

Informatics and computer science, Winter 2025  
User Stories  
“ Comprehensive Human Resource Management System (HRMS)”

## 1 As a System Admin, I should be able to:

- 1) Retrieve complete information for any employee using their ID.

**Signature:**

*Name:* ViewEmployeeInfo.

*Input:* @EmployeeID int.

*Output:* : Row containing all employee details.

- 2) Add a new employee to the system.

**Signature:**

*Name:* AddEmployee.

*Input:* @FullName varchar(200), @NationalID varchar(50), @DateOfBirth date, @CountryOfBirth varchar(100), @Phone varchar(50), @Email varchar(100), @Address varchar(255), @EmergencyContactName varchar(100), @EmergencyContactPhone varchar(50), @Relationship varchar(50), @Biography varchar(max), @EmploymentProgress varchar(100), @AccountStatus varchar(50), @EmploymentStatus varchar(50), @HireDate date, @IsActive bit, @ProfileCompletion int, @DepartmentID int, @PositionID int, @ManagerID int, @ContractID int, @TaxFormID int, @SalaryTypeID int, @PayGrade varchar(50)

*Output:* Confirmation message or new @EmployeeID int.

- 3) Update an employees contact or personal details.

**Signature:**

*Name:* UpdateEmployeeInfo.

*Input:* @EmployeeID int, @Email varchar(100), @Phone varchar(20), @Address varchar(150).

*Output:* Confirmation message.

- 4) Assign system roles to employees (e.g., Payroll Officer, Manager).

**Signature:**

*Name:* AssignRole.

*Input:* '@EmployeeID int', '@RoleID int.

*Output:* Confirmation message .

- 5) Retrieve a summary of employee distribution across departments.

**Signature:**

*Name:* GetDepartmentEmployeeStats.

*Input:* Nothing.

*Output:* Table showing department and number of employees. .

- 6) Reassign an employee to a new manager.

**Signature:**

*Name:* ReassignManager.  
*Input:* @EmployeeID int, @NewManagerID int.  
*Output:* Confirmation message.

- 7) Reassign an employee to a new department or manager within the hierarchy.

**Signature:**

*Name:* ReassignHierarchy.  
*Input:* @EmployeeID int, @NewDepartmentID int, @NewManagerID int.  
*Output:* Confirmation message.

- 8) Notify all affected employees when a structural change occurs or is approved.

**Signature:**

*Name:* NotifyStructureChange.  
*Input:* @AffectedEmployees varchar(500), @Message varchar(200).  
*Output:* Confirmation message.

- 9) View the complete organizational hierarchy.

**Signature:**

*Name:* ViewOrgHierarchy.  
*Input:* **Nothing**.  
*Output:* Table showing employee names, managers, departments, positions, and hierarchy levels.

- 10) Assign shifts (with names and types) to employees for a specified term.

**Signature:**

*Name:* AssignShiftToEmployee.  
*Input:* @EmployeeID int, @ShiftID int, @StartDate date, @EndDate date.  
*Output:* Confirmation message.

- 11) Update shift statuses (Approved, Cancelled, Entered, Expired, Postponed, Rejected, Submitted).

**Signature:**

*Name:* UpdateShiftStatus.  
*Input:* @ShiftAssignmentID int, @Status varchar(20).  
*Output:* Confirmation message.

- 12) Assign shift schedules by department.

**Signature:**

*Name:* AssignShiftToDepartment.  
*Input:* @DepartmentID int, @ShiftID int, @StartDate date, @EndDate date.  
*Output:* Confirmation message.

- 13) Assign custom shifts to individual employees for unique contracts.

**Signature:**

*Name:* AssignCustomShift.  
*Input:* @EmployeeID int, @ShiftName varchar(50), @ShiftType varchar(50), @StartTime time, @EndTime time, @StartDate date, @EndDate date.  
*Output:* Confirmation message.

- 14) Configure split shifts (e.g., 812, 48).

**Signature:**

*Name:* ConfigureSplitShift.  
*Input:* @ShiftName varchar(50), @FirstSlotStart time, @FirstSlotEnd time, @SecondSlotStart time, @SecondSlotEnd time.

*Output:* Confirmation message.

- 15) Enable first in/last out attendance processing so that calculations are simplified.

**Signature:**

*Name:* EnableFirstInLastOut.

*Input:* @Enable bit.

*Output:* Confirmation message.

- 16) Tag attendance by device, terminal ID, or GPS so that records can be verified.

**Signature:**

*Name:* TagAttendanceSource.

*Input:* @AttendanceID int, @SourceType varchar(20), @DeviceID int, @Latitude decimal(10,7), @Longitude decimal(10,7).

*Output:* Confirmation message.

- 17) Allow attendance devices to store records offline and sync later so that no data is lost.

**Signature:**

*Name:* SyncOfflineAttendance.

*Input:* @DeviceID int, @EmployeeID int, @ClockTime datetime, @Type varchar(10).

*Output:* Confirmation message.

- 18) Log all clock edits with timestamps so that manipulation is prevented.

**Signature:**

*Name:* LogAttendanceEdit.

*Input:* @AttendanceID int, @EditedBy int, @OldValue datetime, @NewValue datetime, @EditTimestamp datetime.

*Output:* Confirmation message.

- 19) Apply holiday overrides to employee shifts.

**Signature:**

*Name:* ApplyHolidayOverrides.

*Input:* @HolidayID int, @employeeID int

*Output:* Confirmation message.

- 20) Create and manage user accounts and roles for payroll access.

**Signature:**

*Name:* ManageUserAccounts.

*Input:* @UserID int, @Role varchar(50), @Action varchar(20).

*Output:* Confirmation message.

## 2 As an HR Admin, I should be able to:

- 1) Create a new employment contract for an employee.

**Signature:**

*Name:* CreateContract.

*Input:* @EmployeeID int, @Type varchar(50), @StartDate date, @EndDate date.

*Output:* Confirmation message.

- 2) Renew or extend an existing contract.

**Signature:**

*Name:* RenewContract.  
*Input:* @ContractID int, @NewEndDate date.  
*Output:* Confirmation message.

- 3) Approve or reject leave requests from employees.

**Signature:**

*Name:* ApproveLeaveRequest.

*Input:* @LeaveRequestID int, @ApproverID int, @Status varchar(20).

*Output:* Confirmation message.

- 4) Assign missions or business trips to employees.

**Signature:**

*Name:* AssignMission.

*Input:* @EmployeeID int, @ManagerID int, @Destination varchar(50), @StartDate date, @EndDate date.

*Output:* Confirmation message.

- 5) Approve or reject reimbursement claims.

**Signature:**

*Name:* ReviewReimbursement.

*Input:* @ClaimID int, @ApproverID int, @Decision varchar(20).

*Output:* Confirmation message.

- 6) View all active employment contracts.

**Signature:**

*Name:* GetActiveContracts.

*Input:* nothing.

*Output:* Table of active contracts.

- 7) Retrieve a list of employees under a specific manager.

**Signature:**

*Name:* GetTeamByManager.

*Input:* @ManagerID int.

*Output:* Table containing employee names and IDs.

- 8) Update leave policy details.

**Signature:**

*Name:* UpdateLeavePolicy.

*Input:* @PolicyID int, @EligibilityRules varchar(200), @NoticePeriod int.

*Output:* Confirmation message.

- 9) Retrieve contracts nearing expiration.

**Signature:**

*Name:* GetExpiringContracts.

*Input:* @DaysBefore int.

*Output:* Table showing contracts nearing expiration.

- 10) Assign a department head.

**Signature:**

*Name:* AssignDepartmentHead.

*Input:* @DepartmentID int, @ManagerID int.

*Output:* Confirmation message.

- 11) Create a new employee profile from a hiring form.

**Signature:**

*Name:* CreateEmployeeProfile.

*Input:* @FirstName varchar(50), @LastName varchar(50), @DepartmentID int, @RoleID int, @HireDate date, @Email varchar(100), @Phone varchar(20), @NationalID varchar(50), @DateOfBirth

- date, @CountryOfBirth varchar(100).**  
*Output:*Confirmation message with new EmployeeID.
- 12) Edit or update any part of an employee profile.  
**Signature:**  
*Name:* UpdateEmployeeProfile.  
*Input:* @EmployeeID int, @FieldName varchar(50), @NewValue varchar(255).  
*Output:*Confirmation message.
- 13) Set and track employee profile completeness percentage.  
**Signature:**  
*Name:* SetProfileCompleteness.  
*Input:*@EmployeeID int, @CompletenessPercentage int.  
*Output:*Confirmation message or updated completeness value.
- 14) Search and generate compliance or diversity reports (e.g., by gender, department).  
**Signature:**  
*Name:* GenerateProfileReport.  
*Input:*@FilterField varchar(50), @FilterValue varchar(100).  
*Output:*Table showing employee data by filter.
- 15) Define multiple shift types (Normal, Split, Overnight, Mission, etc.).  
**Signature:**  
*Name:* CreateShiftType.  
*Input:* @ShiftID int, @Name varchar(100), @Type varchar(50), @Start\_Time time, @End\_Time time, @Break\_Duration int, @Shift\_Date date, @Status varchar(50).  
*Output:*Confirmation message.
- ~~–Create and manage shift names (Core Hours, Flex Time, Rotational, etc.).~~  
**Signature:**  
*Name:* CreateShiftName.  
*Input:*@ShiftName varchar(50), @ShiftTypeID int, @Description varchar(200).  
*Output:*Confirmation message.
- 16) Assign employees to rotational shifts (Morning/Evening/Night).  
**Signature:**  
*Name:* AssignRotationalShift.  
*Input:*@EmployeeID int, @ShiftCycle int, @StartDate date, @EndDate date, @status varchar(20).  
*Output:*Confirmation message.
- 18) Receive notifications when a shift assignment is nearing expiry.  
**Signature:**  
*Name:* NotifyShiftExpiry.  
*Input:*@EmployeeID int, @ShiftAssignmentID int, @ExpiryDate date.  
*Output:*Confirmation message.
- 19) Define rules for short time (late arrivals, early outs) so that deductions are consistent.  
**Signature:**  
*Name:* DefineShortTimeRules.  
*Input:*@RuleName varchar(50), @LateMinutes int, @EarlyLeaveMinutes int, @PenaltyType varchar(50).  
*Output:*Confirmation message.

- 20) Set grace periods (e.g., first 10 mins free) so that minor lateness is tolerated.  
**Signature:**  
*Name:* SetGracePeriod.  
*Input:*@Minutes int.  
*Output:*Confirmation message.
- 21) Set thresholds (e.g., >15 mins late = half-day deduction) so that penalties are automatic.  
**Signature:**  
*Name:* DefinePenaltyThreshold.  
*Input:*@LateMinutes int, @DeductionType varchar(50).  
*Output:*Confirmation message.
- 22) Define minimum/maximum hours allowed for permissions so that misuse is prevented.  
**Signature:**  
*Name:* DefinePermissionLimits.  
*Input:*@MinHours int, @MaxHours int.  
*Output:*Confirmation message.
- 23) Escalate pending requests to higher managers after a deadline so that nothing is left unresolved.  
**Signature:**  
*Name:* EscalatePendingRequests.  
*Input:*@Deadline datetime.  
*Output:*Confirmation message.
- 24) Link vacation packages to employee schedules.  
**Signature:**  
*Name:* LinkVacationToShift.  
*Input:*@VacationPackageID int, @EmployeeID int.  
*Output:*Confirmation message.
- 25) Initiate the leave configuration process.  
**Signature:**  
*Name:* ConfigureLeavePolicies.  
*Input:*Nothing.  
*Output:*Confirmation message.
- 26) Authenticate administrator credentials for leave management.  
**Signature:**  
*Name:* AuthenticateLeaveAdmin.  
*Input:*@AdminID int, @Password varchar(100).  
*Output:*Confirmation message.
- 27) Apply validated leave configurations.  
**Signature:**  
*Name:* ApplyLeaveConfiguration.  
*Input:*Nothing.  
*Output:*Confirmation message.
- 28) Update entitlement calculations and scheduling logic.  
**Signature:**  
*Name:* UpdateLeaveEntitlements.  
*Input:*@EmployeeID int.

*Output:*Confirmation message.

- 29) Set eligibility rules for each leave type.

**Signature:**

*Name:* ConfigureLeaveEligibility.

*Input:*@LeaveType varchar(50), @MinTenure int, @EmployeeType varchar(50).

*Output:*Confirmation message.

- 30) Create and manage different leave types.

**Signature:**

*Name:* ManageLeaveTypes.

*Input:*@LeaveType varchar(50), @Description varchar(200).

*Output:*Confirmation message.

- 31) Assign personalized leave entitlements.

**Signature:**

*Name:* AssignLeaveEntitlement.

*Input:*@EmployeeID int, @LeaveType varchar(50), @Entitlement decimal(5,2).

*Output:*Confirmation message.

- 32) Configure leave parameters (max duration, notice periods, approval workflows).

**Signature:**

*Name:* ConfigureLeaveRules.

*Input:*@LeaveType varchar(50), @MaxDuration int, @NoticePeriod int, @WorkflowType varchar(50).

*Output:*Confirmation message.

- 33) Configure special absence types (bereavement, jury duty).

**Signature:**

*Name:* ConfigureSpecialLeave.

*Input:*@LeaveType varchar(50), @Rules varchar(200).

*Output:*Confirmation message.

- 34) Define legal leave year and reset rules.

**Signature:**

*Name:* SetLeaveYearRules.

*Input:*@StartDate date, @EndDate date.

*Output:*Confirmation message.

- 35) Manually adjust employee leave balances.

**Signature:**

*Name:* AdjustLeaveBalance.

*Input:*@EmployeeID int, @LeaveType varchar(50), @Adjustment decimal(5,2).

*Output:*Confirmation message.

- 36) Manage user roles and permissions for leave actions.

**Signature:**

*Name:* ManageLeaveRoles.

*Input:*@RoleID int, @Permissions varchar(200).

*Output:*Confirmation message.

- 37) Finalize approved leave requests.  
**Signature:**  
*Name:*FinalizeLeaveRequest.  
*Input:*@LeaveRequestID int.  
*Output:*Confirmation message.
- 38) Override a managers decision in special cases.  
**Signature:**  
*Name:*OverrideLeaveDecision.  
*Input:*@LeaveRequestID int, @Reason varchar(200).  
*Output:*Confirmation message.
- 39) Process multiple leave requests at once.  
**Signature:**  
*Name:*BulkProcessLeaveRequests.  
*Input:*@LeaveRequestIDs varchar(500).  
*Output:*Confirmation message.
- 40) Verify medical leave documents.  
**Signature:**  
*Name:*VerifyMedicalLeave.  
*Input:*@LeaveRequestID int, @DocumentID int.  
*Output:*Confirmation message.
- 41) Update leave balances automatically after final approval.  
**Signature:**  
*Name:*SyncLeaveBalances.  
*Input:*@LeaveRequestID int.  
*Output:*Confirmation message.
- 42) Carry-forward and year-end leave processing.  
**Signature:**  
*Name:*ProcessLeaveCarryForward.  
*Input:*@Year int.  
*Output:*Confirmation message.
- 43) Sync leave with attendance system in real-time. *hint: use exception to be reflected in the attendance*  
**Signature:**  
*Name:*SyncLeaveToattendance.  
*Input:*@LeaveRequestID int.  
*Output:*Confirmation message.
- 44) Review and update insurance brackets when policies or regulations change.  
**Signature:**  
*Name:*UpdateInsuranceBrackets.  
*Input:*@BracketID int, @NewMinSalary decimal(10,2), @NewMaxSalary decimal(10,2), @NewEmployeeContribution decimal(5,2), @NewEmployerContribution decimal(5,2), @UpdatedBy int.  
*Output:*Notification message.
- 45) Review and confirm changes to payroll-related benefits and policies to ensure compliance.  
**Signature:**  
*Name:*ApprovePolicyUpdate.



*Input:*@PolicyID int, @ApprovedBy int.  
*Output:*Notification message.

### 3 As a Payroll Officer, I should be able to:

- 1) Generate payroll for a specific pay period.

**Signature:**

*Name:* GeneratePayroll.

*Input:*@StartDate date, @EndDate date.

*Output:* Table containing calculated payroll records.

- 2) Add or modify allowances and deductions for an employee.

**Signature:**

*Name:* AdjustPayrollItem.

*Input:*@PayrollID int, @Type varchar(50), @Amount decimal(10,2), @duration(mins) int, @time-zone varchar(20).

*Output:* Confirmation message.

- 3) Compute net salary for a specific payroll record.

**Signature:**

*Name:* CalculateNetSalary.

*Input:* @PayrollID int.

*Output:* '@NetSalary decimal(10,2).

- 4) Apply payroll policies (bonus, overtime, deductions).

**Signature:**

*Name:* ApplyPayrollPolicy. .

*Input:* @PolicyID int, @PayrollID int, @type varchar(20), @description varchar(50).

*Output:* Confirmation message.

- 5) Retrieve payroll summary for a given month. .

**Signature:**

*Name:*GetMonthlyPayrollSummary.

*Input:* @Month int, @Year int .

*Output:* Total salary expenditure.

~~Add a new allowance or deduction to payroll.~~

**Signature:**

~~*Name:* AddAllowanceDeduction.~~

~~*Input:* @PayrollID int, @Type varchar(50), @Amount decimal(10,2).~~

~~*Output:* Confirmation message.~~

- 6) Retrieve payroll history for a specific employee.

**Signature:**

*Name:*GetEmployeePayrollHistory.

*Input:* @EmployeeID int.

*Output:* Table of past payroll records.

- 8) Get list of employees eligible for bonuses.

**Signature:**

*Name:*GetBonusEligibleEmployees.

*Input:* @Eligibility\_criteria.

*Output:* Table of eligible employees.

- 9) Update the salary type of an employee.

**Signature:**

- Name:* UpdateSalaryType.  
*Input:* @EmployeeID int, @SalaryTypeID int.  
*Output:* Confirmation message.
- 10) Retrieve payroll summary for a specific department.  
**Signature:**  
*Name:* GetPayrollByDepartment.  
*Input:* @DepartmentID int, @Month int, @Year int.  
*Output:* Table showing department-level payroll totals.
- 11) Block payroll processing if missed punches remain unresolved so that salary calculations are accurate.  
**Signature:**  
*Name:* ValidateAttendanceBeforePayroll.  
*Input:* @PayrollPeriodID int.  
*Output:* Table of employees with unresolved punches.
- 12) Sync attendance records daily to payroll and benefits so that all systems remain aligned.  
**Signature:**  
*Name:* SyncAttendanceToPayroll.  
*Input:* @SyncDate date.  
*Output:* Confirmation message.
- 13) Ensure only accepted permissions affect payroll so that calculations remain accurate.  
**Signature:**  
*Name:* SyncApprovedPermissionsToPayroll.  
*Input:* @PayrollPeriodID int.  
*Output:* Confirmation message.
- 14) Configure pay grades and salary bands.  
**Signature:**  
*Name:* ConfigurePayGrades.  
*Input:* @GradeName varchar(50), @MinSalary decimal(10,2), @MaxSalary decimal(10,2).  
*Output:* Confirmation message.
- 15) Configure shift differentials and special allowances.  
**Signature:**  
*Name:* ConfigureShiftAllowances.  
*Input:* @ShiftType varchar(50), @AllowanceName varchar(50), @Amount decimal(10,2).  
*Output:* Confirmation message.
- 16) Enable multi-currency payroll for international employees.  
**Signature:**  
*Name:* EnableMultiCurrencyPayroll.  
*Input:* @CurrencyCode varchar(10), @ExchangeRate decimal(10,4).  
*Output:* Confirmation message.
- 17) Define and update tax rules for payroll compliance.  
**Signature:**  
*Name:* ManageTaxRules.  
*Input:* @TaxRuleName varchar(50), @CountryCode varchar(10), @Rate decimal(5,2), @Exemption decimal(10,2).  
*Output:* Confirmation message.
- 18) Approve payroll configuration changes to prevent unauthorized adjustments.  
**Signature:**  
*Name:* ApprovePayrollConfigChanges.  
*Input:* @ConfigID int, @ApproverID int, @Status varchar(20).  
*Output:* Confirmation message.

- 19) Configure signing bonuses for new hires.  
**Signature:**  
*Name:* ConfigureSigningBonus.  
*Input:*@EmployeeID int, @BonusAmount decimal(10,2), @EffectiveDate date.  
*Output:* Confirmation message.
- 20) Configure termination and resignation compensations.  
**Signature:**  
*Name:* ConfigureTerminationBenefits.  
*Input:*@EmployeeID int, @CompensationAmount decimal(10,2), @EffectiveDate date, @Reason varchar(50).  
*Output:* Confirmation message.
- 21) Configure insurance brackets with contribution percentages.  
**Signature:**  
*Name:* ConfigureInsuranceBrackets.  
*Input:*@InsuranceType varchar(50), @MinSalary decimal(10,2), @MaxSalary decimal(10,2), @EmployeeContribution decimal(5,2), @EmployerContribution decimal(5,2).  
*Output:* Confirmation message.
- 22) Update existing insurance brackets when policies change.  
**Signature:**  
*Name:* UpdateInsuranceBrackets.  
*Input:*@BracketID int, @MinSalary decimal(10,2), @MaxSalary decimal(10,2), @EmployeeContribution decimal(5,2), @EmployerContribution decimal(5,2).  
*Output:* Confirmation message.
- 23) Configure payroll rules and structure (salary types, deductions, bonuses, etc.) so the system enforces organizational policy.  
**Signature:**  
*Name:* ConfigurePayrollPolicies.  
*Input:*@PolicyType varchar(50), @PolicyDetails nvarchar(max), @effectivedate date  
*Output:* Confirmation message.
- 24) Define and manage pay grades, salary bands, and compensation limits to ensure consistency and fairness.  
**Signature:**  
*Name:* DefinePayGrades.  
*Input:*@GradeName varchar(50), @MinSalary decimal(10,2), @MaxSalary decimal(10,2), @CreatedBy int.  
*Output:* Confirmation message.
- 25) Configure escalation workflows for deductions or overpayments requiring higher-level approvals.  
**Signature:**  
*Name:* ConfigureEscalationWorkflow.  
*Input:*@ThresholdAmount decimal(10,2), @ApproverRole varchar(50), @CreatedBy int.  
*Output:* Confirmation message.
- 26) Define employee pay types (hourly, daily, monthly, etc.) for correct salary calculations.  
**Signature:**  
*Name:* DefinePayType.  
*Input:*@EmployeeID int, @PayType varchar(50), @EffectiveDate date.  
*Output:* Confirmation message.

- 27) Configure overtime rules (e.g., rate multipliers) to ensure fair compensation.

**Signature:**

*Name:* ConfigureOvertimeRules.

*Input:*@DayType varchar(20), @Multiplier decimal(3,2), @hourspermonth int.

*Output:* Confirmation message.

- 28) Set shift differentials and special condition allowances (e.g., night shift, hazard pay).

**Signature:**

*Name:* ConfigureShiftAllowance.

*Input:*@ShiftType varchar(20), @AllowanceAmount decimal(10,2), @CreatedBy int.

*Output:* Confirmation message.

~~Enable multi-currency payroll for international employees.~~

**Signature:**

~~*Name:* ConfigureMultiCurrency.~~

~~*Input:*@CurrencyCode varchar(10), @ExchangeRate decimal(10,4), @EffectiveDate date.~~

~~*Output:* Confirmation message.~~

- 30) Configure policies for signing bonuses and payroll initiation for new hires.

**Signature:**

*Name:* ConfigureSigningBonusPolicy.

*Input:*@BonusType varchar(50), @Amount decimal(10,2), @EligibilityCriteria nvarchar(max).

*Output:* Confirmation message.

~~Configure insurance brackets and contributions for employees and employers.~~

**Signature:**

~~*Name:* ConfigureInsuranceBrackets.~~

~~*Input:*@BracketName varchar(50), @MinSalary decimal(10,2), @MaxSalary decimal(10,2), @EmployeeContribution decimal(5,2), @EmployerContribution decimal(5,2).~~

~~*Output:* Confirmation message.~~

- 32) Generate tax statements for employees annually.

**Signature:**

*Name:* GenerateTaxStatement.

*Input:*@EmployeeID int, @TaxYear int.

*Output:* Table containing tax summary for the employee.

- 33) Approve configuration changes made by Payroll Specialists before they take effect.

**Signature:**

*Name:* ApprovePayrollConfiguration.

*Input:*@ConfigID int, @ApprovedBy int.

*Output:* Confirmation message.

- 34) Modify or correct payroll entries when authorized.

**Signature:**

*Name:* ModifyPastPayroll.

*Input:*@PayrollRunID int, @EmployeeID int, @FieldName varchar(50), @NewValue decimal(10,2), @ModifiedBy int.

*Output:* Confirmation message.

## 4 As a Line Manager, I should be able to:

- 1) Approve or deny leave requests from team members.  
**Signature:**  
*Name:* ReviewLeaveRequest .  
*Input:* @LeaveRequestID int, @ManagerID int, @Decision varchar(20).  
*Output:* @LeaveRequestID int, @ManagerID int, @Decision varchar(20).
- 2) Assign employees to work shifts.  
**Signature:**  
*Name:* AssignShift.  
*Input:* @EmployeeID int, @ShiftID int .  
*Output:* Confirmation message.
- 3) View attendance summary of team members.  
**Signature:**  
*Name:* ViewTeamAttendance.  
*Input:* @ManagerID int, @DateRangeStart date, @DateRangeEnd date .  
*Output:* Table of attendance records.
- 4) Send notifications to team members.  
**Signature:**  
*Name:* SendTeamNotification.  
*Input:* @ManagerID int, @MessageContent varchar(255), @UrgencyLevel varchar(50) .  
*Output:* Confirmation message.
- 5) Approve completion of a mission assigned to an employee.  
**Signature:**  
*Name:* ApproveMissionCompletion.  
*Input:* @MissionID int, @ManagerID int, @Remarks varchar(200). .  
*Output:* Confirmation message.
- 6) Request a replacement for an unavailable employee.  
**Signature:**  
*Name:* RequestReplacement.  
*Input:* @EmployeeID int, @Reason varchar(150). .  
*Output:* Confirmation message.
- 7) Retrieve department summary including active projects.  
**Signature:**  
*Name:* ViewDepartmentSummary.  
*Input:* @DepartmentID int. .  
*Output:* Summary table showing employee count and projects.
- 8) Reassign a shift for an employee.  
**Signature:**  
*Name:* ReassignShift.  
*Input:* @EmployeeID int, @OldShiftID int, @NewShiftID int. .  
*Output:* Confirmation message.
- 9) Retrieve list of pending leave requests.  
**Signature:**  
*Name:* GetPendingLeaveRequests.  
*Input:* @ManagerID int. .  
*Output:* Table of pending requests.
- 10) View team-level statistics and reporting metrics.  
**Signature:**  
*Name:* GetTeamStatistics.  
*Input:* @ManagerID int. .  
*Output:* Table showing team size, average salary, and span of control.

- 11) View team members profiles (excluding sensitive data).  
**Signature:**  
*Name:*ViewTeamProfiles.  
*Input:* @ManagerID int. .  
*Output:* Table showing team members basic info.
- 12) View summary of team (roles, tenure, departments).  
**Signature:**  
*Name:*GetTeamSummary.  
*Input:* @ManagerID int. .  
*Output:* Table summarizing role distribution and tenure.
- 13) Filter team members by skill or role.  
**Signature:**  
*Name:*FilterTeamProfiles.  
*Input:* @ManagerID int, @Skill varchar(50), @RoleID int. .  
*Output:* Table showing matching employees.
- 14) View certifications and skills of team members.  
**Signature:**  
*Name:*ViewTeamCertifications.  
*Input:* @ManagerID int. .  
*Output:* Table showing skill and certification info.
- 15) Add manager-specific notes (visible only to HR).  
**Signature:**  
*Name:*AddManagerNotes.  
*Input:* @EmployeeID int, @ManagerID int, @Note varchar(500). .  
*Output:* Confirmation message.
- 16) Record attendance manually (with an audit trail) so that missing punches can be corrected.  
**Signature:**  
*Name:*RecordManualAttendance.  
*Input:* @EmployeeID int, @Date date, @ClockIn time, @ClockOut time, @Reason varchar(200), @RecordedBy int. .  
*Output:* Confirmation message.
- 17) Review automatically flagged missed punches (Clock in / Clock out) so that I can take action.  
**Signature:**  
*Name:*ReviewMissedPunches.  
*Input:* @ManagerID int, @Date date. .  
*Output:* Table of flagged attendance exceptions.
- 18) Approve or reject time management requests so that exceptions are controlled.  
**Signature:**  
*Name:*ApproveTimeRequest.  
*Input:* @RequestID int, @ManagerID int, @Decision varchar(20), @Comments varchar(200). .  
*Output:* Confirmation message.
- 19) Review leave requests assigned to me.  
**Signature:**  
*Name:*ViewLeaveRequest.  
*Input:* @LeaveRequestID int, @ManagerID int. .  
*Output:* Leave request details.
- 20) Approve a leave request.  
**Signature:**  
*Name:*ApproveLeaveRequest.  
*Input:*@LeaveRequestID int, @ManagerID int. .  
*Output:* Confirmation message.

- 21) Reject a leave request.  
**Signature:**  
*Name:* RejectLeaveRequest.  
*Input:* @LeaveRequestID int, @ManagerID int, @Reason varchar(200). .  
*Output:* Confirmation message.
- 22) Delegate leave approval authority.  
**Signature:**  
*Name:* DelegateLeaveApproval.  
*Input:* @ManagerID int, @DelegateID int, @StartDate date, @EndDate date. .  
*Output:* Confirmation message.
- 23) Flag irregular leave patterns in team members.  
**Signature:**  
*Name:* FlagIrregularLeave.  
*Input:* @EmployeeID int, @ManagerID int, @PatternDescription varchar(200). .  
*Output:* Confirmation message.
- 24) Receive notifications when a new leave request is assigned to me.  
**Signature:**  
*Name:* NotifyNewLeaveRequest.  
*Input:* @ManagerID int, @RequestID int. .  
*Output:* Notification message.

## 5 As an Employee, I should be able to:

- 1) Submit a leave request.  
**Signature:**  
*Name:* SubmitLeaveRequest.  
*Input:* @EmployeeID int, @LeaveTypeID int, @StartDate date, @EndDate date, @Reason varchar(100).  
*Output:* Confirmation message.
- 2) Check my current leave balance.  
**Signature:**  
*Name:* GetLeaveBalance.  
*Input:* @EmployeeID int.  
*Output:* Remaining leave days.
- 3) Record my attendance for a workday.  
**Signature:**  
*Name:* RecordAttendance.  
*Input:* @EmployeeID int, @ShiftID int, @EntryTime time, @ExitTime time.  
*Output:* Confirmation message.
- 4) Submit a reimbursement request.  
**Signature:**  
*Name:* SubmitReimbursement.  
*Input:* @EmployeeID int, @ExpenseType varchar(50), @Amount decimal(10,2).  
*Output:* Confirmation message.
- 5) Add a personal skill.  
**Signature:**  
*Name:* AddEmployeeSkill.  
*Input:* @EmployeeID int, @SkillName varchar(50).  
*Output:* Confirmation message.
- 6) View assigned shifts.  
**Signature:**

- Name:* ViewAssignedShifts.  
*Input:* @EmployeeID int.  
*Output:* Table showing shift dates, timings, and assigned locations.
- 7) View all my active and past contracts.  
**Signature:**  
*Name:* ViewMyContracts.  
*Input:* @EmployeeID int.  
*Output:* Table of contracts.
- 8) View my payroll history.  
**Signature:**  
*Name:* ViewMyPayroll.  
*Input:* @EmployeeID int.  
*Output:* Table of payroll records.
- 9) Update my contact details.  
**Signature:**  
*Name:* UpdatePersonalDetails.  
*Input:* @EmployeeID int, @Phone varchar(20), @Address varchar(150).  
*Output:* Confirmation message.
- 10) View my assigned missions.  
**Signature:**  
*Name:* ViewMyMissions.  
*Input:* @EmployeeID int.  
*Output:* Table of assigned missions and their status.
- 11) View full employee profile.  
**Signature:**  
*Name:* ViewEmployeeProfile.  
*Input:* @EmployeeID int.  
*Output:* Row with personal details, job info.
- 12) Update contact information (phone, address).  
**Signature:**  
*Name:* Update contact information (phone, address).  
*Input:* @EmployeeID int, @RequestType varchar(50), @NewValue varchar(100).  
*Output:* Confirmation message.
- 13) View employment timeline (hire date, promotions, transfers).  
**Signature:**  
*Name:* ViewEmploymentTimeline.  
*Input:*@EmployeeID int.  
*Output:* Table showing historical records.
- 14) Add or update emergency contact details.  
**Signature:**  
*Name:* UpdateEmergencyContact.  
*Input:*@EmployeeID int, @ContactName varchar(100), @Relation varchar(50), @Phone varchar(20).  
*Output:* Confirmation message.
- 15) Request employment verification or HR letters.  
**Signature:**  
*Name:* RequestHRDocument.  
*Input:*@EmployeeID int, @DocumentType varchar(50).  
*Output:* Confirmation message.
- 16) Get notified of profile updates or document changes.  
**Signature:**



- Name:* NotifyProfileUpdate.  
*Input:* @EmployeeID int, @notificationType varchar(50).  
*Output:* Confirmation message.
- 17) Work under flex-in/flex-out rules to complete required hours within flexible timing.  
**Signature:**  
*Name:* LogFlexibleAttendance.  
*Input:* @EmployeeID int, @Date date, @CheckIn time, @CheckOut time.  
*Output:* Confirmation message or calculated total working hours.
- 18) Be notified when I miss a punch so that I can correct it.  
**Signature:**  
*Name:* NotifyMissedPunch.  
*Input:* @EmployeeID int, @Date date.  
*Output:* Notification message.
- 19) Clock in/out multiple times a day so that breaks and split shifts are tracked.  
**Signature:**  
*Name:* RecordMultiplePunches.  
*Input:* @EmployeeID int, @ClockInOutTime datetime, @Type varchar(10).  
*Output:* Confirmation message.
- 20) Submit a correction request for missed or incorrect punches so that my attendance is accurate.  
**Signature:**  
*Name:* SubmitCorrectionRequest.  
*Input:* @EmployeeID int, @Date date, @CorrectionType varchar(50), @Reason varchar(200).  
*Output:* Confirmation message.
- 21) View the status of my correction or overtime requests so that I know if they are approved or rejected.  
**Signature:**  
*Name:* ViewRequestStatus.  
*Input:* @EmployeeID int.  
*Output:* Table of request statuses.
- ~~Submit a new leave request.~~  
**Signature:**  
~~*Name:* SubmitLeaveRequest.~~  
~~*Input:* @EmployeeID int, @LeaveTypeID int, @StartDate date, @EndDate date, @Reason varchar(100).~~  
~~*Output:* Confirmation message.~~
- 22) Attach documents to a leave request.  
**Signature:**  
*Name:* AttachLeaveDocuments.  
*Input:*@LeaveRequestID int, @FilePath varchar(200).  
*Output:* Confirmation message.
- 24) Modify an existing leave request.  
**Signature:**  
*Name:* ModifyLeaveRequest.  
*Input:*@LeaveRequestID int, @StartDate date, @EndDate date, @Reason varchar(100).  
*Output:* Confirmation message.
- 25) Cancel a leave request before final approval.  
**Signature:**  
*Name:* CancelLeaveRequest.  
*Input:*@LeaveRequestID int.  
*Output:* Confirmation message.

- 26) View current leave balance.  
**Signature:**  
*Name:* ViewLeaveBalance.  
*Input:*@EmployeeID int.  
*Output:* Remaining leave days.
- 27) View past leave requests and their status.  
**Signature:**  
*Name:* ViewLeaveHistory.  
*Input:*@EmployeeID int.  
*Output:* Table of leave requests.
- 28) Submit leave request after leave taken (within allowed period).  
**Signature:**  
*Name:* SubmitLeaveAfterAbsence.  
*Input:*@EmployeeID int, @LeaveTypeID int, @StartDate date, @EndDate date, @Reason varchar(100).  
*Output:* Confirmation message.
- 29) Receive notifications when my leave request is approved, rejected, returned for correction, or modified.  
**Signature:**  
*Name:* NotifyLeaveStatusChange.  
*Input:*@EmployeeID int, @RequestID int, @Status varchar(20).  
*Output:* Notification message.
- ~~Be notified when I miss a punch (Clock in/ Clock out) so that I can correct it.~~  
**Signature:**  
*Name:* NotifyMissedPunch.  
*Input:*@EmployeeID int, @Date date.  
*Output:* Notification message.
- ~~Get notified of profile updates or document changes.~~  
**Signature:**  
*Name:* NotifyProfileUpdate.  
*Input:*@EmployeeID int, @ChangeType varchar(50).  
*Output:* Notification message.