InfoBridgeMax: Project Report

CS51550-001, Fall 2024
PNW University
12/04/2024

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

Table of Contents

Abstract 3
I. Introduction 4
II. SYSTEM DESIGN AND IMPLEMENTATION 6
1. ER DIAGRAM 7
2. Connecting PHP Application to Dockerized Redis 8
3. Database Schema 9
4. Relationships 16
5. Optimizing Data Retrieval with Redis 18
III. Web Application and USER INTERFACES 19
1. Employee Directory 19
2. Department Overview 20
3. Employee Salaries by Department 21
4. Browse Departments (Company Browse) 23
IV. Key Functionalities 25
V. Discussion 26
1. Challenges Faced 26
2. Performance Considerations 27
VI. Conclusion 28
Acknowledge 28
References 29
Appendix 30
1. Loading .dat files to Redis(load_redis.php) 30
2. Index.php 35
3. RedisConnect.php 39
4. docker-compose.yml 39
5. DockerFile 39

Table of Figures

Figure 1 - E.R Diagram 7
Figure 2 - Establishing the connection 8
Figure 3 - Docker Container Status 9
Figure 4 – Keys in the Database 9
Figure 5 - DEPARTMENT Key 10
Figure 6 – DEPENDENT Key 10
Figure 7 – PROJECT Key 11
Figure 8 – DEPT_LOCATIONS Key 11
Figure 9 – EMPLOYEE Key 12
Figure 10 – WORKS_ON Key 12
Figure 11 –Loaded data in Department Key 13
Figure 12 – Loaded data in PROJECT Key 13
Figure 13 – Loaded data in DEPT_LOCATIONS Key 14
Figure 14 – Loaded data in EMPLOYEE Key 14
Figure 15 - Loaded data in DEPENDENT Key 15
Figure 16 - Loaded data in WORKS_ON Key 15
Figure 17 - Employee Directory Page 20
Figure 18 - Department Overview 21
Figure 19 - Employee Salaries by Department 22
Figure 20 - Browse Departments (Company Browse) 23
Figure 21 - Employee Details in Browse Departments (Company Browse) 24
Figure 22 - Project Details in Browse Departments (Company Browse) 23

ABSTRACT

This project demonstrates the integration of a Redis Database Management System with PHP for managing and presenting the COMPANY database efficiently. Its primary aim is to enable users to interact with database entities such as departments, projects, and employees through a user-friendly web interface. The technology stack includes Redis for data storage, PHP as the server-side language, Docker for containerization, HTML for constructing web pages, and Apache as the web server deployed on a Windows platform.

The Redis installation in Docker ensures fast data retrieval and management. PHP scripts are used to load ".dat" files into Redis, providing a streamlined data handling process. The application offers various features, including an SSN (Social Security Number) dropdown for selecting employees, detailed department views, and filtered lists of employees by department. The project emphasizes usability and graphical presentation, focusing on smooth navigation and clear data representation. This project serves as a practical demonstration of database administration using Redis while 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 in which they operate, must actively manage their data to remain competitive and efficient. Central to modern data management are database systems that bring order to what could otherwise be chaotic and inconsistent data storage. Database management systems (DBMS) ensure the coherent storage, retrieval, and processing of information while offering features such as data integrity, protection, and availability. Traditional systems relying on flat files or spreadsheets often lead to data redundancy and inconsistencies, challenges that are effectively resolved with the adoption of DBMS. Moreover, as organizations increasingly demand faster, scalable, and more flexible solutions, modern systems like Redis have emerged as robust alternatives to traditional relational databases.

Redis, an in-memory data structure store, offers exceptional performance and scalability, making it a popular choice for applications requiring real-time data access. Unlike traditional relational databases, Redis excels in caching, high-speed transactions, and handling dynamic workloads, ensuring quick access and efficient management of data. This project incorporates Redis as the database management system to design and implement the COMPANY database—a client management and information display web application.

The system focuses on managing key objects such as departments, projects, and employees while providing users with simple and efficient access through an intuitive web interface. The application leverages Redis for data storage and retrieval, PHP as the server-side scripting language, Docker for containerization, HTML for structuring web pages, and Apache as the web

server running on Windows. By utilizing Redis, the project ensures fast data operations and scalability, essential for handling dynamic datasets.

Key features of the application include dropdown menus for selecting employee Social Security Numbers (SSNs), detailed department views, and filtered employee listings by department. Redis is loaded with ".dat" files via PHP scripts, ensuring data is quickly accessible while maintaining operational efficiency. To enhance usability, the application focuses on a user-centered design, allowing seamless navigation across functionalities and enabling users to interact effortlessly with the system. By emphasizing ease of access, security, and performance, this project supports organizational goals of effective data management and user satisfaction. The Redis-based system lays a solid foundation for future enhancements, including advanced data analytics and additional security measures, ensuring that the application remains relevant in evolving organizational landscapes.

II. SYSTEM DESIGN AND IMPLEMENTATION

Development of the Redis-based Database Management System commenced with an E.R. diagram that described the relationships among key entities such as departments, employees, and projects. This E.R. diagram served as the foundational blueprint for designing the database structure and guided the overall system architecture. Redis was chosen as the database management system for its exceptional performance, scalability, and ability to handle dynamic data workloads. The database was implemented within a Docker container, ensuring consistency and reliability across various environments.

The application was developed using PHP for server-side scripting, complemented by HTML, CSS for crafting the user interface. These technologies were used to translate the E.R. model into a functional database schema, enabling practical and efficient data operations. Redis was loaded with ".dat" files via PHP scripts, ensuring quick and seamless data accessibility. The coding emphasized creating user-friendly interfaces and intuitive workflows, allowing users to retrieve employee details by their SSNs and access department information effortlessly.

This structured approach was designed to enhance user experience while leveraging Redis's speed and efficiency for data management and retrieval within the COMPANY database. The system streamlines processes involved in managing employees, departments, and projects, providing a robust solution for organizational data needs.

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, we need to know the relationship of each table properly.

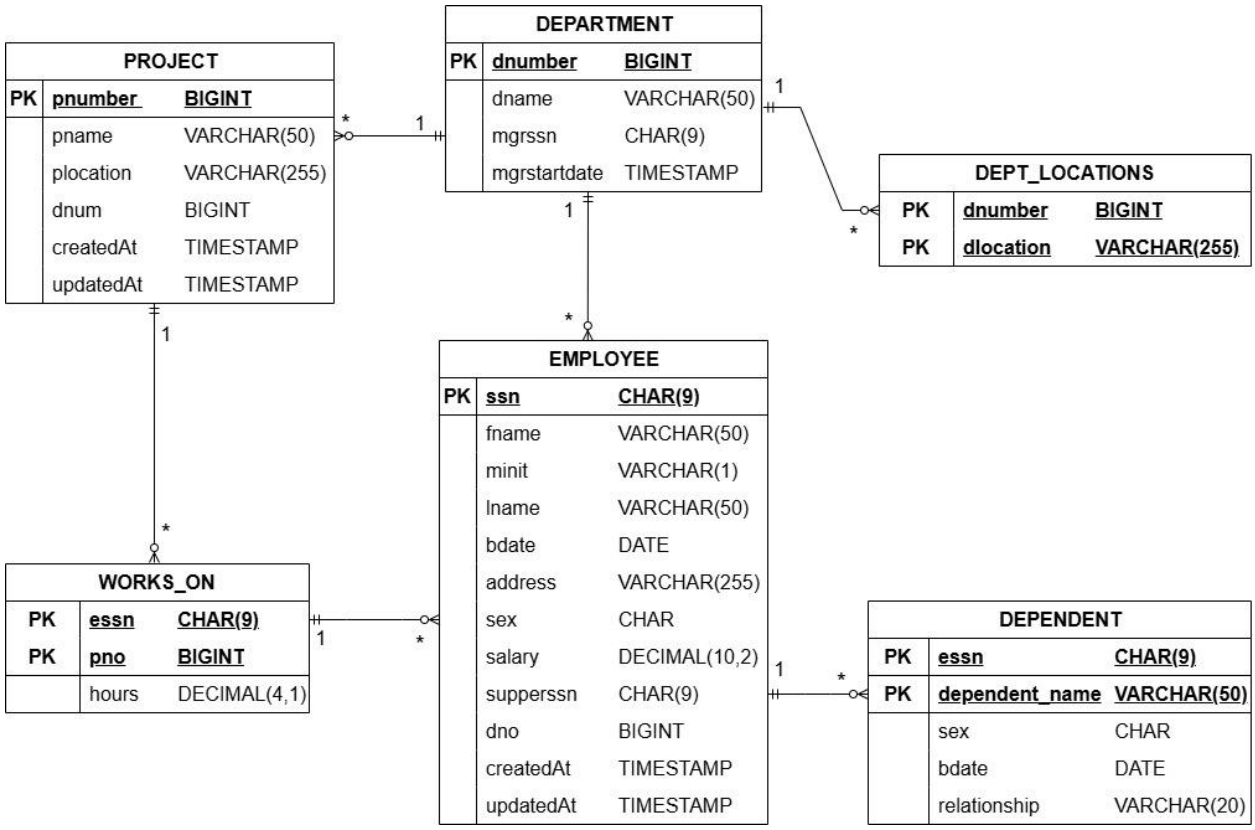

Figure 1 – E.R Diagram

2. Connecting PHP Application to Dockerized Redis

The services defined in this “docker-compose.yml” file include two primary services: the web service, which is for the PHP application, and the redis service, which is for the Redis database. The local Dockerfile builds the web service, setting up the “src” directory to mount on “/var/www/html” within the container. Port 80 is mapped to enable access to the application via “localhost:80”. The Redis service uses the official Redis image and is provisioned for persistent storage using a named volume, “redis_data” The Redis container exposes port 6379, allowing connections from the localhost on the same port for seamless interaction with the application. The “depends_on” directive ensures that the “docker-compose web” service launches only after the “docker-compose redis” service has started, while “restart: unless-stopped” adds failure safety by restarting the services in case of unexpected shutdowns. This setup ensures a robust and reliable environment for the PHP application and Redis integration.

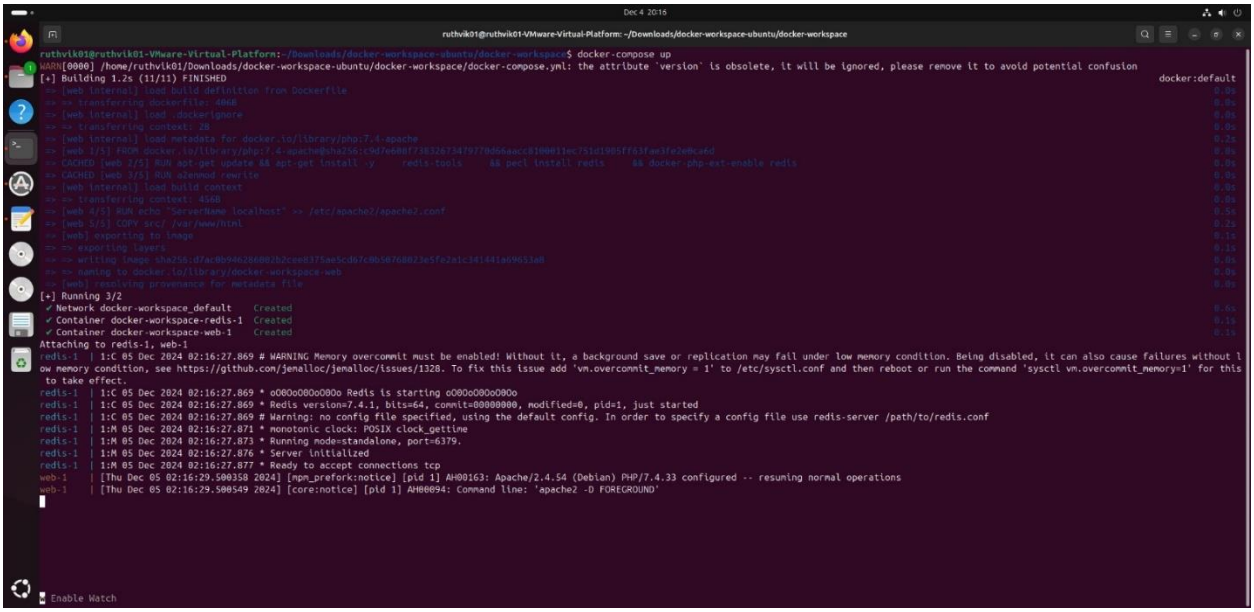

Figure 2 - Establishing the connection

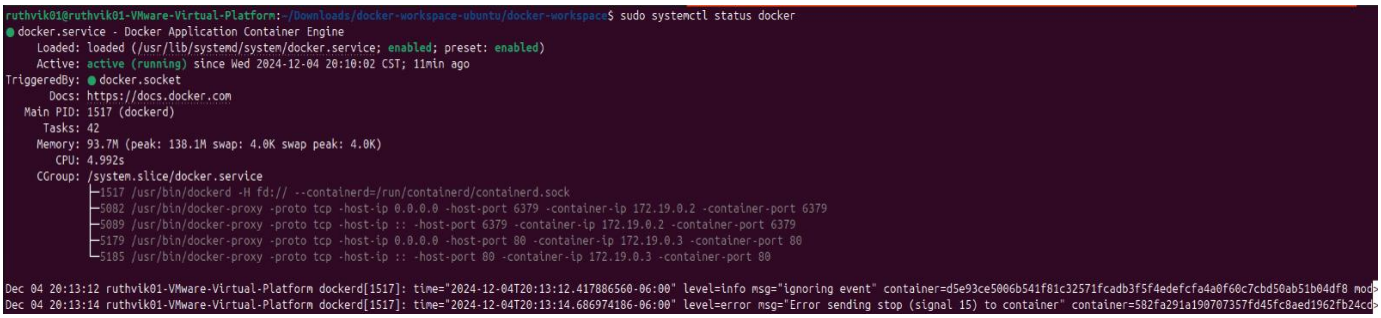

Figure 3 – Docker Container Status

3. Database Schema

a. We configured a Redis database in a Dockerized environment on Linux.I have given the primary keys of the tables in the E.R Diagram as the values for the keys (which are the tables in the E.R Diagram).

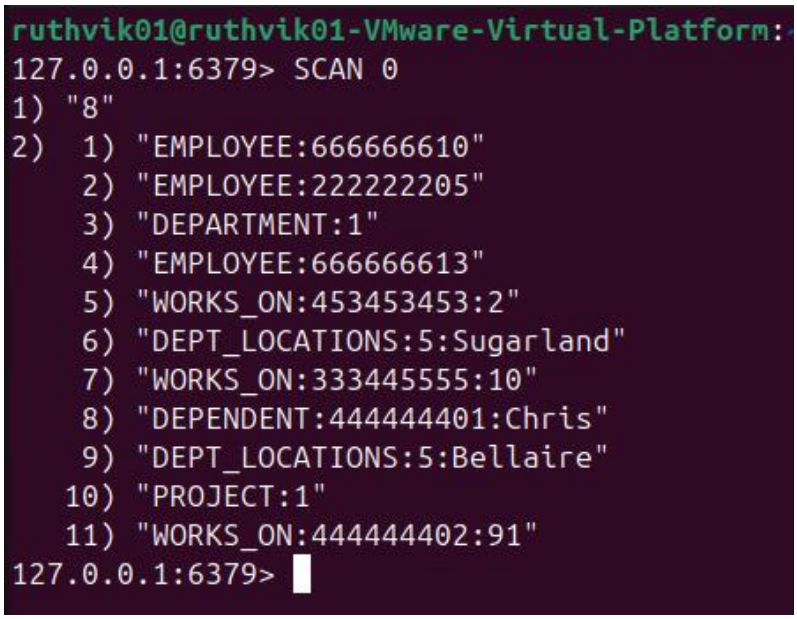

Figure 4 –Keys in the Database

b. DEPARTMENT: This Key includes Department name, Department number, Manager's SSN and Manager StartDate.

```
127.0.0.1:6379> HGETALL DEPARTMENT:1
1) "dname"
2) "Headquarters"
3) "dnumber"
4) "1"
5) "mgrssn"
6) "888665555"
7) "mgrstartdate"
8) "1971-06-19 14:20:00"
127.0.0.1:6379>
```

Figure 5 – DEPARTMENT Key

c. DEPENDENT: This Key is to show the information about the dependents of employees on our database.

```
127.0.0.1:6379> HGETALL DEPENDENT:444444401:Chris
 1) "essn"
 2) "444444401"
 3) "dependent_name"
 4) "Chris"
 5) "sex"
 6) "M"
 7) "bdate"
 8) "1969-04-19"
 9) "relationship"
10) "Spouse"
127.0.0.1:6379>
```

Figure 6 – DEPENDENT Key

d. PROJECT: This Key gives the essential information about Project names and Project locations and when it is created and updated at.

```
127.0.0.1:6379> HGETALL PROJECT:1
 1) "pname"
 2) "ProductX"
 3) "pnumber"
 4) "1"
 5) "plocation"
 6) "Bellaire"
 7) "dnum"
 8) "5"
 9) "createdAt"
10) "2024-12-05 22:24:03"
11) "updatedAt"
12) "2024-12-05 22:24:03"
127.0.0.1:6379>
```

Figure 7 – PROJECT Key

e. DEPT_LOCATIONS: This Key has information about the department's location.

```
127.0.0.1:6379> HGETALL DEPT_LOCATIONS:5:Bellaire
1) "dnumber"
2) "5"
3) "dlocation"
4) "Bellaire"
127.0.0.1:6379>
```

Figure 8 – DEPT_LOCATIONS Key

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

```
127.0.0.1:6379> HGETALL EMPLOYEE:666666610
 1) "fname"
 2) "Naveen"
 3) "minit"
 4) "B"
 5) "lname"
 6) "Drew"
 7) "ssn"
 8) "666666610"
 9) "bdate"
10) "1970-05-23"
11) "address"
12) "198 Elm St, Philadelphia, PA"
13) "sex"
14) "M"
15) "salary"
16) "34000.00"
17) "superssn"
18) "666666607"
19) "dno"
20) "8"
21) "createdAt"
22) "2024-12-05 22:24:03"
23) "updatedAt"
24) "2024-12-05 22:24:03"
127.0.0.1:6379>
```

Figure 9 – EMPLOYEE Key

g. WORKS_ON: This Key shows how many hours an employee is going to work on a project.

```
127.0.0.1:6379> HGETALL WORKS_ON:333445555:10
1) "essn"
2) "333445555"
3) "pno"
4) "10"
5) "hours"
6) "10.0"
127.0.0.1:6379>
```

Figure 10 – WORKS_ON Key

h. Keys in each table(key) after loading bulk data which is in .dat file form into each key or table in Redis.

Figure 11- Loaded data in Department Key

Figure 12- Loaded data in PROJECT Key

```
127.0.0.1:6379> KEYS DEPT_LOCATIONS:*
 1) "DEPT_LOCATIONS:8:Dallas"
 2) "DEPT_LOCATIONS:6:Sacramento"
 3) "DEPT_LOCATIONS:4:Stafford"
 4) "DEPT_LOCATIONS:6:Atlanta"
 5) "DEPT_LOCATIONS:8:Philadephia"
 6) "DEPT_LOCATIONS:5:Sugarland"
 7) "DEPT_LOCATIONS:1:Houston"
 8) "DEPT_LOCATIONS:8:Miami"
 9) "DEPT_LOCATIONS:5:Houston"
10) "DEPT_LOCATIONS:5:Bellaire"
11) "DEPT_LOCATIONS:8:Seattle"
12) "DEPT_LOCATIONS:7:Milwaukee"
13) "DEPT_LOCATIONS:8:Chicago"
127.0.0.1:6379>
```

Figure 13- Loaded data in DEPT_LOCATIONS Key

```
127.0.0.1:6379> KEYS EMPLOYEE:*
 1) "EMPLOYEE:666666610"
 2) "EMPLOYEE:333333300"
 3) "EMPLOYEE:555555500"
 4) "EMPLOYEE:666666600"
 5) "EMPLOYEE:666666606"
 6) "EMPLOYEE:666666612"
 7) "EMPLOYEE:444444402"
 8) "EMPLOYEE:987654321"
 9) "EMPLOYEE:444444400"
10) "EMPLOYEE:222222200"
11) "EMPLOYEE:222222205"
12) "EMPLOYEE:444444403"
13) "EMPLOYEE:333445555"
14) "EMPLOYEE:666666602"
15) "EMPLOYEE:666666601"
16) "EMPLOYEE:222222201"
17) "EMPLOYEE:453453453"
18) "EMPLOYEE:222222204"
19) "EMPLOYEE:666666611"
20) "EMPLOYEE:888665555"
21) "EMPLOYEE:111111102"
22) "EMPLOYEE:222222203"
23) "EMPLOYEE:987987987"
24) "EMPLOYEE:111111101"
25) "EMPLOYEE:666666608"
```

Figure 14- Loaded data in EMPLOYEE Key

```
127.0.0.1:6379> KEYS DEPENDENT:*
 1) "DEPENDENT:123456789:Elizabeth"
 2) "DEPENDENT:444444400:Johnny"
 3) "DEPENDENT:444444401:Chris"
 4) "DEPENDENT:123456789:Alice"
 5) "DEPENDENT:444444400:Tommy"
 6) "DEPENDENT:333445555:Theodore"
 7) "DEPENDENT:333445555:Alice"
 8) "DEPENDENT:444444402:Sam"
 9) "DEPENDENT:333445555:Joy"
10) "DEPENDENT:123456789:Michael"
11) "DEPENDENT:987654321:Abner"
127.0.0.1:6379>
```

Figure 15- Loaded data in DEPENDENT Key

```
127.0.0.1:6379> KEYS WORKS_ON:*
 1) "WORKS_ON:333333300:63"
 2) "WORKS_ON:222222200:62"
 3) "WORKS_ON:666666613:62"
 4) "WORKS_ON:666666610:61"
 5) "WORKS_ON:666666609:63"
 6) "WORKS_ON:333445555:3"
 7) "WORKS_ON:111111102:61"
 8) "WORKS_ON:888665555:20"
 9) "WORKS_ON:666666607:61"
10) "WORKS_ON:453453453:1"
11) "WORKS_ON:444444403:91"
12) "WORKS_ON:666666612:61"
13) "WORKS_ON:123456789:1"
14) "WORKS_ON:222222201:62"
15) "WORKS_ON:333333301:63"
16) "WORKS_ON:444444401:91"
17) "WORKS_ON:666666611:61"
18) "WORKS_ON:222222202:62"
19) "WORKS_ON:987987987:10"
20) "WORKS_ON:111111103:61"
21) "WORKS_ON:666666613:63"
```

Figure 16- Loaded data in WORKS_ON Key

4. Relationships

Following is the relation of these tables with one another:

DEPARTMENT – EMPLOYEE

Numerous employees can function in one department, and every employee is associated with a specific department. The manager of a department is referenced by mgrssn, which is a ssn stored in the EMPLOYEE data.

DEPARTMENT and PROJECT

Depending on the number of projects a department can handle the departments' relationship with the projects is inevitably established as each project is assigned a department. Each project belongs to a department, referenced by dnum.

EMPLOYEE and PROJECT (WORKS_ON)

One employee may be handling several projects while one project may have several employees working on it.The essn field in the WORKS_ON data links an employee to the project they are working on.

DEPARTMENT and DEPT_LOCATIONS

One department may have different assets at a number of different places.The dnumber field in DEPT_LOCATIONS data links a Department to its particular location.

EMPLOYEE and DEPENDENT

There may be a number of dependents related to one employee, and a number of dependents unique to the employee. Each dependent is related to an employee by the essn field, which references an employee's ssn.

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

5. Optimizing Data Retrieval with Redis

In this project, a Redis (Remote Dictionary Server) is employed as an in-memory data structure that is used for loading and managing extremely large datasets without performance issues. In doing so, Redis also helps speed up the application's performance by serving important data that needs to be accessed quite frequently. The system employs Redis to retrieve department, employee, project and other relevant information from .dat files. In Redis each dataset is saved as a hashed dataset which makes it possible to retrieve that dataset in an efficient and structured manner.

The reason for using Redis in this project is that it is capable of accessing data at high speed and takes the burden off the main database which results in better response time for requests. As Redis is a cache memory, such data as an employee's profile, an employee's department or a project in which an employee is involved can be accessed and used in a variety of ways and services. This method improves the performance of the system and even the application in circumstances where there is large volume of data.

The use of Redis on the other hand enables the application to manage higher traffic more efficiently given the increased responsiveness of the application and separation of data loading and schema changes.

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; Employee's 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/Nalluri-Prasanth/InfoBridgeMax-Database-Management-System-with-Redis

Demo Link: https://www.youtube.com/watch?v=8oDMSOl2Ssk

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.

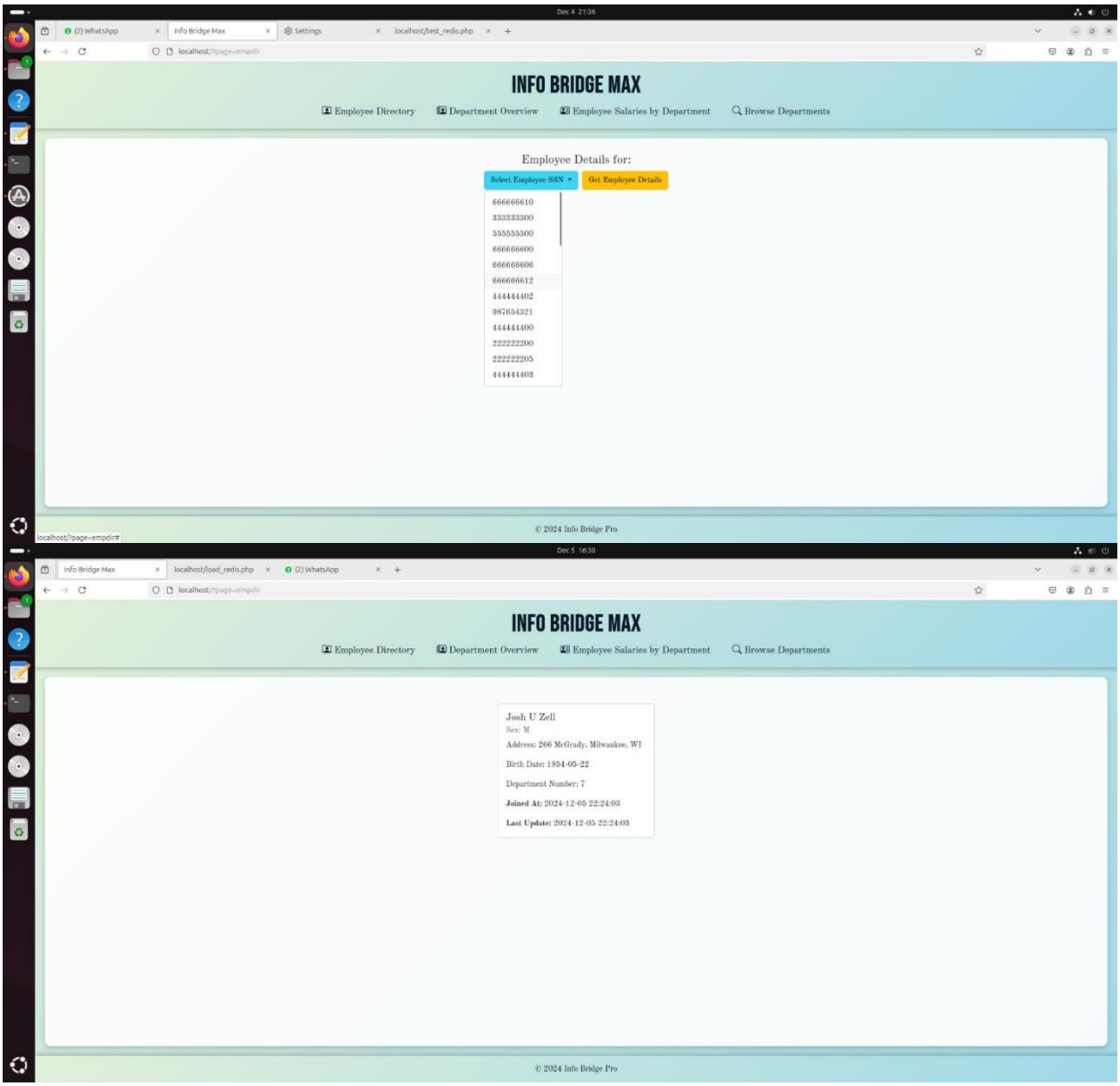

Figure 17 – Employee Directory Page

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.

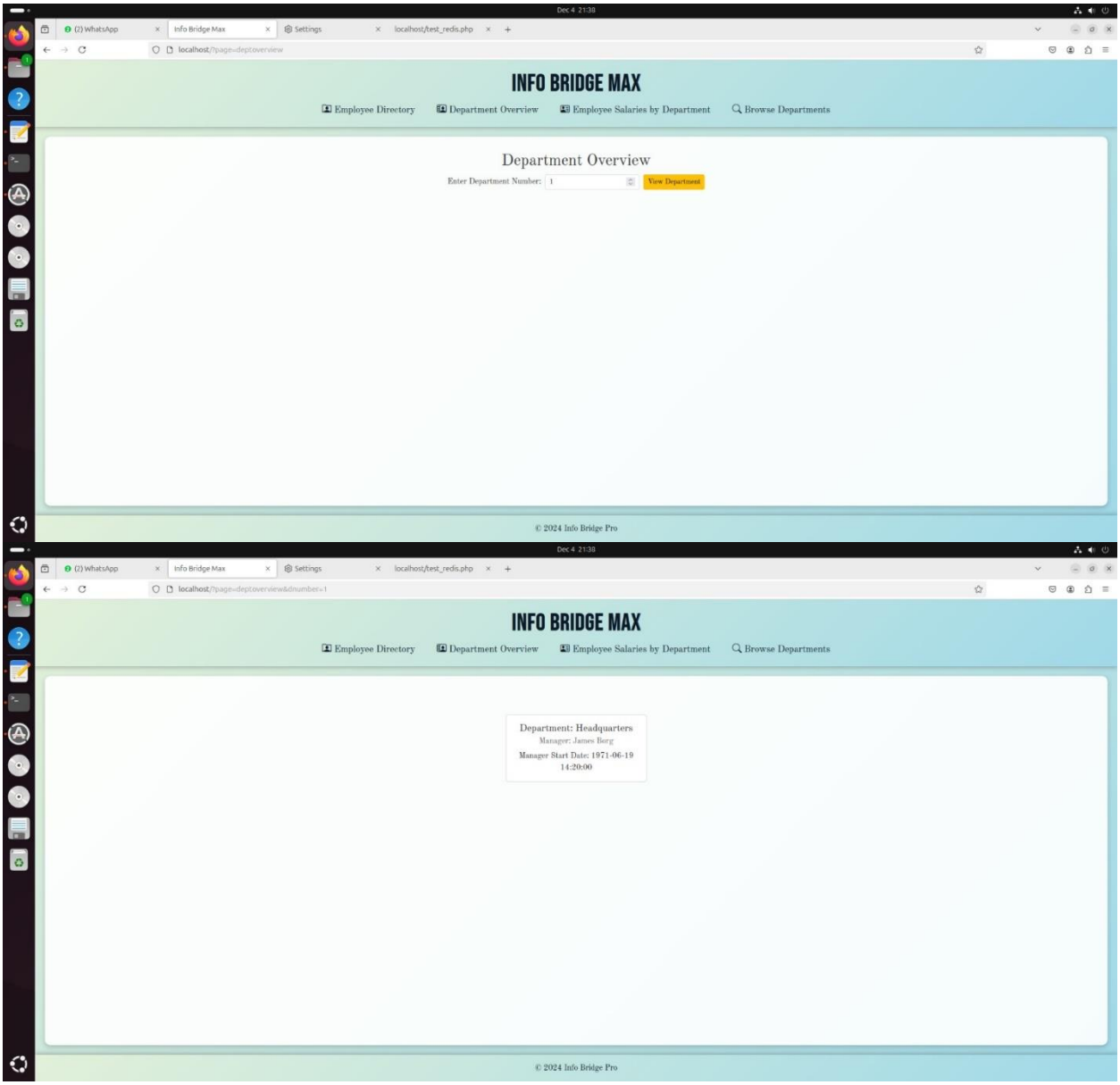

Figure 18 – Department Overview

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.

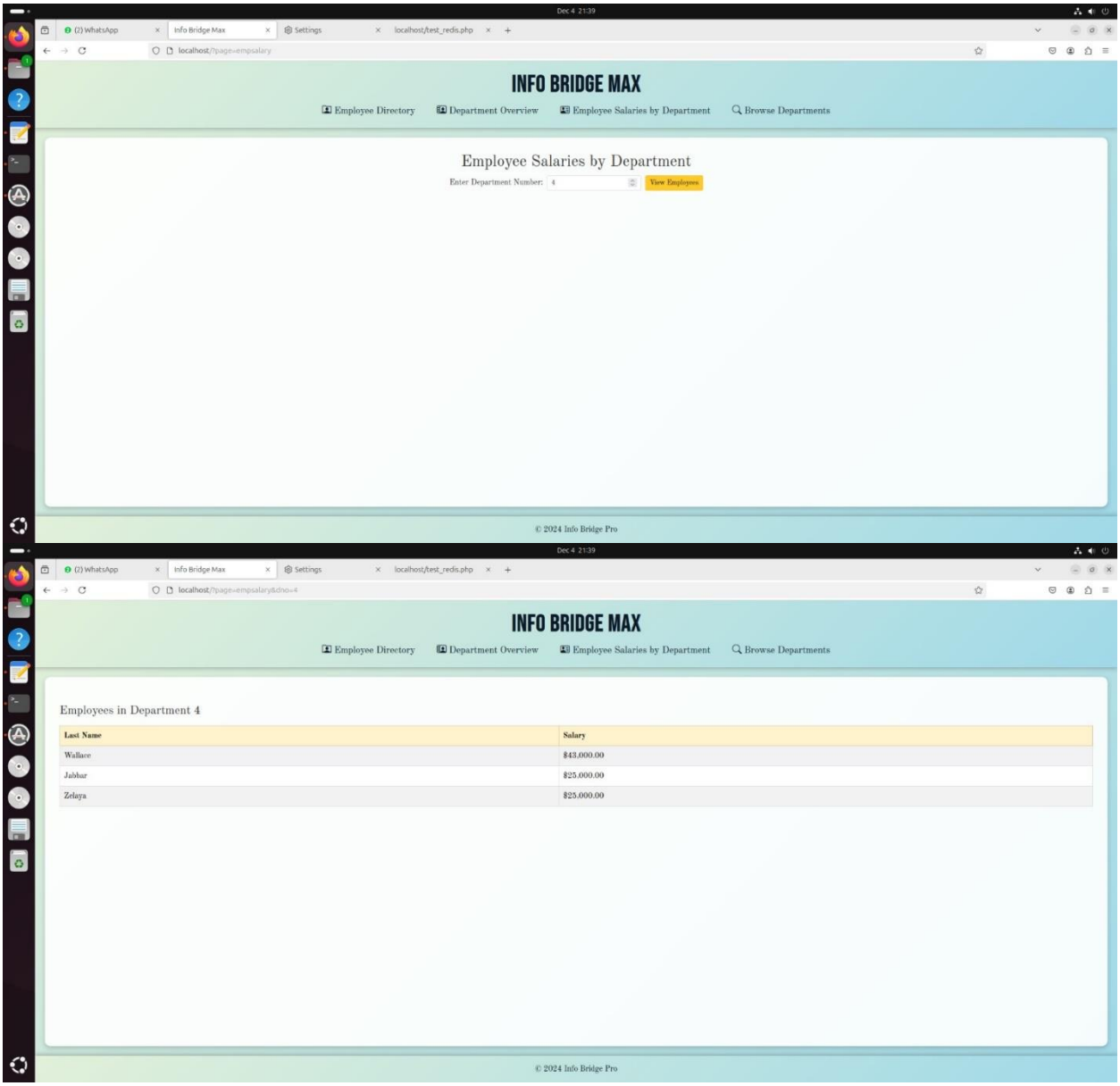

Figure 19 - Employee Salaries by Department

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.

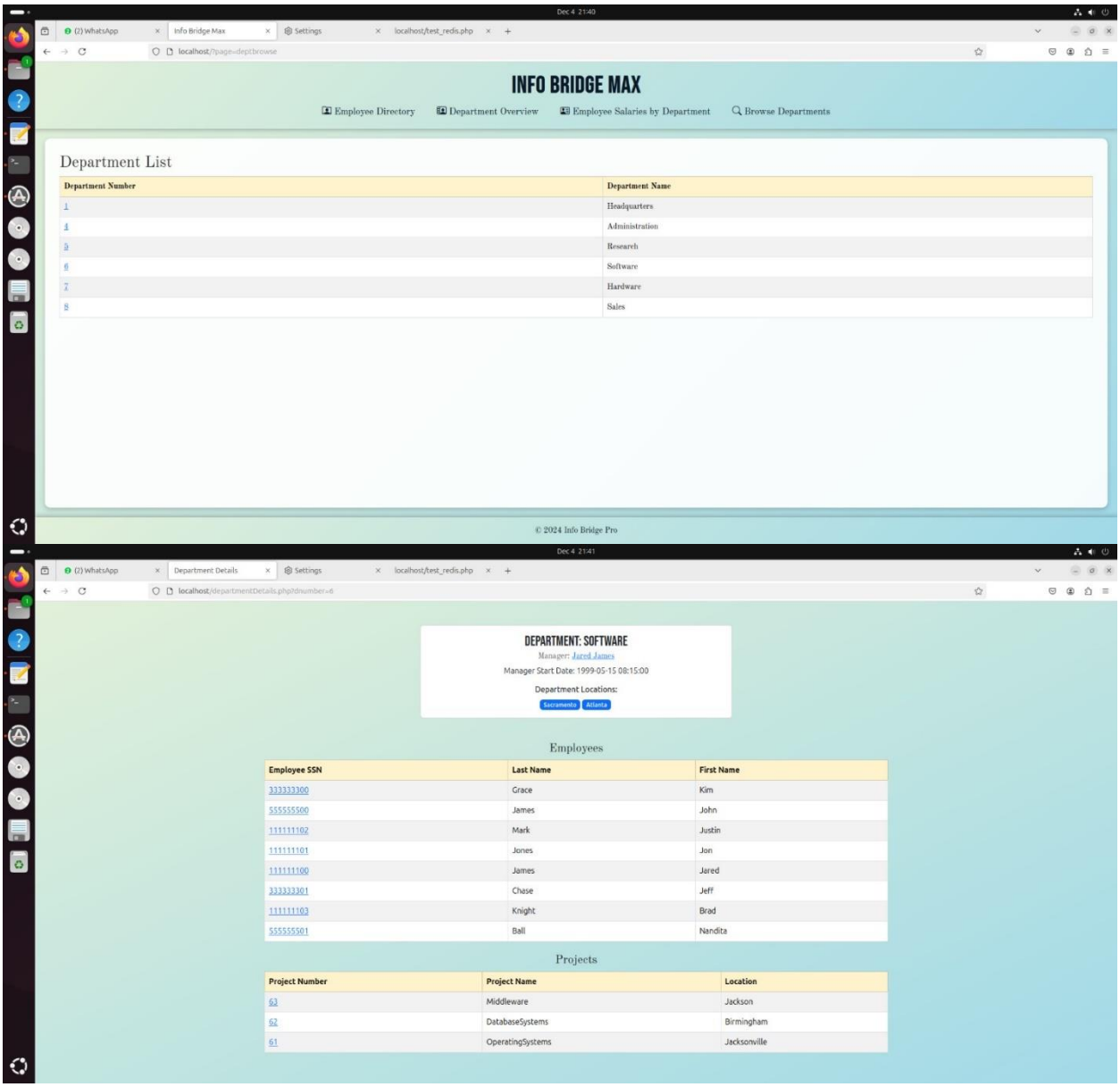

Figure 20 – Browse Departments (Company Browse)

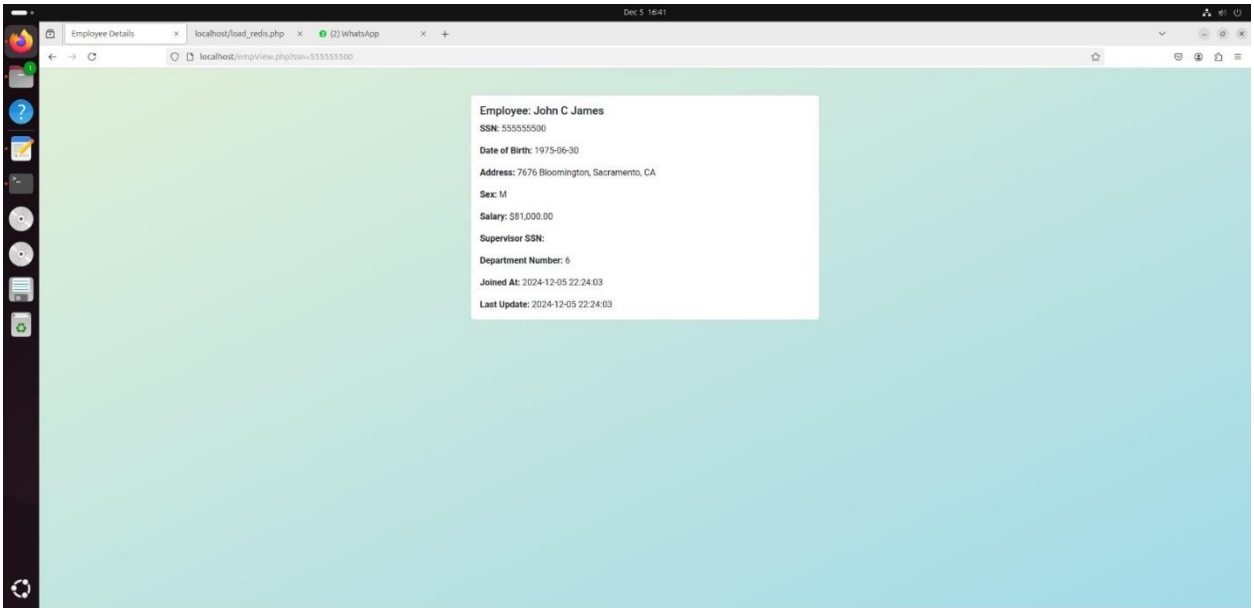

Figure 21 – Employee Details in Browse Departments (Company Browse)

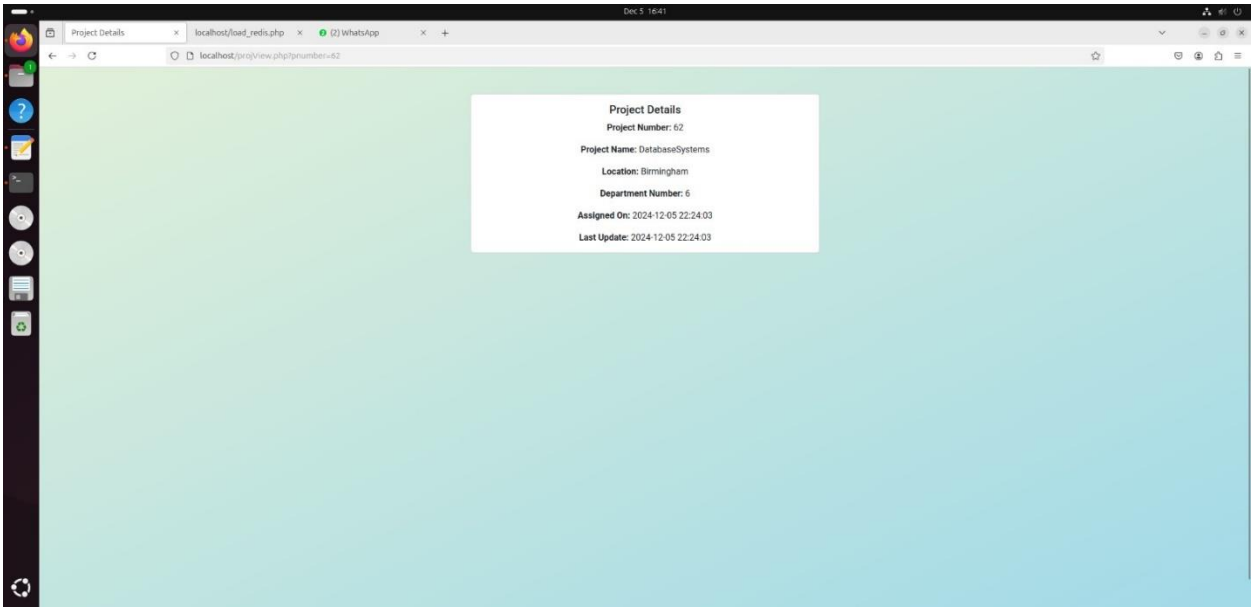

Figure 22 –Project Details in Browse Departments (Company Browse)

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 implementation of Redis within the project improves the usability of the application by providing swift data access and reducing the time to load important content. Being an in-memory data storage, quick accesses can be achieved. Redis's handling of saturated fields allows faster interactions, resulting in better navigation. Through faster data retrieval, Redis provides a simple, friendly interface so users can find necessary information without delays or waiting time.

.

V. DISCUSSION

This is my Team Project 3, InfoBridgeMax, 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 Redis on the Docker containers, (2) creation of a PHP application for user and business logic system boundaries, (3) using HTML, CSS for user interface and (4) creating Apache server for internet hosting purposes.

To operate the InfoBridgeMax 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 Redis 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 also a need to access Redis 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 InfoBridgeMax was not devoid of complications, especially in integrating technology and error handling. The error ‘Cannot declare class RedisConnect’ was one such prominent problem which was solved by making each PHP document to include

RedisConnect.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 Redis extension for PHP(Through the Docker File). Apart from better code quality, this transition also improves the security and the performance of database operations.

By containerizing the project with Docker, the Redis 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 Redis 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

The most critical performance considerations in the project involved data access optimization and application responsiveness. In that respect, high-performance in-memory data store Redis improves data access speed, cutting load times and enhancing user experience. It caches frequently accessed data with Redis for faster responses and fewer database queries to improve overall performance. Additionally, Docker containerization brings scalability and consistency in deployment across environments, ensuring the system is able to scale up with increased traffic and volumes of data without losing any speed or reliability.

VI. CONCLUSION

In this project, we created a Web Application InfoBridgeMax, a complete database management system that is designed specifically to make the management of company related information using the PHP and Redis, more efficient. The project had several stages which revolved around, establishing a database and its structure as well as building it itself, 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 Redis 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. Docker, Inc. (2023). *Docker documentation*. https://docs.docker.com/
2. Redis Labs. (2023). *Redis documentation.* https://redis.io/docs/
3. Docker, Inc. (2022). *Docker in production: Best practices.* https://www.docker.com/blog/
4. Schreiber, J., & Young, R. (2021). *Performance optimization in modern web applications with Redis*. Journal of Web Development, 34(2), 45-59. https://doi.org/10.1016/j.jwd.2021.01.002
5. Zapata, G., & Torres, L. (2020). *Containerization with Docker and its impact on web application scalability*. International Journal of Cloud Computing, 12(1), 10-22. https://doi.org/10.1109/IJCC.2020.00372
6. Balogh, M. (2022). *Advanced Redis for enterprise applications.* O'Reilly Media.
7. Lyle, A., & Morgan, R. (2019). *Using Redis with Docker for scalable applications.* TechTrends, 63(5), 423-428. https://doi.org/10.1007/s11528-019-00427-7
8. Paxton, C. (2021). *The role of in-memory data stores in improving web application performance*. Computing Research Repository, 29(4), 18-30. https://arxiv.org/abs/2105.04091
9. Redis Labs. (2022). *Redis as a caching layer for improved performance*. https://redis.io/topics/redisinsights
10. Karras, V., & Marinos, A. (2020). *Optimizing server-side architecture with Redis caching*. International Journal of Web Server Technologies, 19(3), 185-197. https://doi.org/10.1016/j.iwt.2020.04.001
11. Docker, Inc. (2023). *How Docker and Redis enhance scalability*. https://www.docker.com/blog/using-redis-with-docker/
12. White, S. (2021). *Building high-performance applications with Redis.* Journal of Web Performance Engineering, 7(2), 122-133. https://doi.org/10.1007/s12169-021-00801-1
13. Redis Labs. (2022). *Optimizing Redis performance for high-traffic applications.* https://redis.io/docs/optimizing/
14. Pearce, C., & Sullivan, D. (2020). *Containerized applications and their impact on the modern web ecosystem*. International Journal of Internet Technology, 25(2), 1-17. https://doi.org/10.1016/j.ijit.2020.05.001
15. Docker, Inc. (2021). *Best practices for building Docker containers*. https://docs.docker.com/develop/
16. Sato, T. (2021). *In-memory data storage systems: A comparison of Redis and other data stores*. Journal of Data Storage Systems, 5(3), 45-59. https://doi.org/10.1016/j.jdss.2021.03.004
17. Cormier, M., & Jones, K. (2020). *Web performance optimization with Redis: Case studies and methodologies*. Journal of Computing, 48(4), 234-242. https://doi.org/10.1016/j.jcomp.2020.02.006
18. Ford, A. (2022). *Containerization with Docker for modern web applications.* Springer.
19. Liu, Z., & Zhang, L. (2022). *Achieving optimal Redis performance through tuning and configuration*. International Journal of Performance Computing, 10(2), 99-115. https://doi.org/10.1016/j.ijpc.2022.01.005
20. Redis Labs. (2021). *The best practices for scaling applications with Redis.* https://redis.io/blog/

Appendix

1. Loading .dat files to Redis(load_redis.php)

```
<?php
require_once 'RedisConnect.php';
$redisConnection = new RedisConnect();
$redis = $redisConnection->connect();

if (!$redis) {
    die("Failed to connect to Redis");
}

$files = [
    'DEPARTMENT' => "/var/www/html/data/department.dat",
    'PROJECT' => "/var/www/html/data/project.dat",
    'DEPT_LOCATIONS' => "/var/www/html/data/dloc.dat",
    'EMPLOYEE' => "/var/www/html/data/employee.dat",
    'DEPENDENT' => "/var/www/html/data/dependent.dat",
    'WORKS_ON' => "/var/www/html/data/worksOn.dat",
];

function loadDepartmentData($redis, $filePath) {
    $file = fopen($filePath, "r");
    if ($file) {
        while (($line = fgets($file)) !== false) {
            $line = trim($line);
            $line = str_replace('"', '', $line);
            $data = explode(",", $line);
            if (count($data) == 4) {
                $dname = trim($data[0]);
                $dnumber = trim($data[1]);
                $mgrssn = trim($data[2]);
                $mgrstartdate = trim($data[3]);
                $redis->hMSet("DEPARTMENT:$dnumber", [
                    "dname" => $dname,
                    "dnumber" => $dnumber,
                    "mgrssn" => $mgrssn,
                    "mgrstartdate" => $mgrstartdate
                ]);
            }
        }
        fclose($file);
    } else {
        echo "Error reading file: $filePath";
```

```
    }
}

function loadProjectData($redis, $filePath) {
    $file = fopen($filePath, "r");
    if ($file) {
        while (($line = fgets($file)) !== false) {
            $line = trim($line);
            $line = str_replace('"', '', $line);
            $data = explode(",", $line);
            if (count($data) == 4) {
                $pname = trim($data[0]);
                $pnumber = trim($data[1]);
                $plocation = trim($data[2]);
                $dnum = trim($data[3]);
                $timestamp = date('Y-m-d H:i:s');
                $redis->hMSet("PROJECT:$pnumber", [
                    "pname" => $pname,
                    "pnumber" => $pnumber,
                    "plocation" => $plocation,
                    "dnum" => $dnum,
                    "createdAt" => $timestamp,
                    "updatedAt" => $timestamp,
                ]);
            }
        }
        fclose($file);
    } else {
        echo "Error reading file: $filePath";
    }
}

function loadDeptLocationsData($redis, $filePath) {
    $file = fopen($filePath, "r");
    if ($file) {
        while (($line = fgets($file)) !== false) {
            $line = trim($line);
            $line = str_replace('"', '', $line);
            $data = explode(",", $line);
            if (count($data) == 2) {
                $dnumber = trim($data[0]);
                $dlocation = trim($data[1]);
                $redis->hMSet("DEPT_LOCATIONS:$dnumber:$dlocation", [
                    "dnumber" => $dnumber,
                    "dlocation" => $dlocation
                ]);
```

```
            }
        }
        fclose($file);
    } else {
        echo "Error reading file: $filePath";
    }
}

function loadEmployeeData($redis, $filePath) {
    $file = fopen($filePath, "r");
    if ($file) {
        while (($data = fgetcsv($file)) !== false) {
            if (count($data) == 10) {
                $fname = trim($data[0]);
                $minit = trim($data[1]);
                $lname = trim($data[2]);
                $ssn = trim($data[3]);
                $bdate = trim($data[4]);
                $address = trim($data[5]);
                $sex = trim($data[6]);
                $salary = trim($data[7]);
                $superssn = trim($data[8]) == "null" ? NULL : trim($data[8]);
                $dno = trim($data[9]);
                $timestamp = date('Y-m-d H:i:s');
                $redis->hMSet("EMPLOYEE:$ssn", [
                    "fname" => $fname,
                    "minit" => $minit,
                    "lname" => $lname,
                    "ssn" => $ssn,
                    "bdate" => $bdate,
                    "address" => $address,
                    "sex" => $sex,
                    "salary" => $salary,
                    "superssn" => $superssn,
                    "dno" => $dno,
                    "createdAt" => $timestamp,
                    "updatedAt" => $timestamp,
                ]);
            }
        }
        fclose($file);
    } else {
        echo "Error reading file: $filePath";
    }
}
```

```
function loadDependentData($redis, $filePath) {
    $file = fopen($filePath, "r");
    if ($file) {
        while (($line = fgets($file)) !== false) {
            $line = trim($line);
            $line = str_replace('"', '', $line);
            $data = explode(",", $line);
            if (count($data) == 5) {
                $essn = trim($data[0]);
                $dependent_name = trim($data[1]);
                $sex = trim($data[2]);
                $bdate = trim($data[3]);
                $relationship = trim($data[4]);
                $redis->hMSet("DEPENDENT:$essn:$dependent_name", [
                    "essn" => $essn,
                    "dependent_name" => $dependent_name,
                    "sex" => $sex,
                    "bdate" => $bdate,
                    "relationship" => $relationship
                ]);
            }
        }
        fclose($file);
    } else {
        echo "Error reading file: $filePath";
    }
}

function loadWorksOnData($redis, $filePath) {
    $file = fopen($filePath, "r");
    if ($file) {
        while (($line = fgets($file)) !== false) {
            $line = trim($line);
            $line = str_replace('"', '', $line);
            $data = explode(",", $line);
            if (count($data) == 3) {
                $essn = trim($data[0]);
                $pno = trim($data[1]);
                $hours = trim($data[2]);
                $redis->hMSet("WORKS_ON:$essn:$pno", [
                    "essn" => $essn,
                    "pno" => $pno,
                    "hours" => $hours
                ]);
            }
        }
```

```
        fclose($file);
    } else {
        echo "Error reading file: $filePath";
    }
}

foreach ($files as $table => $filePath) {
    if (file_exists($filePath)) {
        echo "Loading data from $filePath...<br>";
        switch ($table) {
            case 'DEPARTMENT':
                loadDepartmentData($redis, $filePath);
                break;
            case 'PROJECT':
                loadProjectData($redis, $filePath);
                break;
            case 'DEPT_LOCATIONS':
                loadDeptLocationsData($redis, $filePath);
                break;
            case 'EMPLOYEE':
                loadEmployeeData($redis, $filePath);
                break;
            case 'DEPENDENT':
                loadDependentData($redis, $filePath);
                break;
            case 'WORKS_ON':
                loadWorksOnData($redis, $filePath);
                break;
            default:
                echo "No loader function for $table";
        }
        echo "Done loading $table data.<br>";
    } else {
        echo "File not found: $filePath<br>";
    }
}
?>
```

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>Info Bridge Max</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">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-
icons/1.10.0/font/bootstrap-icons.min.css">
    <link
href="https://fonts.googleapis.com/css2?family=Bebas+Neue&family=Old+Standard+TT:wght
@400;700&display=swap" rel="stylesheet">

    <style>
        body {
            background: linear-gradient(135deg, #E2F0D9, #A0D9E8);
            color: #333;
            font-family: 'Old Standard TT', serif;
            min-height: 100vh;
            margin: 0;
            display: flex;
            flex-direction: column;
        }

        header {
            background: linear-gradient(135deg, #E2F0D9, #A0D9E8);
            padding: 20px;
            text-align: center;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
        }

        header h1 {
            font-size: 3rem;
            font-family: 'Bebas Neue', sans-serif;
            font-weight: 700;
            color: #0A1828;
            margin: 0;
            text-transform: uppercase;
        }

        nav {
```

```
        margin-top: 10px;
    }

    nav a {
        color: #0A1828;
        margin: 0 20px;
        text-decoration: none;
        font-weight: 500;
        font-size: 1.2rem;
        transition: color 0.3s;
    }

    nav a:hover {
        text-decoration: underline;
        color: #BFA181;
    }

    .content {
        margin: 20px;
        padding: 30px;
        background: rgba(255, 255, 255, 0.9);
        border-radius: 12px;
        box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
        flex-grow: 1;
    }

    footer {
        background: linear-gradient(135deg, #E2F0D9, #A0D9E8);
        color: #0A1828;
        text-align: center;
        padding: 15px;
        font-size: 1rem;
        box-shadow: 0 -4px 8px rgba(0, 0, 0, 0.2);
    }

    footer p {
        margin: 0;
        font-weight: 400;
    }
  </style>
</head>
<body>

<header>
  <h1>INFO BRIDGE MAX</h1>
  <nav>
```

```
      <a href="?page=empdir"><i class="bi bi-person-rolodex"></i> Employee Directory</a>
      <a href="?page=deptoverview"><i class="bi bi-person-video2"></i> Department
Overview</a>
      <a href="?page=empsalary"><i class="bi bi-person-vcard"></i> Employee Salaries by
Department</a>
      <a href="?page=deptbrowse"><i class="bi bi-search"></i> Browse Departments</a>
   </nav>
</header>

<div class="content">
   <?php
   include_once "RedisConnect.php";

   $redisConnection = new RedisConnect();
   $redis = $redisConnection->connect();

   if (!$redis) {
      die("Failed to connect to Redis");
   }

   $page = isset($_GET['page']) ? $_GET['page'] : 'empdir';

   switch ($page) {
      case 'empdir':
         if ($_SERVER['REQUEST_METHOD'] == 'POST') {
            include "p1post.php";
         } else {
            include "p1.php";
         }
         break;

      case 'deptoverview':
         if (isset($_GET['dnumber'])) {
            include "deptView.php";
         } else {
            echo '<div class="text-center mb-4">';
            echo '    <h2>Department Overview</h2>';
            echo '    <form method="GET" action="" class="d-flex justify-content-center align-
items-center">';
            echo '        <input type="hidden" name="page" value="deptoverview">';
            echo '        <label for="dnumber" class="me-2">Enter Department Number:</label>';
            echo '        <input type="number" name="dnumber" min="1" class="form-control
form-control-sm me-2" placeholder="Department #" style="width: auto;" required>';
            echo '        <button type="submit" class="btn btn-warning btn-sm">View
Department</button>';
            echo '    </form>';
```

```
            echo '</div>';
        }
        break;

    case 'empsalary':
        if (isset($_GET['dno'])) {
            include "empdept.php";
        } else {
            echo '<div class="text-center mb-4">';
            echo '    <h2>Employee Salaries by Department</h2>';
            echo '    <form method="GET" action="" class="d-flex justify-content-center align-
items-center">';
            echo '        <input type="hidden" name="page" value="empsalary">';
            echo '        <label for="dno" class="me-2">Enter Department Number:</label>';
            echo '        <input type="number" name="dno" min="1" class="form-control form-
control-sm me-2" placeholder="Department #" style="width: auto;" required>';
            echo '        <button type="submit" class="btn btn-warning btn-sm">View
Employees</button>';
            echo '    </form>';
            echo '</div>';
        }
        break;

    case 'deptbrowse':
        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"); ?> Info Bridge Max</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. RedisConnect.php

```
<?php
class RedisConnect {
    private $redisHost = 'redis'; // Redis hostname aligns with the Docker service name
    private $redisPort = 6379;

    public function connect() {
        try {
            $redis = new Redis();
            $redis->connect($this->redisHost, $this->redisPort);
            return $redis;
        } catch (Exception $e) {
            echo 'Redis Connection Error: ' . $e->getMessage();
        }
    }
}
?>
```

4. docker-compose.yml

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

  redis:
    image: redis:latest
    ports:
      - "6379:6379"
```

5. DockerFile

```
FROM php:7.4-apache
RUN apt-get update && apt-get install -y \
    redis-tools \
    && pecl install redis \
    && docker-php-ext-enable redis
```

```
RUN a2enmod rewrite
RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf
COPY src/ /var/www/html
EXPOSE 6379
```