TASK MANAGEMENT SYSTEM REPORT

DO BY:

➢ Farah El Khatib 108791
➢ Hadi Baghdadi 109035
➢ Bahaa Mezhir 107667

INTRODUCTION

The Task Management System (TMS) is a powerful and comprehensive solution designed to streamline workforce management. It provides organizations with an efficient way to track and manage employee tasks, attendance, and performance. By integrating key features such as user management, task assignment, skill tracking, and attendance monitoring, the system ensures smooth operations and improved productivity across teams.

In today's fast-paced work environment, managing multiple employees and ensuring accountability can be a challenging task. The Task Management System addresses these challenges by offering a centralized platform where administrators can assign tasks, monitor progress, and analyze employee performance. Employees can efficiently manage their work assignments, log attendance, and track their progress in real-time.

The system incorporates automated validation and security mechanisms, ensuring that only qualified employees receive specific task assignments. Additionally, with built-in performance analytics, organizations can make data-driven decisions to optimize resource allocation and improve efficiency.

With robust database architecture, optimization techniques, and security features such as role-based access control and audit trails, the Task Management System is built for scalability and reliability. Whether for small teams or large enterprises, this system provides an all-in-one solution to task and performance management challenges.

SYSTEM FEATURES

- **User Role Management:** Admins & Employees
- **Skill-Based Task Assignment**
- **Task Status Tracking** (Incomplete, In Progress, Complete)
- **Attendance Monitoring & Logs**

- **Performance Analytics**

1. User Role Management

The system supports different user roles:
- Role 1: Administrators with full system access
- Role 2: Regular employees who can manage tasks and track attendance

2. Skill-Based Task Assignment

Tasks can be assigned based on employee skills, with validation to ensure employees have the required skills before assignment.

3. Task Status Tracking

Tasks have three status levels:

- 0: Incomplete
- 1: In progress
- 2: Complete

Additionally, task assignments have their own status workflow:

- pending
- accepted
- rejected

4. Attendance Monitoring

The system tracks:

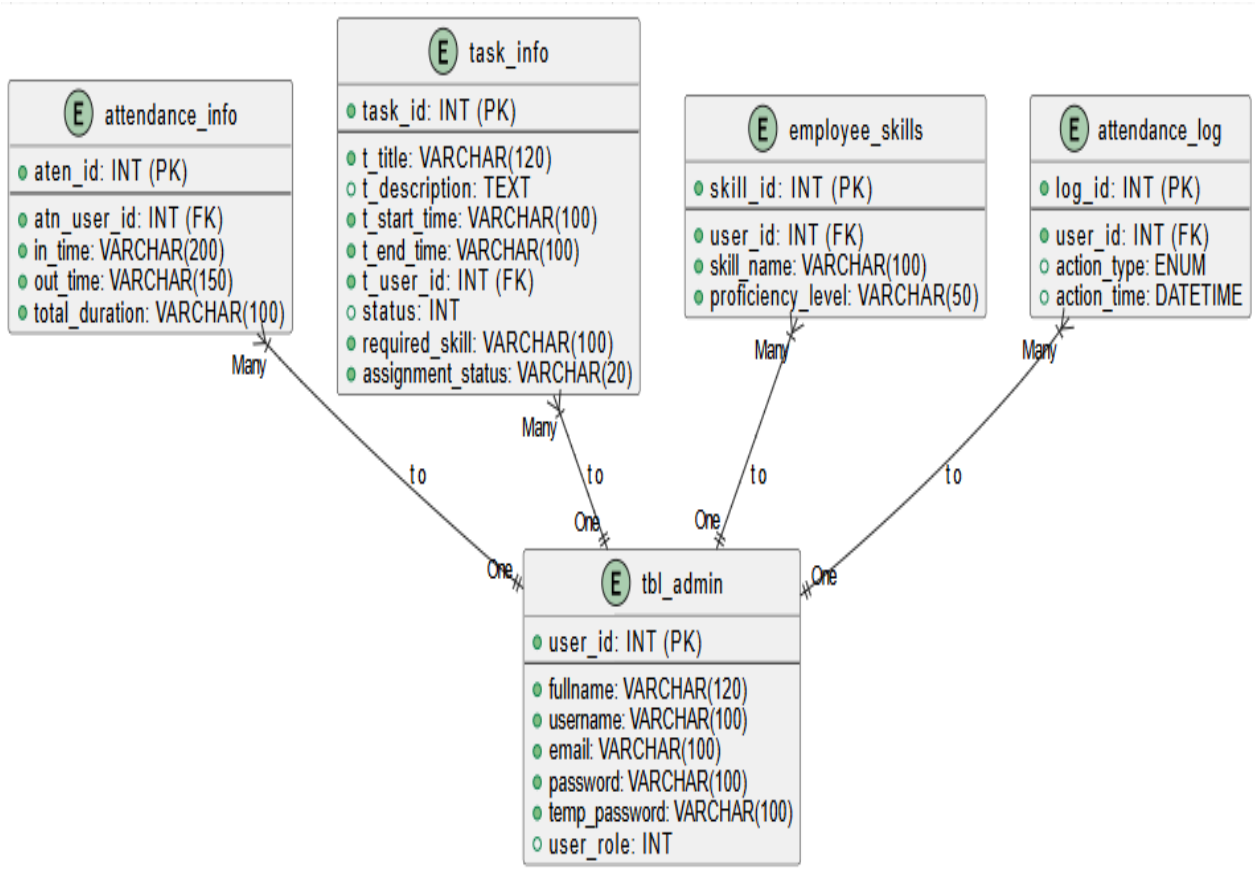
- Check-in times
- Check-out times
- Total work duration
- Attendance history through audit logs

5. Performance Analytics

Through the employee_performance view, managers can analyze:

- Total tasks assigned
- Completion rates
- Skill diversity

RELATIONAL SCHEMA:



DATABASE STRUCTURE

Database Name: etmsh (Employee Task Management System Hub)

Core Tables:

1. tbl_admin (User Management)
2. task_info (Task Assignment)
3. attendance_info (Attendance Tracking)
4. employee_skills (Skill Management)
5. attendance_log (Audit Trail)
6. login_attempts - Tracks login activity (Success/Failure).

```
CREATE TABLE IF NOT EXISTS `tbl_admin` (
  `user_id` int(20) NOT NULL AUTO_INCREMENT,
  `fullname` varchar(120) NOT NULL,
  `username` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `password` varchar(100) NOT NULL,
  `temp_password` varchar(100) DEFAULT NULL,
  `user_role` int(10) NOT NULL,
  PRIMARY KEY (`user_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
CREATE TABLE IF NOT EXISTS `task_info` (
  `task_id` int(50) NOT NULL AUTO_INCREMENT,
  `t_title` varchar(120) NOT NULL,
  `t_description` text,
  `t_start_time` varchar(100) DEFAULT NULL,
  `t_end_time` varchar(100) DEFAULT NULL,
  `t_user_id` int(20) NOT NULL,
  `status` int(11) NOT NULL DEFAULT '0' COMMENT '0 = incomplete, 1
= In progress, 2 = complete',
  `required_skill` varchar(100) DEFAULT NULL,
  `assignment_status` varchar(20) DEFAULT 'pending' COMMENT
'pending, accepted, rejected',
  PRIMARY KEY (`task_id`)
FOREIGN KEY (t_user_id) REFERENCES tbl_admin(user_id) ON
DELETE CASCADE
```

```
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
CREATE TABLE IF NOT EXISTS `attendance_info` (
  `aten_id` int(20) NOT NULL AUTO_INCREMENT,
  `atn_user_id` int(20) NOT NULL,
  `in_time` varchar(200) DEFAULT NULL,
  `out_time` varchar(150) DEFAULT NULL,
  `total_duration` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`aten_id`)
    FOREIGN KEY (user_id) REFERENCES tbl_admin(user_id)
ON DELETE CASCADE
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
CREATE TABLE IF NOT EXISTS `employee_skills` (
  `skill_id` int(20) NOT NULL AUTO_INCREMENT,
  `user_id` int(20) NOT NULL,
  `skill_name` varchar(100) NOT NULL,
  `proficiency_level` varchar(50) NOT NULL, -- 'Beginner',
'Intermediate', 'Expert'
  PRIMARY KEY (`skill_id`),
  FOREIGN KEY (`user_id`) REFERENCES `tbl_admin`(`user_id`) ON
DELETE CASCADE
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
CREATE TABLE attendance_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    action_type ENUM('CLOCK_IN', 'CLOCK_OUT') NOT NULL,
```

```
    action_time DATETIME NOT NULL,
    INDEX (user_id, action_time)
);
```

➢ We create a table to store login attempts, tracking **success and failure**.
Table: login_attempts

```
CREATE TABLE IF NOT EXISTS login_attempts (
    attempt_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    username VARCHAR(100) NOT NULL,
    attempt_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status ENUM('SUCCESS', 'FAILURE') NOT NULL,
    FOREIGN KEY (user_id) REFERENCES tbl_admin(user_id) ON
DELETE SET NULL
);
```

DATABASE OPTIMIZATION

Indexes:

```
CREATE INDEX idx_task_user ON task_info(t_user_id);

CREATE INDEX idx_task_status ON task_info(status);

CREATE INDEX idx_attendance_user ON attendance_info(atn_user_id);

CREATE INDEX idx_employee_skills ON employee_skills(user_id, skill_name);
```

FUNCTIONS

1. Login Authentication:

This function validates user login credentials by checking if the username and password match an existing record.

```
DELIMITER //
CREATE FUNCTION check_login(username VARCHAR(100), pass
VARCHAR(100))
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE valid INT;
    SELECT COUNT(*) INTO valid
    FROM tbl_admin
    WHERE username = username
    AND password = MD5(pass);
    RETURN valid;
END //
DELIMITER ;
```

2. Task Completion Rate Calculator:

This function calculates an employee's task completion percentage for performance tracking.

```
DELIMITER //
CREATE FUNCTION calculate_completion_rate(user_id INT)
RETURNS DECIMAL(5,2)
DETERMINISTIC
```

```
BEGIN
    DECLARE total INT;
    DECLARE completed INT;
    DECLARE rate DECIMAL(5,2);

    SELECT COUNT(*) INTO total FROM task_info WHERE t_user_id =
user_id;
    SELECT COUNT(*) INTO completed FROM task_info WHERE
t_user_id = user_id AND status = 2;

    IF total = 0 THEN RETURN 0; END IF;

    SET rate = (completed / total) * 100;
    RETURN rate;
END //
DELIMITER ;
```

VIEWS

Login activity:

This view generates a report of user login attempts, showing timestamps and statuses.

```
CREATE OR REPLACE VIEW user_login_activity AS
SELECT
    la.attempt_id,
    a.fullname,
    la.username,
    la.attempt_time,
    la.status
FROM login_attempts la
```

```
LEFT JOIN tbl_admin a ON la.user_id = a.user_id;
```

Task Statistics:

This view provides insights into task distribution and progress for each employee.

```
CREATE OR REPLACE VIEW task_statistics AS
SELECT
    t_user_id,
    COUNT(*) as total_tasks,
    SUM(CASE WHEN status = 0 THEN 1 ELSE 0 END) as
incomplete_tasks,
    SUM(CASE WHEN status = 1 THEN 1 ELSE 0 END) as
in_progress_tasks,
    SUM(CASE WHEN status = 2 THEN 1 ELSE 0 END) as
completed_tasks
FROM task_info
GROUP BY t_user_id;
```

Employee Performance:
This view aggregates employee performance metrics, including task completion rates and skills.

```
CREATE OR REPLACE VIEW employee_performance AS
SELECT
    a.user_id,
    a.fullname,
```

```
    COUNT(DISTINCT t.task_id) as total_tasks,
    COUNT(DISTINCT CASE WHEN t.status = 2 THEN t.task_id
END) as completed_tasks,
    calculate_completion_rate(a.user_id) as completion_rate,
    COUNT(DISTINCT s.skill_id) as total_skills
FROM tbl_admin a
LEFT JOIN task_info t ON a.user_id = t.t_user_id
LEFT JOIN employee_skills s ON a.user_id = s.user_id
WHERE a.user_role = 2
GROUP BY a.user_id;
```

TRIGGERS

Locking Accounts After 3 Failed Attempts:

This trigger locks an account after three consecutive failed login attempts to enhance security.

```
DELIMITER //
CREATE TRIGGER lock_account_after_failures
AFTER INSERT ON login_attempts
FOR EACH ROW
BEGIN
    DECLARE failed_attempts INT;

    -- Count failed attempts in the last 10 minutes
    SELECT COUNT(*) INTO failed_attempts
    FROM login_attempts
    WHERE username = NEW.username AND status = 'FAILURE'
    AND attempt_time > NOW() - INTERVAL 10 MINUTE;

    -- Lock account if 3 failures occur
    IF failed_attempts >= 3 THEN
        UPDATE tbl_admin SET temp_password = 'LOCKED'
        WHERE username = NEW.username;
```

```
    END IF;
END;
//
DELIMITER ;
```

Task Assignment Validation:

This trigger prevents task assignments to employees who lack the required skills.

```
DELIMITER //
CREATE TRIGGER before_task_assignment
BEFORE INSERT ON task_info
FOR EACH ROW
BEGIN
    DECLARE skill_exists INT;

    IF NEW.required_skill IS NOT NULL THEN
        SELECT COUNT(*) INTO skill_exists FROM employee_skills
        WHERE user_id = NEW.t_user_id AND skill_name =
NEW.required_skill;

        IF skill_exists = 0 THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Employee
does not have the required skill';
        END IF;
    END IF;
END //
DELIMITER ;
```

Attendance Tracking:

This trigger logs every attendance action to maintain a comp

```
DELIMITER //
CREATE TRIGGER after_attendance_insert
AFTER INSERT ON attendance_info
FOR EACH ROW
BEGIN
   -- Log attendance entry
   INSERT INTO attendance_log (user_id, action_type, action_time)
   VALUES (NEW.atn_user_id, 'CLOCK_IN', NOW());
END //
DELIMITER ;
```

Employee Insert:

This trigger automatically assigns a default skill when a new employee is added.

```
DELIMITER //
CREATE TRIGGER after_employee_insert
AFTER INSERT ON tbl_admin
FOR EACH ROW
```

```
BEGIN
   IF NEW.user_role = 2 THEN
      -- Insert default skill entry for new employee
      INSERT INTO employee_skills (user_id, skill_name,
proficiency_level)
      VALUES (NEW.user_id, 'General', 'Beginner');
   END IF;
END //
DELIMITER ;
```

Employee Update:

This trigger ensures that if an existing user is promoted to an employee role, a default skill is added.

```
DELIMITER //
CREATE TRIGGER after_employee_update
AFTER UPDATE ON tbl_admin
FOR EACH ROW
BEGIN
   IF NEW.user_role = 2 AND OLD.user_role != 2 THEN
      -- Insert default skill when employee role is changed to 2
      INSERT INTO employee_skills (user_id, skill_name,
proficiency_level)
      VALUES (NEW.user_id, 'General', 'Beginner');
   END IF;
END //
DELIMITER ;
```

Employee Delete:

This trigger deletes all skills associated with an employee before their account is removed.

```
DELIMITER //
CREATE TRIGGER before_employee_delete
BEFORE DELETE ON tbl_admin
FOR EACH ROW
```

```
BEGIN
   -- Delete all skills associated with the employee
   DELETE FROM employee_skills WHERE user_id = OLD.user_id;
END //
DELIMITER ;
```

Attendance Insert (Clock-In):

This trigger logs employee check-ins for attendance tracking.

```
DELIMITER //
CREATE TRIGGER after_attendance_clockin
AFTER INSERT ON attendance_info
FOR EACH ROW
BEGIN
   -- Log the clock in event
   INSERT INTO attendance_log
   (user_id, action_type, action_time)
   VALUES
   (NEW.atn_user_id, 'CLOCK_IN', NOW());

   -- You can add additional logging or notifications here
   -- For example, update employee status, send notifications, etc.
END //
DELIMITER ;
```

Auto Logout on Employee Deletion:

This trigger automatically logs out an employee when their account is deleted.

```
DELIMITER //
CREATE TRIGGER before_employee_delete_sessions
BEFORE DELETE ON tbl_admin
FOR EACH ROW
BEGIN
   DELETE FROM session_logs WHERE user_id = OLD.user_id;
```

```
END;
//
DELIMITER ;
```

Sending Task Notifications:

This trigger sends a notification to an employee when a new task is assigned.

```
DELIMITER //
CREATE TRIGGER after_task_assignment
AFTER INSERT ON task_info
FOR EACH ROW
BEGIN
    INSERT INTO notifications (user_id, message)
    VALUES (NEW.t_user_id, CONCAT('New Task Assigned: ',
NEW.t_title));
END;
//
DELIMITER ;
```

STORED PROCEDURES

Login Check:

This procedure verifies login credentials and logs the attempt for security purposes.

```
DELIMITER //
CREATE PROCEDURE sp_check_login(
    IN p_username VARCHAR(100),
    IN p_password VARCHAR(100)
)
BEGIN
    DECLARE v_user_id INT;
    DECLARE v_valid INT;

    -- Check if the user exists and password is correct
    SELECT user_id INTO v_user_id
    FROM tbl_admin
    WHERE username = p_username AND password =
MD5(p_password);

    SET v_valid = IFNULL(v_user_id, 0);

    -- Log the attempt
    INSERT INTO login_attempts (user_id, username, status)
    VALUES (v_user_id, p_username, IF(v_valid > 0, 'SUCCESS',
'FAILURE'));

    -- Return user ID if successful, otherwise return 0
    SELECT v_valid AS user_id;
END;
//
DELIMITER ;
```

Task Reassignment (Using Cursor):

This procedure reassigns tasks from one employee to another if the original assignee is unavailable.

```
DELIMITER //
CREATE PROCEDURE reassign_tasks(IN old_user_id INT, IN
new_user_id INT)
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE task_id INT;
   DECLARE cur CURSOR FOR SELECT task_id FROM task_info
WHERE t_user_id = old_user_id AND status != 2;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
TRUE;

   START TRANSACTION;
   OPEN cur;
   read_loop: LOOP
      FETCH cur INTO task_id;
      IF done THEN LEAVE read_loop; END IF;
      UPDATE task_info SET t_user_id = new_user_id,
assignment_status = 'pending' WHERE task_id = task_id;
   END LOOP;

   CLOSE cur;
   COMMIT;
END //
DELIMITER ;
```

Log User Logout:
This procedure logs the logout time of a user session for audit purposes.

```
DELIMITER //
CREATE PROCEDURE sp_user_logout(IN p_user_id INT)
BEGIN
   UPDATE session_logs SET logout_time = NOW()
   WHERE user_id = p_user_id AND logout_time IS NULL;
END;
//
```

```
DELIMITER ;
```

GRANT PERMISSIONS

```
CREATE ROLE 'manager';
GRANT SELECT, INSERT, UPDATE, DELETE ON etmsh.* TO 'manager';

CREATE ROLE 'employee';
GRANT SELECT, UPDATE ON etmsh.task_info TO 'employee';
GRANT SELECT, INSERT, UPDATE ON etmsh.attendance_info TO
'employee';
GRANT SELECT ON etmsh.employee_skills TO 'employee';

GRANT 'manager' TO 'task_manager'@'localhost';
SET DEFAULT ROLE 'manager' FOR 'task_manager'@'localhost';
ALTER USER 'task_manager'@'localhost' IDENTIFIED BY
'securepassword';

GRANT 'employee' TO 'employee'@'localhost';
SET DEFAULT ROLE 'employee' FOR 'employee'@'localhost';

FLUSH PRIVILEGES;
```

SAMPLE DATA

Users

```
INSERT INTO `tbl_admin` (`user_id`, `fullname`, `username`, `email`, `password`, `temp_password`, `user_role`) VALUES
(1, 'Admin', 'admin', 'admin@gmail.com', '21232f297a57a5a743894a0e4a801fc3', NULL, 1),
(2, 'Bahaa', 'bahaa', 'bahaa@gmail.com', NULL, '123456', 2),
(3, 'Hadi', 'hadi', 'hadi@gmail.com', NULL, '123456', 2),
(4, 'Farah', 'farah', 'farah@gmail.com', NULL, '123456', 2);
```

Tasks

```
INSERT INTO `task_info` (`task_id`, `t_title`, `t_description`, `t_start_time`, `t_end_time`, `t_user_id`, `status`, `required_skill`) VALUES
(20, 'Communications', 'You''re assigned to handle incoming calls and other communications within the office.', '2021-03-22 12:00', '2021-03-22 13:00', 17, 2, NULL),
(21, 'Filing', 'You''re assigned to management of filing system.', '2021-03-22 10:00', '2021-03-22 15:10', 22, 0, NULL),
(22, 'Virtual Meeting', 'Please join the virtual meeting with your senior manager regarding your works on this placement.', '2021-03-22 15:00', '2021-03-22 15:20', 24, 0, NULL),
(23, 'Data Entry', 'Go through some data!', '2021-03-22 14:00', '2021-03-22 17:00', 25, 1, NULL);
```

Attendance

```
INSERT INTO `attendance_info` (`aten_id`, `atn_user_id`, `in_time`, `out_time`, `total_duration`) VALUES
(1, 1, '22-03-2021 08:00:00', '22-03-2021 17:00:00', '9 hours'),
(2, 2, '22-03-2021 08:30:00', '22-03-2021 16:30:00', '8 hours'),
(3, 3, '22-03-2021 09:00:00', NULL, NULL),
(4, 4, '22-03-2021 08:15:00', '22-03-2021 16:45:00', '8.5 hours');
```

Skills

```
INSERT INTO employee_skills (user_id, skill_name, proficiency_level) VALUES

(17, 'PHP', 'Expert'),

(17, 'MySQL', 'Intermediate'),

(18, 'JavaScript', 'Expert'),

(18, 'Python', 'Intermediate'),

(19, 'Java', 'Expert'),

(20, 'C++', 'Intermediate');
```

LOGIN MANAGEMENT:

1. Table for Login Attempts

We create a table to store login attempts, tracking **success and failure**.

Table: login_attempts

2. Stored Procedure for Login Check

This procedure checks login credentials and logs each attempt.

Stored Procedure: sp_check_login

3. View for Login Reports

We can generate a **login activity report** using a view.

View: user_login_activity

4. Trigger for Locking Accounts After 3 Failed Attempts

If a user enters the wrong password **3 times in a row**, we can **lock their account**.

Trigger: lock_account_after_failures

EMPLOYEE MANAGEMENT (RUD):

1. Adding an Employee

Tables Used:

- tbl_admin

- employee_skills (trigger adds a default skill)

SQL Statement:

```
INSERT INTO tbl_admin (fullname, username, email, password, user_role)
VALUES ('John Doe', 'johndoe', 'johndoe@gmail.com', MD5('password123'), 2);
```

Effect:

- Inserts a new employee into tbl_admin.
- A trigger (after_employee_insert) automatically adds a default skill to employee_skills.

2. Updating an Employee

Tables Used:

- tbl_admin
- employee_skills (if role changes to employee)

SQL Statement:

```
UPDATE tbl_admin
SET fullname = 'John D.', email = 'john.doe@company.com', user_role = 2
WHERE user_id = 5;
```

Effect:

- Updates the employee's name, email, or role.
- If user_role changes to 2, the trigger (after_employee_update) adds a default skill.

3. Deleting an Employee

Tables Used:

- tbl_admin
- employee_skills (deleted automatically via trigger)

SQL Statement:

```
DELETE FROM tbl_admin WHERE user_id = 5;
```

Effect:

- Removes the employee from tbl_admin.

- The trigger (before_employee_delete) deletes associated skills from employee_skills.

4. Employee Statistics Report

Tables Used:

- tbl_admin
- task_info
- employee_skills

SQL Statement:

```
SELECT * FROM employee_performance;
```

Effect:

Displays each employee's total tasks, completed tasks, completion rate, and total skills.

TASK MANAGEMENT:

1. Updating Task with Required Skill

If a task has a required skill, the before_task_assignment trigger ensures the assigned employee has the necessary skill.

Tables Used:

- task_info
- employee_skills (for skill validation)

SQL Statement:

```
UPDATE task_info
SET required_skill = 'PHP',
   t_user_id = 17
WHERE task_id = 24;
```

Effect:

- Assigns the task to a user with the "PHP" skill.
- If the user doesn't possess the required skill, the trigger raises an error:
  ➜ *"Employee does not have the required skill."*

2. Updating Task Completion Time

You might want to update the t_end_time once a task is completed.

Tables Used:

- task_info

SQL Statement:

```
UPDATE task_info
SET status = 2,
   t_end_time = NOW()
WHERE task_id = 20;
```

Effect:

- Marks the task as completed (status = 2).
- Logs the current time as the completion time.

3. Update Task Based on Employee Skills

Check and update the task's required skill if the assigned employee gains a new skill.

Tables Used:

- task_info
- employee_skills

SQL Statement:

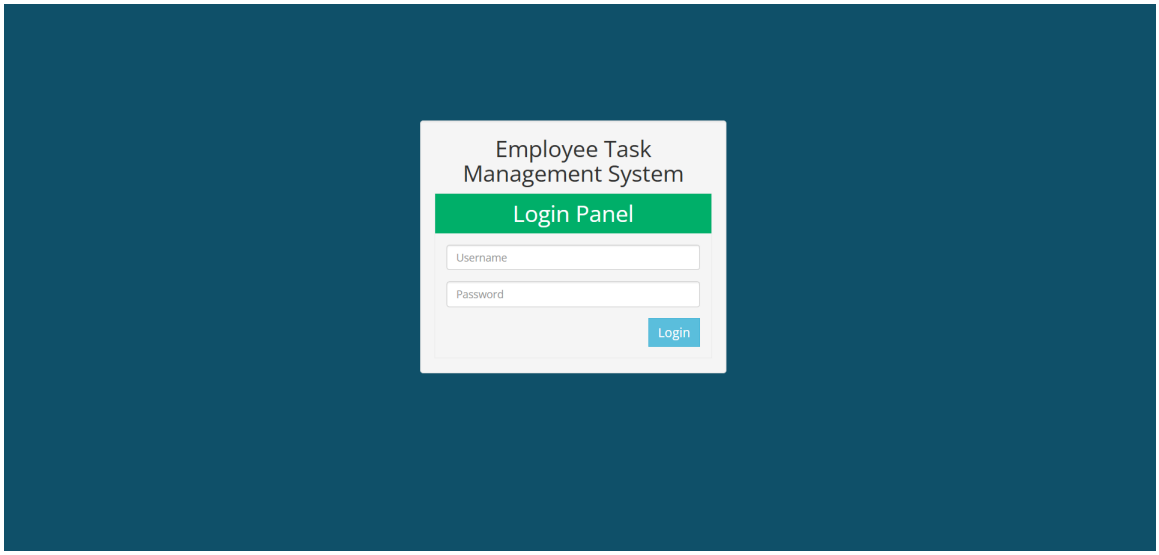
```
UPDATE task_info t
JOIN employee_skills e ON t.t_user_id = e.user_id
SET t.required_skill = 'Python'
WHERE t.task_id = 23 AND e.skill_name = 'Python';
```

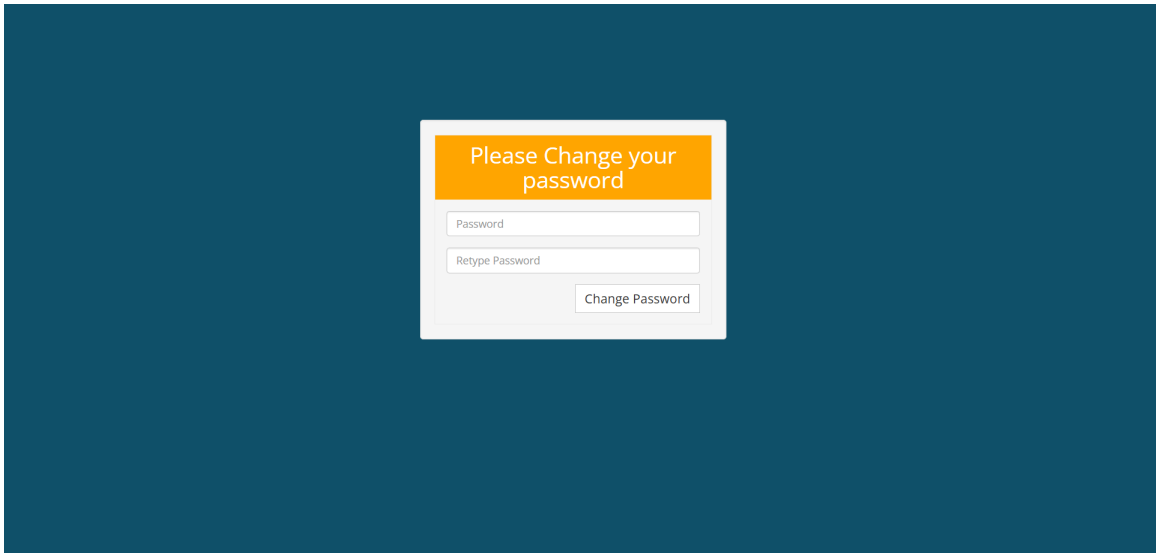
Effect:

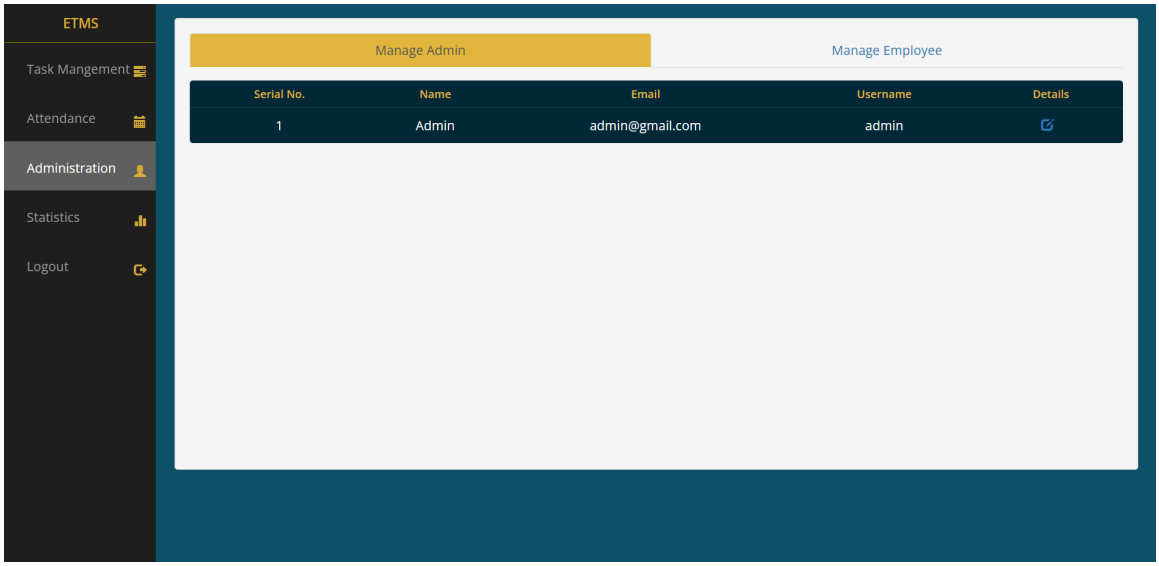
- Updates the task to require "Python" if the employee has that skill.

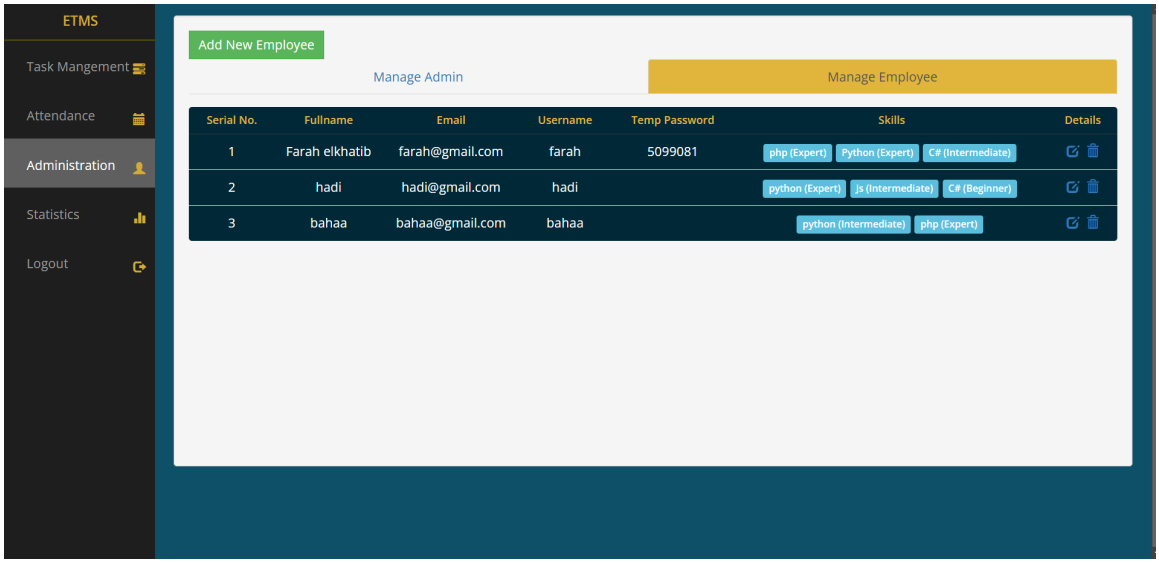
AUTOMATED DATABASE MANAGEMENT BASH SCRIPT

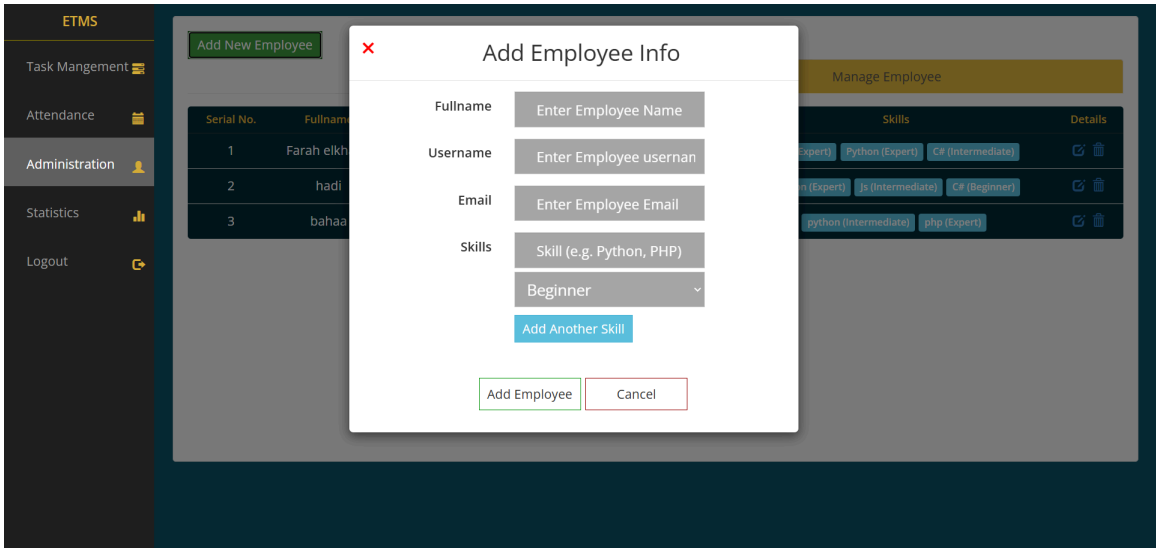
```
#!/bin/bash
DB_USER="root"
DB_PASS=""
DB_NAME="etmsh"

# Create database
echo "Creating database..."
mysql -u $DB_USER -e "CREATE DATABASE IF NOT EXISTS
$DB_NAME;"

# Import schema
echo "Importing database schema..."
mysql -u $DB_USER $DB_NAME < etmsh.sql

# Backup function
backup_database() {
    BACKUP_PATH="database_backups"
    mkdir -p $BACKUP_PATH
    BACKUP_FILE="$BACKUP_PATH/etmsh_backup_$(date
+%Y%m%d_%H%M%S).sql"

    echo "Creating backup at $BACKUP_FILE..."
    mysqldump -u $DB_USER $DB_NAME > $BACKUP_FILE
    echo "Backup completed!"
}

# Create backup
backup_database

echo "Setup completed!"
```

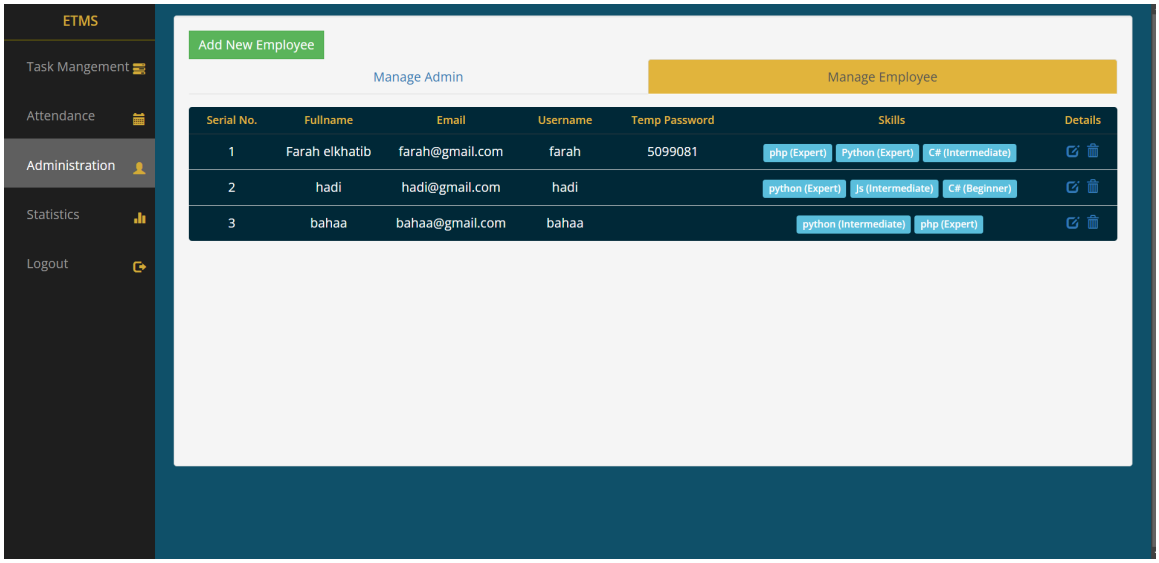

Employee Task Management System
Login Panel
Username
Password
Login

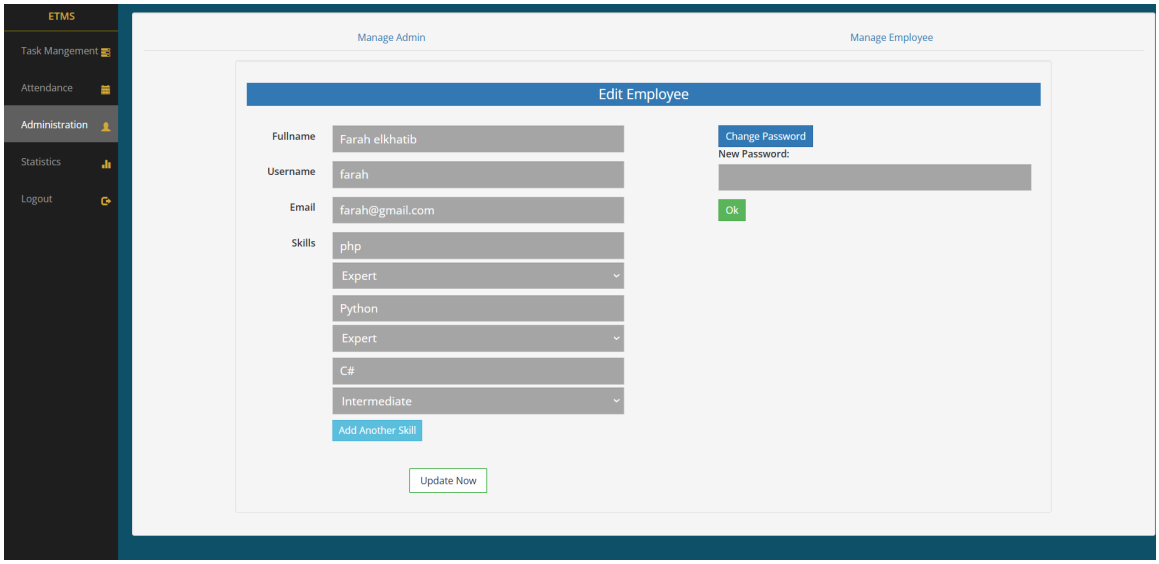

Please Change your password
Password
Retype Password
Change Password

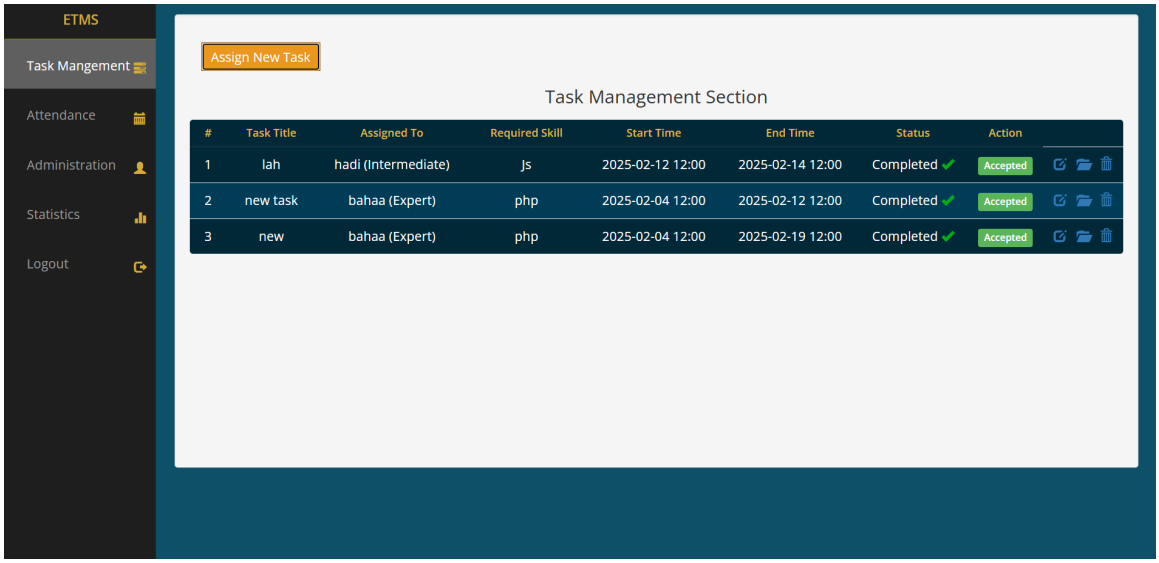

ETMS
Task Mangement
Attendance
Administration
Statistics
Logout
Manage Admin
Manage Employee
Serial No.
Name
Email
Username
Details
1
Admin
admin@gmail.com
admin

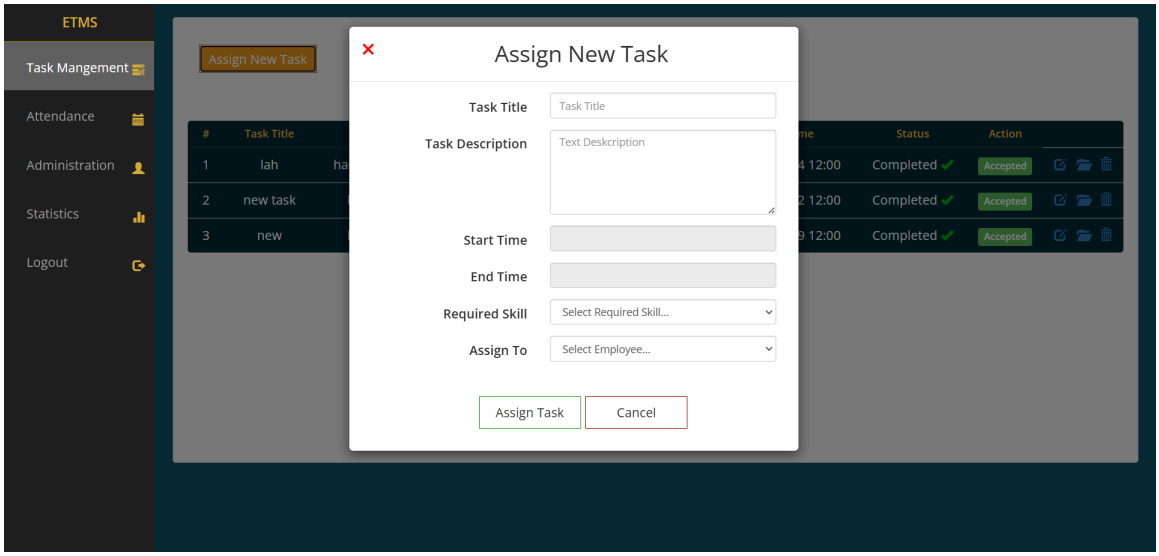

ETMS
Task Mangement
Attendance
Administration
Statistics
Logout
Add New Employee
Manage Admin
Manage Employee
Serial No.
Fullname
Email
Username
Temp Password
Skills
Details
1
Farah elkhatib
farah@gmail.com
farah
5099081
php (Expert)
Python (Expert)
C# (Intermediate)
2
hadi
hadi@gmail.com
hadi
python (Expert)
Js (Intermediate)
C# (Beginner)
3
bahaa
bahaa@gmail.com
bahaa
python (Intermediate)
php (Expert)

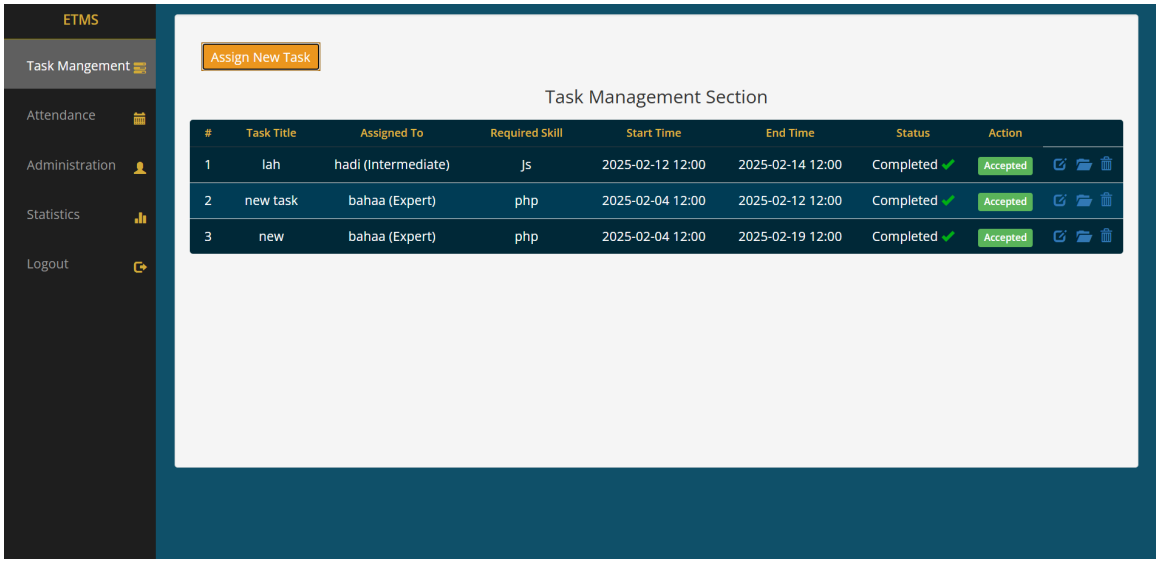

ETMS
Task Mangement
Attendance
Administration
Statistics
Logout
Add New Employee
Manage Employee
Add Employee Info
Fullname
Enter Employee Name
Username
Enter Employee usernan
Email
Enter Employee Email
Skills
Skill (e.g. Python, PHP)
Beginner
Add Another Skill
Add Employee
Cancel
Serial No.
Skills
Details

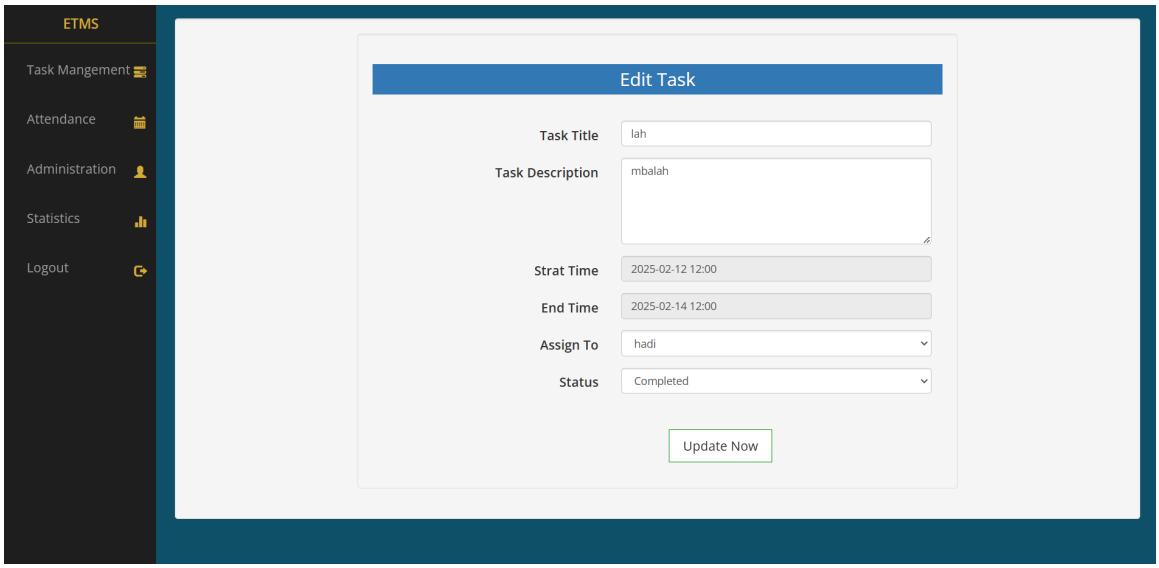

ETMS
Task Mangement
Attendance
Administration
Statistics
Logout
Add New Employee
Manage Admin
Manage Employee
Serial No.
Fullname
Email
Username
Temp Password
Skills
Details
1
Farah elkhatib
farah@gmail.com
farah
5099081
php (Expert)
Python (Expert)
C# (Intermediate)
2
hadi
hadi@gmail.com
hadi
python (Expert)
Js (Intermediate)
C# (Beginner)
3
bahaa
bahaa@gmail.com
bahaa
python (Intermediate)
php (Expert)

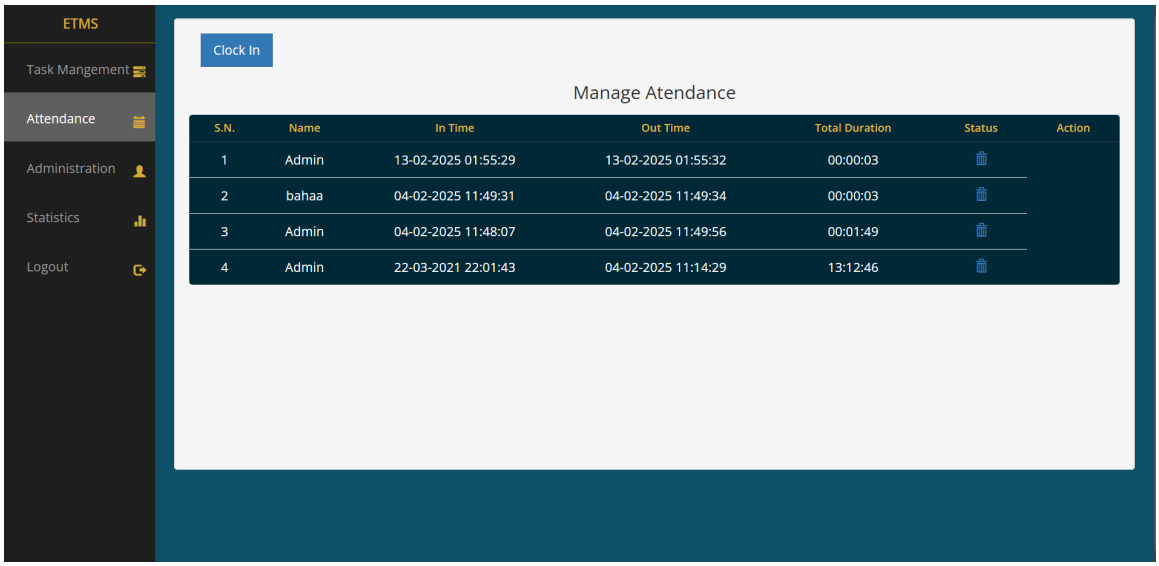

ETMS
Task Mangement
Attendance
Administration
Statistics
Logout
Manage Admin
Manage Employee
Edit Employee
Fullname
Farah elkhatib
Username
farah
Email
farah@gmail.com
Skills
php
Expert
Python
Expert
C#
Intermediate
Add Another Skill
Update Now
Change Password
New Password:
Ok

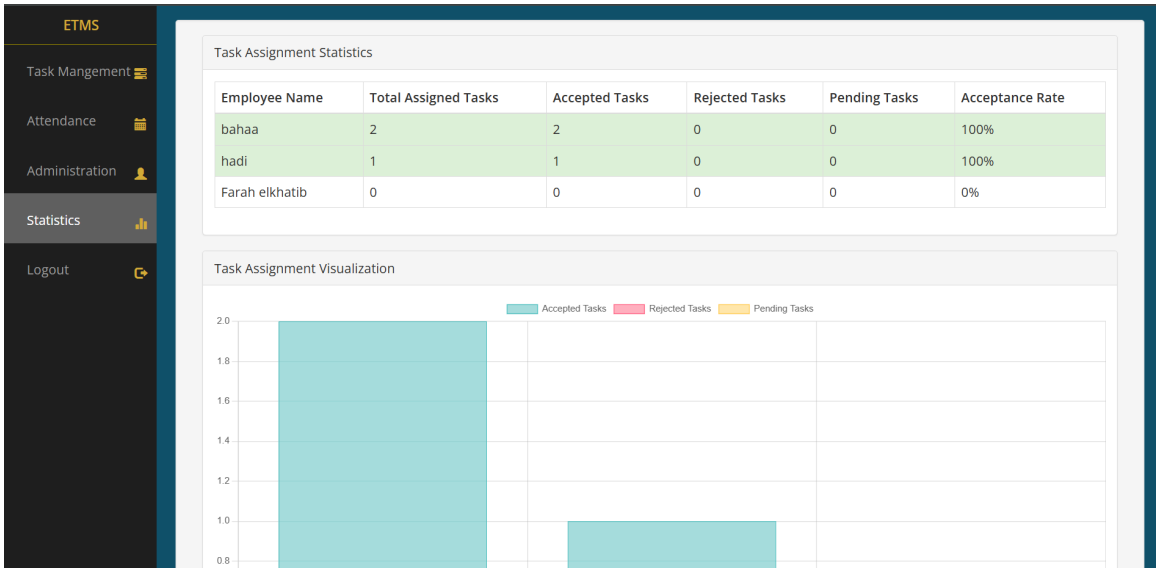

ETMS
Task Mangement
Attendance
Administration
Statistics
Logout
Assign New Task
Task Management Section
Task Title Assigned To Required Skill Start Time End Time Status Action
1 lah hadi (Intermediate) Js 2025-02-12 12:00 2025-02-14 12:00 Completed Accepted
2 new task bahaa (Expert) php 2025-02-04 12:00 2025-02-12 12:00 Completed Accepted
3 new bahaa (Expert) php 2025-02-04 12:00 2025-02-19 12:00 Completed Accepted


ETMS
Task Mangement
Attendance
Administration
Statistics
Logout
Assign New Task
Assign New Task
Task Title
Task Description
Start Time
End Time
Required Skill
Assign To
Assign Task
Cancel
Status
Action
Completed
Accepted


ETMS
Task Mangement
Attendance
Administration
Statistics
Logout
Assign New Task
Task Management Section
Task Title Assigned To Required Skill Start Time End Time Status Action
1 lah hadi (Intermediate) Js 2025-02-12 12:00 2025-02-14 12:00 Completed Accepted
2 new task bahaa (Expert) php 2025-02-04 12:00 2025-02-12 12:00 Completed Accepted
3 new bahaa (Expert) php 2025-02-04 12:00 2025-02-19 12:00 Completed Accepted


ETMS
Task Mangement
Attendance
Administration
Statistics
Logout
Edit Task
Task Title
lah
Task Description
mbalah
Strat Time
2025-02-12 12:00
End Time
2025-02-14 12:00
Assign To
hadi
Status
Completed
Update Now

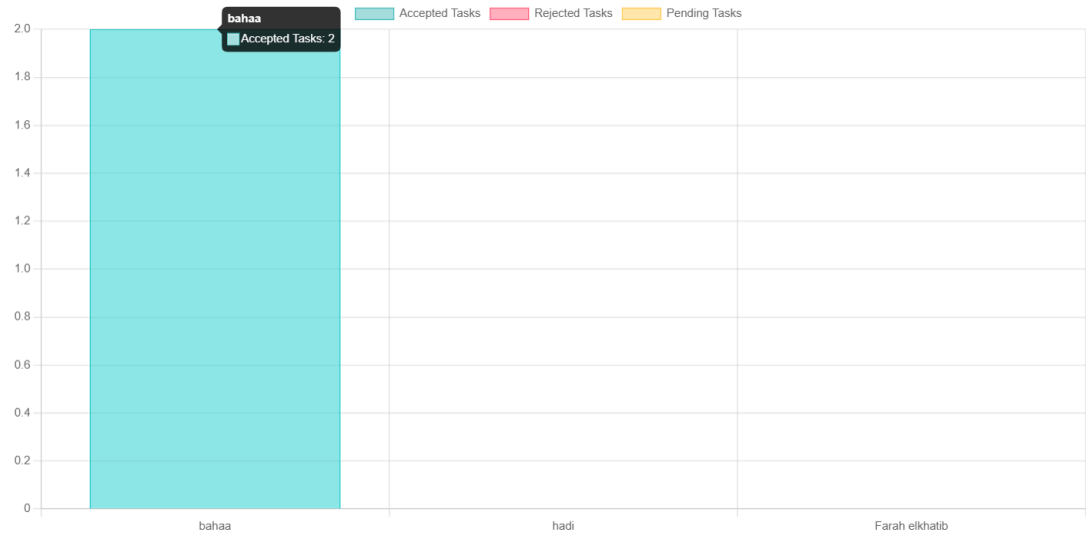

ETMS
Task Mangement
Attendance
Administration
Statistics
Logout
Clock In
Manage Atendance
S.N. Name In Time Out Time Total Duration Status Action
1 Admin 13-02-2025 01:55:29 13-02-2025 01:55:32 00:00:03
2 bahaa 04-02-2025 11:49:31 04-02-2025 11:49:34 00:00:03
3 Admin 04-02-2025 11:48:07 04-02-2025 11:49:56 00:01:49
4 Admin 22-03-2021 22:01:43 04-02-2025 11:14:29 13:12:46



Task Assignment Statistics

Employee Name	Total Assigned Tasks	Accepted Tasks	Rejected Tasks	Pending Tasks	Acceptance Rate
bahaa	2	2	0	0	100%
hadi	0	0	0	0	0%
Farah elkhatib	0	0	0	0	0%

Task Assignment Visualization



CONCLUSION

The Task Management System (TMS) offers a robust platform for employee management with an optimized database, stored procedures, triggers, and security enhancements. It ensures efficient task allocation, skill-based assignments, and attendance monitoring, making it a comprehensive workforce management tool.

Key strengths of the system include:

1. **Efficient Task Assignment and Tracking** – Assign tasks based on employee skills and track progress using stored procedures and views.
2. **Comprehensive Attendance Monitoring** – Logs in-time, out-time, and total duration for effective work hour tracking.
3. **Skill-Based Resource Allocation** – Ensures employees are assigned tasks matching their expertise with enforced validation via triggers.
4. **Performance Measurement and Analytics** – Uses functions and views to calculate task completion rates and employee productivity.
5. **Secure and Scalable Architecture** – Implements stored procedures, user roles, and audit logs for enhanced security and maintainability.
6. **Automated Task Reassignment** – A stored procedure with a cursor enables seamless task reassignment when an employee is unavailable.
7. **Index Optimization for Performance** – Uses indexed queries to speed up task retrieval and employee performance reports.

The implemented triggers, stored procedures, and views enhance the system's functionality while maintaining data integrity and providing valuable insights for management decision-making. This makes the Task Management System a powerful tool in optimizing workforce efficiency and ensuring seamless task and attendance tracking.