Employee Onboarding Process for a Consulting Firm

By:Maitreyie Adep

1.INTRODUCTION

The **Employee Onboarding System** is a web application designed to streamline the onboarding process for new employees in a consulting firm. It incorporates features for **filling employee details**, **tracking progress**, **timesheet management**, **visa status validation**, and **case management** for queries directed to HR or managers.

2.KEY FUNCTIONALITIES

2.1. Employee Profile Creation

- **Purpose:** Collect and display all necessary details about the new employee.
- **Steps:**
  - The employee logs into the onboarding portal.
  - Fills in personal details:
    - Full Name
    - Contact Information
    - Date of Birth
    - Address
  - Provides professional details:
    - Years of Experience
    - Expertise in programming languages, tools, or frameworks
    - Uploads supporting documents like certifications or resumes.

1. Use Case Diagram

This diagram focuses on the interaction between the actors (employee and system) and the system's processes.

Actors:

- **New Employee**: The person creating the profile.
- **System**: The onboarding system where the profile is created.

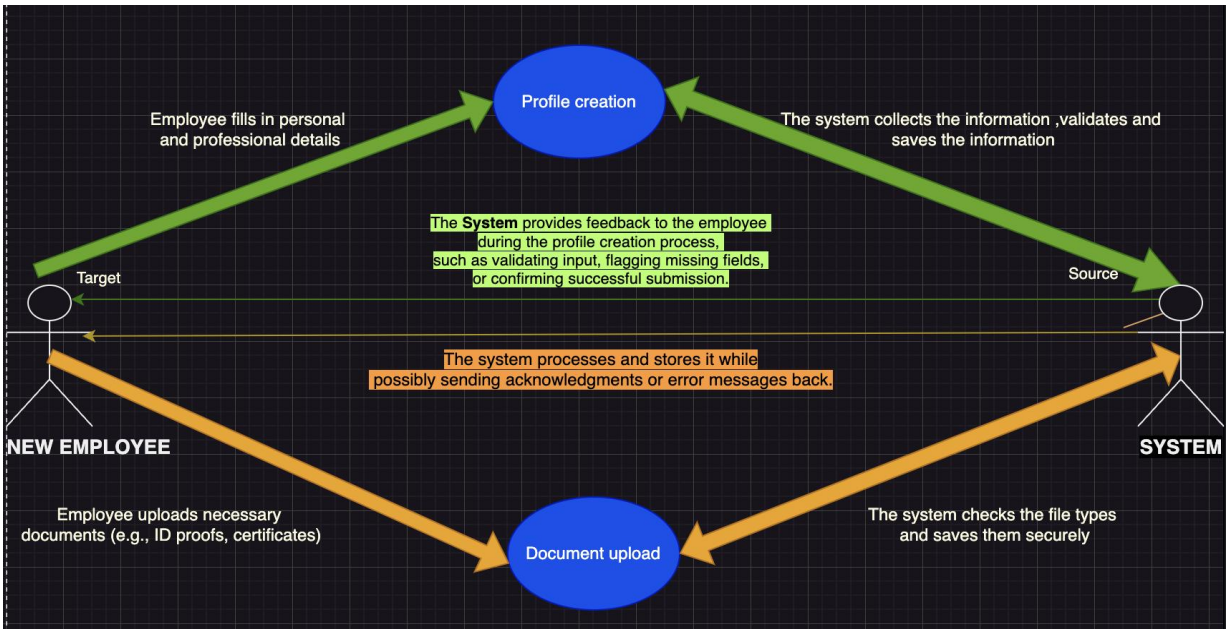
Use Cases:

1. **Profile Creation**:
   - Employee provides basic details such as name, address, phone number, and email.

    - The system validates and saves the information.
2. **Document Upload**:
    - Employee uploads necessary documents (e.g., ID proofs, certificates).
    - The system verifies the file types and saves them.

Relationships:

- The **New Employee** has an association with both **Profile Creation** and **Document Upload** use cases.
- The **System** is responsible for processing these actions.



2. Class Diagram

This diagram defines the structure of the system in terms of classes, their attributes, methods, and relationships.

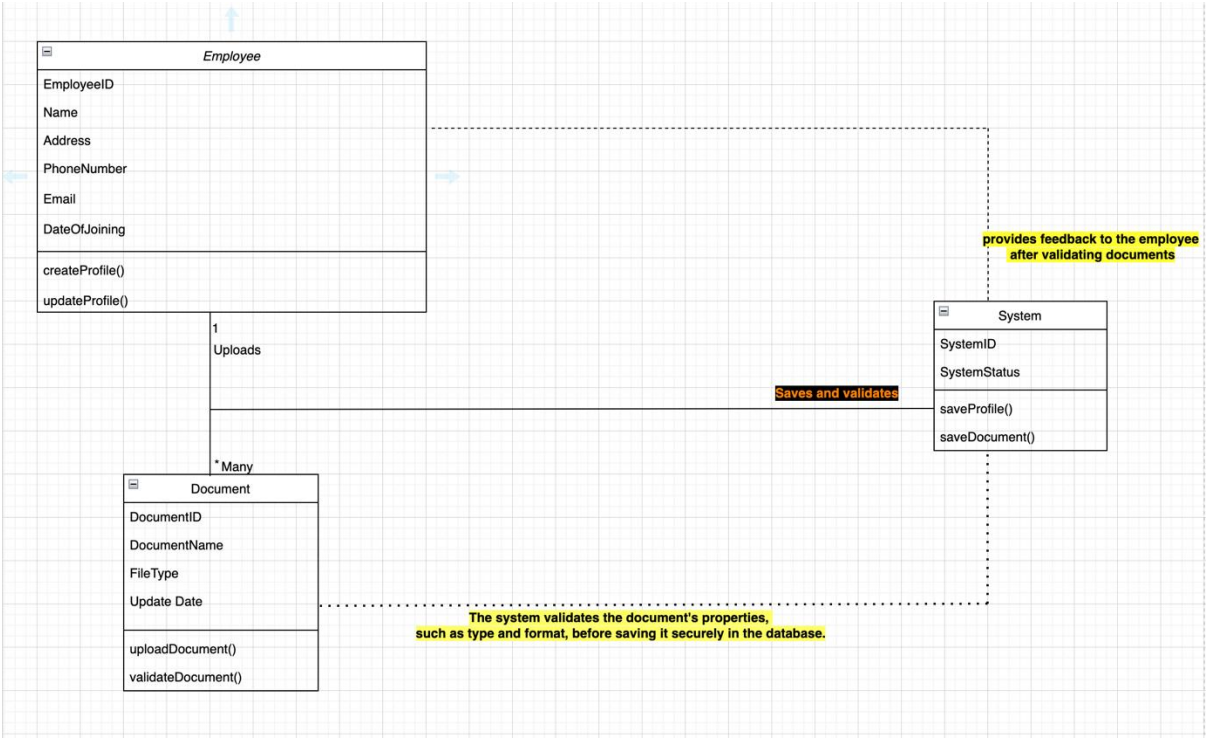
Key Classes:

1. **Employee**:
    - **Attributes**:
        - EmployeeID (Primary Key)
        - Name
        - Address
        - PhoneNumber
        - Email
        - DateOfJoining
    - **Methods**:

        - createProfile()
        - updateProfile()
2. **Document**:
    - **Attributes**:
        - DocumentID (Primary Key)
        - DocumentName
        - FileType
        - UploadDate
    - **Methods**:
        - uploadDocument()
        - validateDocument()
3. **System**:
    - **Attributes**:
        - SystemID
        - SystemStatus
    - **Methods**:
        - saveProfile()
        - saveDocument()

Relationships:

- **Employee** uploads **Document** (one-to-many relationship).
- **System** validates and saves both **Employee** and **Document** data.



The class diagram illustrates the **Employee Profile Creation** process within an onboarding system, showcasing the interaction between the **Employee**, **Document**, and **System**. The **Employee** class manages attributes like EmployeeID, Name, and DateOfJoining and includes methods for profile creation and updates. The **Document** class represents files uploaded by employees, with attributes like DocumentID and methods for uploading and validating documents. The **System** class processes and stores both employee data and documents, validating document properties and providing feedback to employees. Relationships include a one-to-many link between employees and documents, while the system connects with both to ensure validation, secure storage, and user feedback, ensuring a seamless onboarding experience.

3. Activity Diagram

This diagram shows the step-by-step workflow for profile creation.

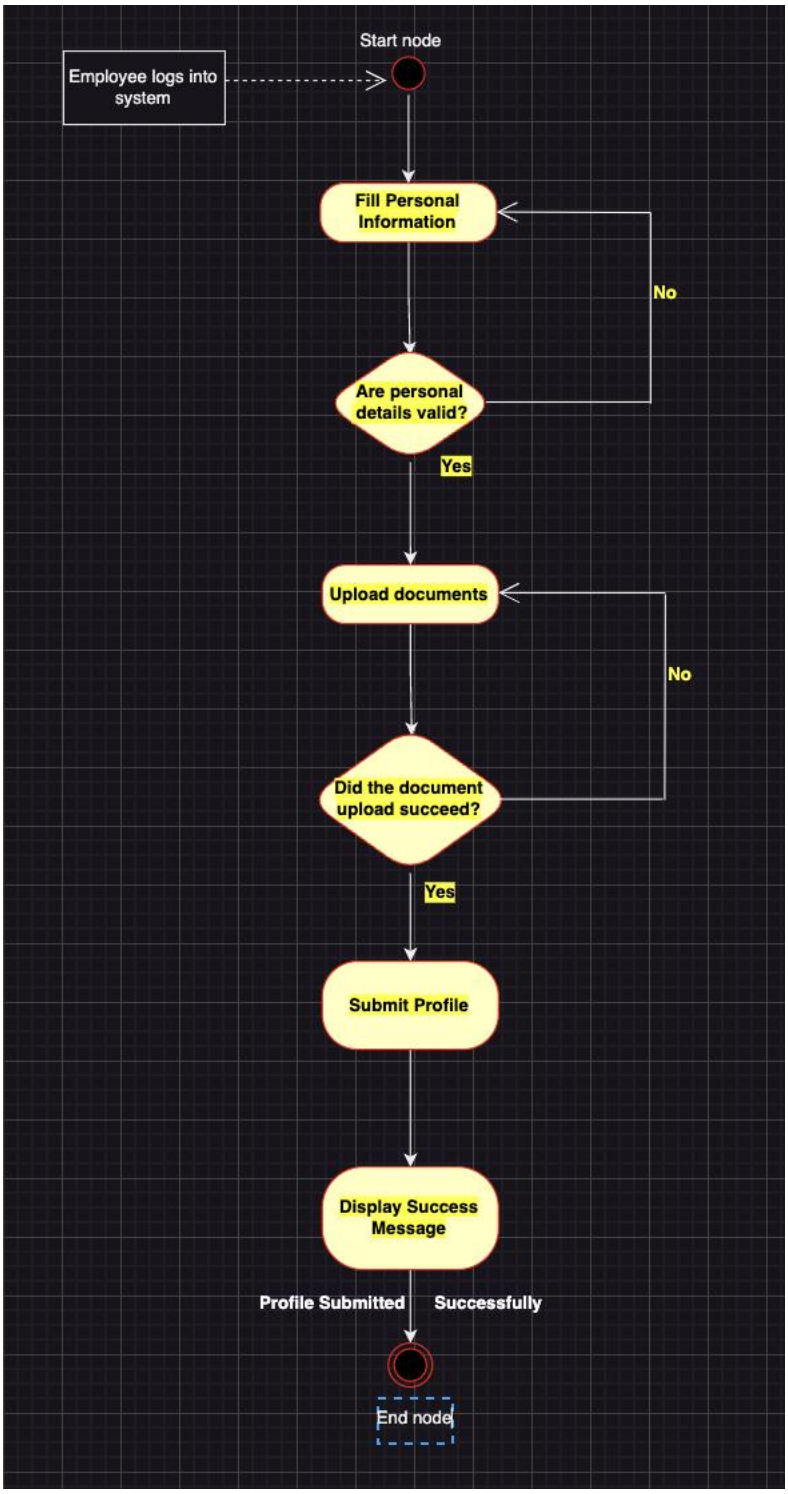
Steps:

1. **Start**:
    - The employee logs into the system.
2. **Fill Personal Information**:
    - The system presents a form.
    - The employee inputs details like name, address, and contact information.
    - The system validates the input fields (e.g., phone number format).
3. **Upload Documents**:
    - The employee uploads required documents.
    - The system checks file types and sizes.
4. **Submit Profile**:
    - The employee submits the profile for verification.
    - The system saves the data and displays a success message.
5. **End**:
    - Profile creation is completed.

Visual Representation:

- **Start Node**: Employee initiates the process.
- **Decision Nodes**:
    - If personal details are invalid, the system prompts corrections.
    - If document upload fails, the system requests re-upload.
- **Action Nodes**:
    - Filling personal details.
    - Uploading documents.
    - Submitting the profile.

- **End Node**: Success confirmation is displayed.



This activity diagram illustrates the step-by-step workflow for the employee profile creation process. The process begins at the **Start Node**, where the employee logs into the system. The first action involves the employee filling in personal information such as name, address, and contact details. A **decision node** validates the entered personal details. If the details are invalid, the process loops back to the "Fill Personal

Information" step for correction (**No path**), while valid details (**Yes path**) proceed to the next step. The employee then uploads the necessary documents like ID proofs and certificates. Another **decision node** checks if the document upload was successful. If the upload fails (**No path**), the system requests the employee to re-upload the documents. Upon successful upload (**Yes path**), the employee submits the profile for final verification. The system then validates and securely saves the data. Finally, a success message confirming the profile submission is displayed to the employee before the process ends at the **End Node**.

2.2 Visa Status Validation

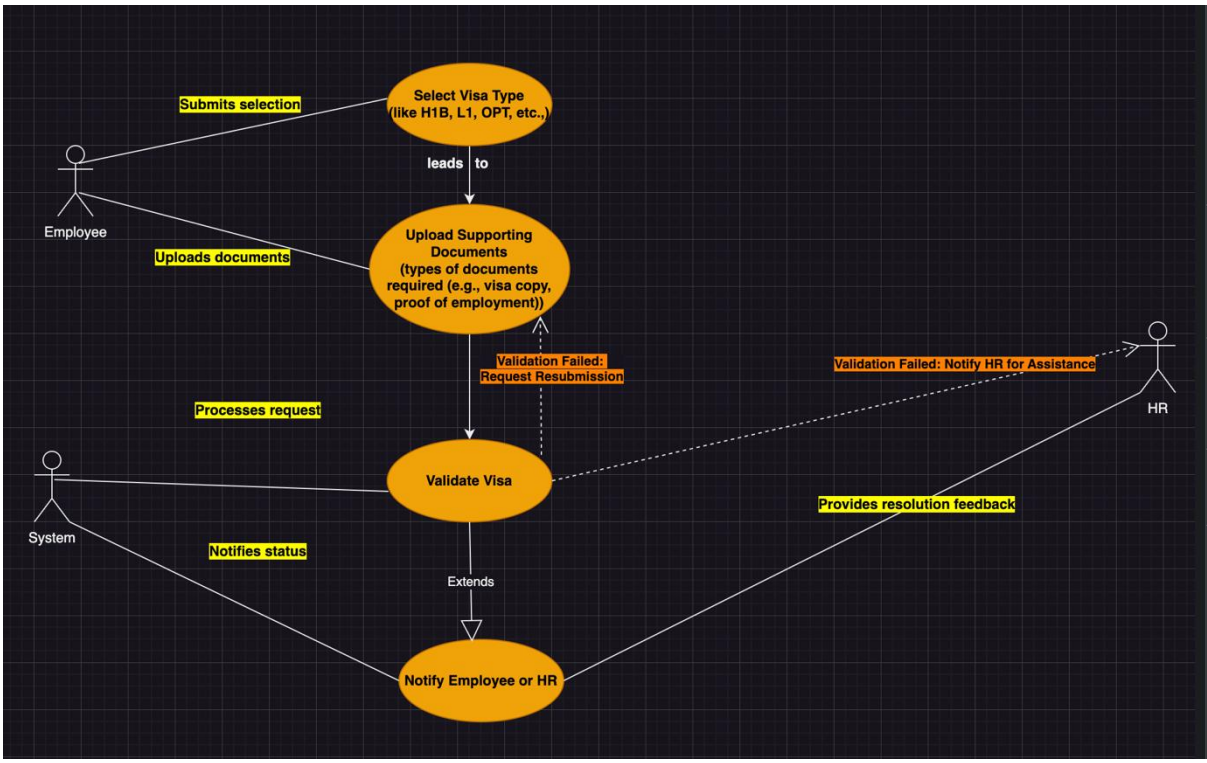
- **Purpose:** Ensure the employee satisfies visa requirements before starting work.
- **Steps:**
  - The employee is prompted to select their visa type:
    - H1B
    - L1
    - OPT
    - Others
  - If the employee has an **H1B visa**, the system validates and proceeds.
  - If the employee does not have an H1B, they are prompted to:
    - Mention their current visa type.
    - Upload supporting visa documentation.
    - Confirm acknowledgment of visa processing timelines (if applicable).

1. Use Case Diagram

This diagram illustrates the interactions between the employee, system, and HR for visa validation.

- **Actors:**
  - **Employee:** Selects visa type and uploads necessary documents.
  - **System:** Validates visa status and documents.
  - **HR:** Resolves issues if validation fails.
- **Use Cases:**
  - Select Visa Type.
  - Upload Supporting Documents.

- Validate Visa.
- Notify Employee or HR in case of issues.



This use case diagram illustrates the visa validation process for new employees. The employee begins by selecting their visa type (H1B, L1, OPT, etc.) and uploading required supporting documents, such as visa copies or proof of employment. The system validates the uploaded documents and processes the request. If the validation is successful, the process proceeds to notify the employee or HR. In case of validation failure, the system sends a resubmission request to the employee for correcting or re-uploading documents. Simultaneously, the system notifies HR to assist in resolving issues. HR provides resolution feedback as needed to ensure the validation process is successfully completed. The use of alternate flows ensures that all scenarios, including validation failures, are effectively handled

2. Class Diagram

Defines the entities and their relationships.

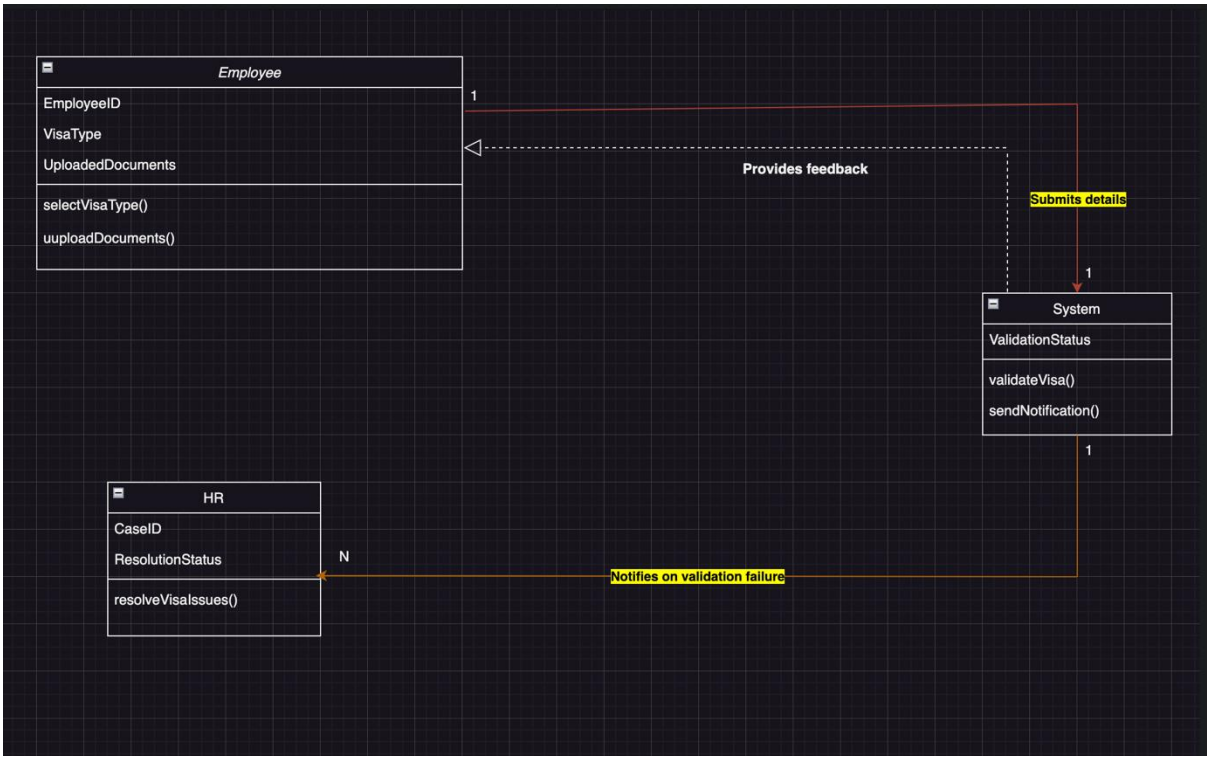
- **Classes:**
  1. **Employee:**
     - Attributes:
       - EmployeeID

        - VisaType
        - UploadedDocuments
    - Methods:
        - selectVisaType()
        - uploadDocuments()
2. **System:**
    - Attributes:
        - ValidationStatus
    - Methods:
        - validateVisa()
        - sendNotification()
3. **HR:**
    - Attributes:
        - CaseID
        - ResolutionStatus
    - Methods:
        - resolveVisaIssues()

- **Relationships:**

    - Employee interacts with the System for visa validation.
    - System notifies HR if validation fails.

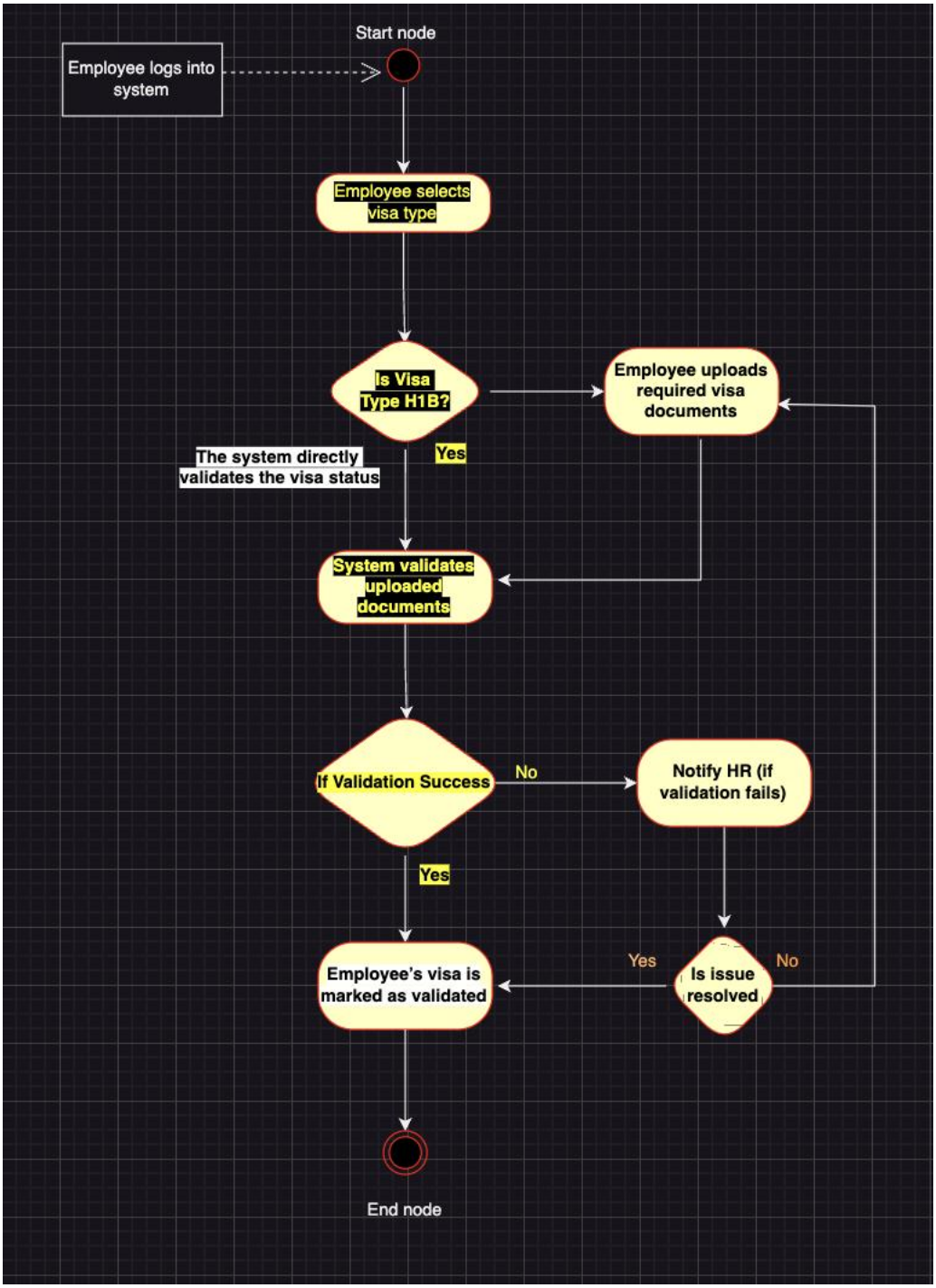
    The class diagram depicts the interaction between the **Employee**, **System**, and **HR** for visa validation and feedback. The **Employee** class has attributes like EmployeeID, VisaType, and UploadedDocuments, with methods such as selectVisaType() and uploadDocuments() to provide visa-related details to the **System**. The **System** class manages visa validation through its ValidationStatus attribute and methods like validateVisa() and sendNotification(). It communicates with the employee by providing feedback on the validation status or requesting resubmissions for errors. A dotted line indicates that feedback flows back to the employee for corrections if necessary. If the validation fails, the **System** notifies the **HR** (through resolveVisaIssues()), which addresses issues and provides resolution feedback to the employee. The multiplicity relationships are clearly shown: one **System** handles many employees (1 to N), and one **HR** can handle multiple visa cases (N to 1). This structure ensures a streamlined and efficient process for visa validation and resolution



3. Activity Diagram

Step-by-step workflow for visa validation.

- **Nodes:**
    1. **Start Node:** Employee logs into the system.
    2. **Decision Node 1:** Employee selects visa type.
        - **H1B:** System validates and proceeds.
        - **Non-H1B:** Employee uploads documents.
    3. **Action Node:** System validates uploaded documents.
    4. **Decision Node 2:**
        - **Validation Success:** Process continues.
        - **Validation Failure:** Notify HR.
    5. **End Node:** Visa status validated.



This activity diagram represents the workflow for validating an employee's visa during the onboarding process. It begins with the employee logging into the system (start node) and selecting their visa type. If the visa type is H1B, the system directly validates it without requiring additional input. For non-H1B visa types, the employee uploads the necessary supporting documents, which are then validated by the system. If the validation is successful, the visa is marked as validated, and the process concludes. In case of a validation failure, the system notifies HR to review the issue. HR

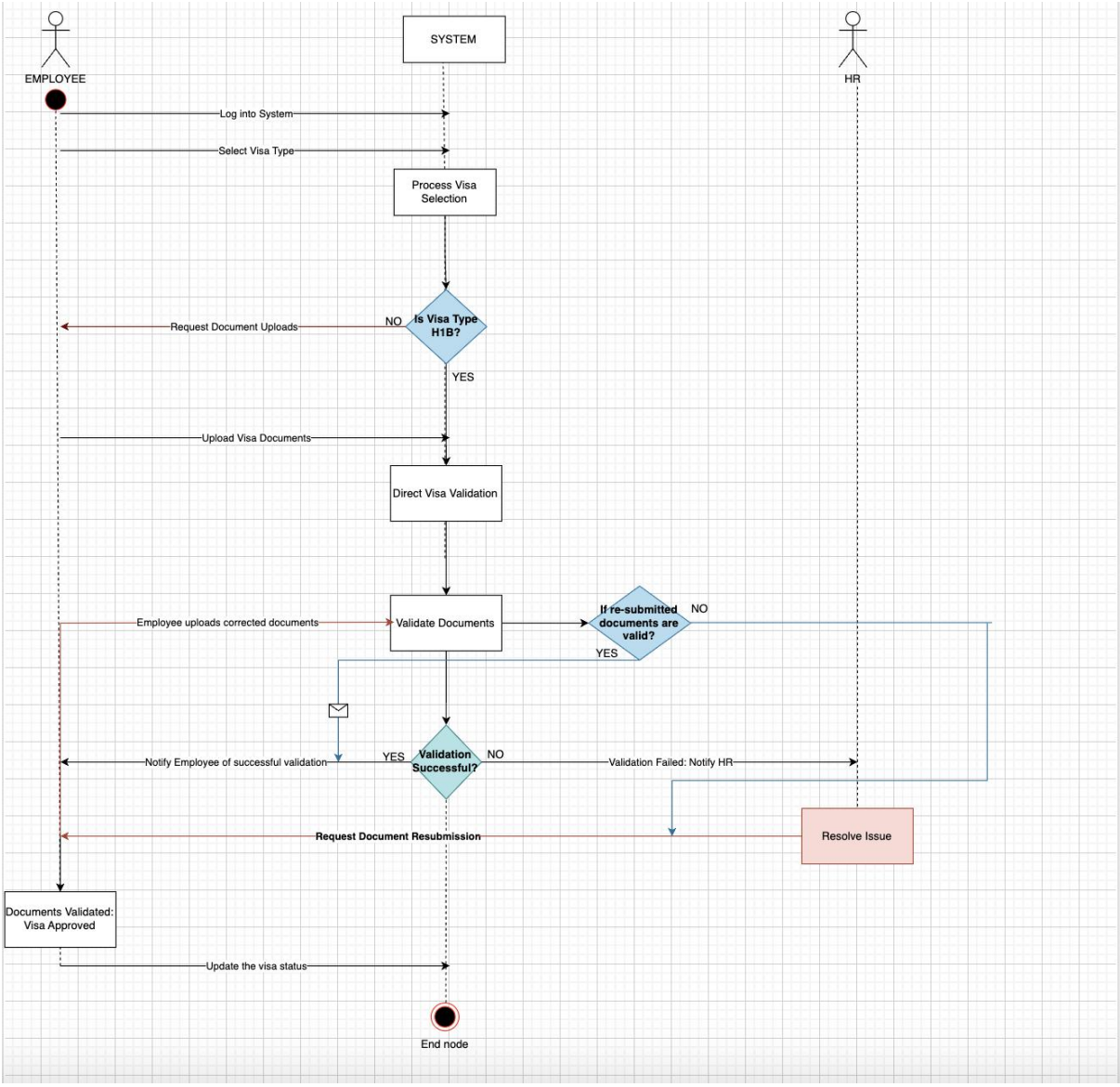
evaluates the situation and determines if the issue can be resolved. If HR resolves the issue, the process loops back to mark the visa as validated. If unresolved, the employee is notified to re-upload the required documents, continuing the loop until resolution. The process ends once the visa is successfully validated (end node).

4. Sequence Diagram

Illustrates the sequence of interactions.

- **Actors:** Employee, System, HR.
- **Interactions:**
    1. Employee selects visa type.
    2. System requests document upload (if needed).
    3. Employee uploads documents.
    4. System validates documents and visa status.
    5. System notifies employee or HR based on the result.

This sequence diagram illustrates the comprehensive workflow for visa validation in an onboarding process. The employee starts by logging into the system and selecting their visa type. If the visa type is H1B, the system directly validates the visa status. For non-H1B visas, the system requests the upload of supporting documents. Once the employee uploads the required documents, the system validates them. If the validation fails, the system prompts the employee to resubmit corrected documents. The resubmitted documents undergo a second round of validation, and if they are valid, the system notifies the employee of successful validation. However, if the validation fails again, the system escalates the issue to HR for manual intervention. HR resolves the issue and updates the system. The process concludes with the system updating the employee's visa status to "Validated" and notifying the employee.



2.3 Timesheet Management

- **Purpose:** Enable employees to log their working hours during onboarding and track productivity.
- **Steps:**
  - Employees access the timesheet section.
  - Fill in daily/weekly working hours:
    - Project allocation
    - Tasks performed
    - Break hours (if any)
  - Submit the timesheet for review.
  - HR or managers can approve or request changes.

1. Use Case Diagram

Actors:

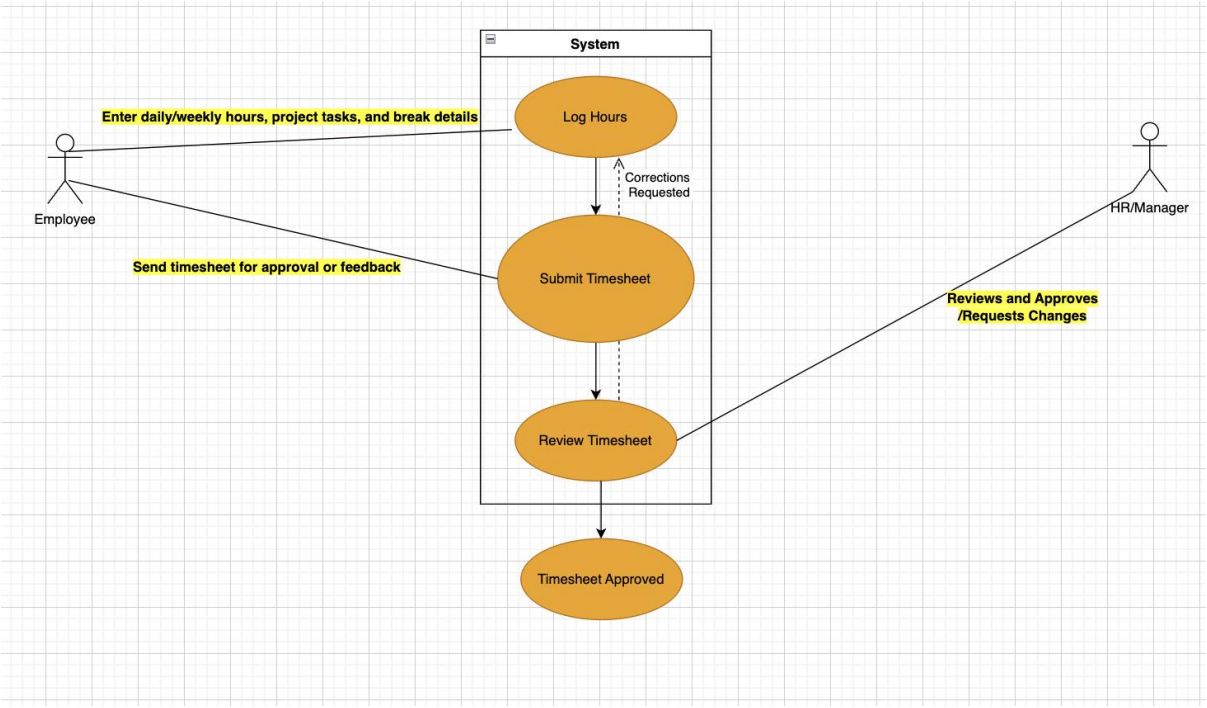
- **Employee:** Logs hours and submits timesheets.
- **System:** Processes the timesheet data.
- **HR/Manager:** Reviews and approves/rejects timesheets.

Use Cases:

- **Log Hours:** Employees input daily/weekly hours, tasks, and breaks.
- **Submit Timesheet:** Employees finalize and submit the timesheet for review.
- **Review Timesheet:** HR/Manager reviews, approves, or requests changes.

Relationships:

- **Employee interacts** with the **System** for logging and submitting timesheets.
- **System interacts** with **HR/Manager** for approval or changes.



This use case diagram illustrates the **Timesheet Management** process. The **Employee** interacts with the system by logging hours, including daily/weekly working hours, project tasks, and break details, as indicated by the **Log Hours** use case. Once the timesheet is complete, the employee submits it to the system via the **Submit Timesheet** use case.

The **HR/Manager** reviews the submitted timesheet through the **Review Timesheet** use case, with the option to approve it or request corrections. If corrections are needed, the system notifies the employee to make adjustments. The process concludes successfully with the **Timesheet Approved** use case, indicating that the employee's hours and tasks are validated and accepted

2. Class Diagram

Classes:

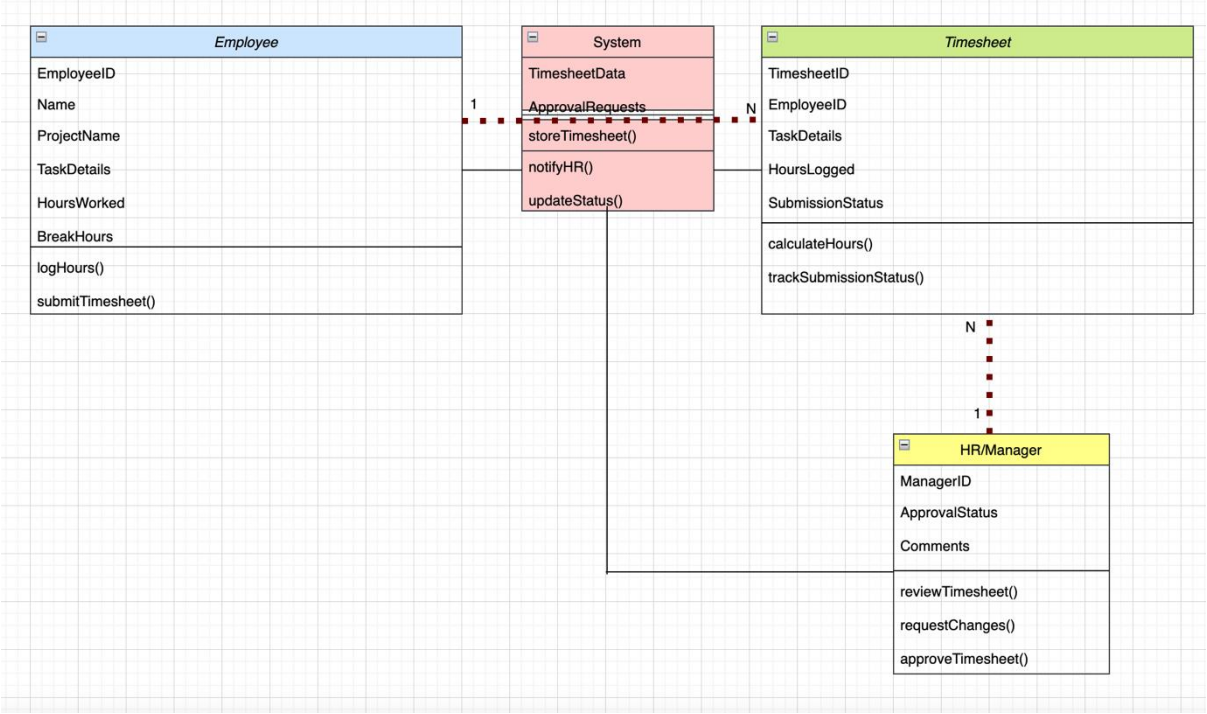
1. **Employee**
   - **Attributes:** EmployeeID, Name, ProjectName, TaskDetails, HoursWorked, BreakHours
   - **Methods:** logHours(), submitTimesheet()
2. **Timesheet**
   - **Attributes:** TimesheetID, EmployeeID, TaskDetails, HoursLogged, SubmissionStatus
   - **Methods:** calculateHours(), trackSubmissionStatus()
3. **HR/Manager**
   - **Attributes:** ManagerID, ApprovalStatus, Comments
   - **Methods:** reviewTimesheet(), requestChanges(), approveTimesheet()
4. **System**
   - **Attributes:** TimesheetData, ApprovalRequests
   - **Methods:** storeTimesheet(), notifyHR(), updateStatus()

Relationships:

- An **Employee** is associated with **Timesheet** (one-to-many).
- **HR/Manager** reviews and approves/rejects the **Timesheet** (many-to-one).
- **System** connects all entities and processes timesheet data.

This **class diagram** represents the Timesheet Management system, detailing the roles and their interactions. The **Employee** logs hours and submits timesheets using methods like logHours() and submitTimesheet(). Each employee is associated with multiple timesheets (1:N relationship). The **Timesheet** class tracks records with attributes like TimesheetID, TaskDetails, and SubmissionStatus, and methods such as calculateHours() and trackSubmissionStatus() for productivity monitoring. The **System** connects all entities, storing timesheet data (TimesheetData), notifying HR via notifyHR(), and updating the status (updateStatus()). The

HR/Manager reviews, approves, or requests changes using methods like reviewTimesheet(), requestChanges(), and approveTimesheet(), with comments and approval statuses recorded. Relationships are established to ensure seamless interaction: employees submit timesheets, the system processes them, and HR finalizes approvals, creating an efficient workflow.



3. Activity Diagram

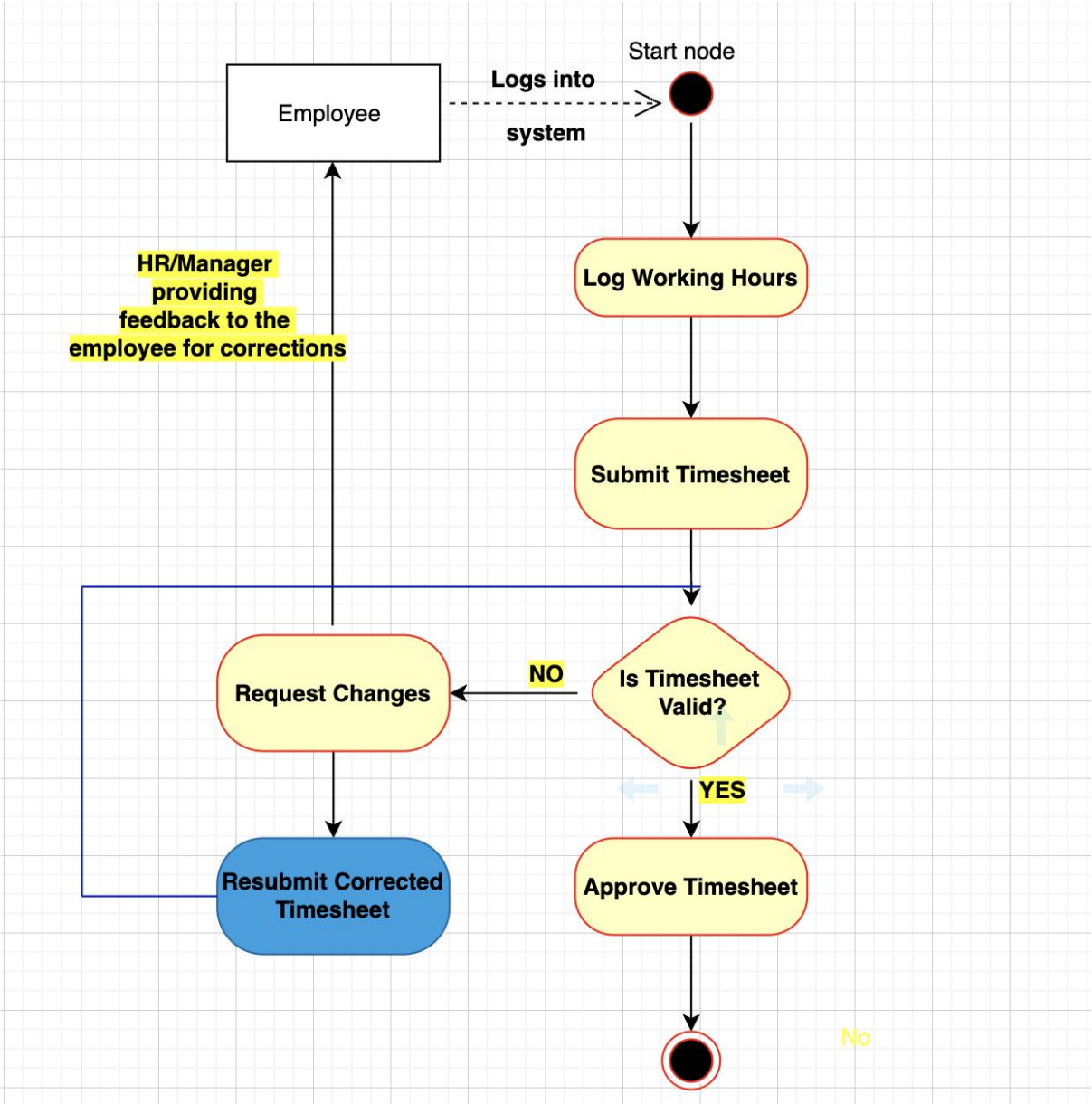
Workflow:

1. **Start:**
   Employee logs into the system and navigates to the timesheet section.
2. **Log Working Hours:**
   - Employee inputs:
     - **Project allocation.**
     - **Tasks performed.**
     - **Break hours (if any).**
3. **Submit Timesheet:**
   - Employee reviews and submits the timesheet.
   - System validates entries.
4. **Review by HR/Manager:**
   - HR/Manager reviews the timesheet:

- **If valid:** Approve the timesheet.
- **If invalid:** Request changes and send feedback to the employee.

5. **End:**
   Timesheet is approved, and productivity is logged.

   This **Activity Diagram** represents the step-by-step workflow for timesheet management. The process begins with the employee logging into the system and navigating to the timesheet section to log their working hours, project allocation, tasks performed, and break hours (if any). After completing the entries, the employee submits the timesheet. The system validates the entries, and if the timesheet is valid, it proceeds to the HR/Manager for approval, which marks the process as complete. However, if the timesheet is invalid, the system notifies the HR/Manager, who provides feedback and requests changes. The employee then resubmits the corrected timesheet, which goes through the validation process again. This cycle continues until the timesheet is valid, ensuring accuracy and proper tracking of employee productivity.

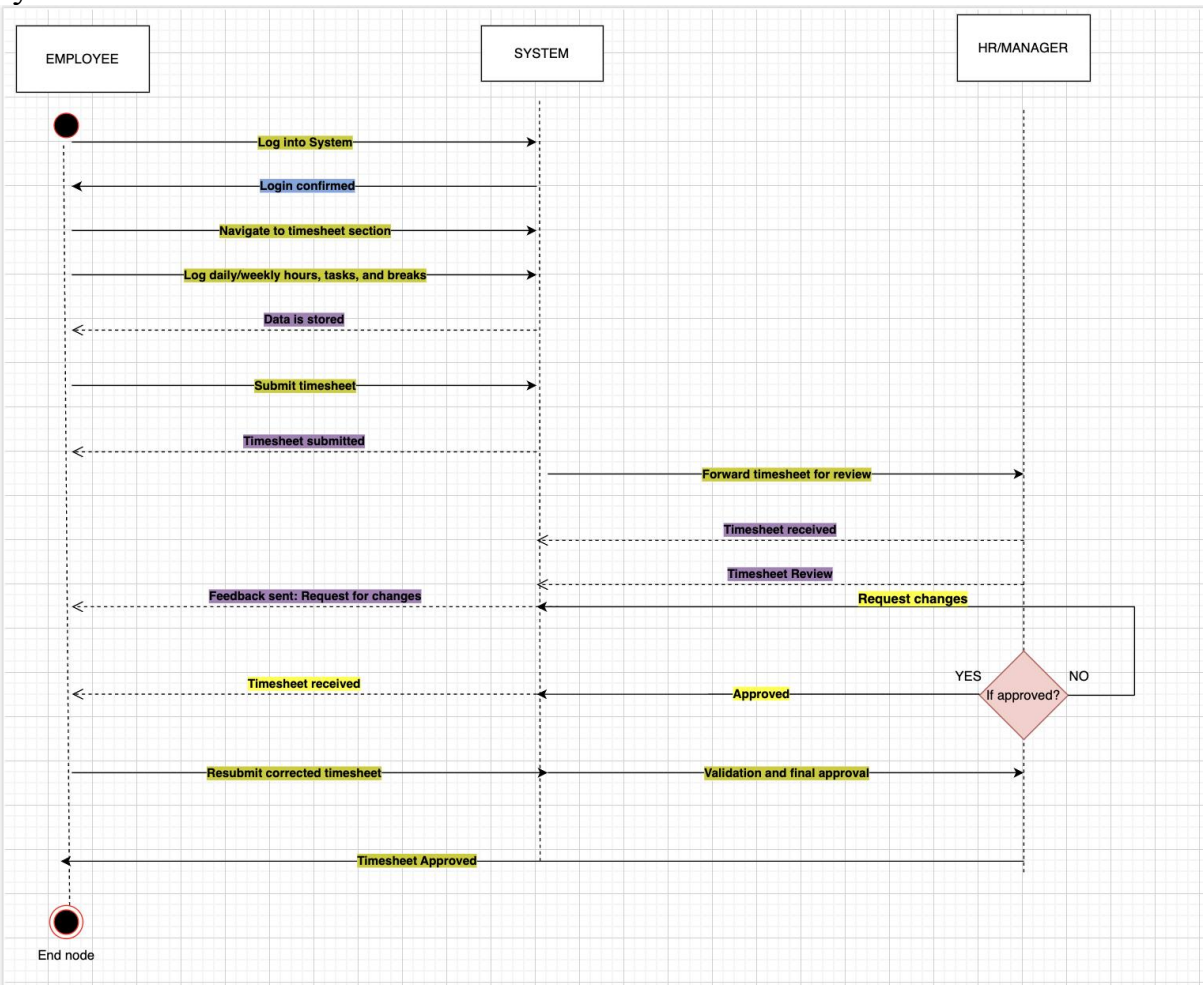


4. Sequence Diagram

Interactions:

1. Employee logs into the system and selects the timesheet section.
2. Employee logs daily/weekly hours, project, tasks, and breaks.
3. System stores the data and allows submission.
4. Employee submits the timesheet.
5. System sends it to HR/Manager for review.
6. HR/Manager reviews the timesheet and provides feedback:
    - If approved: Updates the system and notifies the employee.
    - If rejected: Sends feedback and requests changes.
7. Employee resubmits the updated timesheet if needed.

This sequence diagram illustrates the workflow of timesheet management between the **Employee**, **System**, and **HR/Manager**. The process begins with the employee logging into the system and navigating to the timesheet section to log their daily or weekly hours, project tasks, and breaks. The system stores this data and allows the employee to submit the timesheet. Upon submission, the system forwards the timesheet to the HR/Manager for review. The HR/Manager evaluates the timesheet, and if approved, the system notifies the employee of the approval. If the timesheet is invalid, the HR/Manager requests changes, and the system sends feedback to the employee. The employee resubmits the corrected timesheet, which undergoes another round of review. The process concludes once the timesheet is approved and the final status is updated in the system



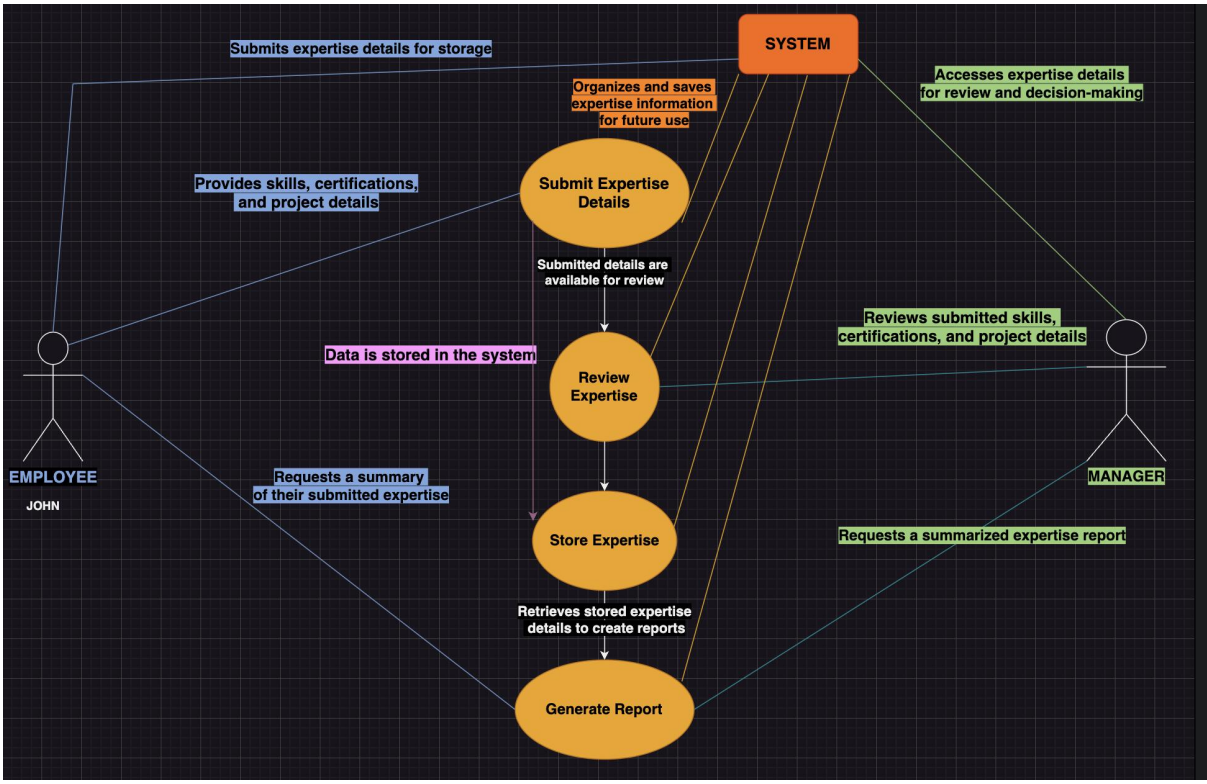
2.4 Expertise

When a new employee joins a company, HR/Manager needs to evaluate their expertise to effectively align them with relevant roles and responsibilities. The expertise evaluation process includes gathering information about the employee's skills, certifications, project experience, and areas of specialization. This allows the organization to assess how the employee's strengths can be utilized to meet organizational goals. The system should provide a structured way to log and store the employee's skills and generate insights for HR/Manager to make informed decisions.

1. Use Case Diagram:

Steps:

1. **Actors**:
    - Employee: Provides expertise details.
    - HR/Manager: Reviews the expertise data.
    - System: Stores and organizes the expertise information.
2. **Use Cases**:
    - Submit Expertise Details: Employee enters their skills, certifications, and past projects.
    - Review Expertise: HR/Manager evaluates the provided data.
    - Store Expertise: System saves the details for future reference.
    - Generate Report: System generates a summary of the employee's expertise.
3. **Relationships**:
    - Employee interacts with the system to submit expertise details.
    - HR/Manager interacts with the system to review expertise and generate reports.



This Use Case Diagram illustrates the process of capturing and managing an employee's expertise within the system. The **Employee** provides details about their skills, certifications, and project experience through the **Submit Expertise Details** use case, which the **System** stores and organizes for future reference. The **Manager** accesses these stored details via the **Review Expertise** use case, enabling a comprehensive evaluation of the employee's capabilities. Additionally, the **System** supports the **Generate Report** use case, where both the employee and manager can request summarized reports of the expertise data for decision-making or showcasing progress. Relationships include the employee interacting with the system to input and retrieve their expertise data, the manager leveraging the system to review and generate insights, and the system ensuring the secure and organized handling of all information.

2. Activity Diagram:

Steps:

1. **Start Node**: Employee begins the process by submitting their expertise details.
2. **Action Nodes**:
    - System logs the expertise details.
    - HR/Manager reviews the skills and past projects.
    - HR/Manager evaluates the data and provides feedback or updates.
3. **Decision Node**:
    - Is the data sufficient?
        - Yes: Generate a summary report.
        - No: Notify the employee to provide additional information.
4. **End Node**: Expertise evaluation is complete, and the data is logged in the system.

Action Nodes in Detail:

1. System Logs the Expertise Details

- **What Happens:**
  - The employee provides their expertise data, such as:
    - Key skills they possess (e.g., programming, leadership, problem-solving).
    - Certifications obtained (e.g., AWS, PMP, or ITIL certifications).
    - Details of past projects (e.g., project titles, roles, achievements).
  - This information is entered into the system using an interface (e.g., a web portal or HR application).
- **System Behavior:**
  - The system categorizes the data under predefined fields (like "Skills," "Certifications," "Projects").
  - It validates the input (e.g., ensuring certification dates are valid).
  - Each expertise entry is saved in the database, making it ready for review by HR/Manager.

2. HR/Manager Reviews the Skills and Past Projects

- **What Happens:**
  - HR/Manager accesses the stored expertise data using the system.
  - HR/Manager evaluates the data for relevance and completeness:
    - **Skills**: Are the listed skills relevant to the current or future organizational needs?
    - **Certifications**: Are the certifications up-to-date and from credible sources?
    - **Projects**: What were the employee's contributions and results in their past projects?
- **System Support:**
  - The system might highlight discrepancies (e.g., missing certification validity dates).
  - Summary sections or prebuilt filters in the system can streamline the review process for HR.

3. HR/Manager Evaluates the Data and Provides Feedback or Updates

- **What Happens:**
  - HR/Manager assesses whether the expertise data meets organizational requirements.
  - If the data is **sufficient**, the HR/Manager approves it for further use (e.g., generating reports).
  - If the data is **insufficient**:
    - HR/Manager provides detailed feedback on what needs improvement.
    - For example:
      - "Please clarify your role in Project X."
      - "Provide certifications for skill Y."

- **System Behavior:**
  - The feedback is saved in the system and made visible to the employee.

4. Notify Employee to Provide Additional Details or Clarifications (If Data is Insufficient)

- **What Happens:**
  - The system sends an automated notification to the employee if HR/Manager marks the data as incomplete or unclear.
- **Details Included in the Notification:**
  - What specific information is missing (e.g., "Add start and end dates for Certification A").
  - Clear instructions on how to update their expertise data.
- **Employee's Next Action:**
  - The employee logs back into the system to add or clarify the requested details.
  - For example:
    - They might upload supporting documents, update certification details, or rewrite project descriptions.

5. Generate a Summary Report (If Data is Sufficient)

- **What Happens:**
  - If the data is complete and accurate, the system generates a **summary report**.
- **Contents of the Report:**
  - **Skills Section**: Lists all categorized skills (e.g., "Technical Skills," "Leadership").
  - **Certifications Section**: Includes certification names, validity periods, and providers.
  - **Projects Section**: Highlights significant contributions and achievements in past projects.
- **System Behavior:**
  - The report is automatically made accessible to:
    - HR/Manager: To guide decision-making for job roles, promotions, or trainings.
    - Employee: To review and understand their expertise profile.

6. Update Query Status to "Resolved"

- **What Happens:**
  - After completing all steps (e.g., clarifications or generating the summary), the system marks the expertise query as "Resolved."
- **Final Notifications:**
  - The system sends a confirmation to:
    - The employee: Notifying them that their expertise data has been finalized and is ready for use.
    - HR/Manager: Indicating that the expertise details have been reviewed and are now available in the system.

- **System Logs:**
  - The system archives the expertise data and any generated reports for future use.

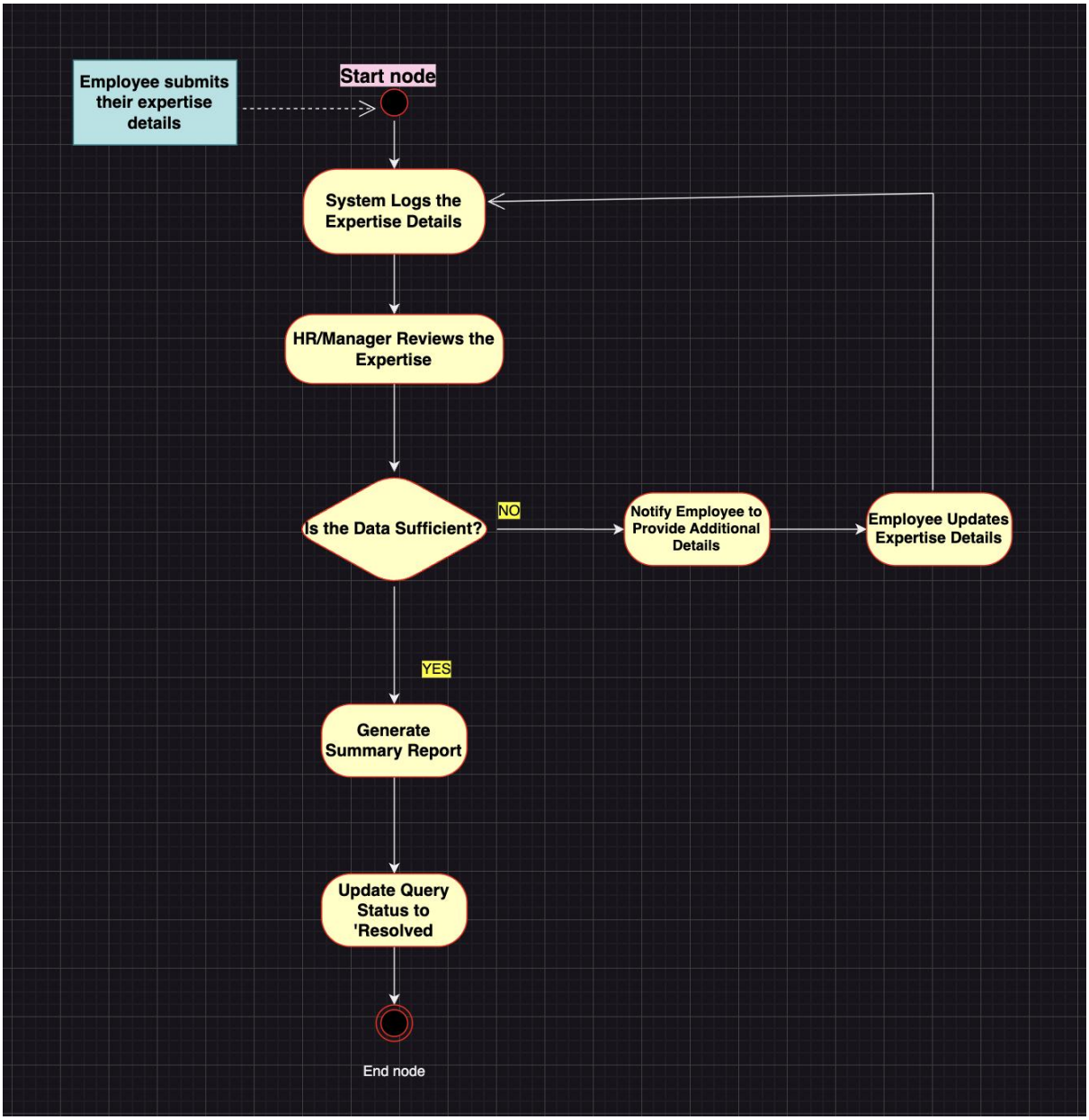
Detailed Explanation of the Decision Node: "Is the Data Sufficient?"

Yes: The Data is Sufficient

1. **System Validation**:
   - The **System** evaluates the submitted data (skills, certifications, and project details) to ensure it meets predefined criteria for sufficiency.
   - Criteria for sufficiency may include:
     - The number of skills listed matches the job requirements.
     - Certifications align with the employee's role or field.
     - Past projects are adequately detailed with outcomes or contributions.
2. **Generate Summary Report**:
   - If the submitted data meets the criteria, the **System** automatically compiles a **summary report**.
   - The summary report includes:
     - A categorized list of **skills** (e.g., technical, managerial, or communication skills).
     - Details of **certifications** (including dates and relevance).
     - Highlights from **past projects** (key contributions, challenges solved, and outcomes).
3. **Access to Report**:
   - The **System** makes the report accessible to both the **Employee** and **HR/Manager**.
   - The **Employee** can review the report for accuracy, while the **HR/Manager** can use it for decision-making (e.g., project assignments, appraisals, or training needs).

No: The Data is Insufficient

1. **System Identification of Gaps**:
   - If the data does not meet sufficiency criteria, the **System** identifies the gaps.
   - For example:
     - Missing certifications or incomplete details (e.g., a project listed without outcomes).
     - Vague descriptions of skills or irrelevant expertise.
2. **Notification to Employee**:
   - The **System** notifies the **Employee** about the insufficiencies in their submission.
   - A message is sent to the employee (e.g., through email or a system alert) detailing:
     - What is missing (e.g., specific certifications, clearer project descriptions).
     - The next steps, such as updating or clarifying the missing information.
3. **Employee Action**:
   - The **Employee** reviews the notification and adds the requested details to their submission.
   - Once updated, the **System** logs the new data, and the process loops back to the **review step** by the **HR/Manager**.



This activity diagram illustrates the workflow for managing employee expertise submissions. The process begins with the employee submitting their expertise details, including skills, certifications, and project information, which the system logs for evaluation. The HR/Manager reviews the submitted data to assess its completeness and relevance. A decision is then made at the decision node labeled "Is the Data Sufficient?" If the data is sufficient, the system generates a summary report highlighting the key skills and certifications and updates the query status to "Resolved," marking the end of the process. However, if the data is insufficient, the system notifies the employee to provide additional details. The employee updates their expertise details, and the system re-logs the information, creating a feedback loop for further evaluation. This structured approach ensures that employee expertise

data is accurately captured, evaluated, and stored while allowing for iterative improvements if required.

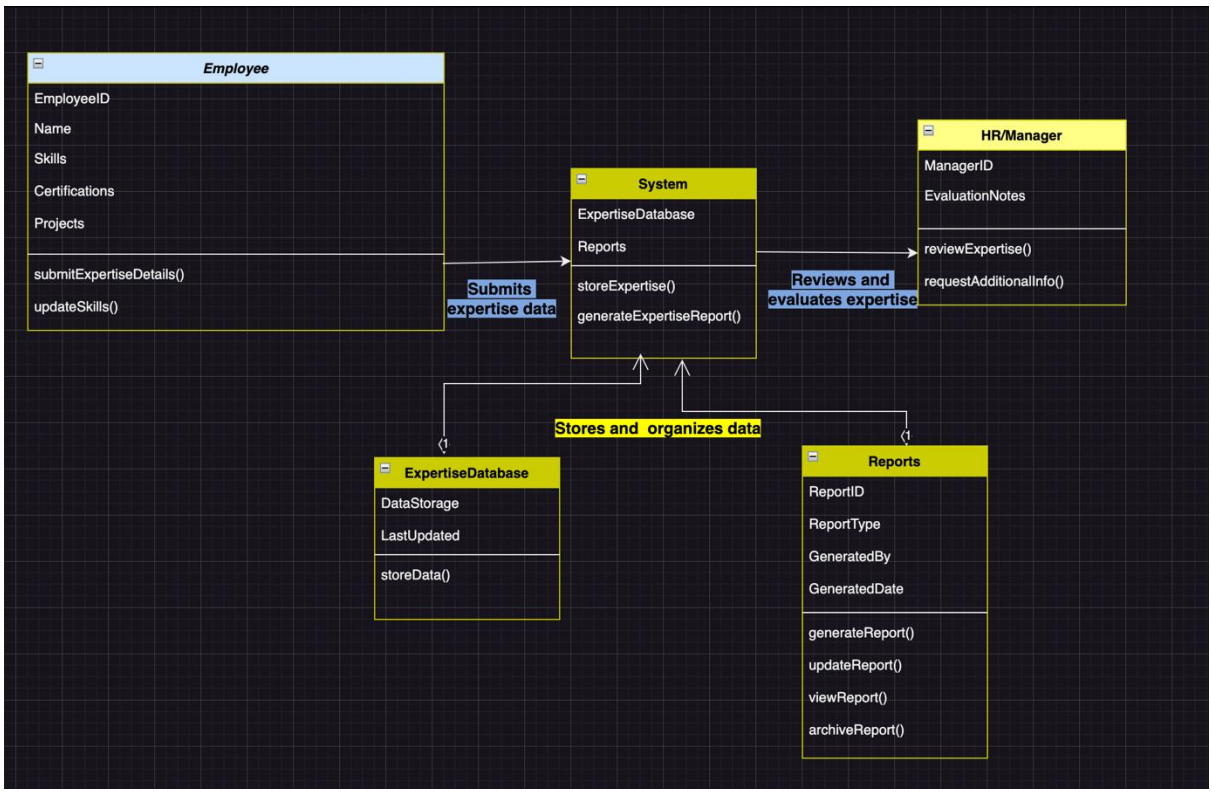
3. Class Diagram:

Classes:

1. **Employee**:
    - **Attributes**: EmployeeID, Name, Skills, Certifications, Projects.
    - **Methods**: submitExpertiseDetails(), updateSkills().
2. **HR/Manager**:
    - **Attributes**: ManagerID, EvaluationNotes.
    - **Methods**: reviewExpertise(), requestAdditionalInfo().
3. **System**:
    - **Attributes**: ExpertiseDatabase, Reports.
    - **Methods**: storeExpertise(), generateExpertiseReport().

Define Relationships:

- **Employee interacts with HR/Manager** via submitExpertiseDetails() and requestAdditionalInfo().
- **HR/Manager interacts with the System** for reviewing and updating expertise data.
- **System interacts with Employee** by storing expertise and generating reports.



This **Class Diagram** provides a detailed representation of the expertise management process for employees.

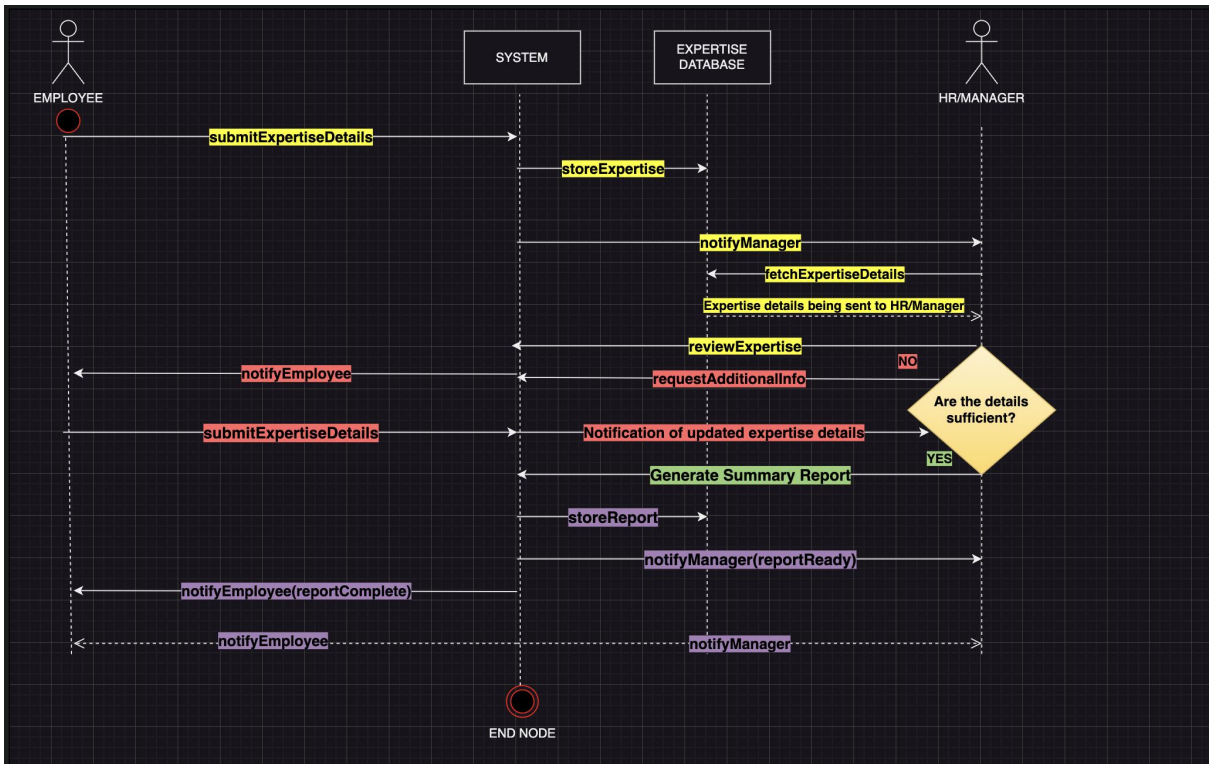
1. **Employee Class**:
   - **Attributes**: Includes details like `EmployeeID`, `Name`, `Skills`, `Certifications`, and `Projects`.
   - **Methods**:
     - `submitExpertiseDetails()`: Allows the employee to submit their expertise information to the system.
     - `updateSkills()`: Enables the employee to update their skillset.
2. **HR/Manager Class**:
   - **Attributes**:
     - `ManagerID`: Identifies the HR or Manager handling the expertise evaluation.
     - `EvaluationNotes`: Stores notes from the expertise review process.
   - **Methods**:
     - `reviewExpertise()`: Allows the manager to evaluate the employee's submitted expertise.
     - `requestAdditionalInfo()`: Used to request further details from the employee if required.
3. **System Class**:
   - **Attributes**:
     - `ExpertiseDatabase`: Links to the storage of expertise data.
     - `Reports`: Connects to the reporting system for expertise summaries.
   - **Methods**:
     - `storeExpertise()`: Stores the submitted expertise details in the database.
     - `generateExpertiseReport()`: Creates a summarized report of the employee's skills and projects.
4. **ExpertiseDatabase Class**:
   - **Attributes**:
     - `DataStorage`: Represents where the data is stored.
     - `LastUpdated`: Tracks the last modification date for the database.
   - **Methods**:
     - `storeData()`: Handles the storage of expertise information.
5. **Reports Class**:
   - **Attributes**:
     - `ReportID`: Unique identifier for each report.
     - `ReportType`: Indicates the type of report (e.g., summary or detailed).
     - `GeneratedBy`: Specifies who generated the report.
     - `GeneratedDate`: Records the date when the report was created.
   - **Methods**:
     - `generateReport()`: Creates a new report.
     - `updateReport()`: Updates an existing report.
     - `viewReport()`: Allows users to view the report.
     - `archiveReport()`: Archives old reports for future reference.
6. **Relationships**:

- The **System** class is linked to both the `ExpertiseDatabase` and `Reports` through **aggregation**, signified by diamond-headed lines, labeled as **"Stores and organizes data."**
- The **Employee** interacts with the **System** to **submit expertise data**, while the **HR/Manager** connects to the **System** for **reviewing and evaluating expertise**.

4. Sequence Diagram:

Steps:

1. Employee submits expertise details to the system.
2. The system logs the data and notifies HR/Manager.
3. HR/Manager reviews the expertise and requests additional information if needed.
4. System generates a report based on the finalized data and shares it with HR/Manager.



The **sequence diagram** represents the flow of actions for managing and evaluating an employee's expertise details within a system. Here's the detailed explanation:

1. **Employee Initiates the Process**: The process begins with the `submitExpertiseDetails` message sent from the **Employee** to the **System**. This signifies that the employee has submitted their skills, certifications, and project details for evaluation.

2. **System Logs the Data**: Upon receiving the expertise details, the **System** invokes the `storeExpertise` operation to save the data in the **Expertise Database** for further processing and retrieval.
3. **System Notifies HR/Manager**: Once the data is logged, the **System** sends a `notifyManager` message to the **HR/Manager**, alerting them that new expertise details are ready for review.
4. **HR/Manager Fetches Expertise Details**: The **HR/Manager** requests the expertise details by sending a `fetchExpertiseDetails` message to the **System**, which retrieves the data from the **Expertise Database**.
5. **System Sends Data to HR/Manager**: The fetched expertise details are returned to the **HR/Manager** for evaluation, represented by the dotted return arrow labeled `Expertise details being sent to HR/Manager`.
6. **HR Reviews Expertise**: The **HR/Manager** evaluates the submitted details through the `reviewExpertise` operation sent to the **System**. At this point, the decision node `Are the details sufficient?` comes into play.
    - **If No**: If the details are insufficient:
        - The **HR/Manager** sends a `requestAdditionalInfo` message to the **System**, which subsequently sends a `notifyEmployee` message to the **Employee**, requesting the missing or incomplete information.
        - The **Employee** resubmits the updated details through another `submitExpertiseDetails` message, and the process loops back to re-evaluation.
    - **If Yes**: If the details are sufficient:
        - The **System** generates a summary report by invoking the `Generate Summary Report` operation.
7. **Storing the Report**: The generated report is stored in the **Expertise Database** using the `storeReport` operation.
8. **Notification of Completion**: The **System** notifies both the **HR/Manager** and the **Employee** about the completion of the process:
    - A `notifyManager(reportReady)` message is sent to the **HR/Manager** to indicate the report is ready for review.
    - A `notifyEmployee(reportComplete)` message is sent to the **Employee**, informing them of the availability of their expertise report.
9. **Process Completion**: The flow ends with the `END NODE`, indicating the successful evaluation and reporting of the expertise details.

This diagram efficiently captures the interactions between the **Employee**, **System**, **HR/Manager**, and **Expertise Database**, ensuring that all possible scenarios—sufficient and insufficient details—are addressed systematically.

2.5 Progress Tracking

- **Purpose:** Enable HR or managers to monitor the performance and growth of employees during their tenure in the company, ensuring they meet project and organizational expectations.
- **Steps:**
- **Employee Performance Metrics:**
    - HR or managers can monitor:

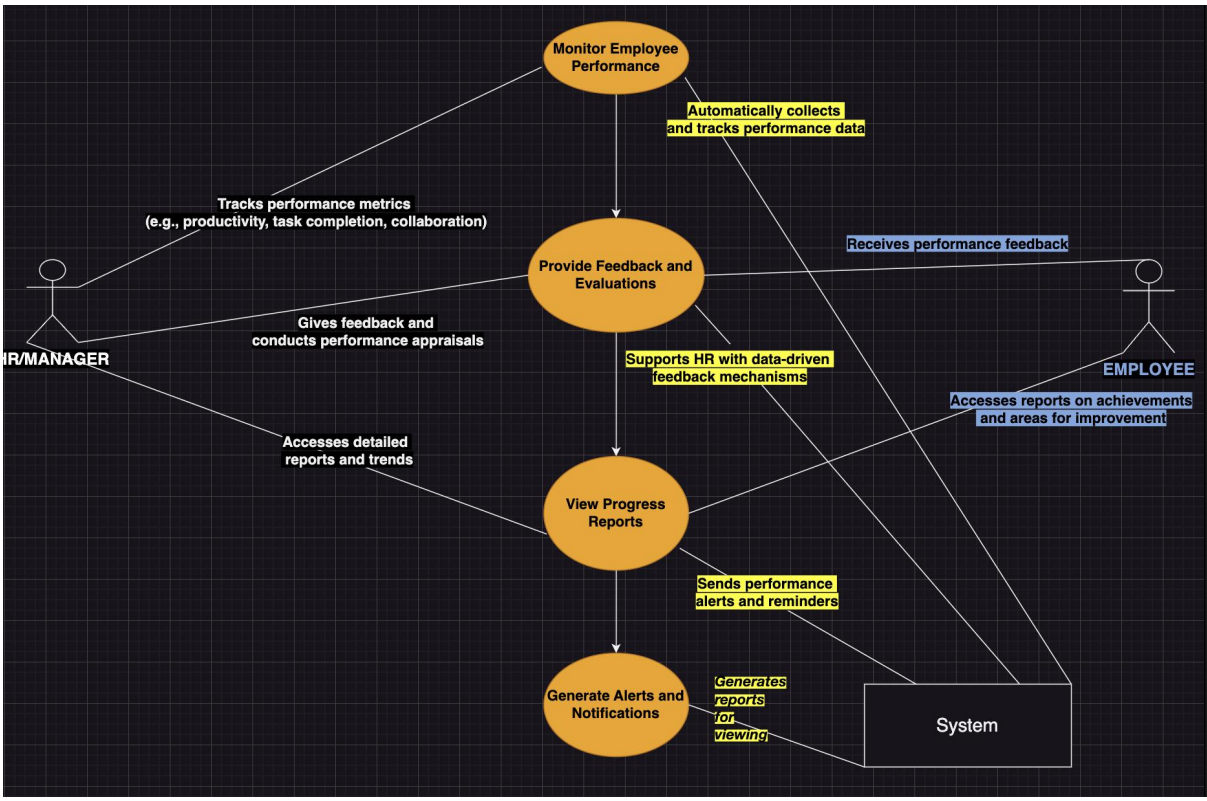
        - Task Completion Rates (e.g., project deadlines, assigned deliverables).
        - Quality of Work (e.g., code quality, client feedback).
        - Productivity (e.g., hours worked versus tasks completed).
        - Team Collaboration (e.g., peer feedback and contributions in group projects).
- **Skills & Expertise Tracking:**
    - Track how employees improve or acquire new skills over time.
    - Monitor certifications, training programs, or new responsibilities undertaken.
- **Project and Task Assignments:**
    - View all projects the employee is currently assigned to.
    - Track performance metrics for each project (e.g., on-time delivery, issue resolution).
- **Feedback and Evaluations:**
    - HR or managers can provide periodic feedback to employees.
    - Record performance evaluations during appraisals or reviews.
- **Progress Visualization:**
    - Utilize dashboards or visual reports to show:
        - Growth trends over time.
        - Achievements and milestones reached.
        - Areas for improvement.
- **Alerts and Notifications:**
    - HR and managers receive automated notifications if an employee is falling behind or excelling in their performance.
    - Employees are reminded of pending evaluations or tasks.

1. Use Case Diagram

Steps to Create:

1. **Identify Actors**:
    - HR/Manager: Monitors performance, provides feedback.
    - System: Tracks progress, generates reports, sends notifications.
    - Employee: Receives feedback, completes tasks, improves skills.
2. **Identify Use Cases**:
    - Monitor Employee Performance.
    - Provide Feedback and Evaluations.
    - View Progress Reports.
    - Generate Alerts and Notifications.
3. **Draw Relationships**:
    - Connect the HR/Manager actor to relevant use cases (e.g., Monitor Employee Performance).

- Connect the System to use cases involving automation (e.g., Generate Alerts).
- Employee interacts with feedback-related use cases.

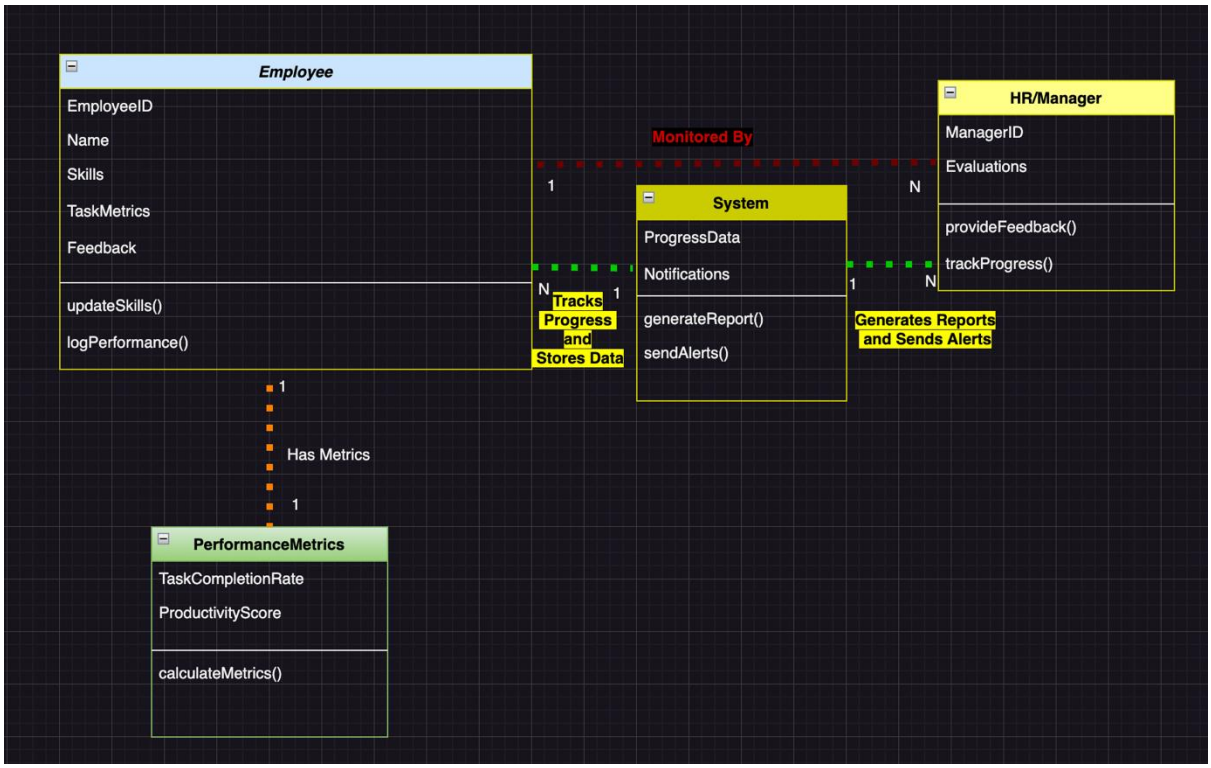


2. Class Diagram

Steps to Create:

1. **Define Classes**:
   - **Employee**: Attributes (EmployeeID, Name, Skills, TaskMetrics, Feedback); Methods (updateSkills(), logPerformance()).
   - **HR/Manager**: Attributes (ManagerID, Evaluations); Methods (provideFeedback(), trackProgress()).
   - **System**: Attributes (ProgressData, Notifications); Methods (generateReport(), sendAlerts()).
   - **PerformanceMetrics**: Attributes (TaskCompletionRate, ProductivityScore); Methods (calculateMetrics()).
2. **Add Relationships**:
   - Employee is monitored by HR/Manager ()
   - System connects to Employee and HR/Manager ()
   - PerformanceMetrics is associated with Employee ()

3. **Define Methods and Attributes for Each Class**.



This class diagram illustrates the interactions between employees, HR/managers, the system, and performance metrics. The **Employee** class includes attributes such as EmployeeID, Name, Skills, TaskMetrics, and Feedback and methods like updateSkills() and logPerformance() to track individual progress. The **HR/Manager** class manages Evaluations and provides feedback using methods like provideFeedback() and trackProgress(). The **System** acts as a central entity connecting employees and managers by storing ProgressData and Notifications, utilizing methods like generateReport() and sendAlerts() to ensure effective tracking and reporting. The **PerformanceMetrics** class is linked to the employee, encapsulating metrics like TaskCompletionRate and ProductivityScore and using the calculateMetrics() method for assessment. Relationships include HR/Managers monitoring employees, the system facilitating communication and data management, and employees having specific performance metrics for evaluation.

3. Activity Diagram

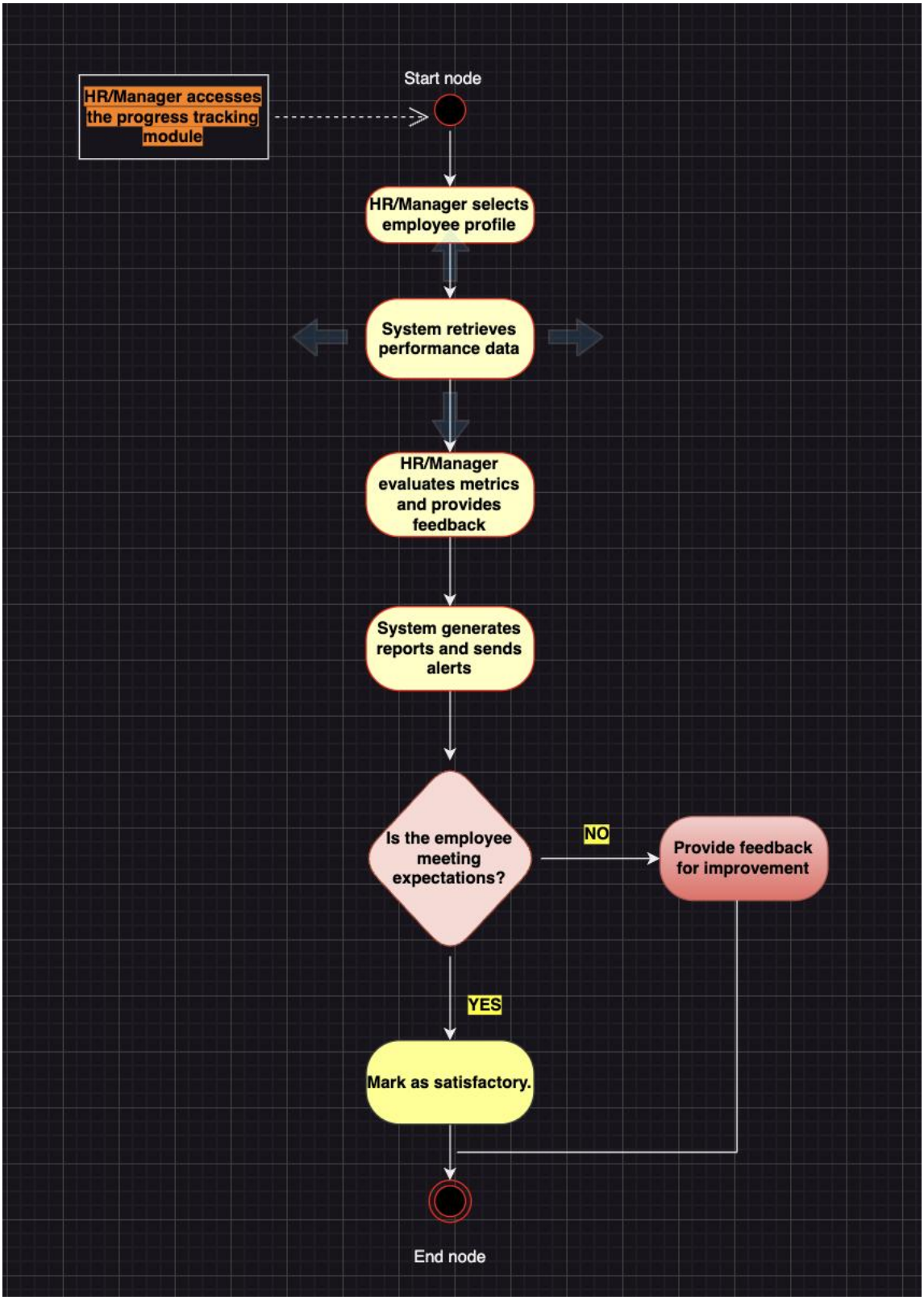
Steps to Create:

1. **Start Node**: Represent HR/Manager accessing the progress tracking module.
2. **Action Nodes**:
    - HR/Manager selects employee profile.
    - System retrieves performance data.
    - HR/Manager evaluates metrics and provides feedback.
    - System generates reports and sends alerts (if required).
3. **Decision Nodes**:

- Is the employee meeting expectations? (Yes: Mark as satisfactory; No: Provide feedback for improvement).

4. **End Node**: Progress is updated in the system.

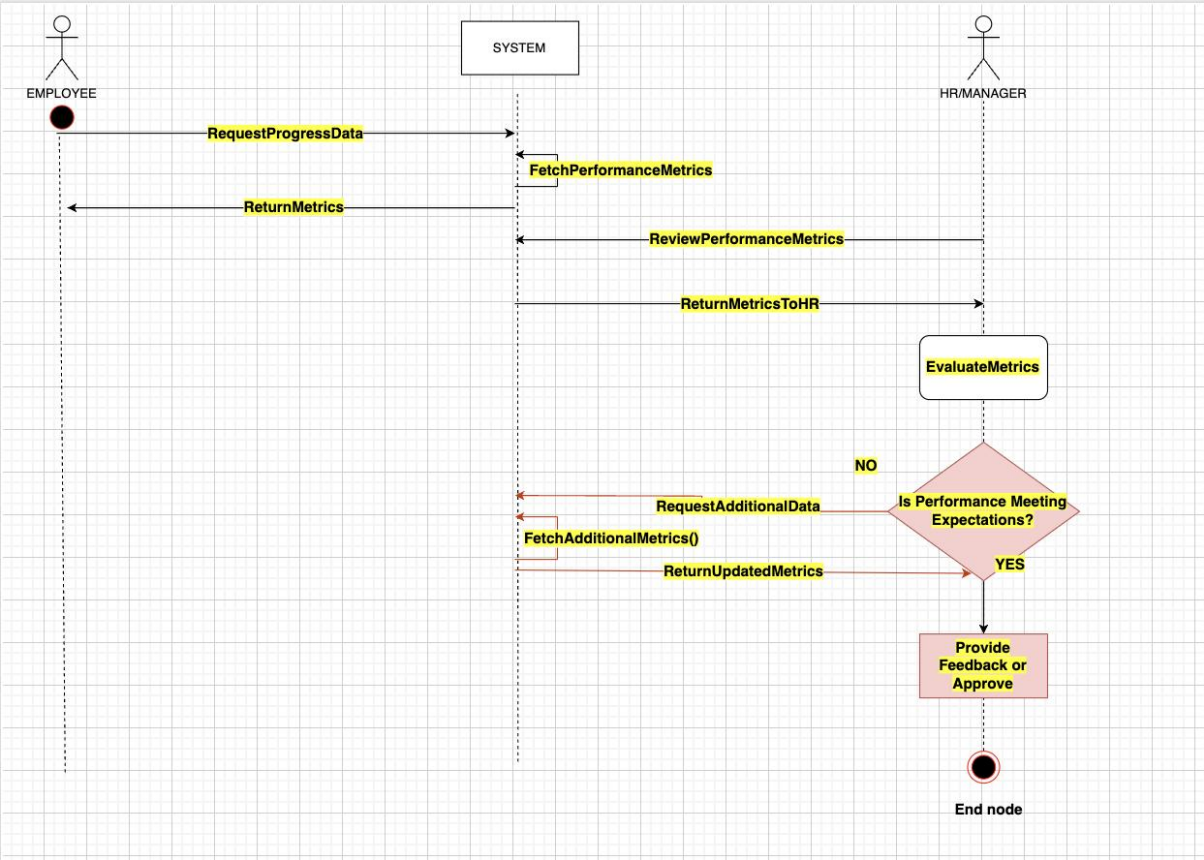
This activity diagram illustrates the progress tracking workflow for employees, starting with the HR/Manager accessing the progress tracking module to evaluate employee performance. The HR/Manager selects an employee profile (**"Select profile"**) and the system retrieves performance data (**"Retrieve data from database"**) such as task completion rates, productivity scores, and collaboration metrics. The HR/Manager evaluates the retrieved metrics (**"Evaluate performance metrics"**) and provides necessary feedback. The system then generates reports and sends alerts if required (**"Generate reports/alerts"**). At the decision node, the system assesses whether the employee is meeting expectations. If the performance is satisfactory (**"Mark performance as satisfactory"**), the process ends with the system updating the records. If not, feedback is provided to the employee for improvement (**"Provide feedback for improvement"**) and the process loops back for reevaluation (**"Re-evaluate after updates"**) once updates are made. This ensures continuous monitoring and growth of employee performance.


HR/Manager accesses the progress tracking module
Start node
HR/Manager selects employee profile
System retrieves performance data
HR/Manager evaluates metrics and provides feedback
System generates reports and sends alerts
Is the employee meeting expectations?
NO
Provide feedback for improvement
YES
Mark as satisfactory.
End node

4. Sequence Diagram

Steps to Create:

1. **Identify Interactions**:
   - HR/Manager requests progress details from the system.
   - System fetches performance metrics.
   - HR/Manager evaluates data and provides feedback.
   - System generates reports and sends alerts/notifications.
2. **Draw Lifelines**:
   - HR/Manager, System, and Employee as participants.
   - Show interactions like `requestProgressData()`, `generateAlerts()`, and `provideFeedback()`.
3. **Conditional Flow**:
   - Add conditions for sending alerts or approving progress.



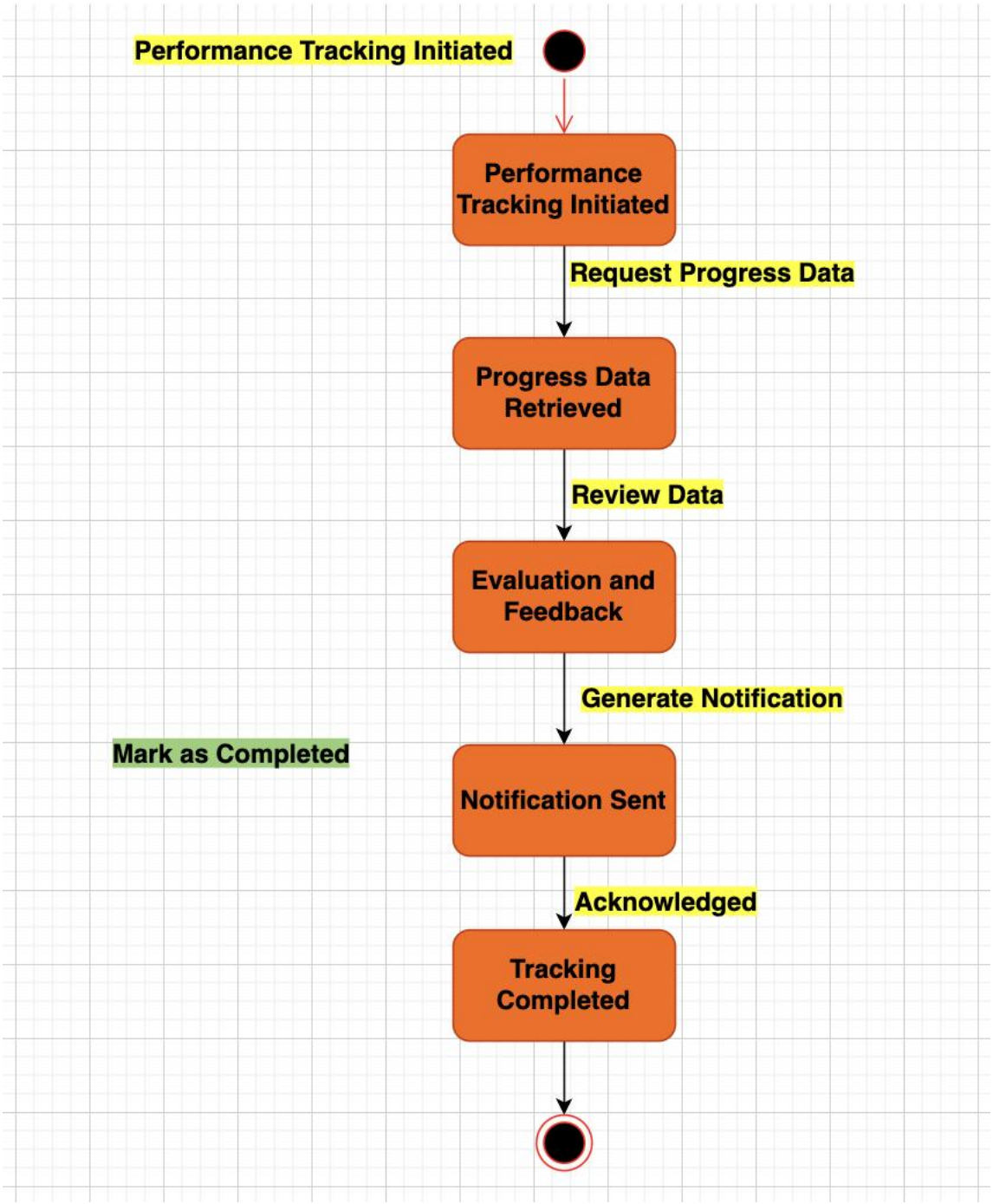
This sequence diagram represents the process of tracking and evaluating employee performance. The workflow begins with the **Employee** requesting their progress data from the **System** via the RequestProgressData message. The **System** internally fetches the relevant performance metrics using FetchPerformanceMetrics and returns the metrics back to the **Employee** via the ReturnMetrics message. Simultaneously, the **System** shares the performance data

with the **HR/Manager** by invoking the ReviewPerformanceMetrics operation, and the metrics are sent using ReturnMetricsToHR. The **HR/Manager** evaluates these metrics via the EvaluateMetrics action. At the decision point labeled **"Is Performance Meeting Expectations?"**, the system determines whether the employee meets expectations. If the data is insufficient or performance expectations are not met, the **HR/Manager** requests additional data from the **System** using the RequestAdditionalData operation. The **System** processes this request through FetchAdditionalMetrics() and sends updated metrics back to the **HR/Manager** via ReturnUpdatedMetrics, forming a feedback loop. If performance is satisfactory, the **HR/Manager** proceeds to Provide Feedback or Approve to finalize the evaluation, marking the end of the process. The clear decision flow and feedback loop ensure comprehensive performance assessment.

5. State Diagram

Steps to Create:

1. **Define States**:
    - Performance Tracking Initiated.
    - Progress Data Retrieved.
    - Evaluation and Feedback.
    - Notification Sent (if required).
    - Tracking Completed.
2. **Add Transitions**:
    - Transitions occur when HR/Manager completes each step (e.g., from "Data Retrieved" to "Evaluation Completed").



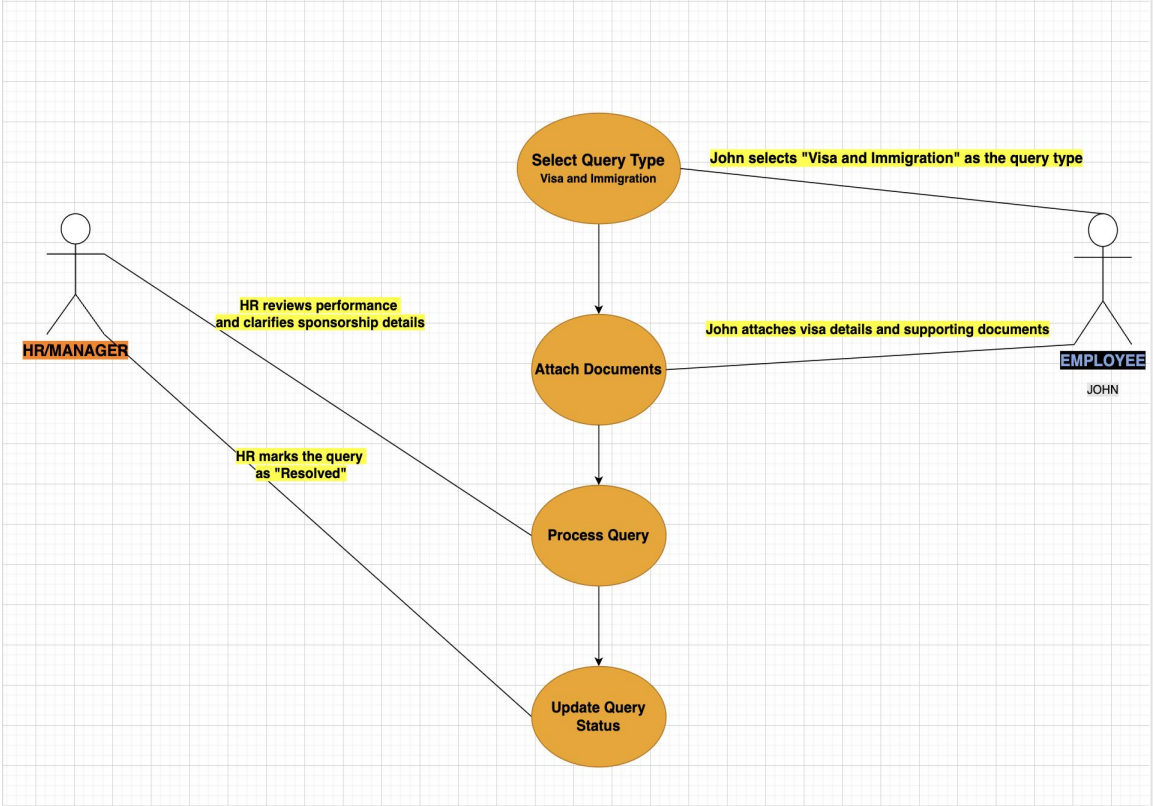
The state diagram illustrates the workflow of performance tracking, starting with the initiation of the process labeled as "Performance Tracking Initiated," where the HR/Manager accesses the system to track employee performance. The process transitions to "Progress Data Retrieved," where the system gathers relevant performance metrics such as task completion rates and productivity scores. This data is then reviewed by the HR/Manager in the "Evaluation and Feedback" state, where they analyze the metrics and provide feedback. If additional action is required, the system generates a notification in the "Notification Sent" state, alerting the necessary stakeholders. Once the notification is acknowledged or the performance evaluation is finalized, the workflow

concludes in the "Tracking Completed" state, marking the end of the process.

2.5.1 Case Management During Onboarding

Case 1: Visa and Immigration Assistance

- **Purpose:** Clarify H1B sponsorship and visa requirements.
- **Scenario:**
  - John selects the query type **"Visa and Immigration."**
  - Description provided: *"When will my H1B sponsorship be initiated? What are the required documents, and is there support for dependent visas?"*
  - Current visa details and supporting documents are attached.
- **Resolution:**
  - HR clarifies that H1B sponsorship begins after six months of employment, pending performance reviews and project contributions.
  - Instructions for document submission are shared.
  - Query status is updated to **"Resolved."**



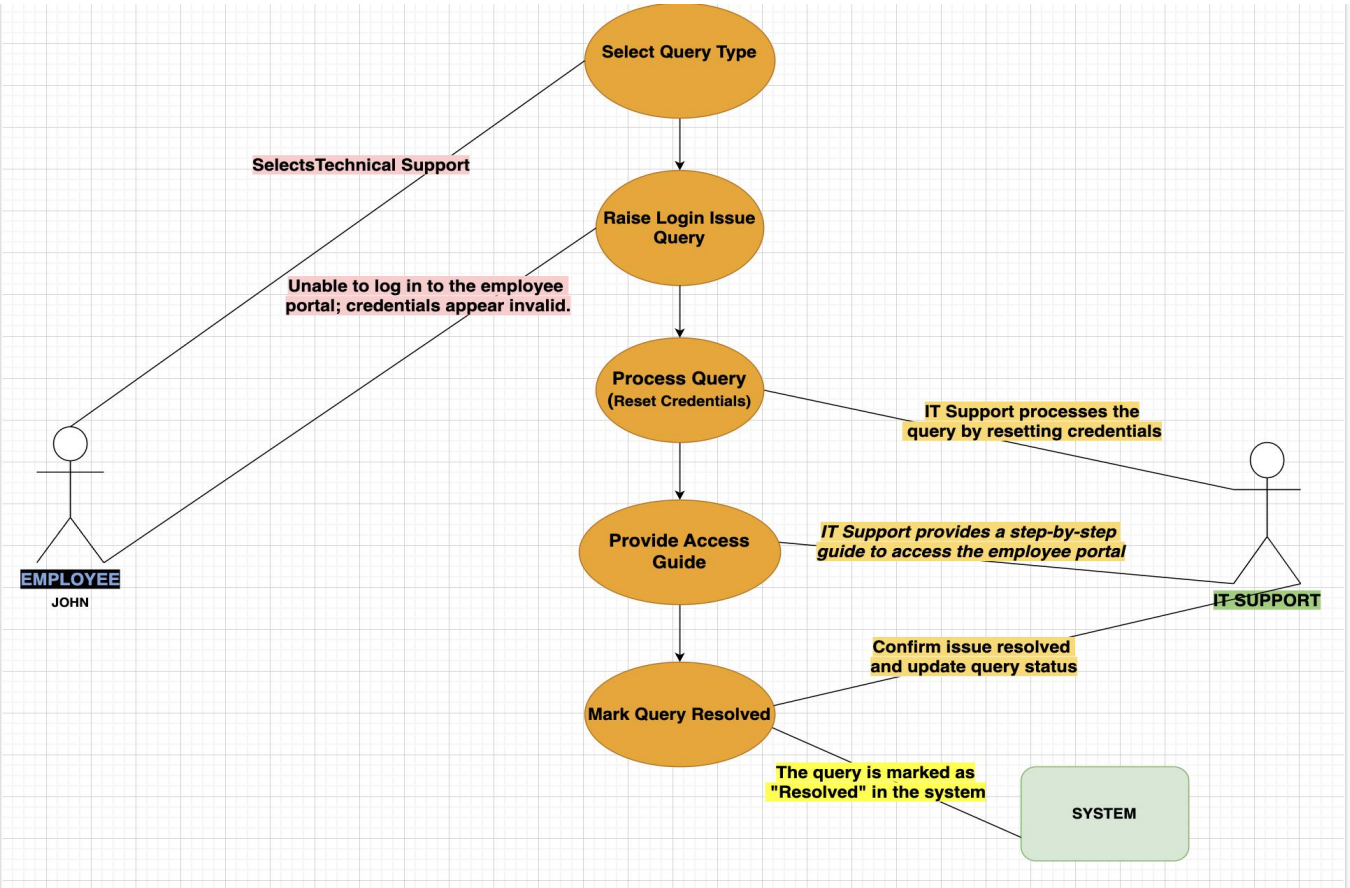
The Use Case Diagram illustrates the process of Visa and Immigration Assistance involving three main actors: the Employee (John), the HR Manager, and the system. John begins the interaction by selecting the query type "Visa and

Immigration" and specifying his query details, such as the H1B sponsorship timeline, required documents, and support for dependent visas. The next step involves attaching supporting documents, including current visa details, which the system processes. The HR Manager evaluates the query, reviewing John's performance and contributions to clarify that H1B sponsorship will begin after six months of employment. They also provide instructions for document submission. Finally, the HR Manager updates the query status to "Resolved," completing the workflow. The diagram clearly depicts the flow of actions, including the association between actors and specific tasks, ensuring seamless communication and resolution within the system.

Case 2: Technical Support for Onboarding Tools

- **Purpose:** Resolve login issues with onboarding portals.
- **Scenario:**
    - John raises a query: *"Unable to log in to the employee portal; credentials appear invalid."*
    - The query type **"Technical Support"** is selected.
- **Resolution:**
    - IT resets credentials and provides a step-by-step guide to access the portal.
    - Query status is marked as **"Resolved."**

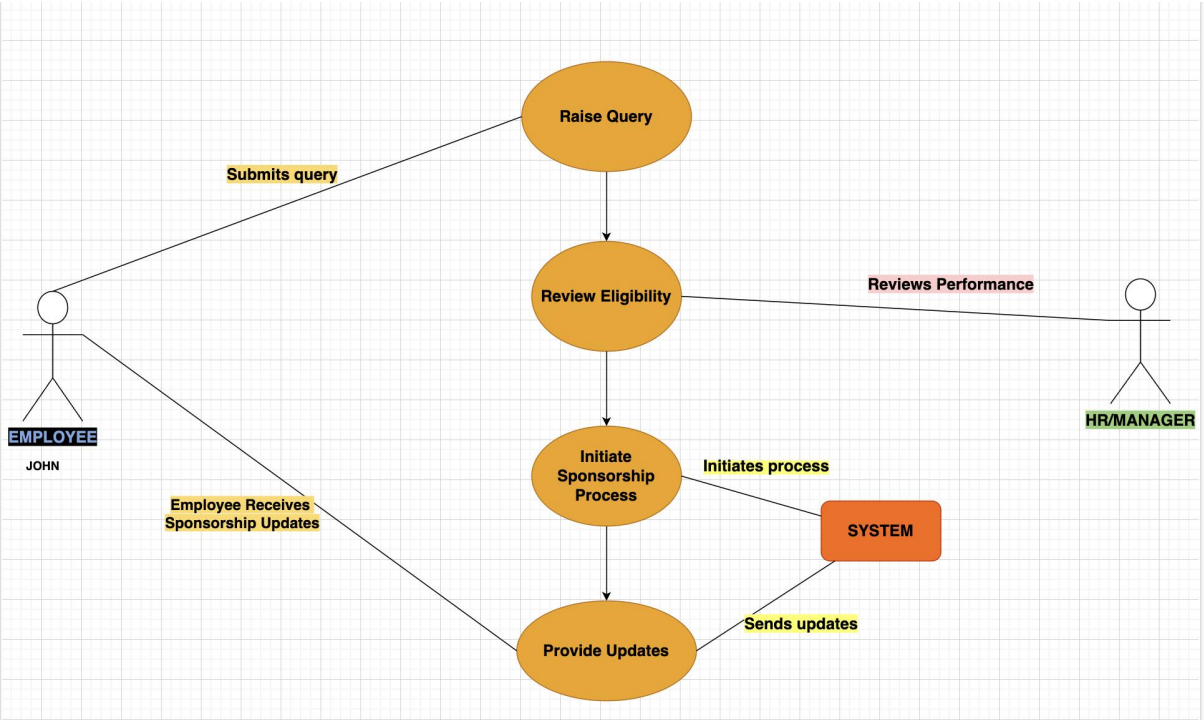
This use case diagram illustrates the resolution process for a technical support query related to login issues with the onboarding portal. The employee, John, selects "Technical Support" as the query type and raises a login issue query, detailing the problem of invalid credentials. IT Support processes the query by resetting the credentials through the system and provides a step-by-step access guide to John to ensure he can log into the portal successfully. Finally, IT Support confirms the resolution of the issue and updates the system, marking the query as "Resolved." The system also facilitates communication and ensures that the resolution process is tracked and documented effectively.



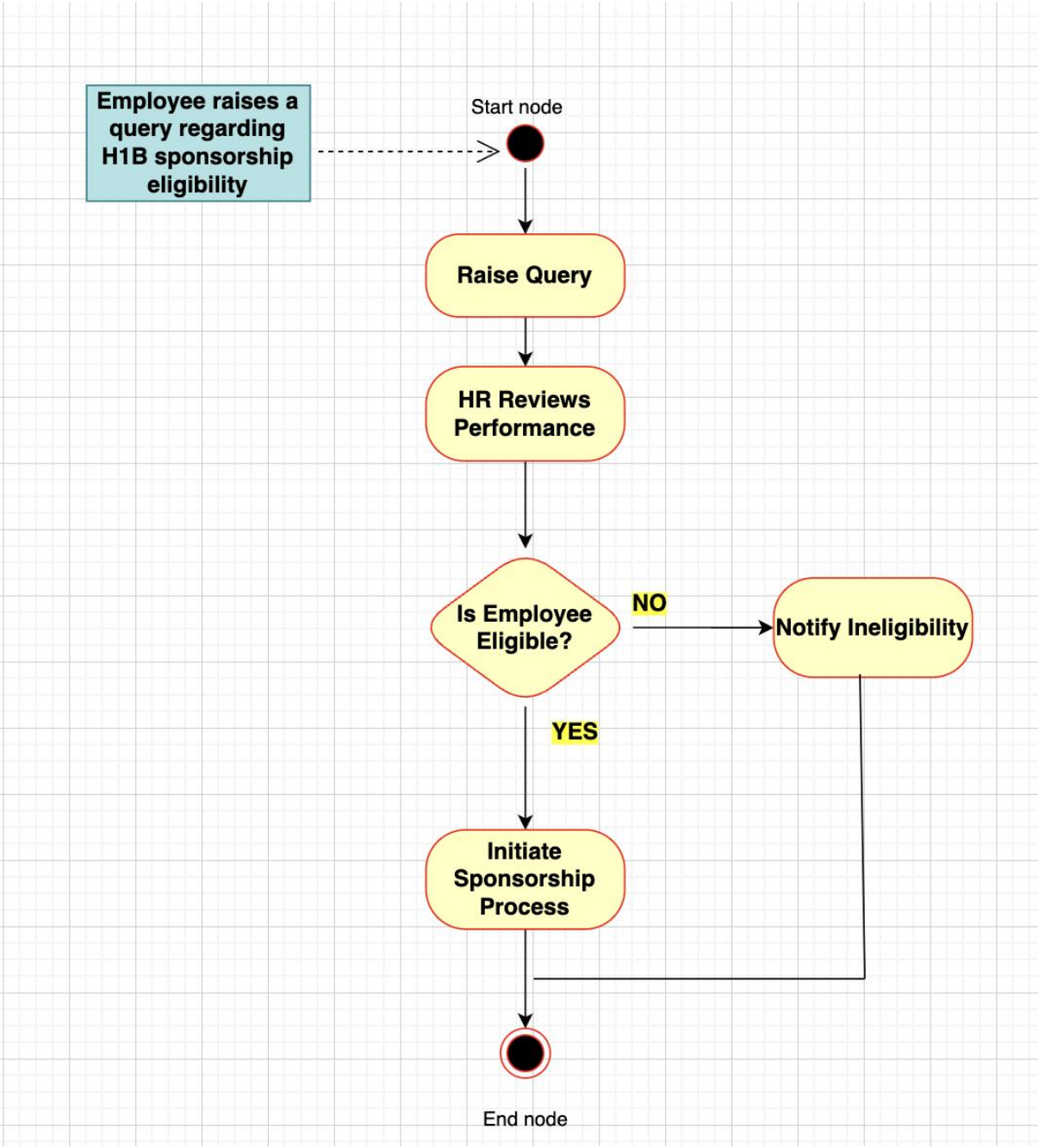
2.5.2 Case Management During Employment

Case 3: Visa Sponsorship Timeline

- **Purpose:** Monitor eligibility and progress for H1B sponsorship.
- **Scenario:**
  - John raises a follow-up query: *"I am nearing six months of employment. How do I confirm if I qualify for H1B sponsorship?"*
- **Resolution:**
  - HR reviews John's performance and confirms eligibility.
  - The sponsorship process is initiated, and updates are provided through the Case Management system.



The Use Case Diagram illustrates the Visa Sponsorship Timeline process, focusing on the interactions between the Employee (John), HR/Manager, and the System. John raises a query regarding his H1B sponsorship eligibility through the "Raise Query" use case. The System forwards the request to HR/Manager, who reviews John's performance and eligibility under the "Review Eligibility" use case. If eligible, HR initiates the sponsorship process via the "Initiate Sponsorship Process" use case, which is tracked and managed by the System. Throughout the process, the System provides periodic updates to John under the "Provide Updates" use case, ensuring he is informed about the progress. The Employee interacts with the System to both submit the query and receive updates, while HR leverages the System to evaluate eligibility and initiate sponsorship, creating a seamless workflow for managing sponsorship requests. The labeled relationships, such as "Submits Query," "Reviews Performance," "Initiates Sponsorship," and "Receives Sponsorship Updates," clarify the responsibilities and interactions between the actors and the system.



1. Start Node

- **Purpose:** Begin the process when the employee raises a query regarding H1B sponsorship eligibility.

2. Action Node: Raise Query

- **Action:** The Employee submits a query about H1B sponsorship.

3. Action Node: HR Reviews Performance

- **Action:** HR accesses the system to review the Employee's performance.
- **Details Reviewed:**

    - Task completion metrics.
    - Feedback from peers or supervisors.
    - Overall contributions to projects.

4. Decision Node: Is the Employee Eligible?

- **Decision:** HR evaluates if the Employee meets the criteria for H1B sponsorship based on performance metrics.
- **Criteria Checked:**
    - Six months of employment.
    - Positive performance reviews.
    - Contribution to critical projects.

5. If YES:

- **Action Node: Initiate Sponsorship Process**
    - HR approves the sponsorship, and the system initiates the required process.
- **Action Node: Send Updates**
    - The system notifies the Employee about sponsorship approval and updates.

6. If NO:

- **Action Node: Notify Ineligibility**
    - HR sends a notification to the Employee explaining the reasons for ineligibility.

7. End Node

- **Purpose:** Conclude the workflow after completing the sponsorship process or providing updates.

Case 4: General Query – Vacation Policy

- **Purpose:** Clarify time-off policy and application process.
- **Scenario:**
    - John queries under **"General Inquiry"**: *"What is the policy for vacation days, and how can I apply for time off?"*
- **Resolution:**
    - HR explains that employees are entitled to 15 annual vacation days and shares the Leave Management portal link.
    - Status is marked as **"Resolved."**

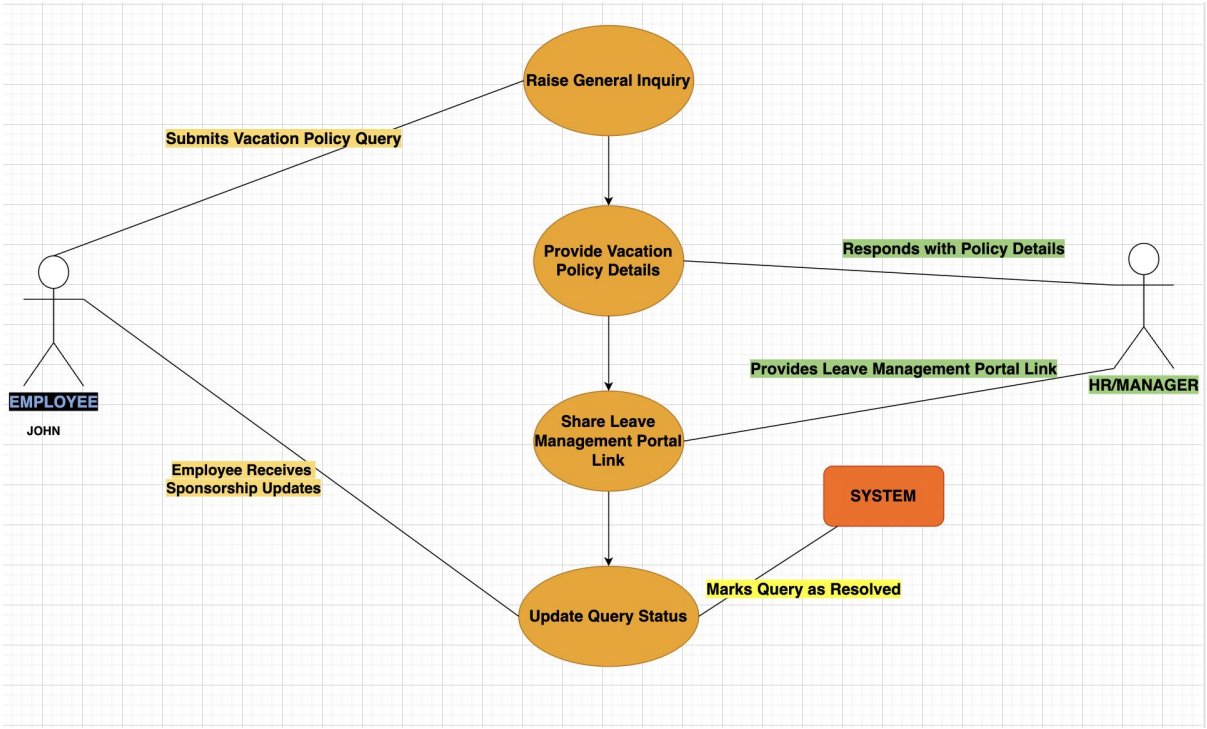
1. Use Case Diagram

This diagram will represent the actors, the system, and the actions involved in handling the vacation policy query.

Steps for Use Case Diagram:

- **Actors:**
  - **Employee (John):** Raises the query.
  - **HR/Manager:** Responds to the query.
  - **System:** Manages the Leave Management portal link and query status.
- **Use Cases:**
  - **Raise General Inquiry:** Employee submits a query about the vacation policy.
  - **Provide Vacation Policy Details:** HR responds to the query.
  - **Share Leave Management Portal Link:** HR provides the link for applying for time off.
  - **Update Query Status:** System marks the query as "Resolved."
- **Relationships:**
  - Employee interacts with the system to raise a query.
  - HR interacts with the system to provide details and resolve the query.

The Use Case Diagram for the vacation policy query represents three key actors: the Employee (John), HR/Manager, and the System. The Employee interacts with the system to **submit a vacation policy query** through the "Raise General Inquiry" use case. HR/Manager then **responds with policy details**, explaining the entitlement of 15 annual vacation days through the "Provide Vacation Policy Details" use case. Additionally, HR/Manager **shares the Leave Management Portal link** to guide the Employee on the time-off application process, represented by the "Share Leave Management Portal Link" use case. The System, upon receiving HR's response, **marks the query as resolved** through the "Update Query Status" use case. The relationships are labeled for clarity: the Employee submits the query, HR responds and provides the portal link, and the System handles query resolution. This workflow ensures that the Employee receives the required vacation policy details and resources efficiently, with the System maintaining query status transparency.

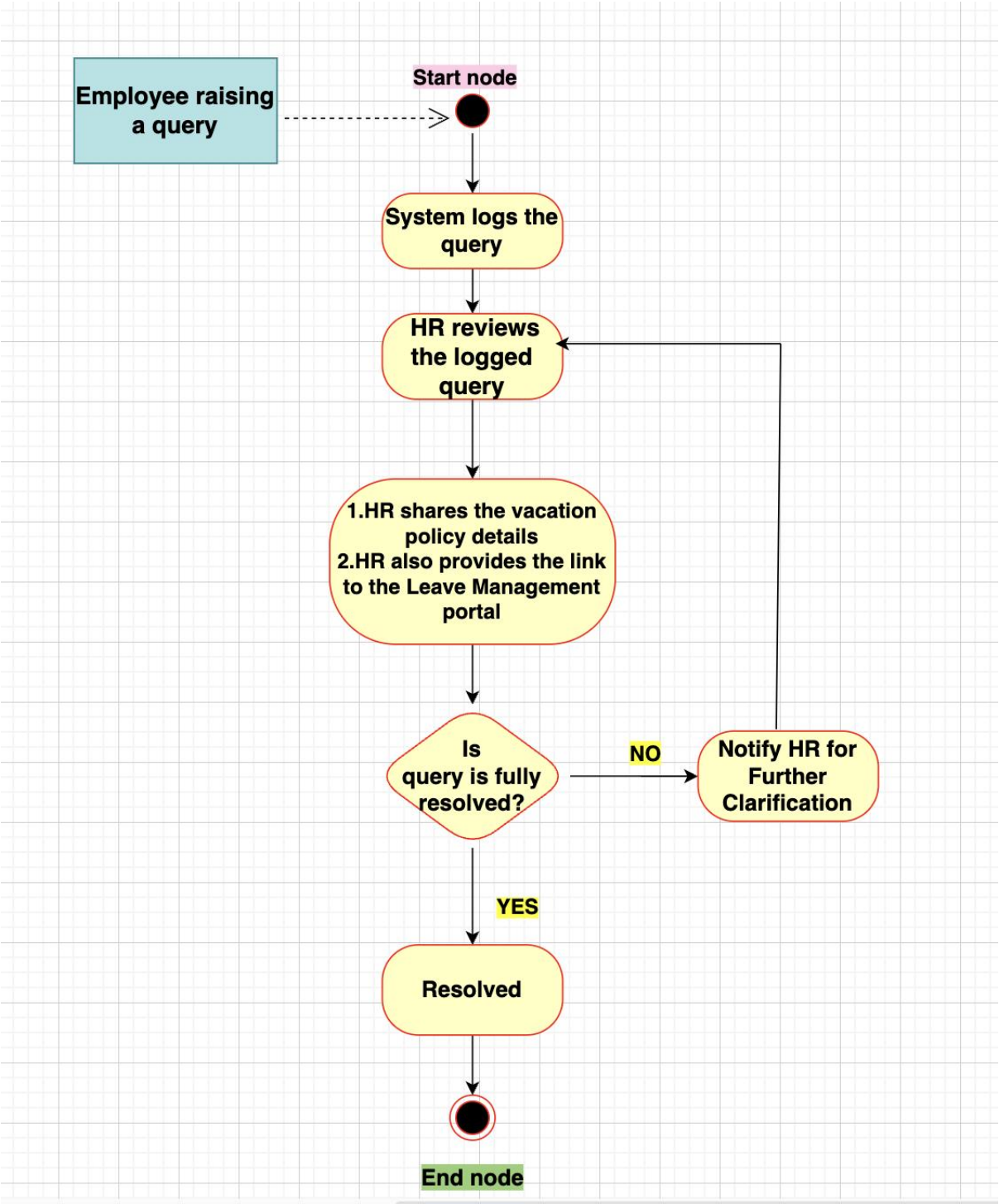


2. Activity Diagram

This diagram will illustrate the step-by-step workflow for processing the vacation policy query.

Steps for Activity Diagram:

1. **Start Node:** Employee raises a query about the vacation policy.
2. **Action Nodes:**
    - System logs the query under "General Inquiry."
    - HR reviews the query and provides details about the vacation policy (15 annual days).
    - HR shares the Leave Management portal link for applying for time off.
3. **Decision Node:**
    - Is the query resolved?
        - **Yes:** Update query status to "Resolved."
        - **No:** Notify HR for further clarification.
4. **End Node:** The query status is updated, and the process is completed.



Detailed Step-by-Step Process for the Activity Diagram

1. **Start Node**:
   - The process begins with the **Employee raising a query** about the vacation policy through the case management system.
   - The query is categorized as a "General Inquiry" in the system.
2. **Action Node 1**:
   - The **System logs the query** under the "General Inquiry" category for tracking and resolution purposes.

    - The query includes the employee's question: "What is the policy for vacation days, and how can I apply for time off?"
3. **Action Node 2**:
    - The **HR reviews the logged query** in the system. They access the details of the query to prepare a response.
    - HR retrieves information about the company's vacation policy. For instance:
        - Employees are entitled to **15 annual vacation days**.
        - Time-off requests must be submitted through the **Leave Management portal**.
4. **Action Node 3**:
    - HR **shares the vacation policy details** with the Employee, including the number of vacation days and any additional guidelines or rules related to time off.
    - HR also **provides the link to the Leave Management portal** so the Employee can apply for vacation days directly.
5. **Decision Node**:
    - The system evaluates whether the query is fully resolved:
        - **If Yes**:
            - The system updates the query status to "Resolved."
            - A confirmation is sent to the Employee indicating that the query has been successfully addressed.
        - **If No**:
            - The system **notifies HR** that further clarification is needed.
            - HR reviews the query again to provide additional details or address follow-up questions.
6. **End Node**:
    - Once the query is fully resolved, the status is updated to "Resolved" in the system.
    - The process concludes with the Employee having complete information about the vacation policy and access to the portal to apply for time off.

Why These Diagrams:

- The **Use Case Diagram** provides a high-level view of the actors and their interactions with the system.
- The **Activity Diagram** captures the detailed workflow for processing and resolving the query, ensuring each point in the scenario is addressed.

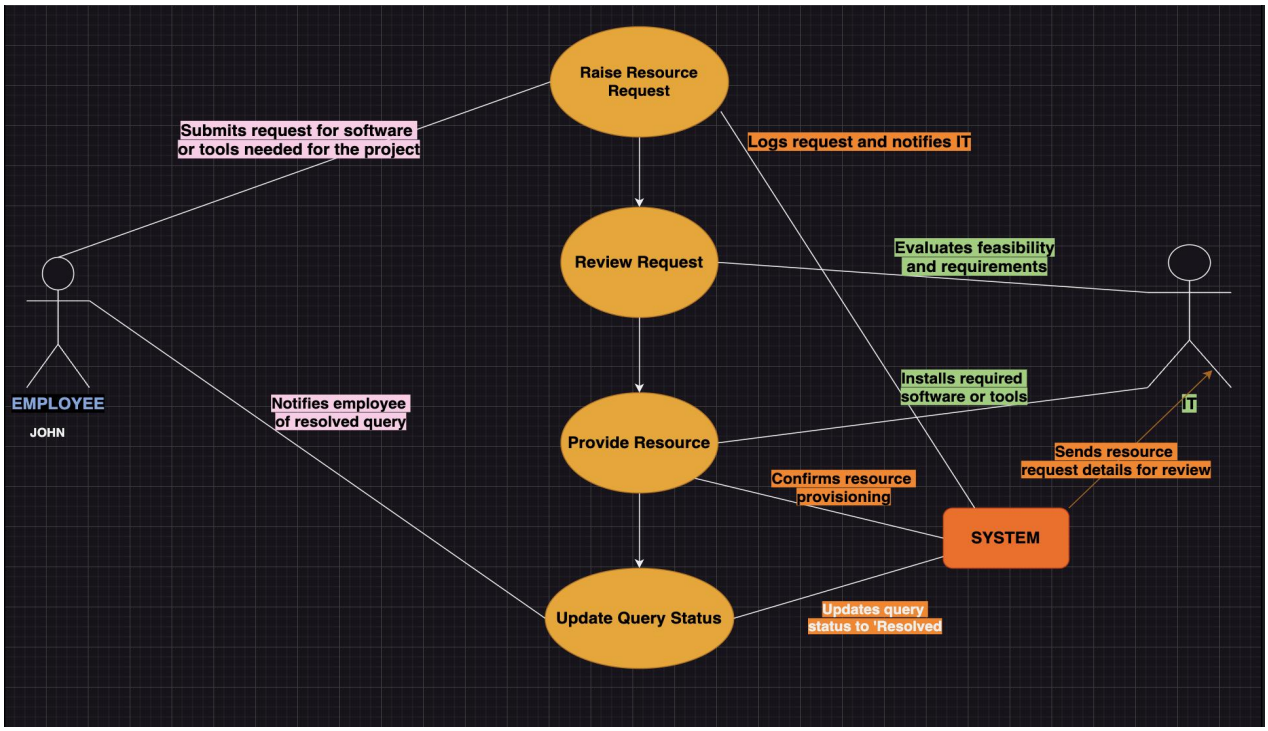
Case 5: Project Resource Request

- **Purpose:** Request tools or software for project work.
- **Scenario:**

- John queries: *"My project requires access to software not currently installed on my workstation. Can this be arranged?"*

- **Resolution:**
  - IT installs the necessary software and updates the query status to **"Resolved."**

This **Use Case Diagram** illustrates the process of handling a project resource request. The employee submits a query for required software or tools through the "Raise Resource Request" use case, which is logged and forwarded to the system. The system notifies IT, who reviews the request to evaluate feasibility and requirements under the "Review Request" use case. If the request is approved, IT proceeds to the "Provide Resource" step, where the required software or tools are installed. Once the resource is provisioned, IT confirms the action to the system, which updates the query status to "Resolved" through the "Update Query Status" use case. Finally, the system notifies the employee that their query has been resolved, ensuring a seamless communication and resolution process.



2. Activity Diagram

This diagram visualizes the workflow for handling the project resource request from start to resolution.

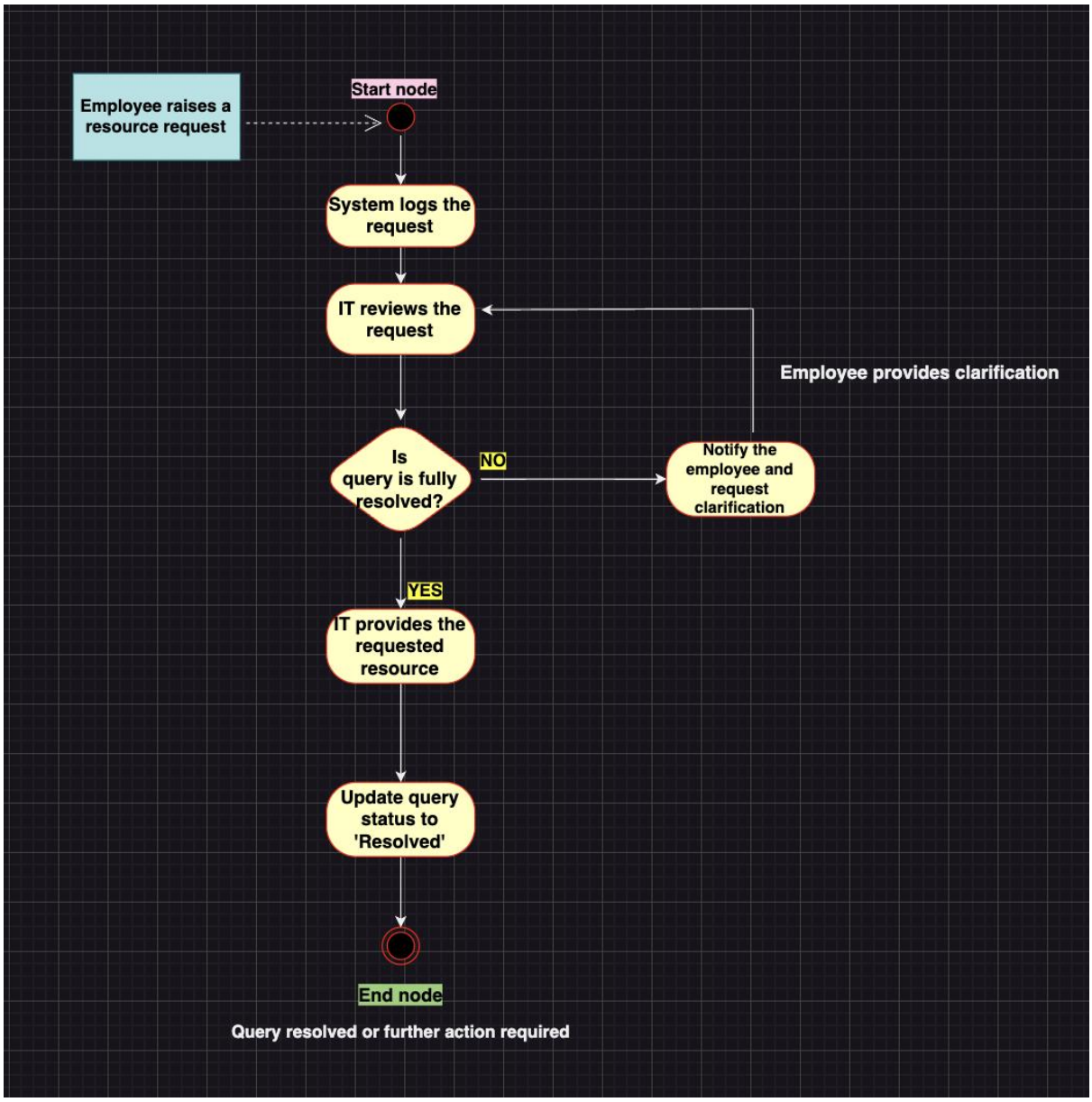
Steps for the Activity Diagram:

1. **Start Node:** Employee raises a resource request.
2. **Action Nodes:**
   - System logs the request.

- IT reviews the request for validity and feasibility.
- IT installs the requested software or provides the necessary tools.

3. **Decision Node:**
   - **Is the Resource Provided?**
     - **Yes:** Update query status to "Resolved."
     - **No:** Notify the Employee and request additional clarification.
4. **End Node:** Query status is updated, and the process ends.

The activity diagram illustrates the process for handling a project resource request. The process begins when an employee raises a request, logged by the system. IT reviews the request to assess its validity and feasibility. If the query is not fully resolved, IT notifies the employee and requests clarification, forming a feedback loop where the employee provides additional information, and IT re-evaluates the request. If the query is valid and fully resolved, IT proceeds to provide the requested resource, such as software installation or tools. Finally, the system updates the query status to "Resolved," and the process ends. This workflow ensures iterative feedback and complete resolution of the resource request.



Case 6: Feedback on Response

- **Purpose:** Request feedback on completed projects.
- **Scenario:**
    - John queries: *"Can I receive feedback on my recent project to understand areas of improvement?"*
- **Resolution:**
    - The manager provides detailed feedback during a one-on-one session, and the case is closed.

For the **Feedback on Response** case, the following two UML diagrams are suitable:

1. Use Case Diagram

This diagram will focus on the interactions between the Employee, Manager, and the System in the feedback request and resolution process.

Steps:

1. **Actors:**
    - Employee (John): Requests project feedback.
    - Manager: Reviews the project and provides feedback.
    - System: Manages the feedback query and its status.
2. **Use Cases:**
    - Raise Feedback Request: Employee submits a query for feedback on a project.
    - Review Project: Manager evaluates the project for performance and areas of improvement.
    - Provide Feedback: Manager delivers feedback during a one-on-one session.
    - Update Query Status: System marks the feedback case as "Resolved."
3. **Relationships:**
    - Employee interacts with the System to raise a feedback request.
    - Manager interacts with the System to review the project and provide feedback.
    - System handles the query lifecycle, including updating its status.

Define the Use Cases

1. **Raise Feedback Request:**
    - Employee (John) logs into the system and submits a query to request feedback on a specific project.
    - This use case captures the interaction where John formally requests feedback via the system.
2. **Review Project:**
    - The manager accesses the system to view the feedback request.

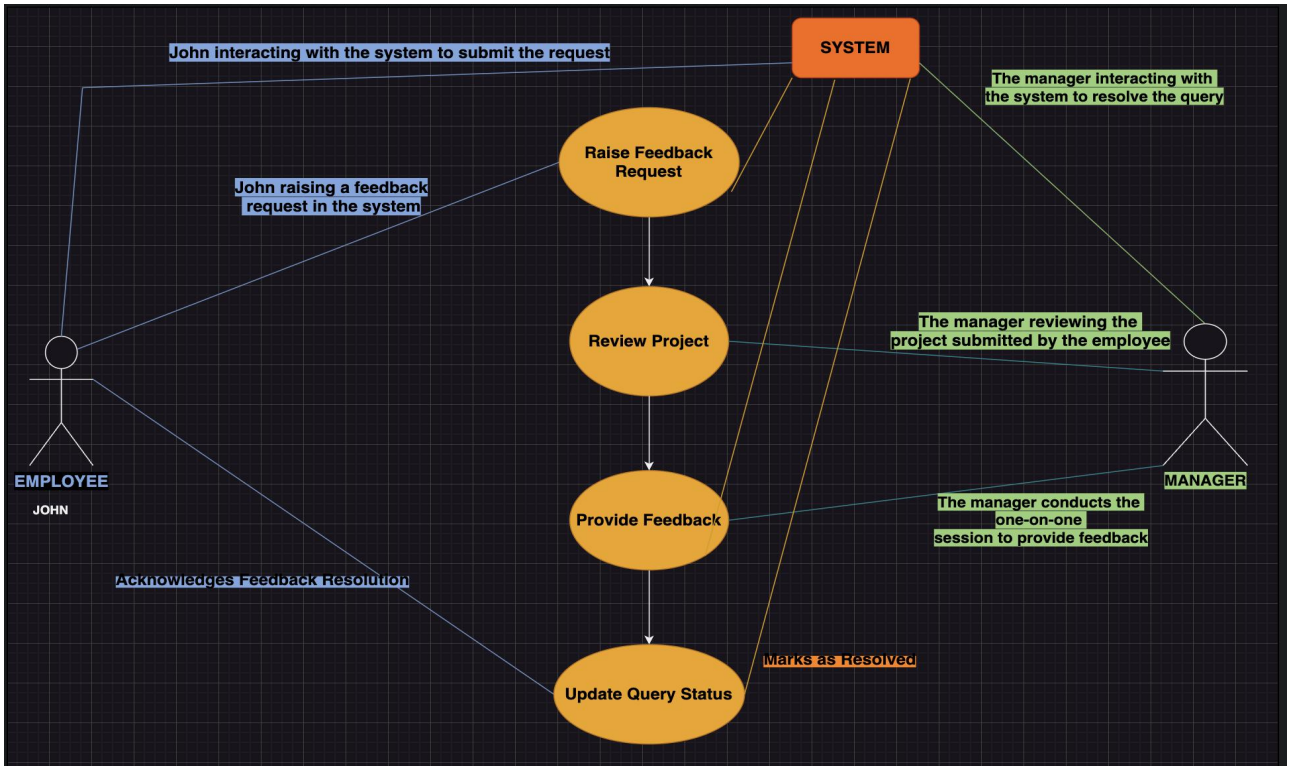
- The manager reviews the details of the submitted project, evaluates performance, and identifies areas for improvement.
- This use case involves the manager preparing feedback for John based on the project details.

3. **Provide Feedback:**
   - The manager conducts a one-on-one session with John to share detailed feedback.
   - This session can include discussing strengths, areas for improvement, and suggestions for growth.
4. **Update Query Status:**
   - Once the feedback session is completed, the system updates the status of the feedback request to "Resolved."
   - This ensures that the query lifecycle is formally closed.

Define the Relationships

1. **Employee to System:**
   - John interacts with the system to submit the feedback request.
   - The system allows John to view the status of his query.
2. **Manager to System:**
   - The manager uses the system to access the feedback request, review the project, and mark the query as resolved after providing feedback.
3. **System Workflow:**
   - The system serves as the central mediator, facilitating the lifecycle of the query:
     - Logging the feedback request.
     - Storing project details for the manager's review.
     - Updating the query status once feedback is provided.

This use case diagram represents the workflow for processing a feedback request from an employee. The **Employee (John)** initiates the process by interacting with the **System** to raise a feedback request for a recently completed project. The **Manager** reviews the project through the system, evaluating performance and identifying areas for improvement. The manager then conducts a one-on-one session with the employee to provide detailed feedback. After the feedback is provided, the employee acknowledges the resolution, and the **System** updates the query status to "Resolved." The diagram illustrates the interactions and relationships between the employee, manager, and system, showcasing the steps involved in addressing and resolving the feedback request efficiently.

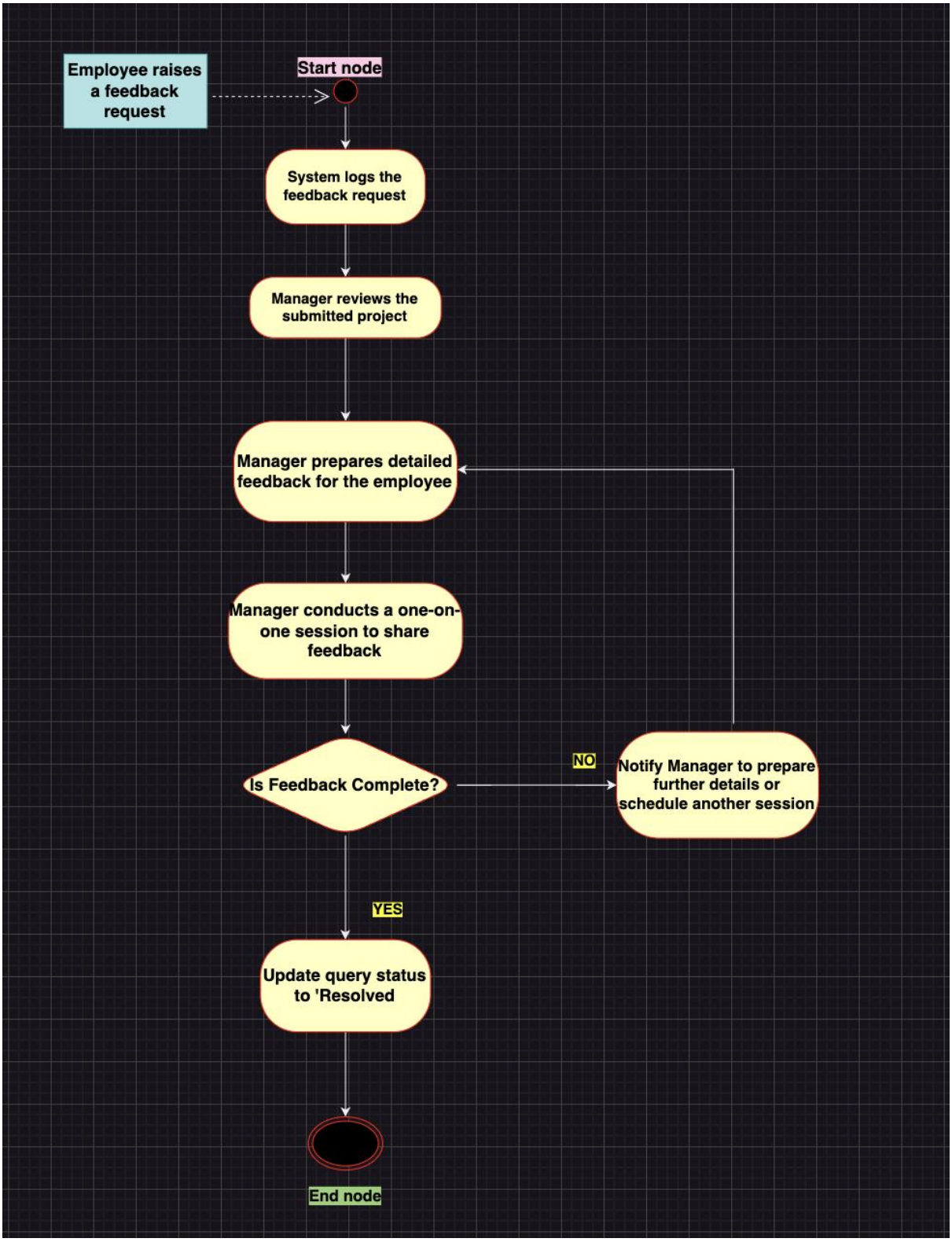


2. Activity Diagram

This diagram will represent the workflow for processing the feedback request.

Steps:

1. **Start Node:** Employee raises a feedback request.
2. **Action Nodes:**
    - System logs the feedback request.
    - Manager reviews the submitted project.
    - Manager prepares detailed feedback for the employee.
    - Manager conducts a one-on-one session to share feedback.
3. **Decision Node:**
    - Is Feedback Complete?
        - Yes: Update query status to "Resolved."
        - No: Notify Manager to prepare further details or schedule another session.
4. **End Node:** The feedback case is marked as resolved.



This activity diagram outlines the workflow for processing an employee's feedback request. The process starts with the employee raising a feedback request, which the system logs for processing. The manager reviews the submitted project and prepares detailed feedback. The manager then conducts a one-on-one session with the employee to share feedback. At the decision node, it is evaluated whether the feedback is complete. If the feedback is deemed complete, the system updates the

query status to "Resolved," ending the process. If the feedback is not complete, the system notifies the manager to prepare further details or schedule another session, creating a feedback loop to ensure all issues are addressed. This iterative process ensures that the employee receives comprehensive feedback and the query is closed satisfactorily.

General Features of the Case Management System

1. **Comprehensive Query Types:**
    - Visa and immigration.
    - Payroll and benefits.
    - Technical support.
    - Project resource allocation.
    - General inquiries.
2. **Real-Time Updates:**
    - Automated notifications for status changes (e.g., "New," "In Progress," "Resolved").
    - Follow-up reminders for unresolved queries.
3. **Document Attachments:**
    - Employees can upload relevant documents for queries (e.g., visa, project deliverables).
4. **Analytics and Insights:**
    - HR and managers can track query trends to address recurring issues proactively.