KNOWLEDGE INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai
Kakapalayam (PO), Salem - 637 504.

DEPARTMENT

of

COMPUTER SCIENCE AND ENGINEERING

MINI PROJECT

Project Title	**- BPO Management System**
Subject Name	**- Object Oriented Software Engineering**
Subject Code	**- CCS356**
Year/Sem/Sec	**- III/VI/C**
Date	**- 30.04.2025**

REGISTER NUMBER	TEAM MEMBERS
611222104162	VIGNESHWARAN M
611222104163	VIGNESHWARAN M
611222104164	VIMAL V R
611222104165	VISHNUPRIYA V

FACULTY SIGNATURE

AIM:

To design and implement a BPO Management System that allows administrative and HR staff to manage projects, employee shifts, tasks, client communication, and performance metrics in an organized and automated manner.

PROBLEM STATEMENT:

The traditional manual method of managing operations in a BPO firm leads to inefficiencies in shift allocation, tracking task progress, handling client requests, and employee performance reviews. The objective is to develop an automated system that streamlines BPO operations, improves transparency, and enhances productivity.

EXPLANATION:

Step 1: Requirements Analysis

- **Employee registration/login**: Employees need a secure login system to access their dashboard, view tasks, and mark attendance. Registration is restricted to authorized admin approval.
- **Client management**: The system should allow adding, updating, and tracking client projects. This helps in maintaining communication records and project deadlines efficiently.
- **Shift scheduling**: Shift management ensures that employees are allocated to proper time slots based on their role, availability, and team size requirements.
- **Task assignment and tracking**: Admins should be able to assign tasks to employees, who in turn can update progress. Tasks need to reflect status such as "pending", "in progress", or "completed".
- **Performance evaluation**: The system must include metrics to evaluate employee performance based on productivity, punctuality, and task quality.
- **Attendance tracking**: Each employee should mark daily attendance, and the system logs it to track presence, lateness, or absences automatically.
- **Reporting tools**: Admins and HR should generate periodic reports on employee performance, attendance, and client task completion for decision-making.

Step 2: System Architecture Design

- **Presentation Layer**: This layer includes web pages or mobile screens where employees, admins, HR, and clients interact with the system using a user-friendly interface.
- **Application Layer**: It acts as the brain of the system, managing task logic, validating actions, assigning shifts, and generating performance scores based on rules.

- **Data Layer**: This stores all persistent data such as employee details, shift timings, client info, and task records in a secure, relational database.

Step 3: Database Design

- **Employees**: Stores basic info like employee ID, name, login credentials, department, and contact details used for authentication and communication.
- **Clients**: Contains client ID, organization name, contact person, and the associated project details to track outsourcing activities.
- **Tasks**: Tracks all tasks with task IDs, assigned employee, deadline, status, and descriptions to monitor ongoing work.
- **Shifts**: Records shift timings, which team is scheduled, and shift status (active/inactive), ensuring proper resource planning.
- **Attendance**: Logs daily check-in/check-out times or presence status of employees, forming the basis for payroll and performance review.
- **Performance**: Contains ratings and evaluation criteria based on work output, client feedback, and attendance trends.
- **Users**: Stores usernames and passwords to verify access levels and roles like admin, HR, or employee.

Step 4: Implementation Plan

Phase 1: Core Modules

- **Setup environment:** Install required software stacks (e.g., database, server, framework) and set up the development workspace.
- **User authentication:** Implement secure login/logout and session control for different user types like admin, employee, and client.
- **Employee and client management:** Develop CRUD operations (Create, Read, Update, Delete) for employee and client profiles.
- **Shift scheduling logic:** Code functionality to assign, view, and update shifts for different teams, with constraints like overlap avoidance.

Phase 2: Task & Attendance Management

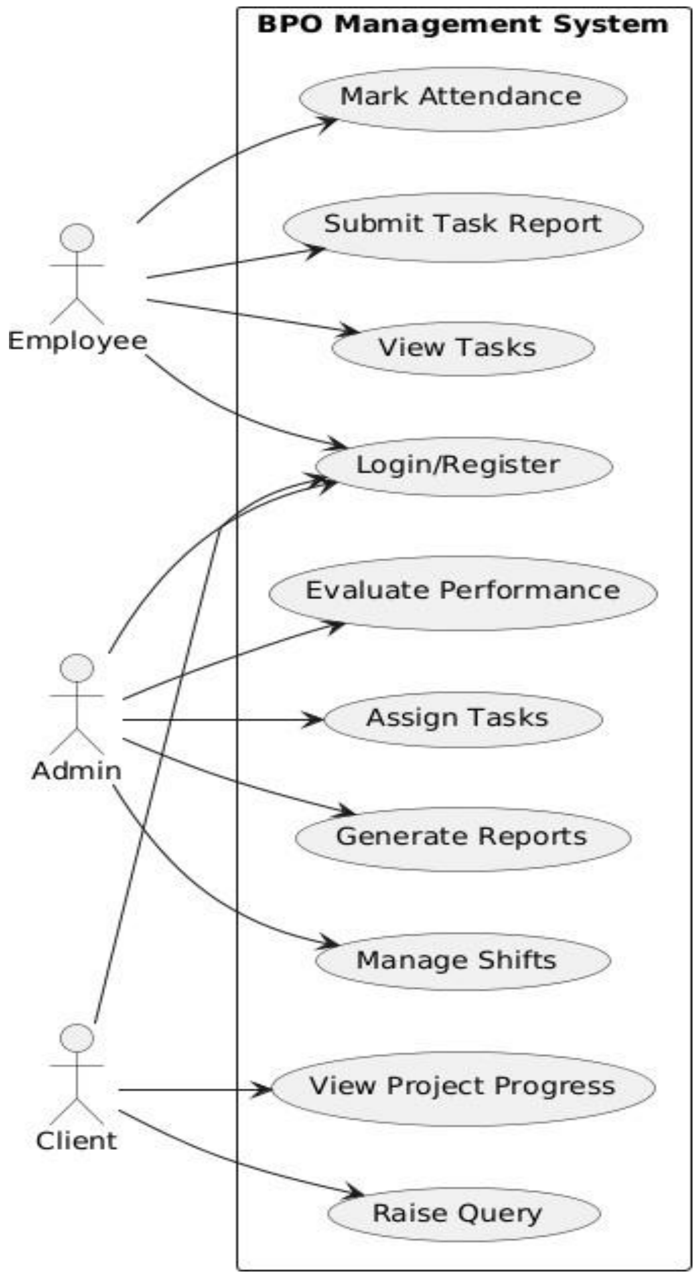
- **Assigning and updating tasks:** Admins can assign tasks with deadlines; employees update progress, and the system tracks completion automatically.
- **Daily attendance marking:** Enable employees to mark presence, which is timestamped and logged into the attendance database daily.

- **Performance metrics generation:** Analyze attendance and task performance to generate periodic ratings and evaluations per employee.

Phase 3: Administrative Features

- **Reports for tasks, attendance, and performance:** Generate visual and downloadable reports that help HR and management with insights.
- **Notifications to employees:** Notify users of new tasks, shift changes, and performance updates via the dashboard or email alerts.
- **Admin dashboard for oversight:** Centralized panel where admins can manage employees, shifts, tasks, reports, and system settings.

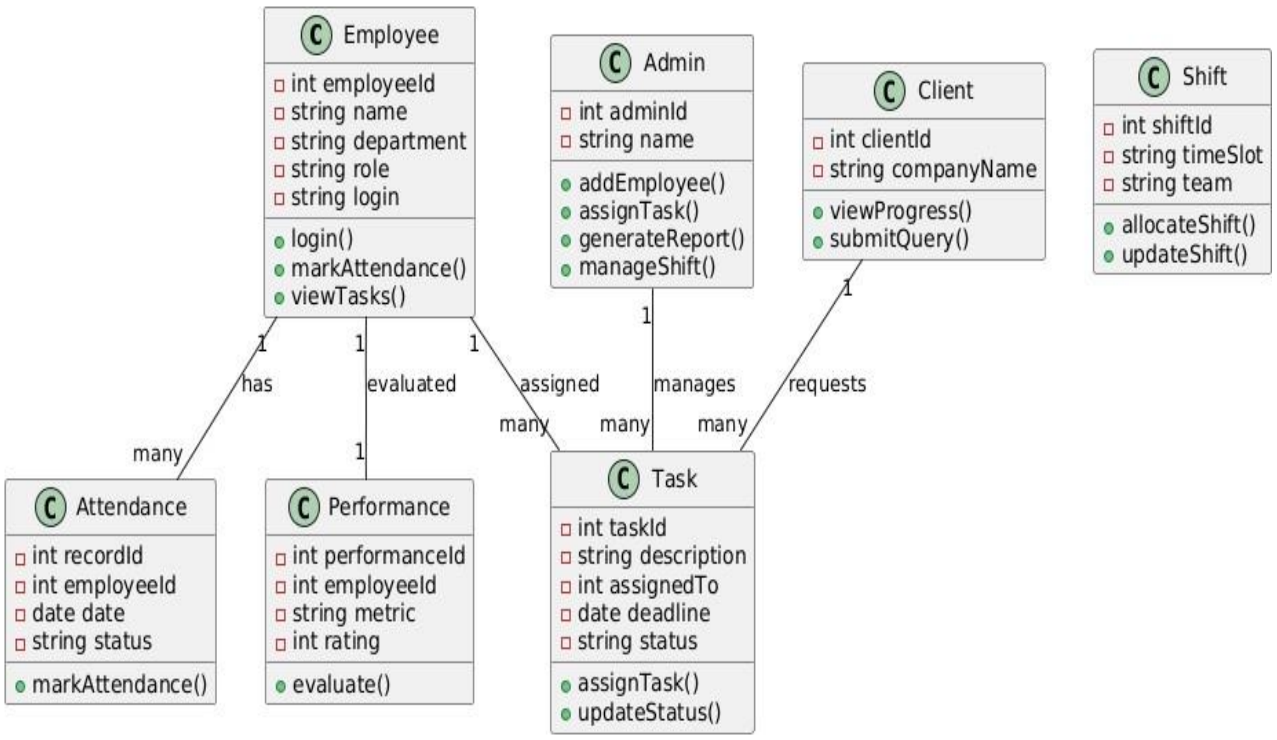
USE CASE DIAGRAM:



CODING:

```
@startuml
left to right direction
skinparam packageStyle rectangle

actor Employee
actor Admin
actor Client

rectangle "BPO Management System" {
  (Login/Register) as UC1
  (View Tasks) as UC2
  (Mark Attendance) as UC3
  (Submit Task Report) as UC4
  (Assign Tasks) as UC5
  (Generate Reports) as UC6
  (Manage Shifts) as UC7
  (Evaluate Performance) as UC8
  (View Project Progress) as UC9
  (Raise Query) as UC10

  Employee --> UC1
  Employee --> UC2
  Employee --> UC3
  Employee --> UC4

  Admin --> UC1
  Admin --> UC5
  Admin --> UC6
  Admin --> UC7
  Admin --> UC8

  Client  -->  UC1
  Client  -->  UC9
  Client --> UC10
}
@enduml
```

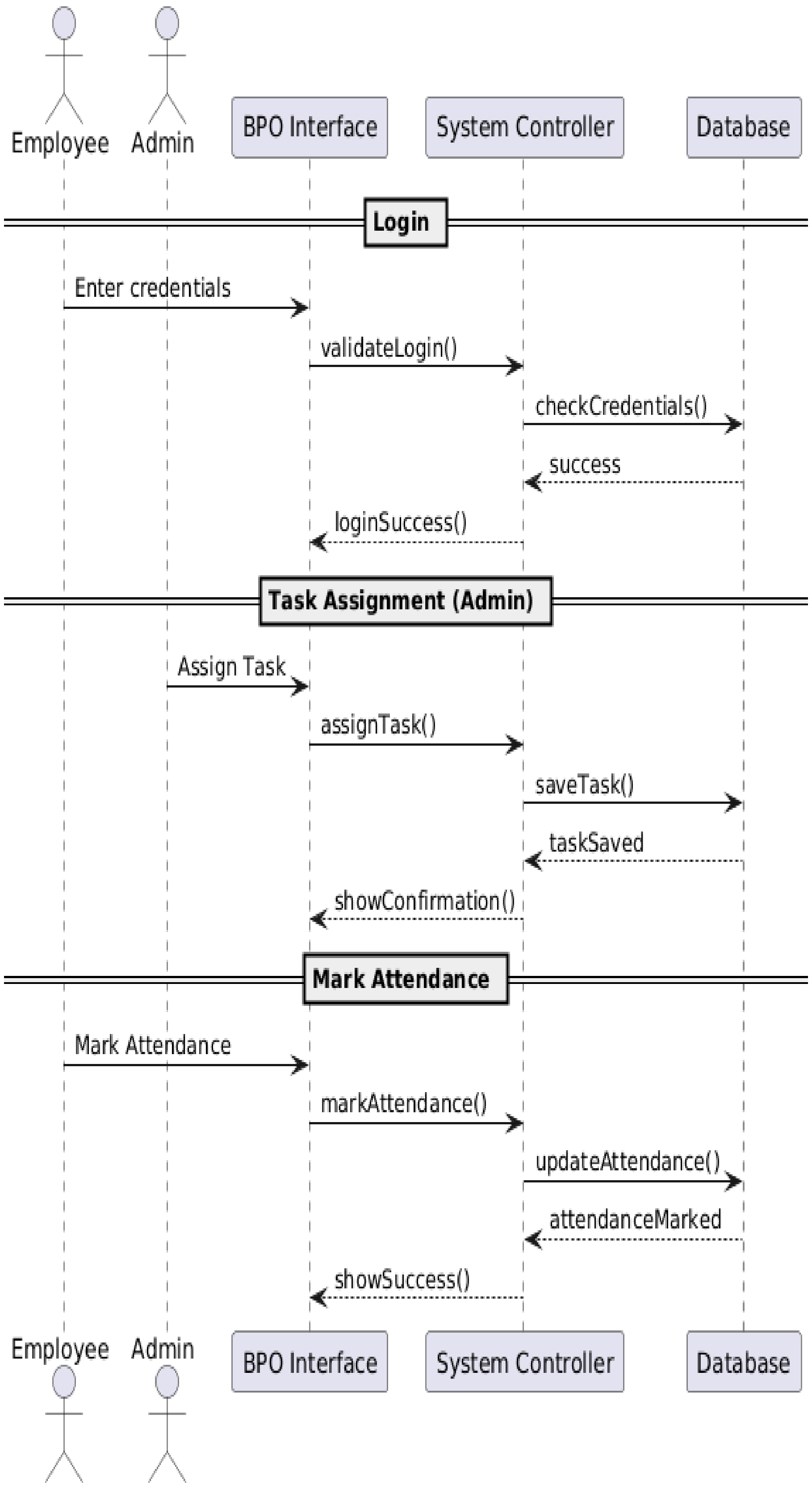
CLASS DIAGRAM:



CODING:

```
@startuml
class Employee {
  - int employeeId
  - string name
  - string department
  - string role
  - string login
  + login()
  + markAttendance()
  + viewTasks()
}

class Admin {
  - int adminId
  - string name
  + addEmployee()
  + assignTask()
  + generateReport()
  + manageShift()
}

class Client {
  - int clientId
  - string companyName
  + viewProgress()
  + submitQuery()
}

class Task {
  - int taskId
  - string description
  - int assignedTo
  - date deadline
  - string status
  + assignTask()
  + updateStatus()
}

class Shift {
  - int shiftId
  - string timeSlot
  - string team
  + allocateShift()
  + updateShift()
}

class Attendance {
  - int recordId
  - int employeeId
  - date date
  - string status
  + markAttendance()
}

class Performance {
  - int performanceId
  - int employeeId
  - string metric
  - int rating
  + evaluate()
}

Employee "1" -- "many" Task : assigned
Employee "1" -- "many" Attendance : has
Employee "1" -- "1" Performance : evaluated
Admin "1" -- "many" Task : manages
Client "1" -- "many" Task : requests
@enduml
```

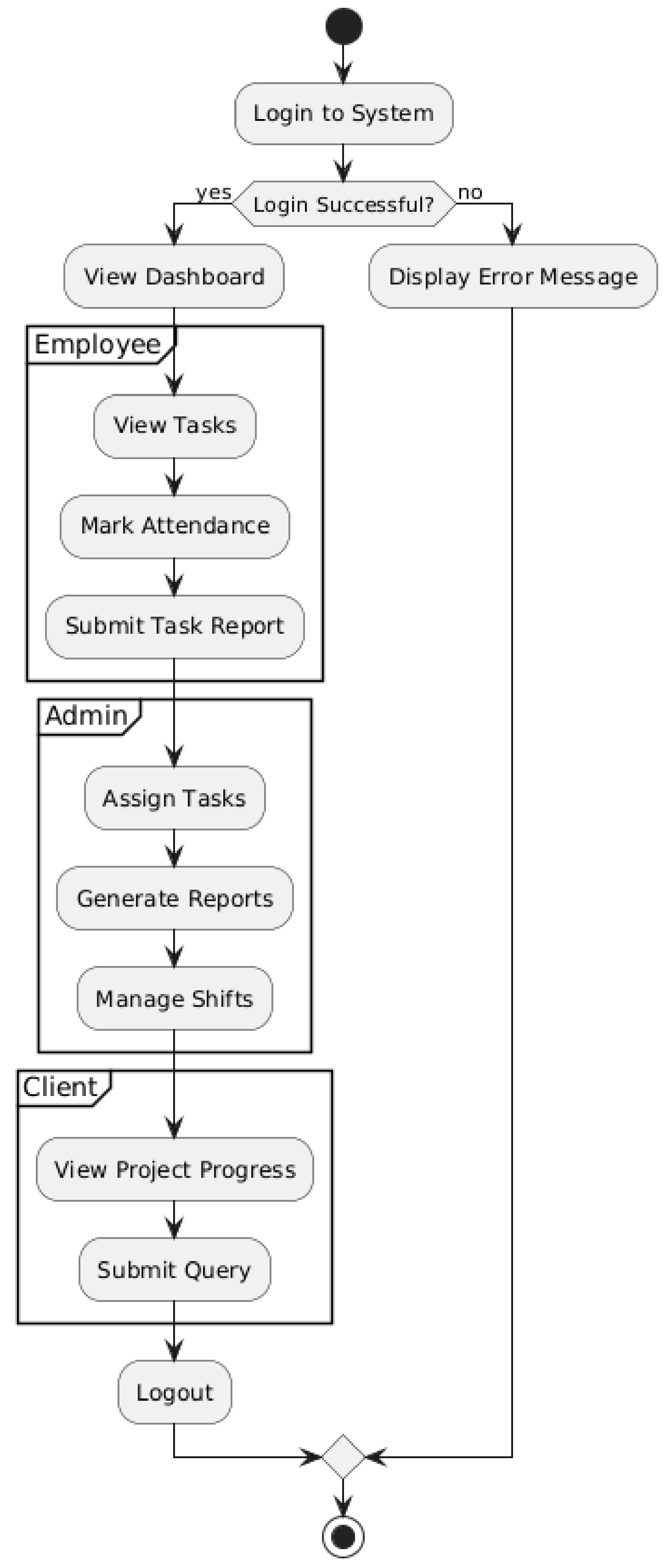
SEQUENCE DIAGRAM:



CODING:

```
@startuml
actor Employee
actor Admin
participant "BPO Interface" as UI
participant "System Controller" as SC
participant "Database" as DB

/’Login’/
Employee -> UI : Enter credentials
UI -> SC : validateLogin()
SC -> DB : checkCredentials()
DB --> SC : success
SC --> UI : loginSuccess()

/’Task Assignment (Admin) ‘/
Admin -> UI : Assign Task
UI -> SC : assignTask()
SC -> DB : saveTask()
DB --> SC : taskSaved
SC --> UI : showConfirmation()

/’Mark Attendance ‘/
Employee -> UI : Mark Attendance
UI -> SC : markAttendance()
SC -> DB : updateAttendance()
DB --> SC : attendanceMarked
SC --> UI : showSuccess()
@enduml
```

ACTIVITY DIAGRAM:



CODING:

```
@startuml
start
:Login to System;
if (Login Successful?) then (yes)
  :View Dashboard;
  partition Employee {
    :View Tasks;
    :Mark Attendance;
    :Submit Task Report;
  }
  partition Admin {
    :Assign Tasks;
    :Generate Reports;
    :Manage Shifts;
  }
  partition Client {
    :View Project Progress;
    :Submit Query;
  }
  :Logout;
else (no)
  :Display Error Message;
endif
stop
@enduml
```

RESULT:

A complete BPO Management System that allows efficient task tracking, employee management, shift scheduling, and reporting. It reduces administrative workload, minimizes human errors, and improves team coordination in BPO operations.