName() {
        return employeeName;
    }
    public void setEmployeeName(String employeeName) {
        this.employeeName = employeeName;
    }
    public String getEmployeeEmail() {
        return employeeEmail;
    }
    public void setEmployeeEmail(String employeeEmail) {
        this.employeeEmail = employeeEmail;
    }
    public Long getEmployeePhone() {
        return employeePhone;
    }
    public void setEmployeePhone(Long employeePhone) {
        this.employeePhone = employeePhone;
    }
    public String getEmployeeGender() {
        return employeeGender;
    }

```

```

    }

    public void setEmployeeGender(String employeeGender) {
        this.employeeGender = employeeGender;
    }

    public String getEmployeeSalary() {
        return employeeSalary;
    }

    public void setEmployeeSalary(String employeeSalary) {
        this.employeeSalary = employeeSalary;
    }

    public String getEmployeeRole() {
        return employeeRole;
    }

    public void setEmployeeRole(String employeeRole) {

        this.employeeRole = employeeRole;
    }

    @Override
    public String toString() {
        return "Employee [id=" + id + ", employeeName=" + employeeName + ",
employeeEmail=" + employeeEmail
        + ", employeePhone=" + employeePhone + ", employeeGender=" +
employeeGender + ", employeeSalary="
        + employeeSalary + ", employeeRole=" + employeeRole + "]";
    }

}

```

EmployeeRepo.java

```

package in.Keshav.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import in.Keshav.entity.Employee;
public interface EmployeeRepo extends JpaRepository<Employee, String>{

}

```

UserServiceImpl.java

```

import java.util.Arrays;
import java.util.Collection;
import java.util.stream.Collectors;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;

```

```

import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import net.javaguides.springboot.dto.UserRegistrationDto;
import net.javaguides.springboot.model.Role;
import net.javaguides.springboot.model.User;
import net.javaguides.springboot.repository.UserRepository;

@Service
public class UserServiceImpl implements UserService {

    private UserRepository userRepository;
    @Autowired
    private BCryptPasswordEncoder passwordEncoder;

    public UserServiceImpl(UserRepository userRepository) {
        super();
        this.userRepository = userRepository;
    }

    @Override
    public User save(UserRegistrationDto registrationDto) {
        User user = new User(registrationDto.getFirstName(),
            registrationDto.getLastName(), registrationDto.getEmail(),
            passwordEncoder.encode(registrationDto.getPassword()),
            Arrays.asList(new Role("ROLE_USER")));

        return userRepository.save(user);
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws
    UsernameNotFoundException {

        User user = userRepository.findByEmail(username);
        if(user == null) {
            throw new UsernameNotFoundException("Invalid username or password.");
        }
        return new org.springframework.security.core.userdetails.User(user.getEmail(),
            user.getPassword(), mapRolesToAuthorities(user.getRoles()));
    }

    private Collection<? extends GrantedAuthority>
    mapRolesToAuthorities(Collection<Role> roles){
        return roles.stream().map(role -> new
        SimpleGrantedAuthority(role.getName())).collect(Collectors.toList());
    }
}

```

```

package in.Keshav.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

```

SecurityConfiguration.java

```

import in.service.UserService;
@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Autowired
    private UserService userService;

    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider auth = new DaoAuthenticationProvider();
        auth.setUserDetailsService(userService);
        auth.setPasswordEncoder(passwordEncoder());
        return auth;
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.authenticationProvider(authenticationProvider());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests().antMatchers(
            "/registration**",

```

```

"/js/**", "/css/**", "/img/**").permitAll().anyRequest().authenticated()
    .and()
    .formLogin()
    .loginPage("/login")

    .permitAll()
    .and()
    .logout()
    .invalidateHttpSession(true)
    .clearAuthentication(true)
    .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
    .logoutSuccessUrl("/login?logout")
    .permitAll();
}

}

```

MainController.java

```

package in.Keshav.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller

public class MainController {

    @GetMapping("/login")

    public String login() {

        return "login";

    }    /*

    * @GetMapping("/") public String home() { return "index"; }

    */}

```

UserRegistrationController.java

```

package in.Keshav.controller;
import org.springframework.stereotype.Controller;

```

```

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import net.javaguides.springboot.dto.UserRegistrationDto;
import net.javaguides.springboot.service.UserService;

@Controller
@RequestMapping("/registration")

public class UserRegistrationController {

    private UserService userService;

    public UserRegistrationController(UserService userService) {

        super();

        this.userService = userService;

    }

    @ModelAttribute("user")

    public UserRegistrationDto userRegistrationDto() {

        return new UserRegistrationDto();

    }

    @GetMapping

    public String showRegistrationForm() {

        return "registration";

    }

    @PostMapping

```

```

public String registerUserAccount(@ModelAttribute("user")
UserRegistrationDto registrationDto) {

    userService.save(registrationDto);

    return "redirect:/registration?success";

}

```

```

}

```

EmployeeService.java

```

import java.util.List;

import org.springframework.data.domain.Page;

import net.javaguides.springboot.model.Employee;

public interface EmployeeService {

    List<Employee> getAllEmployees();

    void saveEmployee(Employee employee);

    Employee getEmployeeById(long id);void deleteEmployeeById(long
id);

    Page<Employee> findPaginated(int pageNo, int pageSize, String
sortField, String sortDirection);

}

```

Role.java

```

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

```



```

@Table(name = "role")

public class Role {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    public Role() {

    }

    public Role(String name) {

        super();

        this.name = name;

    }

    public Long getId() {

        return id;

    }

    public void setId(Long id) {

        this.id = id;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

}

```

User.java

```
import java.util.Collection;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.persistence.UniqueConstraint;
import javax.persistence.JoinColumn;

@Entity

@Table(name = "user", uniqueConstraints = @UniqueConstraint(columnNames
= "email"))

public class User {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @Column(name = "first_name")

    private String firstName;

    @Column(name = "last_name")

    private String lastName;

    private String email;

    private String password;

    @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)

    @JoinTable(        name = "users_roles",

        joinColumns = @JoinColumn(
```

```

        name = "user_id", referencedColumnName = "id"),
inverseJoinColumns = @JoinColumn( name = "role_id",
        referencedColumnName = "id"))
private Collection<Role> roles;

    public User() {
    }

    public User(String firstName, String lastName, String email,
String password, Collection<Role> roles) {

        super();

        this.firstName = firstName;

        this.lastName = lastName;

        this.email = email;

        this.password = password;

        this.roles = roles;
    }

    public Long getId() {

        return id;
    }

    public void setId(Long id) {

        this.id = id    }

    public String getFirstName() {

        return firstName;
    }

    public void setFirstName(String firstName) {

```

```

this.firstName = firstName;

    }

public String getLastName() {

    return lastName;

}

public void setLastName(String lastName) {

    this.lastName = lastName;

}

public String getEmail() {

    return email;

}

    public void setEmail(String email) {

        this.email = email;

    }

public String getPassword() {

    return password;

}

public void setPassword(String password) {

    this.password = password;

}

public Collection<Role> getRoles() {

    return roles;

}

    public void setRoles(Collection<Role> roles) {

```

```
this.roles = roles;
```

```
    }
```

```
}
```

application.properties

```
spring.datasource.url= jdbc:mysql://localhost:3306/project
```

```
spring.datasource.username= root
```

```
spring.datasource.password= password
```

```
spring.datasource.driver= com.mysql.cj.jdbc.Driver
```


```
spring.jpa.hibernate.ddl-auto= create
```


```
spring.jpa.show-sql= true
```


```
spring.mvc.view.prefix=/templates/
```


```
spring.mvc.view.suffix=.html
```


Screenshot image



 Add Employee

 Update Employee

 Delete Employee

 Delete All

SLNO	ID	Name	Email	Phone	Gender	Salary	Role
1	EMP1491	Keshav	cks@gmail.com	6598746985	Male	56000000	Java Developer
2	EMP2962	Aliya	al@gmail.com	4569321452	Female	96800000	Web Developer
3	EMP3509	Santosh	anish@gmail.com	6398562341	Male	1200000	UI Developer
4	EMP3797	John	john12@gmail.com	5896365420	Male	12000	Web Developer
5	EMP4525	Amit	a@gmail.com	1236547896	Male	450000	Python Developer
6	EMP7150	Amrutesh	rikh@gmail.com	5698745693	Male	456220000	Java Developer

EMS

✚Add Employee

⚠Delete All

Sl.NO	ID	Name	Gender	Salary	Role
1	EMP1491	Keshav		56000000	Java Developer
2	EMP3509	Santosh		1200000	UI Developer
3	EMP7150	Amrutesh		456220000	Java Developer

Update Employee

Employee ID

employee id

Employee Name

employee name

Email

email address

Phone

phone number

Gender

select option

Salary

salary

Employee Role

select option

Update Employee

EMS

✚Add Employee

⚠Delete All

Sl.NO	ID	Name	Gender	Salary	Role
1	EMP1491	Keshav		56000000	Java Developer
2	EMP3509	Santosh		1200000	UI Developer
3	EMP7150	Amrutesh		456220000	Java Developer

Update Employee

Employee ID

employee id

Employee Name

employee name

Email

email address

Phone

phone number

Gender

select option

Salary

salary

Employee Role

select option

Update Employee

[49]

✚Add Employee

🔄Update Employee

🗑Delete Employee

🗑Delete All

Delete Employee

Employee ID

employee id

Delete Employee

Sl.NO	ID	Name	Gender	Salary	Role
1	EMP1481	Kesha	Female	5600000	Java Developer
2	EMP1509	Santo	Male	1200000	UI Developer
3	EMP7152	Amrutesh	Male	45622000	Java Developer

✚Add Employee

🔄Update Employee

🗑Delete Employee

⚠Delete All

Delete Employee

Employee ID

Sl.NO	ID	Name	Gender	Salary	Role
1	EMP1481	Kesha	Female	5600000	Java Developer
2	EMP1509	Santo	Male	1200000	UI Developer
3	EMP7152	Amrtesh	Male	45622000	Java Developer

Delete Employee

Chapter 4

Market Potential & Competitive Advantages

The old manual system was suffering from a series of drawbacks. Since whole of the system was to be maintained with hands the process of keeping, maintaining and retrieving the information was very tedious and lengthy. The records were never used to be in a systematic order. there used to be lots of difficulties in associating any particular transaction with a particular con text. If any information was to be found it was required to go through the different registers, documents there would never exist anything like report generation.

Likely Benefits

As the population of the world continues to grow, so does the need for healthcare services. This demand for healthcare, in turn, will increase the number of employees seeking care at medical facilities, employees, wellness centers, physicians' practices and holistic groups. While the employee numbers increase,

Feature scope

Whether you're reserving a table at your favorite restaurant or ensuring that you have a set time to visit a busy fine arts museum, online employee scheduling continues to become further ingrained in our daily lives. The same should be said about the way employees schedule a visit to the employee's office.

Conclusion

In this report, an information system's development has been presented. It was emphasized on the basic steps, consequently taken during the project's development course as a particular attention

References

- [1] – Begg Carolyn, Connolly Thomas, Database systems (a Practical approach to Design, Implementation, and Management), Addison-Wesley, an imprint of Pearson Education, University of Paisley (U.K.), Fourth edition 2005
- [2] – Bodnar George /Duquesne University/, Hopwood William /Florida Atlantic University/, Accounting Information systems, Eighth Edition, Prentice Hall, Upper Saddle River, New Jersey .
- [3] – Andersen Virginia, Access 2000: The Complete Reference, Blacklick, OH, USA: McGraw-Hill Professional Book Group, 2001, <http://site.ebrary.com/lib/vaxjo/Doc?id=5002842> (2006-05-25).
- [4] – Andersson Tobias, [DAB744] C# Course Lectures, School of Mathematics and System Engineering, Växjö University.
- [5] - <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vboritextboxctltasks.asp> (2024-05-25).
- [6]-Harper PR, Gamlin HM. Reduced outemployee waiting times with improved employee scheduling: a simulation modeling approach. *OR Spectrum*. 2003;25:207– 222. doi: 10.1007/s00291-003-0122-x. [[CrossRef](#)] [[Google Scholar](#)]
- [7]-Zhu ZC, Heng BH, Teow KL. Analysis of factors causing long employee waiting time and clinic overtime in outemployee clinics. *J Med Syst*. 2010. in press . [[PubMed](#)]