Institute of
Space Technology

DEPARTMENT OF COMPUTER SCIENCE

DS-01

DataBase Systems

Project Report

Submitted to:

Engr.Shehla Gul

Submitted by:

Bilal Ahmed Khan (241201055)

M.Usman Riaz (241201014)

Table of Contents

1-Introduction:

Currently, it is difficult for modern organizations to oversee tasks done by employees and ensure everyone remains accountable. Manual tracking does not work well for teams that are spread out or handling constantly changing projects. These problems are dealt with by the ETMS through the following actions:

- Organizing and managing tasks from one central point
- I am able to get instant updates and be notified in real-time.
- Managing access rights depending on people's place in the organization
- Analysts review the results of past performances.
- Sending notifications about deadline reminders

All the components are developed using a standard three-tier architecture.
HTML, CSS and JavaScript are the main components for the Presentation Layer
PHP is used for the program's application development with a traditional procedure style.
The database I have built is a **MySQL**

2-Requirment Analysis

- **Functional Requirements**

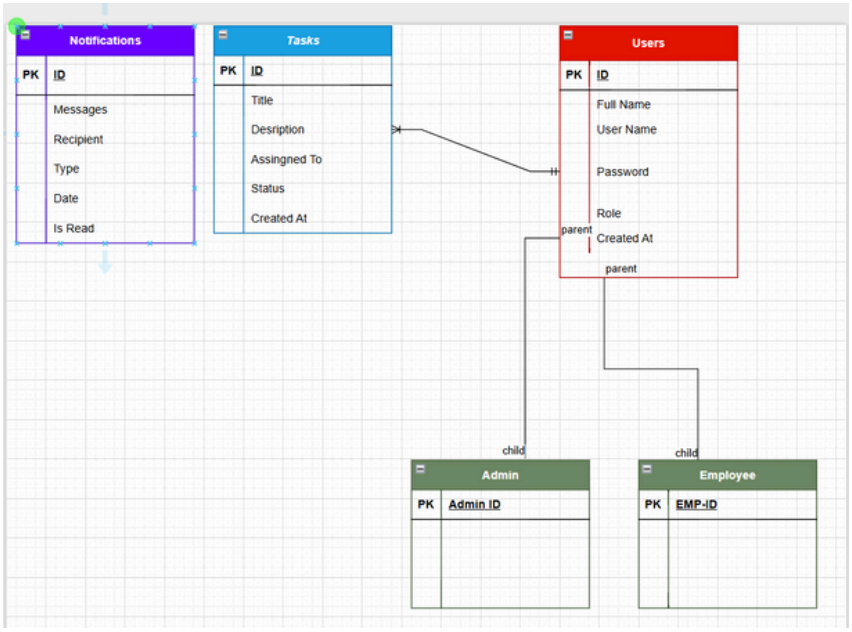
The Functional Requirements of this project includes:
1-Login authentication with assigned credentials
2-Role Based Access control for both employees and Admin
3-Create/Update/Delete tasks with title, description, assignees, and deadline s
4-Real-time alerts for new assignments and deadline reminders

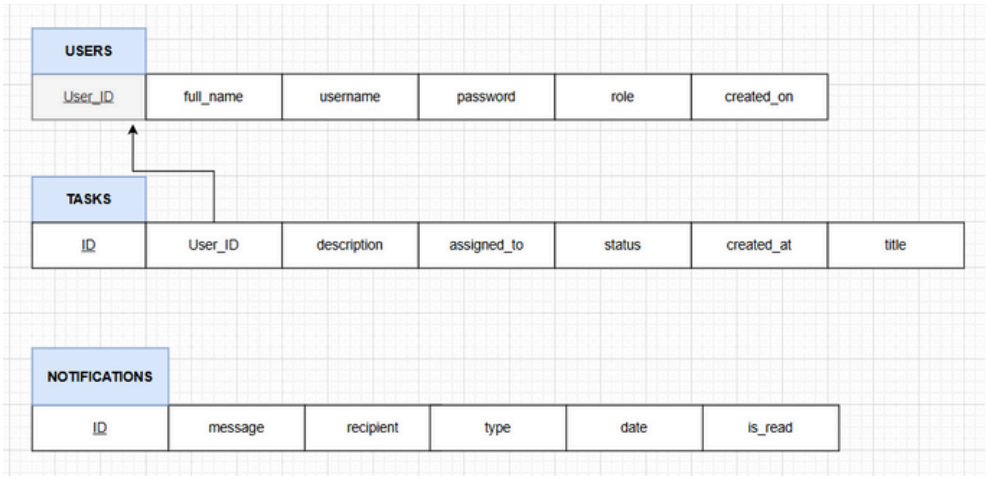
- **Non Functional Requirements**

The Non Functional Requirements of this project includes:
1-Three Tier architecture based Project
2-LocalHost based project using XAAMP
3-Reliable and effective for teams to manage their work
4-This project also includes password encryption increasing security

3-ER-Diagram & Relational Schema:

Below are the attached Images of ER diagram and Relational Schema





Our tables are already in 3rd NF so we don't need to convert . Looking at the 1st NF there are no duplicate values same for the 2nd NF there are no partial dependencies and for the 3rd NF there were no transitive dependencies

By this we don't need to do normalization in out tables

4-Database Design:

The database architecture was designed and implemented using **phpMyAdmin**, a MySQL administration tool, within the **XAMPP** development environment. The **MySQL Server** was configured to operate on **port 3307** to ensure seamless integration with the application framework

The following project contains three tables

- Users
- Task
- Notifications

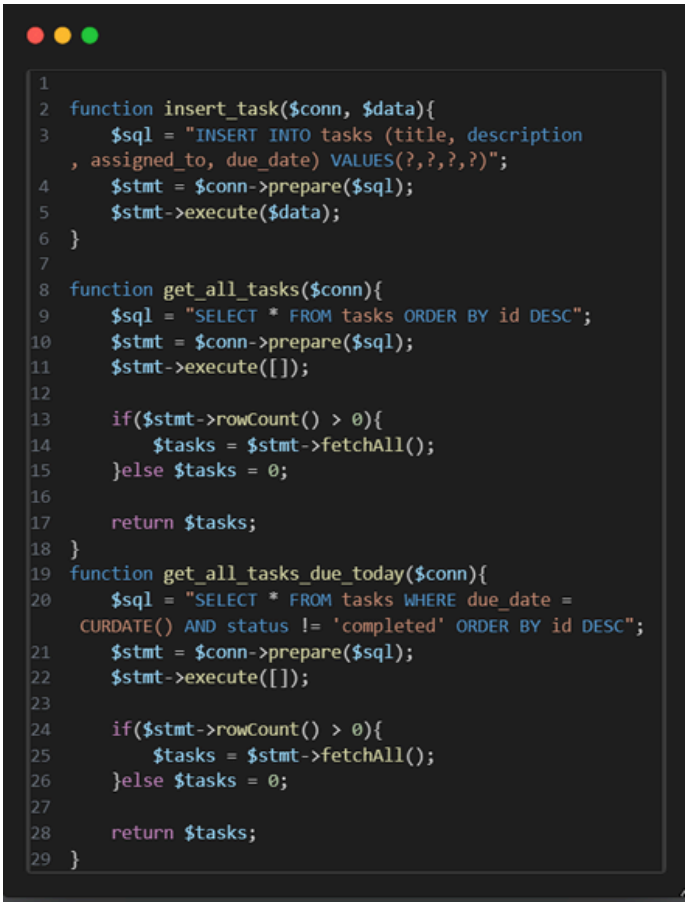
Such that one and many tasks are assigned to one user (employee) and Each employee can have one none and many task assigned to him

5-SQL Scripts

```php

function get_all_users($conn){
    $sql = "SELECT * FROM users WHERE role =? ";
    $stmt = $conn->prepare($sql);
    $stmt->execute(["employee"]);

    if($stmt->rowCount() > 0){
        $users = $stmt->fetchAll();
    }else $users = 0;

    return $users;
}


function insert_user($conn, $data){
    $sql = "INSERT INTO users (full_name, username, password, role) VALUES(?,?,?, ?)";
    $stmt = $conn->prepare($sql);
    $stmt->execute($data);
}
```

```php
function update_user($conn, $data){
    $sql = "UPDATE users SET full_name=?, username=?, password=?, role=? WHERE id=? AND role=?";
    $stmt = $conn->prepare($sql);
    $stmt->execute($data);
}

function delete_user($conn, $data){
    $sql = "DELETE FROM users WHERE id=? AND role=?";
    $stmt = $conn->prepare($sql);
    $stmt->execute($data);
}


function get_user_by_id($conn, $id){
    $sql = "SELECT * FROM users WHERE id =? ";
    $stmt = $conn->prepare($sql);
    $stmt->execute([$id]);

```

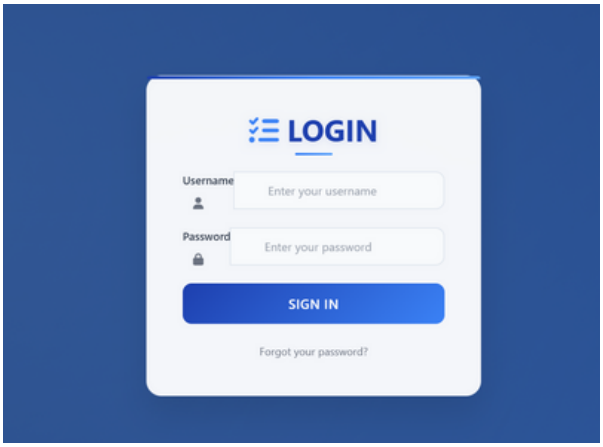
```php

function update_profile($conn, $data){
    $sql = "UPDATE users SET full_name=?,  password=?
WHERE id=? ";
    $stmt = $conn->prepare($sql);
    $stmt->execute($data);
}

function count_users($conn){
    $sql = "SELECT id FROM users WHERE role='employee'";
    $stmt = $conn->prepare($sql);
    $stmt->execute([]);

    return $stmt->rowCount();
}
```



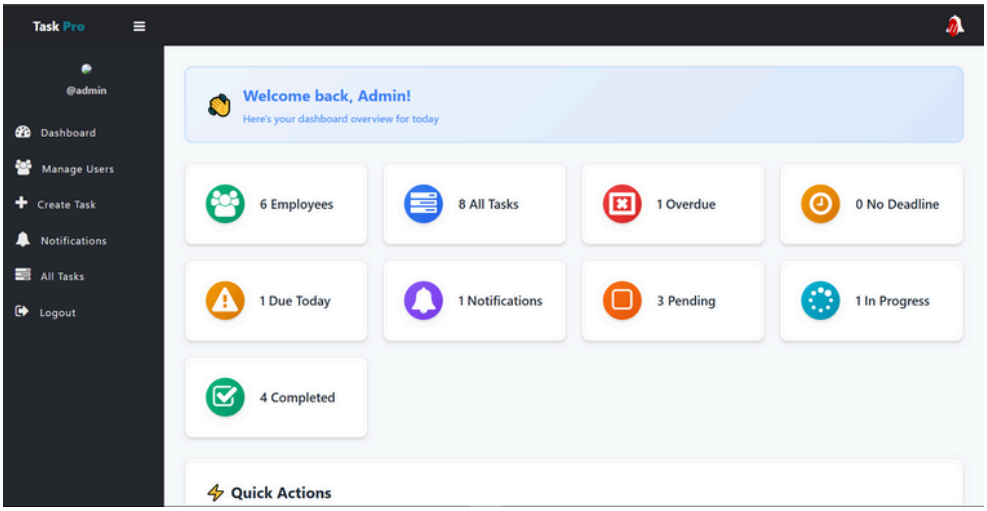
```php

function insert_task($conn, $data){
    $sql = "INSERT INTO tasks (title, description
, assigned_to, due_date) VALUES(?,?,?,?)";
    $stmt = $conn->prepare($sql);
    $stmt->execute($data);
}

function get_all_tasks($conn){
    $sql = "SELECT * FROM tasks ORDER BY id DESC";
    $stmt = $conn->prepare($sql);
    $stmt->execute([]);

    if($stmt->rowCount() > 0){
        $tasks = $stmt->fetchAll();
    }else $tasks = 0;

    return $tasks;
}
function get_all_tasks_due_today($conn){
    $sql = "SELECT * FROM tasks WHERE due_date =
 CURDATE() AND status != 'completed' ORDER BY id DESC";
    $stmt = $conn->prepare($sql);
    $stmt->execute([]);

    if($stmt->rowCount() > 0){
        $tasks = $stmt->fetchAll();
    }else $tasks = 0;

    return $tasks;
}
```

The following code snippets are not all that are used in actual code as it would make report too long

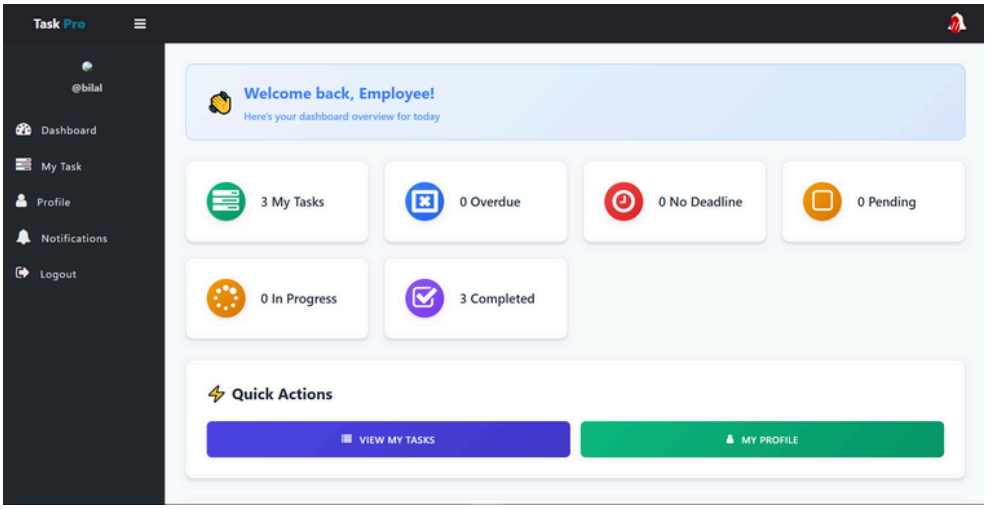
6-Application Interface:



- **Admin UI**



- **Employee UI**



7-Testing and Result:

The following project upon testing gives successful result. Some testing methods include

- Login check (by entering invalid credentials)
- Creating/Updating/Deleting Task from admin side
- Checking the proper working of notifications
- Updation of status of the task upon employee requirement
- Addition and deletion of employees by admin side

8-Challenges & Learnings

Challenges

1. **Session Management Issues** - Getting user sessions to work properly across different pages, ensuring users stay logged in, and handling session security.
2. **Database Connection Problems** - Setting up the database connection, handling prepared statements correctly, and dealing with SQL injection prevention.
3. **Error Handling & Debugging** - The missing exit() statements and URL encoding issues we just fixed are classic examples of bugs that can be frustrating to track down.
4. **Frontend-Backend Integration** - Getting the PHP error messages to display properly in your HTML/CSS interface, handling form submissions correctly.
5. **User Experience Flow** - Managing redirects, ensuring users end up in the right place after login/logout, handling different user roles (admin vs employee).
6. **Security Considerations** - Input validation, preventing unauthorized access, and handling edge cases.

Lessons :

- URL encode parameters when passing them in URLs
- The importance of proper error handling and debugging
- How sessions work in PHP web applications
- Database security with prepared statements
- The value of testing different scenarios (valid/invalid logins, empty fields, etc.)

9-Conclusion & Future Improvement:

Future Improvement

1. Mobile App development
2. Using a proper web hosting
3. Integration of AI
4. Using various frameworks like react,native

Lessons :

To conclude this our project is a full web app locally hosted covering all key functionalities that were proposed despite all the challenges we have successfully created Employee Task Management System