**The National University of Computer and Emerging Sciences**

**Islamabad Campus**



# **Database systems**

# **Vendor & contract system**

**Hamza Asad 22i-1908 / Shahroze Naveed 22i-1922**

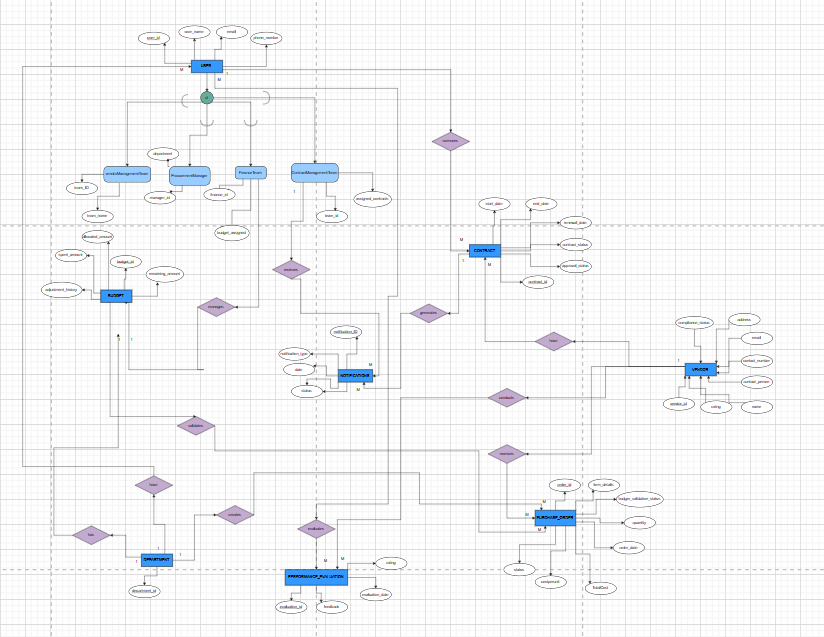
**DB LAB DS – A**

**1. Project Overview**

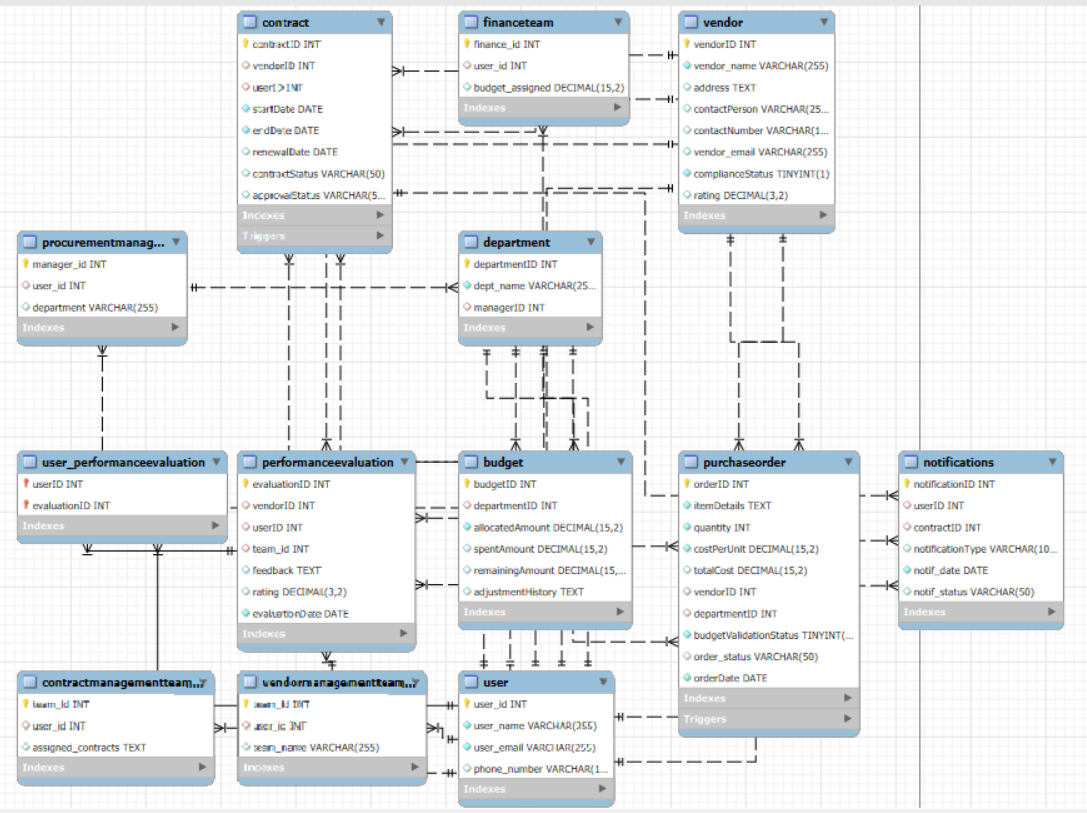
The Procurement Management System is designed to streamline and automate various procurement-related tasks for an organization, ensuring compliance, efficiency, and effective vendor management. The system covers the following use cases:

1. Vendor Registration and Login
2. Contract Management
3. Purchase Order Management
4. Vendor Performance Tracking
5. Compliance and Auditing
6. Task Allocation for Procurement Process
7. Budget Monitoring and Tracking

**2. Entity-Relationship Diagram (ERD)**

****

**3. Relational Schema**

****

**4. Table Descriptions**

1. **User Class**

The User class represents the generalized concept of all users in the system. Each user can belong to a specific role, such as **Vendor Management Team**, **Procurement Manager**, or **Finance Team**. These roles are **disjoint**, meaning each user belongs to exactly one of these specialized roles.

**User (Superclass)**:

* user\_id (PK)
* name
* email
* phone\_number

**Specialized Subclasses**:

* **VendorManagementTeam**
  + team\_id
  + team\_name
* **ProcurementManager**
  + manager\_id
  + department
* **FinanceTeam**
  + finance\_id
  + budget\_assigned
* **ContractManagementTeam**
  + team\_id
  + assigned\_contracts

**Generalization/Specialization Relationship:**

* Generalization: User is the parent entity.
* Specialization: Each role (e.g., VendorManagementTeam, ProcurementManager) is a child entity.
* Disjoint Constraint: A User can belong to **one and only one role**.

1. **Vendor**
   * Attributes:
     + vendorID (PK)
     + name
     + address
     + contactPerson
     + contactNumber
     + email
     + complianceStatus (Boolean)
     + rating (Average rating based on performance evaluations)
2. **Contract**
   * Attributes:
     + contractID (PK)
     + vendorID (FK)
     + userID (FK)
     + startDate
     + endDate
     + renewalDate
     + contractStatus (e.g., Active, Expired, Pending Renewal)
     + approvalStatus (e.g., Approved, Rejected)
3. **Notifications**
   * Attributes:
     + notificationID (PK)
     + userID (FK)
     + contractID (FK)
     + notificationType (e.g., Contract Renewal, Budget Alert) – POSSIBLE SUPERCLASS but don’t wanna overcomplicate it.
     + date
     + status (e.g., Sent, Read, Pending)
4. **PurchaseOrder**
   * Attributes:
     + orderID (PK)
     + itemDetails
     + quantity
     + costPerUnit
     + TotalCost
     + vendorID (FK)
     + departmentID (FK)
     + budgetValidationStatus (Boolean)
     + status (e.g., Pending, Approved, Rejected)
     + orderDate
5. **Budget**
   * Attributes:
     + budgetID (PK)
     + departmentID (FK)
     + allocatedAmount
     + spentAmount
     + remainingAmount
     + adjustmentHistory
6. **PerformanceEvaluation**
   * Attributes:
     + evaluationID (PK)
     + vendorID (FK)
     + userID (FK)
     + feedback
     + rating
     + evaluationDate
7. **Department**
   * Attributes:
     + departmentID (PK)
     + name
     + managerID (FK)

**Relationships**

**User Relationships**

1. **User → Contract**
   * **Relationship**: One-to-Many
   * **Description**: Specialized users (e.g., ProcurementManager, ContractManagementTeam) can oversee multiple contracts.
   * **Key Fields**: userID (PK in User, FK in Contract)
2. **User → Notifications**
   * **Relationship**: One-to-Many
   * **Description**: Specialized users receive notifications related to contracts, budget alerts, or other tasks.
   * **Key Fields**: userID (PK in User, FK in Notifications)
3. **Department → User**
   * **Relationship**: One-to-Many
   * **Description**: Each department can have multiple users (e.g., staff, department head).
   * **Key Fields**: departmentID (PK in Department, FK in User specialization, e.g., ProcurementManager)
4. **User → PerformanceEvaluation**
   * **Relationship**: Many-to-Many
   * **Description**: Users (e.g., VendorManagementTeam) evaluate the performance of vendors.
   * **Key Fields**: userID (PK in User, FK in PerformanceEvaluation)

**Vendor Relationships**

1. **Vendor → Contract**
   * **Relationship**: One-to-Many
   * **Description**: A vendor can have multiple contracts.
   * **Key Fields**: vendorID (PK in Vendor, FK in Contract)
2. **Vendor → PerformanceEvaluation**
   * **Relationship**: One-to-Many
   * **Description**: A vendor can have multiple performance evaluations conducted by the VendorManagementTeam.
   * **Key Fields**: vendorID (PK in Vendor, FK in PerformanceEvaluation)
3. **Vendor → PurchaseOrder**
   * **Relationship**: One-to-Many
   * **Description**: A vendor can receive multiple purchase orders from departments.
   * **Key Fields**: vendorID (PK in Vendor, FK in PurchaseOrder)

**Contract Relationships**

1. **Contract → Vendor**
   * **Relationship**: Many-to-One
   * **Description**: Each contract is associated with a single vendor.
   * **Key Fields**: vendorID (FK in Contract, PK in Vendor)
2. **Contract → Notifications**
   * **Relationship**: One-to-Many
   * **Description**: Each contract can generate multiple notifications (e.g., renewal reminders).
   * **Key Fields**: contractID (PK in Contract, FK in Notifications)

**Budget and Department Relationships**

1. **Department → Budget**
   * **Relationship**: One-to-One
   * **Description**: Each department has a unique budget allocated.
   * **Key Fields**: departmentID (PK in Department, FK in Budget)
2. **Department → PurchaseOrder**
   * **Relationship**: One-to-Many
   * **Description**: Departments create multiple purchase orders.
   * **Key Fields**: departmentID (PK in Department, FK in PurchaseOrder)
3. **Budget → PurchaseOrder**
   * **Relationship**: One-to-Many
   * **Description**: A budget validates and tracks multiple purchase orders within allocated limits.
   * **Key Fields**: budgetID (PK in Budget, FK in PurchaseOrder)

**Constraints**

1. **User Constraints:**
   * Each user must have a unique email and userID.
   * Role-based access ensures specific users manage certain tasks (e.g., contract approvals).
2. **Vendor Constraints:**
   * A vendor cannot be assigned new contracts unless their complianceStatus is true.
3. **Contract Constraints:**
   * A contract cannot be created without a valid vendorID and userID.
   * Contracts nearing their renewalDate should generate a notification.
4. **Notifications Constraints:**
   * Notifications must reference a valid contractID and userID.
   * Notifications should have a status of either Sent, Read, or Pending.
5. **Purchase Order Constraints:**
   * A purchase order must reference a valid vendorID and departmentID.
   * The total cost of purchase orders should not exceed the remainingAmount in the budget.
6. **Budget Constraints:**
   * The sum of all spentAmount in purchase orders cannot exceed the allocatedAmount.
7. **Performance Evaluation Constraints:**
   * Performance evaluations must reference a valid vendorID and userID.
   * Ratings must fall within a predefined range (e.g., 1–5).

**5. Assumptions**

1. Each User can belong to one role only, defined by user\_type.
2. Vendor compliance status is tracked as a Boolean (TRUE for compliant, FALSE otherwise).
3. Budgets are pre-assigned to departments and are adjusted dynamically based on expenditure.
4. Contracts have an automated renewal date update mechanism.
5. Notifications are sent to vendors only when triggered by compliance or performance issues.
6. Performance evaluations are managed exclusively by the Vendor Management Team.

**6. Functional Use Cases**

**Vendor Registration and Login**

* Vendors can register and log in to access a dashboard that displays compliance status, performance ratings, and notifications.

**Contract Management**

* Manage contract creation, amendments, and track renewal status using the Contract table.

**Purchase Order Management**

* Creation and validation of purchase orders with budget checks using triggers and stored procedures.

**Vendor Performance Tracking**

* Track and evaluate vendors using the PerformanceEvaluation table, providing feedback on various metrics.

**Compliance and Auditing**

* The Compliance module uses the Notifications table to notify vendors of compliance breaches.

**Task Allocation**

* Tasks for procurement processes are assigned to relevant departments, ensuring streamlined workflows.

**Budget Monitoring**

* Use the Budget table to track allocated, spent, and remaining funds, ensuring no overspending.

**7. Additional Features**

1. **Dashboards**:
   * Separate dashboards for vendors, procurement managers, finance teams, and contract management teams.
2. **Notification System**:
   * Triggers ensure timely updates and alerts.
3. **Dynamic Budget Validation**:
   * Prevent overspending through automated checks in the PurchaseOrder table.

**Server Details**

* The backend is implemented using **Node.js** with the **Express framework**.
* MySQL database integration is achieved using the **mysql2 package**.
* EJS files are used for rendering user interfaces (e.g., login, dashboards).

**Implemented Features**

**1. User Authentication**

* **Login**: Validates user credentials against the User table and redirects to role-specific dashboards.
* **Sign-Up**: Handles new vendor registration by inserting records into both the User and Vendor tables using transactions.

**2. Role-Specific Dashboards**

* **Vendor Dashboard**:
  + Displays vendor details and pending notifications.
  + Automatically marks notifications as "Sent" once viewed.
* **Vendor Management Team Dashboard**:
  + Provides options to manage vendors, view vendor performance, and audit compliance.
* **Procurement Manager Dashboard**:
  + Enables creating and tracking purchase orders and validating department budgets.
* **Finance Team Dashboard**:
  + Allows monitoring budgets, generating financial reports, and adjusting department budgets.
* **Contract Management Team Dashboard**:
  + Features for managing, creating, assigning, and tracking contracts.

**3. Vendor Management**

* Manage vendors through a CRUD interface:
  + Create, update, and delete vendor details.
* Approve or reject vendor applications.

**4. Procurement Management**

* Create purchase orders with automatic budget validation.
* Track and view purchase order details by department or procurement manager.

**5. Contract Management**

* View, create, update, and delete contracts.
* Assign contracts to specific vendors and teams.
* Manage contract renewal and status changes.

**6. Compliance and Auditing**

* Categorize vendors as "Good" or "Bad" based on performance and compliance.
* Send notifications to non-compliant vendors via a notification system.

**7. Budget Monitoring and Adjustments**

* Monitor department budgets, including allocated, spent, and remaining amounts.
* Adjust budgets with an option to log adjustment history.

**8. Performance Tracking**

* View and evaluate vendor performance with feedback and rating mechanisms.

**9. Notifications**

* Notifications are dynamically generated for various events like contract updates and vendor compliance issues.

**Assumptions & Observations**

1. **User Types**: Each User is tied to a single role (user\_type), ensuring clarity in permissions.
2. **Vendor signup** : Emails to admins are already assigned to them, while vendors can sign up and register.
3. **Notification Workflow**: Notifications are generated in response to specific triggers (e.g., contract expiry, vendor compliance).
4. **Budget Constraints**: Purchase orders are validated against the remaining budget using triggers or application logic.
5. **Contract Assignments**: Contracts can be linked to specific vendors and management teams for better traceability.
6. **Error Handling**: Comprehensive logging for database operations, especially for errors during transactions.