InfoBridgePro: Project Report

CS51550-001, Fall 2024
PNW University
10/31/2024

Name	Email	Contributions
Mekala Ruthvik Reddy	**rmekala@pnw.edu**	**Application Code, Containerized PHP and SQL Setup, Documentation**
Nalluri Prashanth	**pnallur@pnw.edu**	**Application Code, User Interface, Database Design**

Table of Contents

Table of Figures

ABSTRACT

This project demonstrates the integration of a MySQL Database Management System with PHP for managing and depicting the overall information about the COMPANY database effectively. Its main aim is to allow the user to deal with database entities such as departments, projects, and employees through a web interface effortlessly. This technology stack includes MySQL for data storage, PHP as the server-side language, Docker for containerization, HTML for web page construction as well as Apache which is the web server that deploys on a Windows platform.

The MySQL installation in Docker has two distinct users, one for building the database's structural design and the second one for ".dat" file loading to improve security and operational effectiveness. The application offers different options, like an SSN (Social Security Number) drop-down for selecting employees, department views in detail, and lists of employees by department filters. The project also focuses on usability and graphical issues, bringing good navigability and data representation into focus. This project is useful to demonstrate certain practical elements of database administration but also to invite ideas about extending the scope of the future work in which more complex data analytics and additional security are incorporated.

I. INTRODUCTION

Organizations today, regardless of the sector within which they operate, have to take into account, and actively manage, their data. Central to any organization—there are database systems these days, so there is no more chaos in databases, everything is in order—database management systems enable coherent storage, call, and processing of information, with the features of assurance in integrity, protection, and availability. The practice of relying on flat files or spreadsheets solutions has its downsides. These systems basically caused data redundancy and their inconsistencies, which are addressed effectively by DBMS because the systems allow multiple user access to centrally located data.

This project aims to design COMPANY database, a client's management and information display web application running through PHP and managed via MySQL database. The system revolves around main objects such as departments, projects and employees, and providing the user with easy access to them via a simple user interface.

The application utilizes MySQL database for storage, PHP scripting language for the server, Docker for addressing issues of containerization, HTML as the structure of web pages and Apache as server running on windows to provide data management. Other features are providing dropdowns that maintain employee social security nut empirical numbers selection, advanced views of departments and selective employment listings. To enhance security and efficiency, this project has two MySQL users where one is for creating the structure of the database and the second one is employed in the loading of .dat files – the project improves operational efficiency, laying the foundation for future enhancements in data management and analytics.

The project also puts great emphasis on user-centered design-ease of access and ease of use for all categories of users. It means the interface will be so designed that users can switch from one functionality to another in comfort and hence interact with an application easily. Such usability support will enable one to handle information about employees, their departments, and hence assist in arriving at informed decisions and increasing productivity. This project nurtures in its approach supports the organizational goals of effective data management and user satisfaction.

II. SYSTEM DESIGN AND IMPLEMENTATION

Development of the MySQL Database Management System commenced with an E.R diagram that was to describe the association relationships that existed among other entities such as departments, employees, and projects. This E.R diagram laid down the basic template for the structure of the database and thus helped in bringing out the overall system design. MySQL ran the database to make sure it was consistent and reliable across various different environments that allowed the deployment of the Docker container. The application was done in PHP, for server-side scripting, combined with HTML for designing the user interface in practically implementing the E.R. model in the functional database scheme. The coding was done so as to give extremely user-friendly interfaces and business logics for different users in retrieving the details of employees by their SSNs and jumping to department information with ease. This structured approach aimed to enhance user experience while ensuring efficient data management and retrieval within the COMPANY database, thereby streamlining the processes involved in employee and department management.

1. ER DIAGRAM

First, we created an ER Diagram for the whole system to get a better understanding of the tables in the system and to write the query easily. Since there are primary keys and foreign keys we need to know the relationship of each table properly.

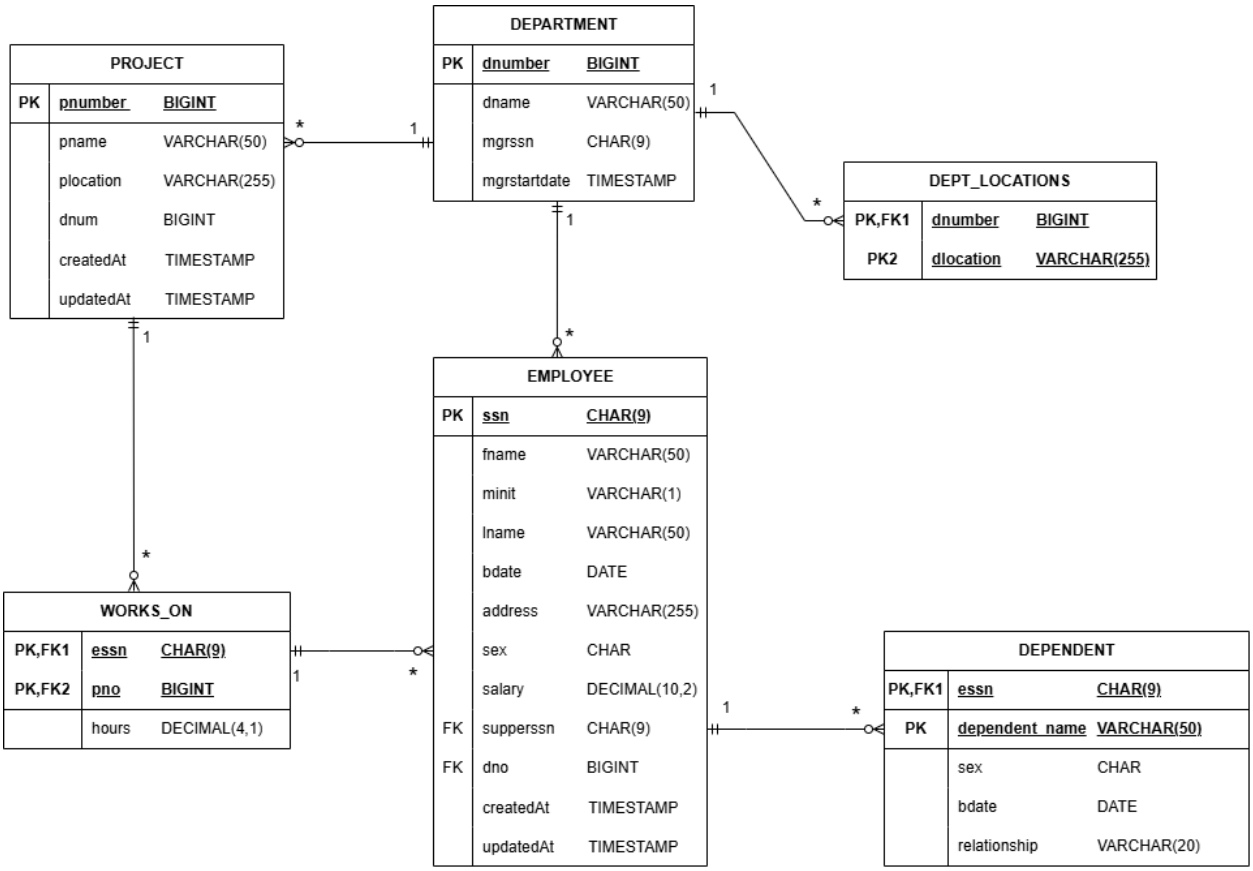

Figure 1 – E.R Diagram

2. Connecting PHP Application to Dockerized MySQL

The services defined in this docker-compose.yml file are two: the web service which is for the PHP application and the db service which is for the MySQL database. The local Dockerfile building the web service is set up to mount src directory on /var/www/html of the container. Port 80 is mapped to allow access to the application on localhost:80 Port. This service uses db_data, The MySQL 8.0 image and provisioned for persistent storage. It creates a database by the name of company, assigns a password to the root user and opens port 3306 for connections from the localhost on port 3306. The depends_on guarantees that the docker-compose web launches only when the docker-compose db has already started, whereas restart: unless-stopped completes the failure safety for the introduced database.

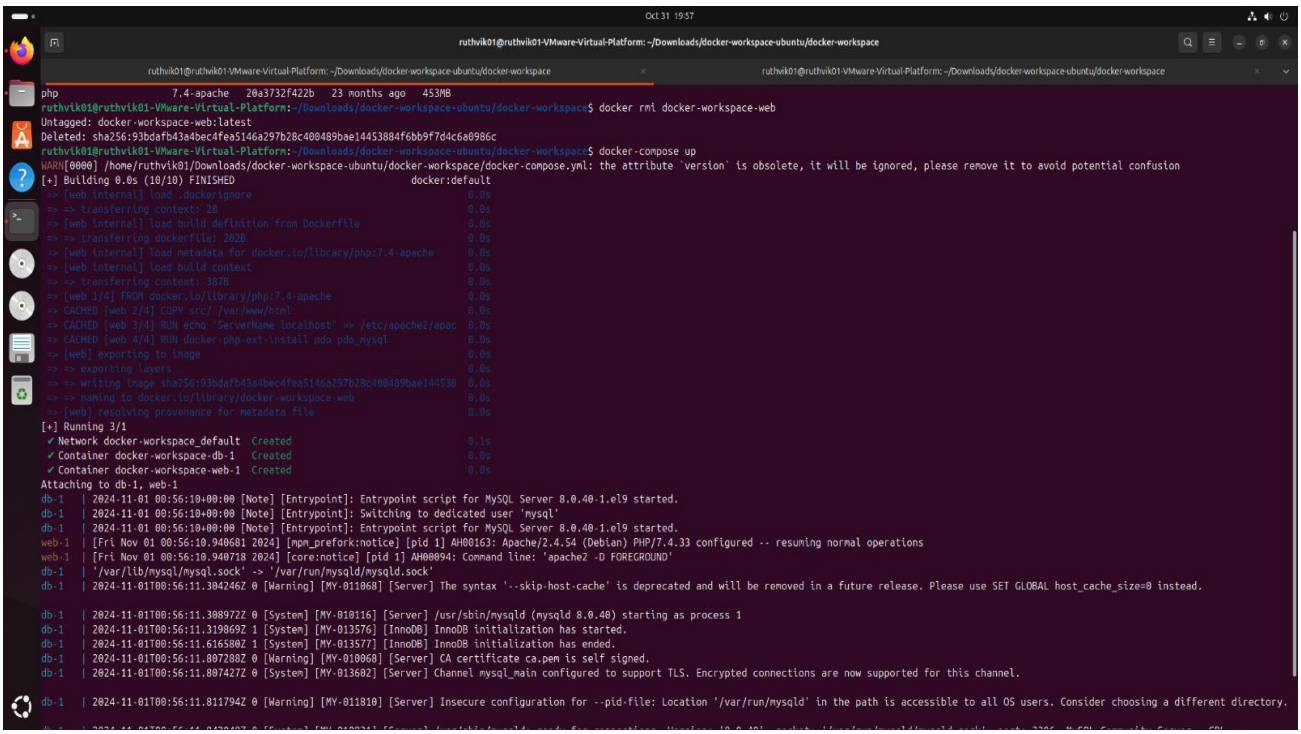

Figure 2 - Establishing the connection

```
ruthvik01@ruthvik01-VMware-Virtual-Platform:~/Downloads/docker-workspace-ubuntu/docker-workspace$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-10-31 22:02:19 CDT; 13s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 6352 (dockerd)
      Tasks: 9
     Memory: 25.7M (peak: 25.8M)
        CPU: 443ms
     CGroup: /system.slice/docker.service
             └─6352 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Figure 3 – Docker Container Status

3. Database Schema

a. We created a database named company in a dockerized MySQL 8 in Linux.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| company            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)
```

Figure 4 – Company Database

b. Here we created two users, The first user is called book, and has granted all rights to the company database to the book user. This user will create tables in the database.

```
mysql> use company;
Database changed
mysql> CREATE USER 'book'@'%' IDENTIFIED BY 'book';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON company.* TO 'book'@'%';
Query OK, 0 rows affected (0.01 sec)
```

```
bash-5.1# mysql -u book -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.40 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| company            |
| information_schema |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)

mysql> use company;
Database changed
mysql>  CREATE TABLE DEPARTMENT (
    ->         dname varchar(50) not null,
    ->         dnumber bigint,
    ->         mgrssn char(9) not null,
    ->         mgrstartdate timestamp,
    ->         primary key (dnumber),
    ->         key (dname)
    ->       );
Query OK, 0 rows affected (0.02 sec)
```

Figure 5 – Book User

c. After that we created all the tables. The system is based on a company database which has tables, DEPARTMENT, DEPENDENT, DEPT_LOCATIONS, EMPLOYEE, PROJECT and WORKS_ON.

```
mysql> show tables;
+-------------------+
| Tables_in_company |
+-------------------+
| DEPARTMENT        |
| DEPENDENT         |
| DEPT_LOCATIONS    |
| EMPLOYEE          |
| PROJECT           |
| WORKS_ON          |
+-------------------+
6 rows in set (0.00 sec)
```

Figure 6 – Tables in the database

d. **DEPARTMENT:** This table includes Department name, Department number, Manager's SSN and Manager StartDate.

```
mysql> describe DEPARTMENT;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| dname        | varchar(50) | NO   | MUL | NULL    |       |
| dnumber      | bigint      | NO   | PRI | NULL    |       |
| mgrssn       | char(9)     | NO   |     | NULL    |       |
| mgrstartdate | timestamp   | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Figure 7 – DEPARTMENT Table

e. **DEPENDENT:** This table is to show the information about the dependents of employees on our database.

```
mysql> describe DEPENDENT;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| essn           | char(9)     | NO   | PRI | NULL    |       |
| dependent_name | varchar(50) | NO   | PRI | NULL    |       |
| sex            | char(1)     | YES  |     | NULL    |       |
| bdate          | date        | YES  |     | NULL    |       |
| relationship   | varchar(20) | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Figure 8 – DEPENDENT Table

f. **PROJECT:** This table gives the essential information about Project names and Project locations and when it is created and updated at.

```
mysql> describe PROJECT;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| pname     | varchar(50)  | NO   | UNI | NULL    |       |
| pnumber   | bigint       | NO   | PRI | NULL    |       |
| plocation | varchar(255) | YES  |     | NULL    |       |
| dnum      | bigint       | NO   | MUL | NULL    |       |
| createdAt | timestamp    | YES  |     | NULL    |       |
| updatedAt | timestamp    | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
```

Figure 9 – PROJECT Table

g. **DEPT_LOCATIONS:** This table has information about the department's location.

```
mysql> describe DEPT_LOCATIONS;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| dnumber   | bigint       | NO   | PRI | NULL    |       |
| dlocation | varchar(255) | NO   | PRI | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Figure 10 – DEPT_LOCATIONS Table

h. **EMPLOYEE:** This table has the essential information about an employee starting with his First, Middle, Last name, Birthday, salary, SSN and gender.

```
mysql> describe EMPLOYEE;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| fname     | varchar(50)  | NO   |     | NULL    |       |
| minit     | varchar(1)   | YES  |     | NULL    |       |
| lname     | varchar(50)  | NO   |     | NULL    |       |
| ssn       | char(9)      | NO   | PRI | NULL    |       |
| bdate     | date         | YES  |     | NULL    |       |
| address   | varchar(255) | YES  |     | NULL    |       |
| sex       | char(1)      | YES  |     | NULL    |       |
| salary    | int          | YES  |     | NULL    |       |
| superssn  | char(9)      | YES  |     | NULL    |       |
| dno       | bigint       | YES  | MUL | NULL    |       |
| createdAt | timestamp    | YES  |     | NULL    |       |
| updatedAt | timestamp    | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
12 rows in set (0.00 sec)
```

Figure 11 – EMPLOYEE Table

i. **WORKS_ON:** This table shows how many hours an employee is going to work on a project.

```
mysql> describe WORKS_ON;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| essn  | char(9)      | NO   | PRI | NULL    |       |
| pno   | bigint       | NO   | PRI | NULL    |       |
| hours | decimal(4,2) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Figure 12 – WORKS_ON Table

j. The second user is called ruthvik, this user can load bulk data which is in .dat file form into the tables.

```
mysql> CREATE USER 'ruthvik'@'%' IDENTIFIED BY 'ruthvik';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON company.* TO 'ruthvik'@'%';
Query OK, 0 rows affected (0.01 sec)
ruthvik01@ruthvik01-VMware-Virtual-Platform:~/Downloads/docker-workspace-ubuntu/docker-workspace$ docker exec -it a910a899b010 bash
bash-5.1# mysql --local-infile=1 -u ruthvik -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 8.0.40 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use company;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> LOAD DATA INFILE '/var/lib/mysql-files/department.dat'
    -> INTO TABLE DEPARTMENT
    -> FIELDS TERMINATED BY ','
    -> OPTIONALLY ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> (dname, dnumber, mgrssn, mgrstartdate);
Query OK, 6 rows affected (0.00 sec)
Records: 6  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from DEPARTMENT;
+---------------+---------+-----------+---------------------+
| dname         | dnumber | mgrssn    | mgrstartdate        |
+---------------+---------+-----------+---------------------+
| Headquarters  |       1 | 888665555 | 1971-06-19 14:20:00 |
| Administration |       4 | 987654321 | 1985-01-01 10:45:00 |
| Research      |       5 | 333445555 | 1978-05-22 09:30:00 |
| Software      |       6 | 111111100 | 1999-05-15 08:15:00 |
| Hardware      |       7 | 444444400 | 1998-05-15 13:50:00 |
| Sales         |       8 | 555555500 | 1997-01-01 16:05:00 |
+---------------+---------+-----------+---------------------+
6 rows in set (0.00 sec)

mysql>
```

Figure 13- Creating ruthvik user and loading data to Department table

```
Database changed
mysql> LOAD DATA INFILE '/var/lib/mysql-files/project.dat'
    -> INTO TABLE PROJECT
    -> FIELDS TERMINATED BY ','
    -> OPTIONALLY ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> (pname, pnumber, plocation, dnum)
    -> SET createdAt = NOW(), updatedAt = NOW();
Query OK, 11 rows affected (0.01 sec)
Records: 11  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from PROJECT;
+------------------+---------+--------------+------+---------------------+---------------------+
| pname            | pnumber | plocation    | dnum | createdAt           | updatedAt           |
+------------------+---------+--------------+------+---------------------+---------------------+
| ProductX         |       1 | Bellaire     |    5 | 2024-10-31 22:14:08 | 2024-10-31 22:14:08 |
| ProductY         |       2 | Sugarland    |    5 | 2024-10-31 22:14:08 | 2024-10-31 22:14:08 |
| ProductZ         |       3 | Houston      |    5 | 2024-10-31 22:14:08 | 2024-10-31 22:14:08 |
| Computerization  |      10 | Stafford     |    4 | 2024-10-31 22:14:08 | 2024-10-31 22:14:08 |
| Reorganization   |      20 | Houston      |    1 | 2024-10-31 22:14:08 | 2024-10-31 22:14:08 |
| Newbenefits      |      30 | Stafford     |    4 | 2024-10-31 22:14:08 | 2024-10-31 22:14:08 |
| OperatingSystems |      61 | Jacksonville |    6 | 2024-10-31 22:14:08 | 2024-10-31 22:14:08 |
| DatabaseSystems  |      62 | Birmingham   |    6 | 2024-10-31 22:14:08 | 2024-10-31 22:14:08 |
| Middleware       |      63 | Jackson      |    6 | 2024-10-31 22:14:08 | 2024-10-31 22:14:08 |
| InkjetPrinters   |      91 | Phoenix      |    7 | 2024-10-31 22:14:08 | 2024-10-31 22:14:08 |
| LaserPrinters    |      92 | LasVegas     |    7 | 2024-10-31 22:14:08 | 2024-10-31 22:14:08 |
+------------------+---------+--------------+------+---------------------+---------------------+
11 rows in set (0.00 sec)
```

Figure 14- Loading data to PROJECT table

```
mysql> LOAD DATA INFILE '/var/lib/mysql-files/dloc.dat'
    -> INTO TABLE DEPT_LOCATIONS
    -> FIELDS TERMINATED BY ','
    -> OPTIONALLY ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> (dnumber, dlocation);
Query OK, 13 rows affected (0.01 sec)
Records: 13  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from DEPT_LOCATIONS;
+---------+-------------+
| dnumber | dlocation   |
+---------+-------------+
|       1 | Houston     |
|       4 | Stafford    |
|       5 | Bellaire    |
|       5 | Houston     |
|       5 | Sugarland   |
|       6 | Atlanta     |
|       6 | Sacramento  |
|       7 | Milwaukee   |
|       8 | Chicago     |
|       8 | Dallas      |
|       8 | Miami       |
|       8 | Philadephia |
|       8 | Seattle     |
+---------+-------------+
13 rows in set (0.00 sec)
```

Figure 15- Loading data to DEPT_LOCATIONS table

```
mysql> LOAD DATA INFILE '/var/lib/mysql-files/employee.dat'
    -> INTO TABLE EMPLOYEE
    -> FIELDS TERMINATED BY ','
    -> OPTIONALLY ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> (fname, minit, lname, ssn, bdate, address, sex, salary, superssn, dno)
    -> SET createdAt = NOW(), updatedAt = NOW();
Query OK, 40 rows affected (0.01 sec)
Records: 40  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from EMPLOYEE;
+---------+------+--------+----------+----------+-------------------------------+------+--------+----------+------+---------------------+---------------------+
| fname   | minit | lname  | ssn      | bdate    | address                       | sex  | salary | superssn | dno  | createdAt           | updatedAt           |
+---------+------+--------+----------+----------+-------------------------------+------+--------+----------+------+---------------------+---------------------+
| Jared   | D    | James  | 111111100 | 1966-10-10 | 123 Peachtree, Atlanta, GA    | M    | 85000 | null     |    6 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
| Jon     | C    | Jones  | 111111101 | 1967-11-14 | 111 Allgood, Atlanta, GA      | M    | 45000 | 111111100 |    6 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
| Justin  | n    | Mark   | 111111102 | 1966-01-12 | 2342 May, Atlanta, GA         | M    | 40000 | 111111100 |    6 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
| Brad    | C    | Knight | 111111103 | 1968-02-13 | 176 Main St., Atlanta, GA     | M    | 44000 | 111111100 |    6 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
| John    | B    | Smith  | 123456789 | 1955-01-09 | 731 Fondren, Houston, TX      | M    | 30000 | 333445555 |    5 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
| Evan    | E    | Wallis | 222222200 | 1958-01-16 | 134 Pelham, Milwaukee, WI     | M    | 92000 | null     |    7 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
| Josh    | U    | Zell   | 222222201 | 1954-05-22 | 266 McGrady, Milwaukee, WI    | M    | 56000 | 222222200 |    7 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
| Andy    | C    | Vile   | 222222202 | 1944-06-21 | 1967 Jordan, Milwaukee, WI    | M    | 53000 | 222222200 |    7 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
| Tom     | G    | Brand  | 222222203 | 1966-12-16 | 112 Third St, Milwaukee, WI   | M    | 62500 | 222222200 |    7 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
```

Figure 16- Loading data to EMPLOYEE table

```
mysql> LOAD DATA INFILE '/var/lib/mysql-files/dependent.dat'
    -> INTO TABLE DEPENDENT
    -> FIELDS TERMINATED BY ','
    -> OPTIONALLY ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> (essn, dependent_name, sex, bdate, relationship);
Query OK, 11 rows affected (0.00 sec)
Records: 11  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from DEPENDENT;
+-----------+----------------+------+------------+--------------+
| essn      | dependent_name | sex  | bdate      | relationship |
+-----------+----------------+------+------------+--------------+
| 123456789 | Alice          | F    | 1978-12-31 | Daughter     |
| 123456789 | Elizabeth      | F    | 1988-11-21 | Spouse       |
| 123456789 | Michael        | M    | 1978-01-01 | Son          |
| 333445555 | Alice          | F    | 1976-04-05 | Daughter     |
| 333445555 | Joy            | F    | 1948-05-03 | Spouse       |
| 333445555 | Theodore       | M    | 1973-10-25 | Son          |
| 444444400 | Johnny         | M    | 1997-04-04 | Son          |
| 444444400 | Tommy          | M    | 1999-06-07 | Son          |
| 444444401 | Chris          | M    | 1969-04-19 | Spouse       |
| 444444402 | Sam            | M    | 1964-02-14 | Spouse       |
| 987654321 | Abner          | M    | 1932-02-29 | Spouse       |
+-----------+----------------+------+------------+--------------+
11 rows in set (0.00 sec)
```

Figure 17- Loading data to DEPENDENT table

```
mysql> LOAD DATA INFILE '/var/lib/mysql-files/worksOn.dat'
    -> INTO TABLE WORKS_ON
    -> FIELDS TERMINATED BY ','
    -> OPTIONALLY ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> (essn, pno, hours);
Query OK, 48 rows affected (0.01 sec)
Records: 48  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from WORKS_ON;
+-----------+-----+-------+
| essn      | pno | hours |
+-----------+-----+-------+
| 111111100 |  61 | 40.00 |
| 111111101 |  61 | 40.00 |
| 111111102 |  61 | 40.00 |
| 111111103 |  61 | 40.00 |
| 123456789 |   1 | 32.50 |
| 123456789 |   2 |  7.50 |
| 222222200 |  62 | 40.00 |
| 222222201 |  62 | 48.00 |
| 222222202 |  62 | 40.00 |
| 222222203 |  62 | 40.00 |
| 222222204 |  62 | 40.00 |
| 222222205 |  62 | 40.00 |
| 333333300 |  63 | 40.00 |
```

Figure 18- Loading data to WORKS_ON table

4. Creating PLSQL for my company database

a. We created a Procedure to get all the projects of a particular Department

```
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE get_department_projects (
    ->     IN dept_no BIGINT
    -> )
    -> BEGIN
    ->     SELECT pnumber, pname, plocation
    ->     FROM PROJECT
    ->     WHERE dnum = dept_no;
    -> END$$
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql>
mysql> CALL get_department_projects(4)
    -> ;
+---------+-----------------+-----------+
| pnumber | pname           | plocation |
+---------+-----------------+-----------+
|      10 | Computerization | Stafford  |
|      30 | Newbenefits     | Stafford  |
+---------+-----------------+-----------+
2 rows in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

Figure 19- Procedure to get Projects from dno

b. We also created a Procedure to update Employee salary on a certain percentage of a particular Department.

```
| Jennifer | S     | Wallace | 987654321 | 1931-06-20 | 291 Berry, Bellaire, TX          | F    |  43000 | 888665555 |    4 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
| Ahmad    | V     | Jabbar  | 987987987 | 1959-03-29 | 980 Dallas, Houston, TX          | M    |  25000 | 987654321 |    4 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
| Alicia   | J     | Zelaya  | 999887777 | 1958-07-19 | 3321 Castle, Spring, TX          | F    |  25000 | 987654321 |    4 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
+----------+-------+---------+-----------+------------+----------------------------------+------+--------+-----------+------+---------------------+---------------------+
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE increase_department_salary (
    ->     IN dept_no BIGINT,
    ->     IN percentage DECIMAL(5,2)
    -> )
    -> BEGIN
    ->     DECLARE EXIT HANDLER FOR SQLEXCEPTION
    ->     BEGIN
    ->         SELECT 'Error in increasing salary' AS message;
    ->     END;
    ->
    ->     UPDATE EMPLOYEE
    ->     SET salary = salary + (salary * (percentage / 100))
    ->     WHERE dno = dept_no;
    ->
    ->     SELECT CONCAT('Salary updated for department: ', dept_no) AS message;
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql> CALL increase_department_salary(4,10);
+----------------------------------+
| message                          |
+----------------------------------+
| Salary updated for department: 4 |
+----------------------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> select * from EMPLOYEE where dno=4;
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+------+---------------------+---------------------+
| fname    | minit | lname   | ssn       | bdate      | address                 | sex  | salary | superssn  | dno  | createdAt           | updatedAt           |
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+------+---------------------+---------------------+
| Jennifer | S     | Wallace | 987654321 | 1931-06-20 | 291 Berry, Bellaire, TX | F    |  47300 | 888665555 |    4 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
| Ahmad    | V     | Jabbar  | 987987987 | 1959-03-29 | 980 Dallas, Houston, TX | M    |  27500 | 987654321 |    4 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
| Alicia   | J     | Zelaya  | 999887777 | 1958-07-19 | 3321 Castle, Spring, TX | F    |  27500 | 987654321 |    4 | 2024-10-31 22:19:36 | 2024-10-31 22:19:36 |
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+------+---------------------+---------------------+
3 rows in set (0.00 sec)

mysql>
```

Figure 20 – Procedure to increase salary

c. Also created a procedure to count the number of employees in a department.

```
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE count_emp (
    ->     IN dept_no BIGINT
    -> )
    -> BEGIN
    ->     SELECT COUNT(*) AS emp_count
    ->     FROM EMPLOYEE
    ->     WHERE dno = dept_no;
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql>
mysql>
mysql> CALL count_emp(6);
+-----------+
| emp_count |
+-----------+
|         8 |
+-----------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

Figure 21 – Procedure for employee count

5. Relationships

Following is the relation of these tables with one another:

DEPARTMENT – EMPLOYEE

The DEPARTMENT to EMPLOYEE relationships can be defined as one to many since each department can have many employees under it. In the EMPLOYEE table, the foreign key dno references the dnumber in the DEPARTMENT table. This type of relation is useful in linking employees with the respective departments they belong to.

DEPARTMENT and PROJECT

As departments can contain a number of projects, hence the DEPARTMENT and PROJECT do have one to many relationships. In the PROJECT table the dnum column is a foreign key to dnumber in the DEPARTMENT table indicating the department in charge of the project.

EMPLOYEE and PROJECT (WORKS_ON)

The WORKS_ON associative entity depicts the many to many relationships of employees working on the projects. In WORKS_ON, employee and project are related through each entry. Essentially, employee subordinates through essn which is the social security number to ssn in EMPLOYEE and project through pno which is project number to pnumber in PROJECT. This enables the recording of hours an employee spends per project.

DEPARTMENT and DEPT_LOCATIONS

It is common for departments to have more than one location that results to a one to many relations between DEPARTMENT and DEPT_LOCATIONS. In this case, the dnumber field in DEPT LOCATIONS that acts as a foreign key is linked to dnumber which belongs to DEPARTMENT letting the departments be attached with different sites.

EMPLOYEE and DEPENDENT

As for the employee having dependents, he will be having a one-to-many relation with the EMPLOYEE and DEPENDENT respectively. The foreign key essn in the DEPENDANT table is joined to the ssn in EMPLOYEE table giving a relation for every dependant to the employee.

These relations provide important frameworks for arranging the data in a systematic form and managing the records in great details in the database.

6. KNIME for Loading Dataset

We used KNIME to load our datasets to our company database in MySQL

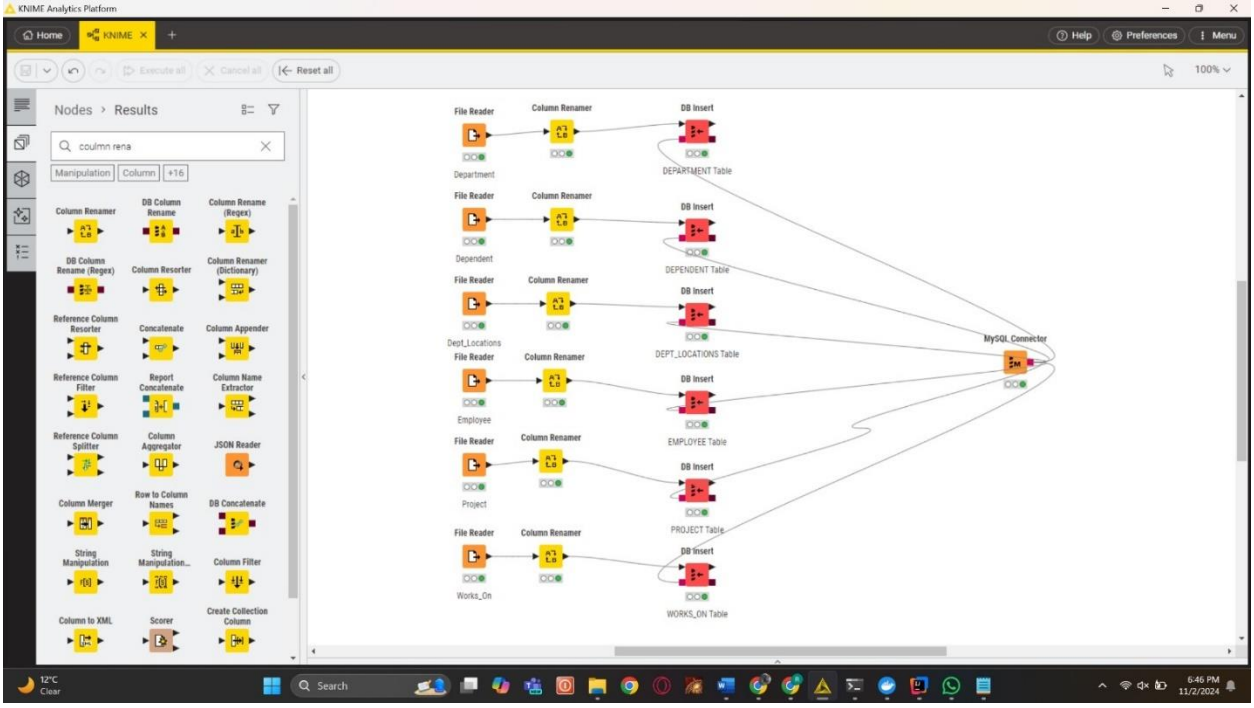

Figure 22 – KNIME Workflow

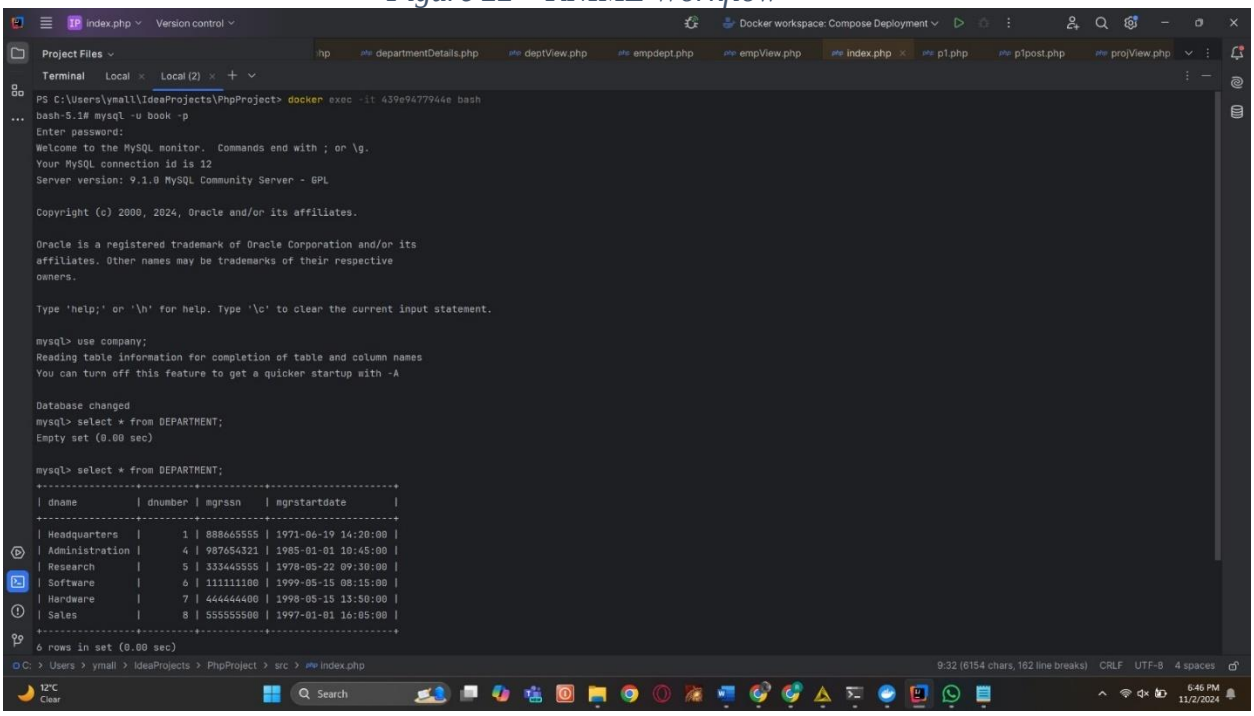

Figure 23 – Data getting added to the database

7. MySQL User Authentication

As a part of this project, user authentication is used to prevent unauthorized access and to avoid overlapping of responsibilities within the database. There are three users of MySQL, namely root, book and ruthvik, none of which is without a role. The root user can be considered an administrative user and can perform unrestricted tasks in the database but is rarely used except for configuration and advanced maintenance purposes. This user is only employed during installation or when system-level changes are required.

The book user's role security is assumed as that role will be used for table creation and the mutations that define the database schema by the er model. Since the role that allows table creation is restricted to book, the project will be able to ensure that there is a clear distinction between schema creation, and data management whereby other users trying to carry out normal operations do not make unintended alterations to the schema.

The ruthvik user, on the other hand, is responsible for loading bulk data it into the tables using the .dat files which do not enable him to affect the structure of the database even though the privileges are limited. This method ensures that database access is defined, and secure by role with an implication of no unauthorized access. It offers a reliable way of performing database operations which mostly try to separate various degrees of loading data into the database and structuring the database.

III. WEB APPLICATION AND USER INTERFACES

The User Interface (UI) pertaining to this project aims at enhancing user convenience for easier interaction with the database. The same comprises of four broad options: the Employee Directory which permits searching for employees by selecting the SSNs; the Department Overview which contains detailed information on all programs; Employees Salaries by Department which provides areas of salary by department; and Browse Departments (Company Browse) which fastest expands the range of departments thanks to the presence of hyperlinks to departments pages.

GitHub Link: https://github.com/Ruthvikr01/InfoBridgePro--Database-Management-System

Demo Link: https://www.youtube.com/watch?v=0k1FNOR7WXk

1. Employee Directory

Employee Directory, for the directory employees, ssn distributions and other records are more than sufficient to assist them in the efficient gathering of information. As there are numerous people to choose from, employees can select an employee from the menu and view their name, location, and employment position. In this case, the employee data management feature is enhanced.

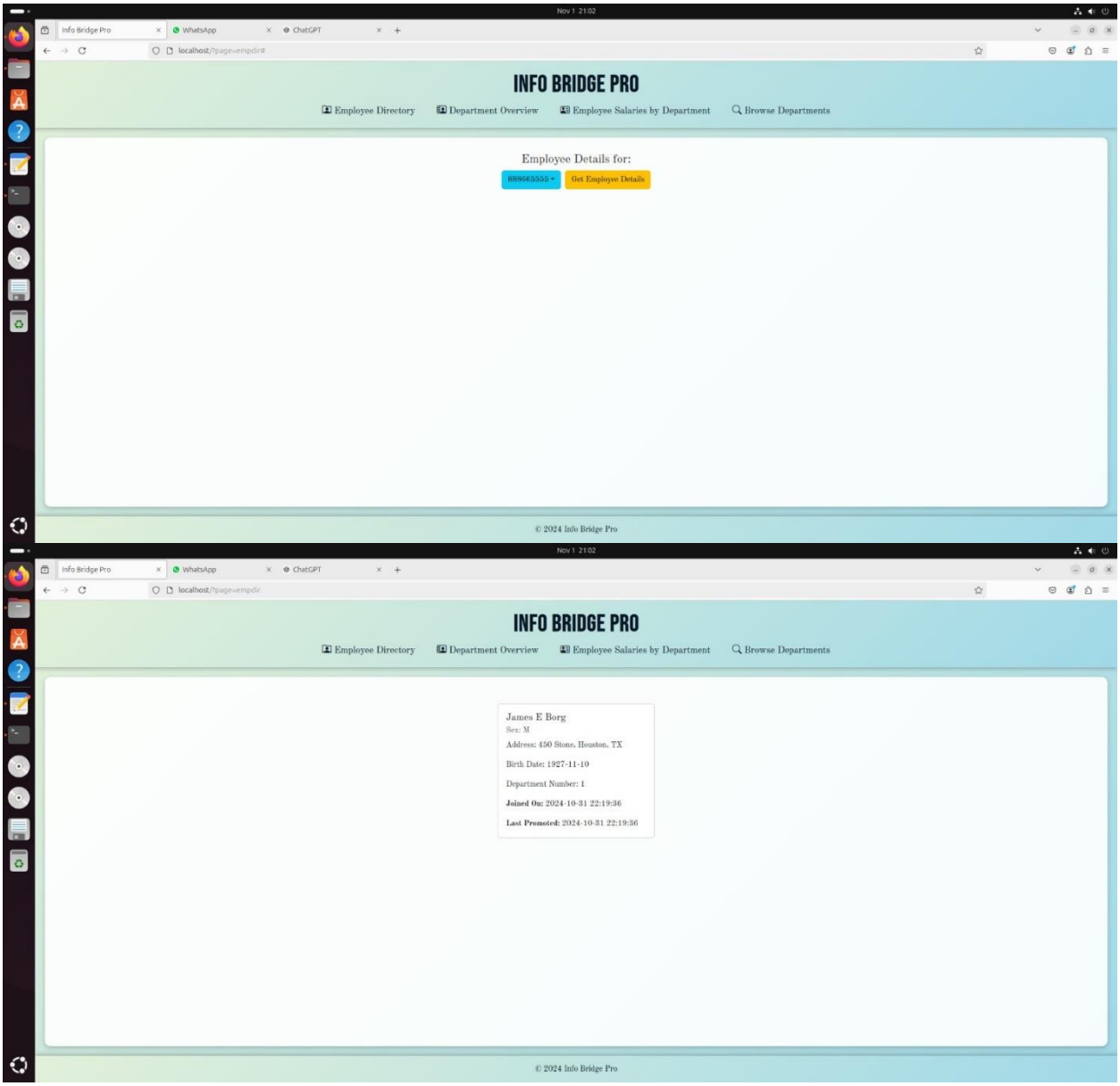

Figure 24 – Employee Directory Page

2. Department Overview

The Department Overview incorporates the department and manager details, including the department title, representative with SSN, and employment date with the manager. The presentation is information rich standardized promoting ease of use to the targeted audience as it

encapsulates all managerial and managerial timelines under a single box thereby sparing the audience's attention to other issues. It is quite beneficial in determining and evaluating the possible organization structure and the hierarchy of the departments in the company.

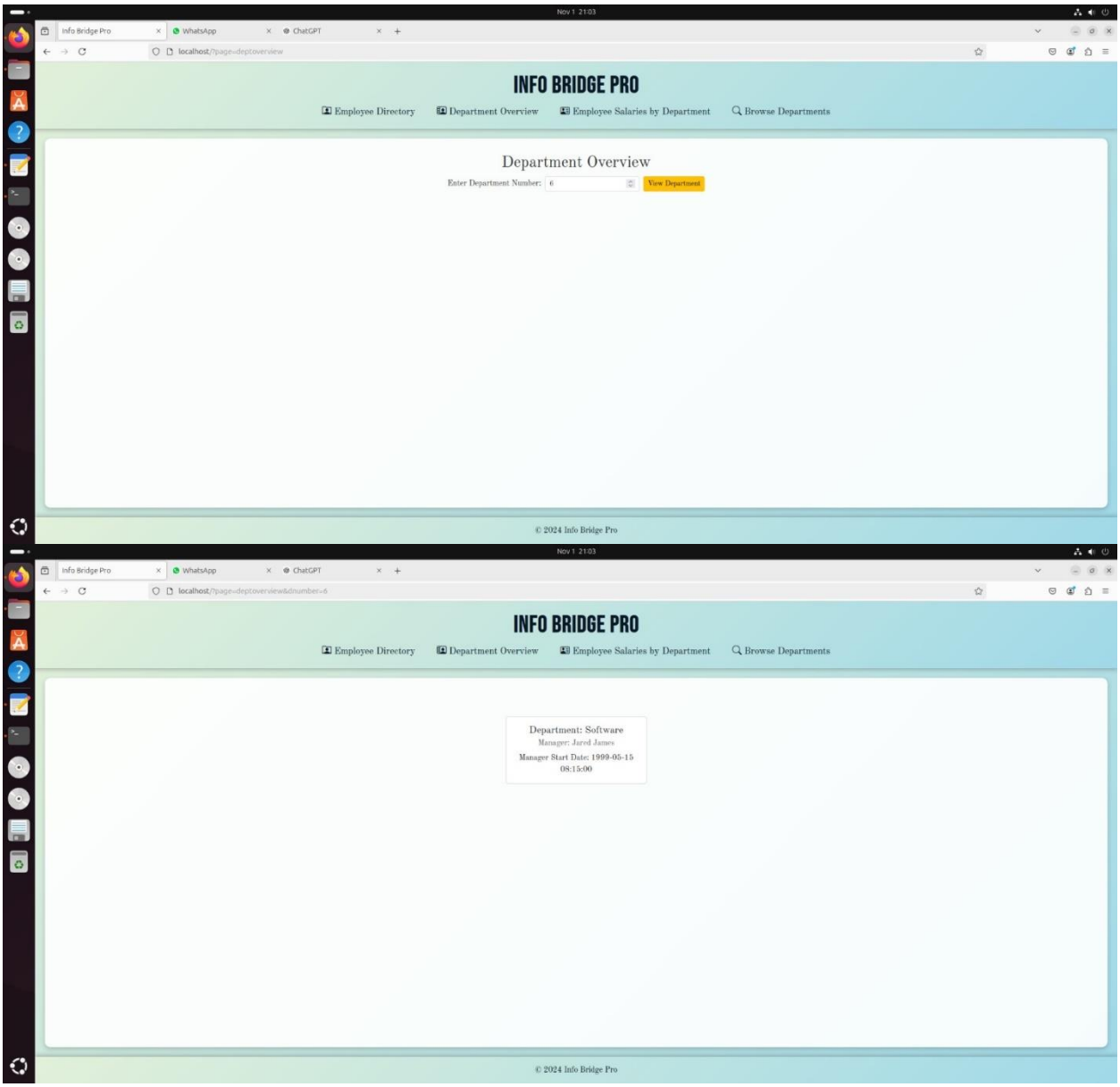

Figure 25 – Department Overview

3. Employee Salaries by Department

This volume groups the employees and their specific salaries in their respective departments. Department numbers or relevant filters can be used to locate department names and

their corresponding number of employees and their salaries. This is a helpful function for providing information on the number of employees within the department and the funds that are expected to be used within the specific departments which can assist in finance and HR functions such as determination of payroll for a whole department.

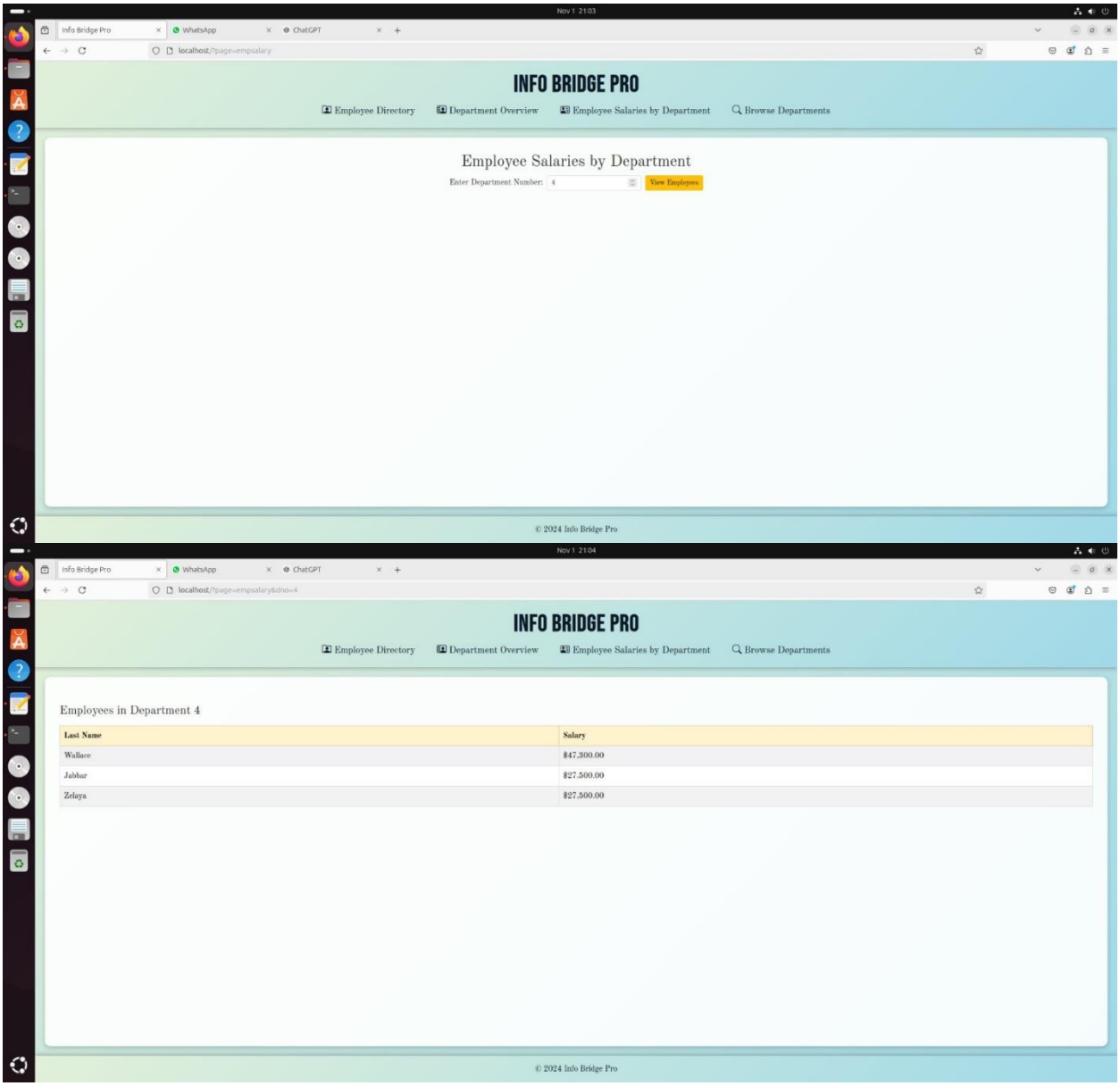

Figure 26 - Employee Salaries by Department

4. Browse Departments (Company Browse)

The Browse Departments feature allows the user to go through the list of departments within the company. Clicking any of the links showing the number of the department takes him to the detailed view of the information concerning the respective department. This feature gives an easy way of exploring the organizational structure for quick access to department-specific data, enhancing navigation and data accessibility for its users.

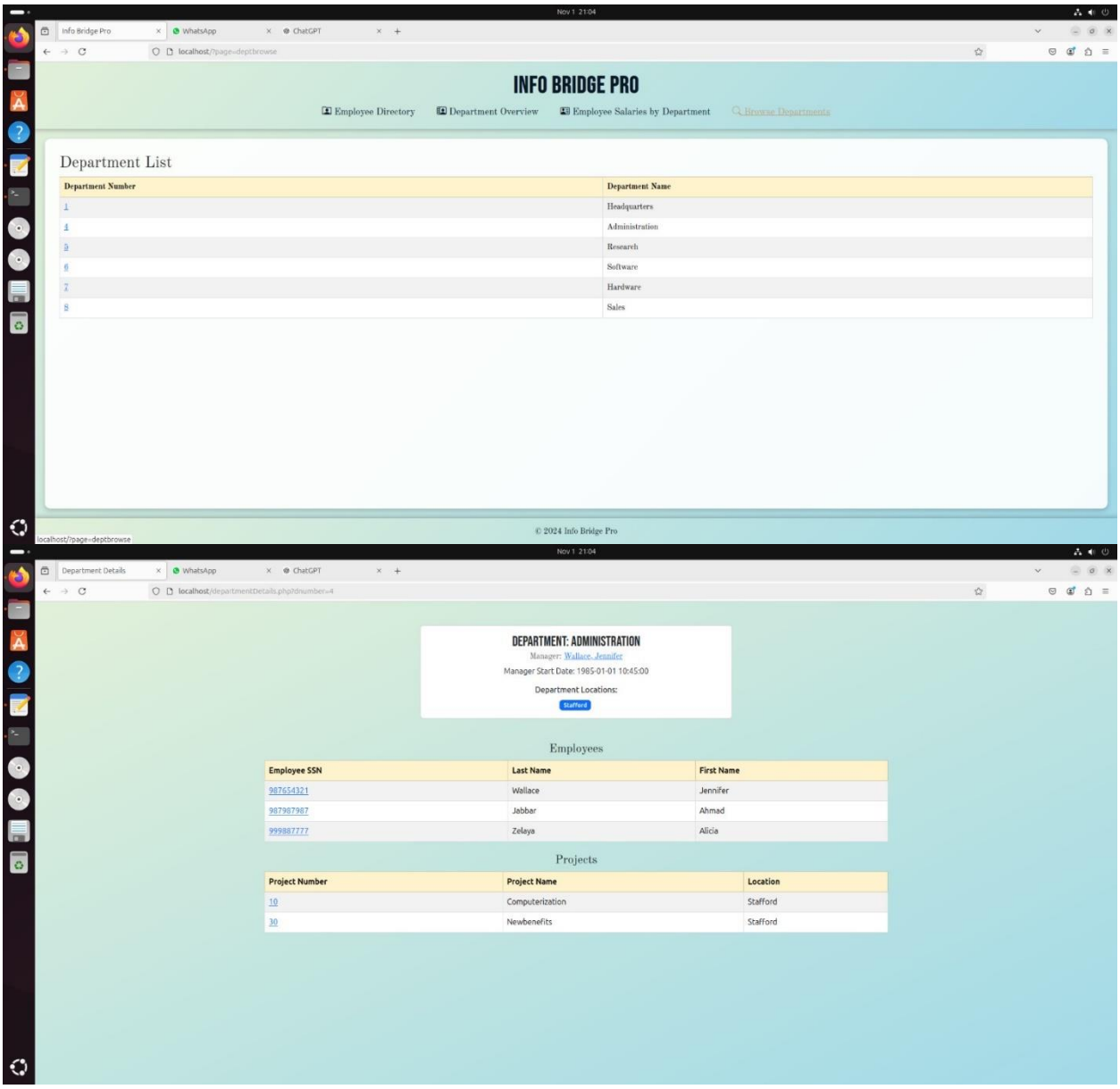

Figure 27 – Browse Departments (Company Browse)

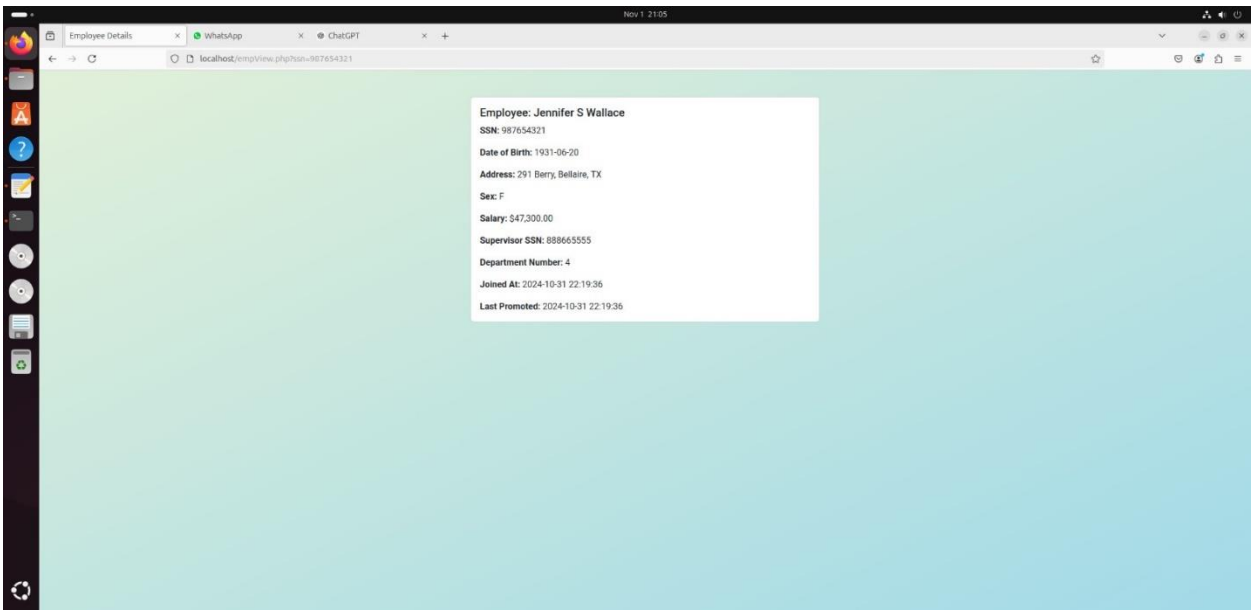

Figure 28 – Employee Details in Browse Departments (Company Browse)

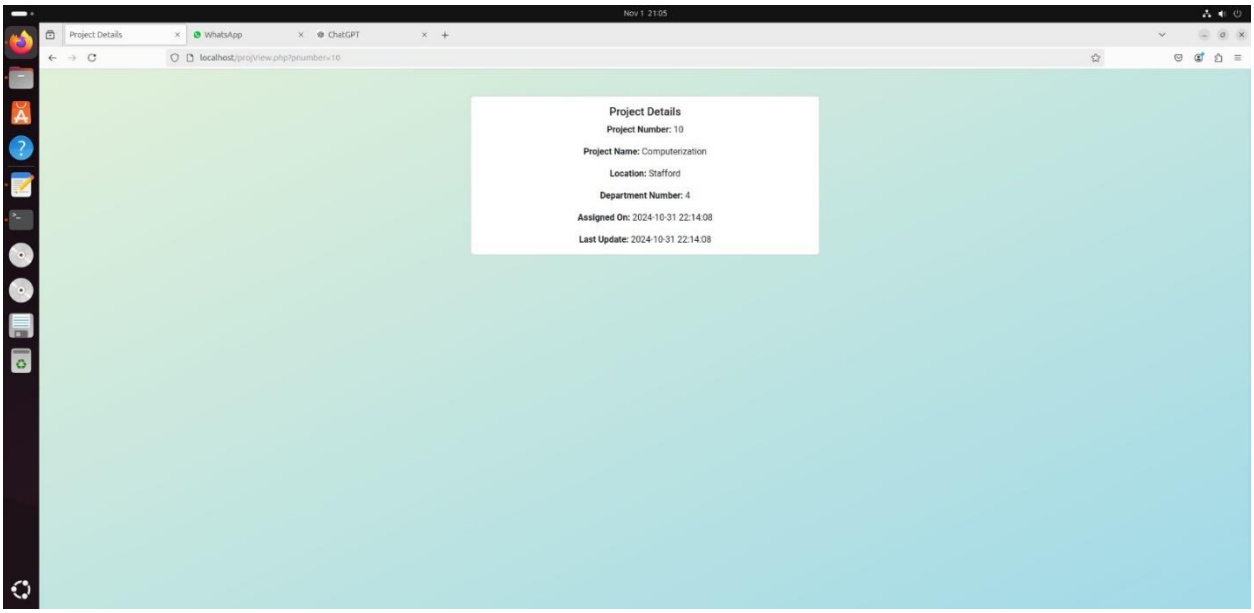

Figure 29 –Project Details in Browse Departments (Company Browse)

IV. KEY FUNCTIONALITIES

The project implements some key functionalities for easy management of company-associated data:

- Employee Information Retrieval:

A drop-down menu selection on Social Security Number will allow the user to fetch detailed information about an employee. The user gets to know the quick summary about an employee, first name, middle initial and the last name to retrieve the information hassle-free.

- Department Overview:

Browse Departments allows viewing of all the company's departments. Clicking any of the department number’s links to the detailed view of the department's information. This enhances the navigational aspect, and the aspect of data access allows a user to view the organization structure and quickly obtain the relevant information related to any department.

- Employee Details and Salary Information, Department-wise:

This will display to the user a filtered list of employees, listing last name and salary by department number-dno-that will enable the end user to evaluate the salaries of employees in a particular department for financial analysis and reporting.

- User Experience and Accessibility:

The design of the web application is based on end-user involvement and navigation. Besides, it has two MySQL users who have different privileges in maintaining the functionalities of the database. It is containerized in Docker, hence assurance of consistency and scalability of the deployment environment. With this attention to accessibility, besides containerization, enhances the user experience such that one is able to find the required information without wasting time exploring the various functionalities of the application.

V. DISCUSSION

This is my Team Project 2, InfoBridgePro, which is a database management system involving multiple technologies to gather and store company data with ease. This particular project incorporated practical skills and integration of several tools and technologies including (1) creation of MySQL on the Docker containers, (2) creation of a PHP application for user and business logic system boundaries, (3) using HTML for user interface and (4) creating Apache server for internet hosting purposes.

To operate the InfoBridgePro application successfully, users have to adhere to certain requirements. The following are the prerequisites for the successful operation of this application:

Docker containerization has to be installed on the system for easier containment of the MySQL database. There is a need to install PHP of version 7.4 or higher in order for the application code to be supported. It is important that the Apache Server be installed in order to perform the required PHP application and access the internet for the web interface. There is an urgent need to access MySQL but within a Docker container for the purpose of database control and management. Every task within every given phase of the project ensures that the team appreciates the whole process of designing the system, integrating to the system and database management. The project called for teamwork for both good and bad aspects presented and improved our web application and database management.

1. Challenges Faced

The process of creating InfoBridgePro was not devoid of complications, especially in integrating technology and error handling. The error ‘Cannot declare class DbConnect’ was one such prominent problem which was solved by making each PHP document to include

DbConnect.php just once. This existence has emphasized the relevance of file handling of PHP files in order to avoid instances of conflicts that may hinder the operations of the application.

Because of the use of the obsolete PHP functions, we ran into difficulties in the first part. To promote sustainability and compatibility of the system, we had to propose a change from the obsolete functions to the modern functions that are PDO (PHP Data Objects which helped to integrate with MySQL). Apart from better code quality, this transition also improves the security and the performance of database operations.

By containerizing the project with Docker, the MySQL database setup was made easier. The setting up of the Docker environment was done in a systematic manner in the docker-compose.yml file, which included port settings, volume settings and environment variables settings. This method guaranteed that MySQL and the PHP application would work together without any challenges at all since there was a uniform and stable environment throughout the different levels of development.

2. Performance Considerations

Performance remained a point of focus throughout the design and the building of InfoBridgePro, particularly in view of the number of users who are often expected to access and interact with the application concurrently. In an attempt to reduce the response time when performing some routine operations like searching for employee or department information, key fields in the MySQL database were provided with indexes. Other strategies were also used such as the reduction of the number of nested queries, the effectiveness of joins and the use of query enhancement techniques. Sustained and timely performance profiling pinpointed weaknesses that resulted in optimizations, which reduced inexpensive query execution and improved database connection handling. There were also approaches in caching that were applied in order to reduce

response times at peak usage levels to make it easier for the system to handle heavy request volumes. Efforts enabled the acquisition of important practical lessons on the design and implementation of the system and on the management of the database, while promoting teamwork in addressing issues. The practical work with PHP, MySQL and Docker improved greatly the installation and operation of the database management system, which led to better user satisfaction and data retrieval.

VI. CONCLUSION

In this project, we created a Web Application InfoBridgePro, a complete database management system that is designed specifically to make the management of company related information using the PHP and MySQL, more efficient. The project had several stages which revolved around, establishing a database and its structure as well as building it itself, authentication of the end users and making it possible for them to obtain relevant information through a web-based interface. Team members worked independently on different tasks according to their strengths to ensure that every aspect of the project was managed satisfactorily.

To complete the project, we made sure that every person involved has important tools and software such as Docker for MySQL containerization and PHP development environments for the development of the application, at hand. This deliberate decision enabled us to adapt certain factors to enhance the users and the design of the database to be more appealing.

It was apparent for this project that several people had to work together to achieve the objectives. Database related issues along with data consistency and data integrity are problems that were solved within the group and are also a clear demonstration of the role of cooperation on intricate assignments. In the final analysis, instead only focused on rounding out the technical skills, but also broadened our skills on working with other people and meeting new requirements

Acknowledge

I acknowledge the invaluable guidance and support of my professor, whose expertise and encouragement were instrumental in completing this assignment. Special thanks to my institution for providing the necessary resources and environment to carry out this work successfully.

REFERENCES

1. Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of database systems* (7th ed.). Pearson.
2. Ullman, L. (2017). *PHP and MySQL Web Development* (4th ed.). Addison-Wesley.
3. Duvall, P. (2016). *PHP for the web: Visual QuickStart Guide* (4th ed.). Peachpit Press.
4. McCool, M. (2020). *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5* (4th ed.). O'Reilly Media.
5. Becker, J., & Knackstedt, R. (2019). *Database management systems: A practical approach*. Springer.
6. Powers, B. (2018). *Learning PHP 7: A practical guide to PHP programming*. Packt Publishing.
7. Schwartz, R., Zandstra, M., & Smith, J. (2016). *PHP 7 for absolute beginners*. Apress.
8. Davidson, S., & Lyman, T. (2018). *Learning PHP, MySQL, JavaScript, and CSS* (4th ed.). O'Reilly Media.
9. W3Schools. (n.d.). *PHP Tutorial*. Retrieved from https://www.w3schools.com/php /
10. MySQL. (n.d.). *MySQL 8.0 Reference Manual*. Retrieved from https://dev.mysql.com/doc/refman/8.0/en/
11. Docker. (n.d.). *Get started with Docker*. Retrieved from https://docs.docker.com/get-started/
12. Rouse, M. (2020). What is Docker? *TechTarget*. Retrieved from https://www.techtarget.com/whatis/definition/Docker
13. Docker, Inc. (n.d.). *Docker for beginners*. Retrieved from https://www.docker.com/101-tutorial
14. Pahl, C. (2015). Containerization and the role of Docker in cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, 4(1), 1-6.
15. McKinsey & Company. (2019). *The state of AI in business: Data-driven insights from experts and leaders*. Retrieved from https://www.mckinsey.com/featured-insights/artificial-intelligence
16. Kuhlman, T. (2017). *Docker for PHP developers*. Packt Publishing.
17. Beighley, L., & Morrison, M. (2010). *Headfirst PHP & MySQL*. O'Reilly Media.
18. Hogue, C. (2017). *PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide* (4th ed.). Peachpit Press.
19. Gorton, I. (2019). *The Essential Guide to PHP and MySQL: Web Development Fundamentals*. CreateSpace Independent Publishing Platform.
20. O'Reilly Media. (2020). *Docker: Up & Running: Shipping Reliable Containers in Production* (2nd ed.). O'Reilly Media.

Appendix

1. Creating tables for the Entities

```
 CREATE TABLE DEPARTMENT (
       dname varchar(50) not null,
       dnumber bigint,
       mgrssn char(9) not null,
       mgrstartdate timestamp,
       primary key (dnumber),
       key (dname)
     );


CREATE TABLE PROJECT (
  pname      varchar(50) not null,
  pnumber    bigint,
  plocation  varchar(255),
  dnum       bigint not null,
  createdAt  timestamp,
  updatedAt  timestamp,
  primary key (pnumber),
  unique (pname),
  foreign key (dnum) references DEPARTMENT(dnumber)
);

CREATE TABLE DEPT_LOCATIONS (
  dnumber   bigint,
  dlocation varchar(255),
  primary key (dnumber,dlocation),
  foreign key (dnumber) references DEPARTMENT(dnumber)
);

CREATE TABLE EMPLOYEE (
  fname    varchar(50) not null,
  minit    varchar(1),
  lname    varchar(50) not null,
  ssn      char(9),
  bdate    date,
  address  varchar(255),
  sex      char,
  salary   int,
  superssn char(9),
  dno      bigint,
  createdAt timestamp,
```

```
  updatedAt timestamp,
  primary key (ssn),
  foreign key (dno) references DEPARTMENT(dnumber)
);

CREATE TABLE DEPENDENT (
  essn           char(9),
  dependent_name varchar(50),
  sex            char,
  bdate          date,
  relationship   varchar(20),
  primary key (essn,dependent_name),
  foreign key (essn) references EMPLOYEE(ssn)
);

CREATE TABLE WORKS_ON (
  essn   char(9),
  pno    bigint,
  hours  decimal(4,2),
  primary key (essn,pno),
  foreign key (essn) references EMPLOYEE(ssn),
  foreign key (pno) references PROJECT(pnumber)
);
```

2. Index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Company Web Application</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">

    <style>
        body { font-family: Arial, sans-serif; }
        header { background: #333; color: white; padding: 10px; text-align: center; }
        nav a { color: white; margin: 0 15px; text-decoration: none; }
        .content { margin: 20px; }
        footer { background: #333; color: white; text-align: center; padding: 10px; position: fixed;
bottom: 0; width: 100%; }
    </style>
</head>
<body>
```

```
<header>
   <h1>Company Database Management</h1>
   <nav>
      <a href="?page=empdir">Employee Directory</a>
      <a href="?page=deptoverview">Department Overview</a>
      <a href="?page=empsalary">Employee Salaries by Department</a>
      <a href="?page=deptbrowse">Browse Departments</a>
   </nav>
</header>

<div class="content">
   <?php
   // Include the database connection file
   include_once "DbConnect.php";
   $db = new DbConnect();
   $conn = $db->connect();

   // Determine which page to show based on the selected menu item
   $page = isset($_GET['page']) ? $_GET['page'] : 'empdir';
   switch ($page) {
      case 'empdir':
         // Employee Directory
         if ($_SERVER['REQUEST_METHOD'] == 'POST') {
            include "p1post.php";  // Process the selected SSN and show details
         } else {
            include "p1.php";  // Show the SSN dropdown
         }
         break;

      case 'deptoverview':
         // Department Overview
         // Check if dnumber is set in the URL
         if (isset($_GET['dnumber'])) {
            include "deptView.php";  // Include deptView.php to show details
         } else {
            // Display form to enter department number
            echo '<h2>Department Overview</h2>';
            echo '<form method="GET" action="">';
            echo '    <input type="hidden" name="page" value="deptoverview">';
            echo '    <label for="dnumber">Enter Department Number:</label>';
            echo '    <input type="number" name="dnumber" min="1" required>';
            echo '     <button type="submit" class="btn btn-warning">View Department</button>';
            echo '</form>';
         }
         break;
```

```
        case 'empsalary':
            // Employee Salaries by Department
            if(isset($_GET['dno'])){
                include "empdept.php";
            }else{
                echo '<h2>Employee Salaries by Department</h2>';
                echo '<form method="GET" action="">';
                echo '    <input type="hidden" name="page" value="empsalary">';
                echo '    <label for="dno">Enter Department Number:</label>';
                echo '    <input type="number" name="dno" min="1" required>';
                echo '   <button type="submit" class="btn btn-warning">View Employees</button>';
                echo '</form>';
            }
            break;

        case 'deptbrowse':
            // Browse Departments
            include "companyBrowse.php";
            break;

        default:
            echo "<h2>Page Not Found</h2><p>The page you're looking for doesn't exist.</p>";
    }
    ?>
</div>

<footer>
    <p>&copy; <?php echo date("Y"); ?> Company Database Management</p>
</footer>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>

</body>
</html>
```

3. DbConnect.php

```
<?php
class DbConnect {
    private $host = 'db'; // Hostname aligns with the MySQL service name
    private $dbName = 'company';
    private $user = 'root';
    private $pass = 'password';


    public function connect() {
        try {
            $conn = new PDO('mysql:host=' . $this->host . ';dbname=' . $this->dbName, $this->user,
$this->pass);
            $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            return $conn;
        } catch(PDOException $e) {
            echo 'Database Error: ' . $e->getMessage();
        }
    }
}
?>


</project>
```

4. docker-compose.yml

```
 version: '3.7'
services:
  web:
    build: .
    volumes:
      - ./src:/var/www/html
    ports:
      - 80:80
    depends_on:
      - db

  db:
    image: mysql:8.0 # Specify a version to ensure compatibility
    volumes:
      - db_data:/var/lib/mysql # Use a named volume instead of a bind mount
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: company
    ports:
```

```
    - 3306:3306
  restart: unless-stopped # Ensure it restarts in case of failures

volumes:
  db_data:
```

5. DockerFile

```
FROM php:7.4-apache
COPY src/ /var/www/html
RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf
RUN docker-php-ext-install pdo pdo_mysql
EXPOSE 90
```