Contents

[Introduction: 3](#_Toc183920701)

[ERD: 4](#_Toc183920702)

[Entities and Their Purpose 4](#_Toc183920703)

[Relational Schema 6](#_Toc183920704)

[Mapped Tables and Their Purpose 6](#_Toc183920705)

[Table showing erd 7](#_Toc183920706)

[Triggers 8](#_Toc183920707)

[Few example screenshots of implemented triggers 10](#_Toc183920708)

[Table description using DESC command 11](#_Toc183920709)

[Stored Procedures 18](#_Toc183920710)

[Backend Connection (Database - db.js) 19](#_Toc183920711)

[HTML pages integrated with front and backend 19](#_Toc183920712)

[1. Login and Signup Page 19](#_Toc183920713)

[Backend Logic (JavaScript - main.js) 21](#_Toc183920714)

[Login Functionality (login() in main.js): 21](#_Toc183920715)

[Signup Functionality (signup() in main.js): 21](#_Toc183920716)

[2. Vendor Management Page Overview 22](#_Toc183920717)

[Front-End (HTML - vendor-management.html) 22](#_Toc183920718)

[Vendor List: 22](#_Toc183920719)

[Backend Logic (server.js and main.js) 23](#_Toc183920720)

[Purpose of This Section 24](#_Toc183920721)

[3. Purchase Order Management Page Overview 24](#_Toc183920722)

[Front-End (HTML - purchase-order.html) 24](#_Toc183920723)

[Purchase Order List: 25](#_Toc183920724)

[Backend Logic (server.js and main.js) 26](#_Toc183920725)

[Purpose of This Section 26](#_Toc183920726)

[4. Reports Page Overview 26](#_Toc183920727)

[Front-End (HTML - reports.html) 26](#_Toc183920728)

[Procurement Report: 27](#_Toc183920729)

[Backend Logic (server.js and main.js) 28](#_Toc183920730)

[Purpose of This Section 28](#_Toc183920731)

[5. Audit Logs Page Overview 28](#_Toc183920732)

[Front-End (HTML - audit-logs.html) 28](#_Toc183920733)

[Backend Logic (server.js) 29](#_Toc183920734)

[6. Budget Tracking Page Overview 29](#_Toc183920735)

[Front-End (HTML - budget-tracking.html) 29](#_Toc183920736)

[Backend Logic (server.js and main.js) 30](#_Toc183920737)

[7. Contract Management Page Overview 31](#_Toc183920738)

[Front-End (HTML - contract-management.html) 31](#_Toc183920739)

[Backend Logic (server.js and main.js) 32](#_Toc183920740)

[8. Dashboard Page Overview 32](#_Toc183920741)

[Front-End (HTML - dashboard.html) 32](#_Toc183920742)

[Backend Logic (server.js) 33](#_Toc183920743)

[9. System Settings Page Overview 34](#_Toc183920744)

[Front-End (HTML - system-settings.html) 34](#_Toc183920745)

[Backend Logic (server.js) 34](#_Toc183920746)

[10. Task Management Page Overview 34](#_Toc183920747)

[Front-End (HTML - task-management.html) 34](#_Toc183920748)

[Backend Logic (server.js) 35](#_Toc183920749)

[11. User Management Page Overview 36](#_Toc183920750)

[Front-End (HTML - user-management.html) 36](#_Toc183920751)

[Backend Logic (server.js) 37](#_Toc183920752)

[Detailed Role Functions 37](#_Toc183920753)

[Table showing Roles 39](#_Toc183920754)

[Charts using D3 showing meaningful insights on Dashboard: 40](#_Toc183920755)

[Average Performance Ratings Graph Type: 40](#_Toc183920756)

[Percentage of Budget Utilized Graph Type: 41](#_Toc183920757)

[Purchase Orders by Status Graph Type: 42](#_Toc183920758)

# Introduction:

The **Vendor Management System** is a robust, role-based web application designed to streamline procurement, vendor relations, contract administration, and task management processes within an organization. By implementing secure user authentication and role-specific access controls, the system ensures that users interact only with the data and functionalities relevant to their responsibilities. Key user roles include Admins, Procurement Managers, Department Heads, and General Users, each granted permissions tailored to their specific needs. The system enables efficient vendor management through functionalities like performance evaluation, compliance tracking, and centralized data storage. Contract administration is simplified with tools for creation, updates, renewals, and proactive notifications for contracts nearing expiry, ensuring business continuity.

The platform also supports comprehensive purchase order workflows, including creation, approval, rejection, and real-time tracking, while integrated budget validation enforces compliance with financial policies. Task allocation features allow team leaders to assign and monitor tasks, ensuring accountability and timely completion. Advanced reporting and analytics provide actionable insights into vendor performance, procurement activities, and departmental budgets, with export options for easy data sharing. The inclusion of audit logging enhances accountability and security by tracking all critical user actions. A centralized dashboard delivers personalized notifications, performance summaries, and critical updates, making it easier for users to stay informed and act promptly.

Developed using modern technologies like Node.js, Express.js, and MySQL, and incorporating responsive design principles with Bootstrap 5.3.0, the system bridges the frontend and backend seamlessly through RESTful APIs. Security measures such as role-based authentication using JWT and encrypted user credentials ensure data integrity and protection. The Corporate Vendor and Contract Management System is a scalable solution that enhances organizational efficiency by automating routine processes, enforcing compliance, and fostering collaboration across departments.

# ERD:

A diagram of a flowchart

Description automatically generated

## Entities and Their Purpose

1. **Vendor**
   * **Attributes**: Vendor\_ID, Name, Contact\_Info, Service\_Category, Certifications, Compliance\_Status, etc.
   * **Purpose**: Central to managing vendor details such as certifications, compliance checks, and service offerings.
2. **Contract**
   * **Attributes**: Contract\_ID, Vendor\_ID (FK), StartDate, EndDate, Terms, Renewal\_Status.
   * **Purpose**: Tracks the lifecycle of vendor contracts, including initiation, terms, renewal, and expiry notifications.
3. **Purchase\_Order**
   * **Attributes**: PO\_ID, Dept\_ID (FK), Vendor\_ID (FK), Quantity, TotalCost, ItemDetails, Status.
   * **Purpose**: Handles purchase orders, ensuring alignment with vendor agreements and departmental budgets.
4. **Budget**
   * **Attributes**: Budget\_ID, Allocated\_Budget, Remaining\_Budget, Expenses.
   * **Purpose**: Monitors departmental budgets for procurement processes and ensures financial responsibility.
5. **Department**
   * **Attributes**: Dept\_ID, Name, Allocated\_Budget, Remaining\_Budget.
   * **Purpose**: Organizes teams or departments managing vendors, contracts, and budgets.
6. **User**
   * **Attributes**: User\_ID, Name, Email, Role, Password, Dept\_ID (FK).
   * **Purpose**: Manages authentication and role-based access within the system.
7. **Task**
   * **Attributes**: Task\_ID, Assigned\_To (User\_ID), Status, UpdatedAt, Due\_Date.
   * **Purpose**: Tracks assignments related to procurement, including approvals and vendor evaluations.
8. **VendorPerformance**
   * **Attributes**: Evaluation\_ID, Vendor\_ID (FK), QualityRating, TimelinessRating, Feedback.
   * **Purpose**: Evaluates vendor performance based on pre-defined metrics, crucial for compliance and decision-making.
9. **Audit\_Log**
   * **Attributes**: Log\_ID, User\_ID (FK), Action, ActionTime, Description.
   * **Purpose**: Tracks user actions for security, compliance, and troubleshooting.
10. **Permissions**
    * **Attributes**: Permission\_ID, Name, Description.
    * **Purpose**: Lists actions authorized for different roles.
11. **Role\_Permission**
    * **Attributes**: Role\_ID, Permission\_ID (FK).
    * **Purpose**: Links roles with specific permissions to enforce role-based access control.
12. **Notifications**
    * **Attributes**: NotificationID, ContractID (FK), NotificationDate, NotificationStatus.
    * **Purpose**: Sends alerts about expiring contracts or budget issues, ensuring proactive management.
13. **SystemSettings**
    * **Attributes**: ID, SettingName, SettingValue, Description.
    * **Purpose**: Stores configurations for system-wide settings, such as notification

intervals.

# Relational Schema

# Mapped Tables and Their Purpose

1. **Vendor**
   * Purpose: To store and manage vendor-related data for easy retrieval and performance evaluations.
2. **Contract**
   * Purpose: Tracks the terms and renewal of contracts, associating vendors and departments.
3. **Purchase\_Order**
   * Purpose: Handles procurement, ensuring departmental budgets and vendor alignments.
4. **Budget**
   * Purpose: Monitors and manages budgetary allocations across departments.
5. **Department**
   * Purpose: Represents functional units responsible for procurement and vendor management.
6. **User**
   * Purpose: Provides role-based access control, linking actions to departments.
7. **Task**
   * Purpose: Manages task allocation and tracking for procurement processes.
8. **VendorPerformance**
   * Purpose: Records and rates vendor performance for compliance and procurement decisions.
9. **Audit\_Log**
   * Purpose: Logs user activities for accountability and troubleshooting.
10. **Permissions**
    * Purpose: Defines actions available in the system.
11. **Role\_Permission**
    * Purpose: Associates permissions with roles, enforcing role-based system controls.
12. **Notifications**
    * Purpose: Manages alerts for contracts and budgets.
13. **SystemSettings** (No Relationships)
    * Purpose: Stores independent settings for system-wide configurations, such as notifications or UI preferences.
    * **Reason for Extension**: Added for flexibility to change system configurations dynamically without altering the schema.

## Table showing erd

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Entity 1** |  |  | | --- | |  | | **Relationship** | **Entity 2** | **Type of relationship** | **Purpose** |
| |  | | --- | | **Vendor** | | |  | | --- | | Initiates | | |  | | --- | | **Contract** | | |  | | --- | | 1:M | | A vendor can have multiple contracts associated with them. |
| **Contract** | Belongs to | **Vendor** | M:1 | Each contract is linked to a single vendor. |
| **Purchase\_Order** | Belongs to | **Vendor** | M:1 | Each purchase order is linked to a vendor supplying the items or services. |
| **Purchase\_Order** | Created by | **Department** | M:1 | Purchase orders are created by specific departments for their procurement needs. |
| **Budget** | Allocated to | **Department** | 1:M | A department has its own allocated budget managed for purchases and contracts. |
| **Department** | Has | **User** | 1:M | Each department consists of multiple users managing various operations. |
| **Task** | Assigned to | **User** | M:1 | Tasks related to procurement and evaluation are assigned to specific users. |
| **VendorPerformance** | Evaluates | **Vendor** | M:1 | Vendor performance is evaluated and stored for individual vendors. |
| **Audit\_Log** | Logged by | **User** | M:1 | User actions are logged in the audit table for accountability and compliance. |
| **Role\_Permission** | Defines | **Permissions** | M:1 | Each role is associated with multiple permissions, defining their actions in the system. |
| **Role\_Permission** | Assigned to | **Role** | M:1 | Permissions are assigned to roles to enforce role-based access controls. |
| **Notifications** | Belongs to | **Contract** | M:1 | Notifications are linked to contracts to track expiration and renewal alerts. |
| **User** | Belongs to | **Department** | M:1 | Users are part of a specific department responsible for certain tasks. |

**SystemSettings**: This table does not have any direct relationships with other entities, as it stores independent configurations for the system. It was added to enhance system flexibility by allowing changes in configuration (like notification intervals) without altering the schema.

# Triggers

1. **Contract Renewal Notification Trigger**
   * **Purpose**: Checks for contracts expiring within 30 days and automatically sends renewal notifications to relevant stakeholders.
   * **Why Added**: Ensures users take timely action to renew contracts, avoiding service interruptions and improving operational efficiency.
2. **Purchase Order Budget Validation Trigger**
   * **Purpose**: Validates whether a purchase order exceeds the department's allocated budget. If exceeded, it generates alerts.
   * **Why Added**: Maintains financial compliance by ensuring departments stay within budget limits, preventing overspending.
3. **Task Overdue Notification Trigger**
   * **Purpose**: Identifies overdue tasks (those with a past due date and incomplete status) and generates notifications for users.
   * **Why Added**: Promotes timely completion of tasks, ensuring workflows progress without unnecessary delays.
4. **Audit Logging Trigger**
   * **Purpose**: Logs significant actions such as creating, updating, or deleting records into the AuditLogs table.
   * **Why Added**: Provides accountability and an audit trail for troubleshooting and tracking critical activities.
5. **Vendor Performance Update Trigger**
   * **Purpose**: Updates the overall performance rating of a vendor whenever a new evaluation is added or an existing evaluation is modified.
   * **Why Added**: Keeps vendor performance metrics accurate and up-to-date for effective decision-making.
6. **Pending Purchase Order Notification Trigger**
   * **Purpose**: Generates notifications for purchase orders in the "Pending" status, highlighting those that require approval.
   * **Why Added**: Ensures timely processing and approval of purchase orders to prevent procurement delays.
7. **System Settings Change Logging Trigger**
   * **Purpose**: Logs updates to system configuration settings into an audit log, capturing the setting ID and new values.
   * **Why Added**: Tracks changes to critical system settings for debugging and maintaining integrity.
8. **Contract Update Logging Trigger**
   * **Purpose**: Records updates to contracts, such as terms, renewal status, or special clauses, into an audit log.
   * **Why Added**: Maintains a history of contract changes for transparency and accountability.
9. **Task Status Change Trigger**
   * **Purpose**: Logs status changes for tasks (e.g., "Pending" to "Completed"), capturing task ID and the new status.
   * **Why Added**: Tracks task progress and provides an audit trail for completed and updated tasks.
10. **Department Budget Update Trigger**
    * **Purpose**: Automatically recalculates a department's remaining budget whenever a new purchase order is created or updated.
    * **Why Added**: Ensures budget tracking is accurate and reflects real-time data.
11. **Vendor Deletion Validation Trigger**
    * **Purpose**: Prevents deletion of vendors if they are associated with active contracts or pending purchase orders.
    * **Why Added**: Maintains data integrity by avoiding orphaned records in related tables.
12. **Purchase Order Status Change Trigger**
    * **Purpose**: Logs changes to purchase order statuses (e.g., "Approved," "Rejected"), capturing details for audit purposes.
    * **Why Added**: Ensures transparency in procurement decisions and provides an audit trail.
13. **Notifications Consolidation Trigger**
    * **Purpose**: Combines all task, contract, and purchase order notifications into a unified feed.
    * **Why Added**: Simplifies the notification process, providing users with a centralized update mechanism.

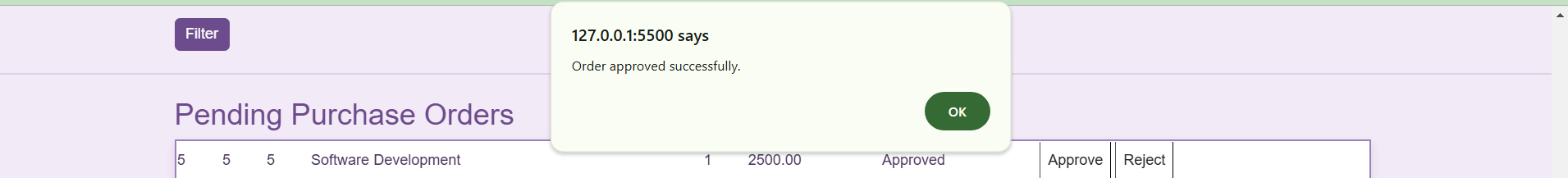
### Few example screenshots of implemented triggers

A white background with black text

Description automatically generated

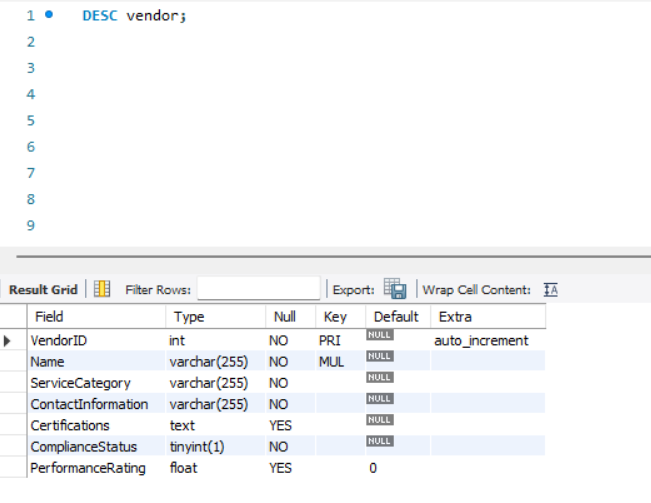
A screenshot of a computer

Description automatically generated

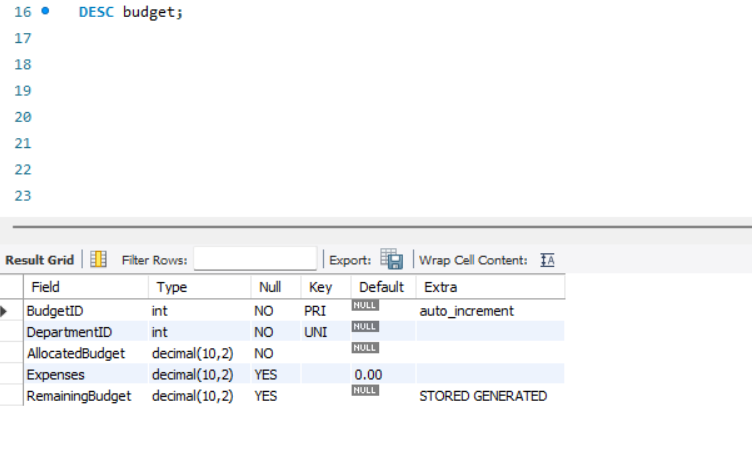


# Table description using DESC command

1. **Vendor:**



1. **Budget:**

****

1. **Contract:**

A screenshot of a computer

Description automatically generated

1. **Auditlogs**

A screenshot of a computer

Description automatically generated

1. **Department**

A screenshot of a computer

Description automatically generated

1. **Notifications**

A screenshot of a computer

Description automatically generated

1. **Permissions**

A screenshot of a computer

Description automatically generated

1. **A screenshot of a computer

   Description automatically generatedRole permissions**
2. A screenshot of a computer

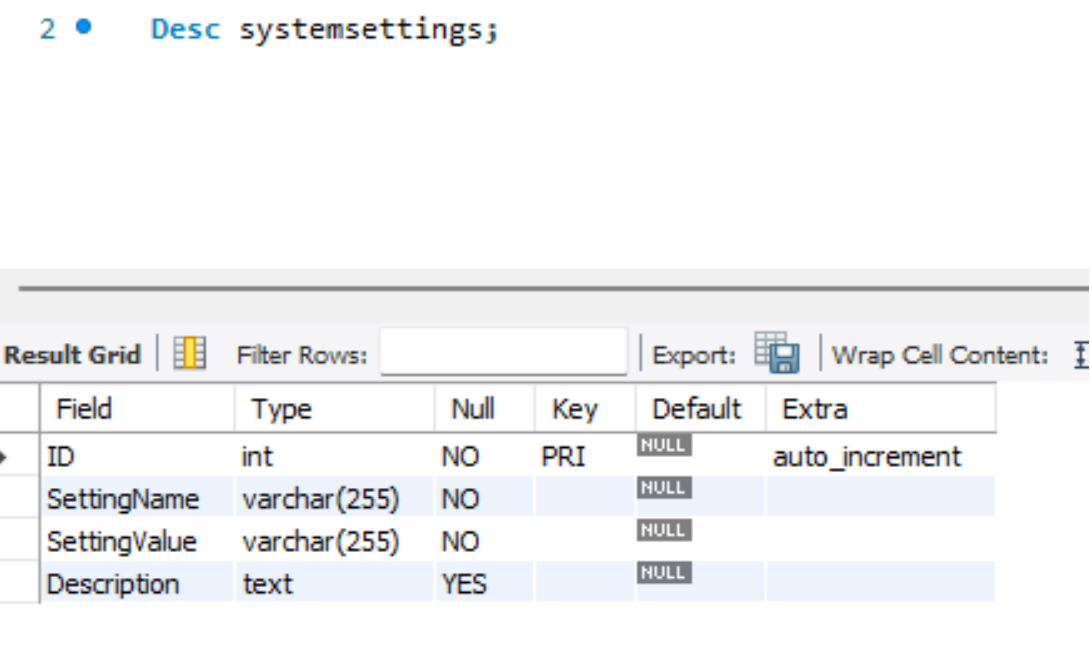
   Description automatically generated**User**
3. A screenshot of a computer

   Description automatically generated**Task**
4. A screenshot of a computer

   Description automatically generated**Vendor Performance**
5. **Purchase Order**

**A screenshot of a computer

Description automatically generated**

1. **System settings**
2. ****

# Stored Procedures

1. **Vendor Registration Procedure**
   * **Purpose**: Simplifies adding new vendors by validating and inserting their data into the database.
   * **Why Added**: Automates vendor creation, reducing manual errors and ensuring consistency.
2. **Contract Renewal Procedure**
   * **Purpose**: Updates contract renewal details, such as renewal status and new terms, in bulk.
   * **Why Added**: Streamlines the renewal process for multiple contracts simultaneously, saving time and effort.
3. **Performance Evaluation Procedure**
   * **Purpose**: Automates the process of assigning performance metrics to vendors based on evaluations.
   * **Why Added**: Ensures vendor performance metrics are updated systematically, maintaining consistency.
4. **Budget Allocation Procedure**
   * **Purpose**: Distributes budgets to departments and calculates remaining balances after expenses.
   * **Why Added**: Automates budget tracking and allocation, ensuring accurate financial reporting.
5. **Purchase Order Approval Procedure**
   * **Purpose**: Updates the status of a purchase order to "Approved" and logs the approval action.
   * **Why Added**: Speeds up the approval process and ensures actions are recorded for accountability.
6. **Task Assignment Procedure**
   * **Purpose**: Assigns tasks to users and sets deadlines while logging the creation details.
   * **Why Added**: Simplifies task allocation and ensures that tasks are systematically assigned.
7. **Audit Log Retrieval Procedure**
   * **Purpose**: Retrieves filtered audit logs based on specific actions, users, or timeframes.
   * **Why Added**: Facilitates quick access to logged actions for troubleshooting and analysis.

**Purpose of Including Triggers and Stored Procedures**

* **Automation**: Reduces manual intervention in repetitive tasks like budget tracking, notifications, and performance updates.
* **Data Integrity**: Ensures consistency across the system, avoiding orphaned or invalid records.
* **Accountability**: Logs critical actions and updates for auditing, improving transparency and traceability.
* **Efficiency**: Streamlines workflows, enabling faster processing of tasks like vendor registration, contract renewals, and purchase order approvals.
* **User Experience**: Simplifies the interaction by providing timely notifications, consistent updates, and automated validations.

# Backend Connection (Database - db.js)

* **Database Connection**:
  + Utilizes mysql2 for connecting to the MySQL database (vendorManagementdb).
  + Provides a shared connection for querying the database during login or signup processes.

# HTML pages integrated with front and backend

## Login and Signup Page

**Front-End (HTML - login.html)**

The login and signup page provides user authentication and role-based user creation functionalities. Here's an overview:

**Design and Layout:**

* + **Framework**: Uses **Bootstrap 5.3.0** for styling and responsive design.
  + **Visual Elements**:
    - Background color: Light purple (#f3eaf7) with dark accents (#4a355a).
    - Border and box-shadow for forms to enhance visibility and style.
  + Two sections: Login and Signup with separate forms.

**Login Form:**

* + **Fields**:
    - Email: Text input (type="email") for user email.
    - Password: Password input (type="password") for authentication.
  + **Submit Button**:
    - Triggers the login() function defined in main.js when the form is submitted.
  + **Styling**:
    - Form is styled with rounded corners, shadows, and a purple theme (#6d4c8d).

A screenshot of a login screen

Description automatically generated

**Signup Form:**

* + **Fields**:
    - Name: Text input for user's full name.
    - Email: Text input for email address.
    - Password: Password input for creating a secure password.
    - Role: Dropdown menu to select the role (Admin, Procurement Manager, Department Head).
  + **Submit Button**:
    - Triggers the signup() function in main.js when the form is submitted.

**Scripts:**

* + Includes the JavaScript file main.js for handling authentication (login() and signup() functions).

A screenshot of a login form

Description automatically generated

### Backend Logic (JavaScript - main.js)

### Login Functionality (login() in main.js):

* + **Purpose**: Authenticates users based on their email and password and redirects them to their role-specific dashboard.
  + **Process**:
    - Captures form data (email and password) from the login form.
    - Sends a POST request to the API endpoint /api/login.
    - If successful:
      * Stores user data (e.g., role, userID) in sessionStorage.
      * Redirects to a role-specific dashboard (Admin, Procurement Manager, or Department Head).
      * Checks for notifications if the user is an Admin and alerts them of new updates.
    - Handles errors with appropriate messages.

### Signup Functionality (signup() in main.js):

* + **Purpose**: Creates new users with roles and credentials.
  + **Process**:
    - Captures form data (name, email, password, and role) from the signup form.
    - Sends a POST request to /api/signup.
    - Alerts the user on successful signup and resets the form.

**Purpose of This Section**

* **Login Page**: Ensures only authenticated users access the system and redirects them to the correct section based on their role.
* **Signup Page**: Enables administrators to add new users with role-based access control.
* **Backend Integration**: Connects the frontend actions to the database via API endpoints, validating and storing user credentials securely.

## 2. Vendor Management Page Overview

### Front-End (HTML - vendor-management.html)

The Vendor Management page enables Admin and Procurement Managers to manage vendor details. Here's an overview:

**Design and Layout:**

* + **Framework**: Bootstrap 5.3.0 for responsive design.
  + **Visual Elements**:
    - Background: Light purple (#f3eaf7) with purple accents.
    - Borders and box-shadow for forms and tables.
  + **Sections**: Vendor registration/editing form and vendor list.

**Vendor Registration/Editing Form:**

* + **Fields**:
    - Name: Vendor name (text input).
    - Service Category: Vendor's service category (text input).
    - Contact Information: Vendor email (email input).
    - Certifications: Vendor certifications (textarea).
    - Compliance Status: Checkbox to indicate compliance.
    - Performance Rating: Dropdown menu (1.0 to 5.0).
  + **Submit Button**:
    - Triggers registerOrUpdateVendor() when the form is submitted.

### Vendor List:

* + **Features**:
    - Displays all vendors in a dynamic table.
    - Includes options for editing and deleting vendors.

1. **Scripts**:
   * Includes main.js for handling dynamic operations like fetching and updating vendor data.

A screenshot of a vendor management

Description automatically generated

A screenshot of a computer

Description automatically generated

### Backend Logic (server.js and main.js)

1. **Fetch Vendors (fetchVendors in main.js)**:
   * **Purpose**: Retrieves vendor data and populates the table dynamically.
   * **Process**: Sends a GET request to /api/vendor.
2. **Add/Update Vendor (registerOrUpdateVendor in main.js)**:
   * **Purpose**: Adds or updates vendor information.
   * **Process**:
     + Sends a POST or PUT request to the backend depending on the context (add or update).
3. **Edit Vendor (editVendor in main.js)**:
   * **Purpose**: Pre-fills the form with existing vendor data for editing.
   * **Process**: Fetches data using /api/vendor/:id.
4. **Delete Vendor (deleteVendor in main.js)**:
   * **Purpose**: Deletes a vendor and refreshes the list.
   * **Process**: Sends a DELETE request to /api/vendor/:id.

### Purpose of This Section

* **Vendor Overview**: Provides a centralized interface to manage vendor details.
* **Efficient Management**: Enables dynamic vendor addition, editing, and deletion.
* **Data Integrity**: Ensures backend updates are reflected immediately on the frontend.

## 3. Purchase Order Management Page Overview

### Front-End (HTML - purchase-order.html)

The Purchase Order page allows Procurement Managers and Department Heads to manage purchase orders. Here's an overview:

**Design and Layout:**

* + **Framework**: Bootstrap 5.3.0.
  + **Visual Elements**:
    - Styled forms and tables for creating and listing purchase orders.
  + **Sections**: Purchase order form and list.

**Purchase Order Form:**

* + **Fields**:
    - Department ID
    - Vendor ID
    - Item Details
    - Quantity
    - Total Cost
  + **Submit Button**:
    - Triggers createPurchaseOrder() to submit data.

### Purchase Order List:

* + Displays all purchase orders in a table.
  + Includes options to approve, reject, or delete orders.

A screenshot of a purchase order management

Description automatically generated

A screenshot of a computer

Description automatically generated

### Backend Logic (server.js and main.js)

1. **Fetch Purchase Orders (fetchPurchaseOrders in main.js)**:
   * Retrieves purchase orders using /api/purchase-orders.
2. **Create Purchase Order (createPurchaseOrder in main.js)**:
   * Submits a new purchase order using /api/purchase-orders.
3. **Approve/Reject Purchase Orders**:
   * Updates the status using /api/purchase-orders/:id/approve or /api/purchase-orders/:id/reject.

### Purpose of This Section

* **Procurement Tracking**: Provides tools to manage purchase orders efficiently.
* **Approval Workflow**: Ensures orders are processed with proper authorization.

## 4. Reports Page Overview

### Front-End (HTML - reports.html)

The Reports page provides users with insights into vendor performance and procurement metrics. Here's an overview:

**Design and Layout:**

* + **Framework**: Bootstrap 5.3.0.
  + **Visual Elements**:
    - Tables styled with alternating row colors for readability.
    - Export buttons for downloading reports.
  + **Sections**:
    - Vendor performance report.
    - Procurement report.

**Vendor Performance Report:**

* + **Features**:
    - Displays average quality and timeliness ratings for each vendor.
    - Shows the total number of evaluations per vendor.
  + **Download Button**:
    - Triggers downloadVendorPerformanceReport() to export as a CSV.

### Procurement Report:

* + **Features**:
    - Displays purchase order details, including department name, vendor name, item details, and costs.
  + **Download Button**:
    - Triggers downloadProcurementReport().

**Scripts**:

* + Includes main.js for fetching and displaying reports dynamically.

A screenshot of a report

Description automatically generated

A screenshot of a report

Description automatically generated

### Backend Logic (server.js and main.js)

1. **Fetch Vendor Performance (fetchVendorPerformanceReport in main.js)**:
   * **Purpose**: Retrieves vendor performance metrics using /api/reports/vendor-performance.
2. **Fetch Procurement Report (fetchProcurementReport in main.js)**:
   * **Purpose**: Fetches procurement details from /api/reports/procurement.
3. **Export Reports**:
   * Downloads data in CSV format via endpoints like /api/report/department-budgets.

### Purpose of This Section

* **Insights**: Provides detailed data for decision-making on vendors and procurement.
* **Accountability**: Tracks performance metrics and procurement details.

## 5. Audit Logs Page Overview

### Front-End (HTML - audit-logs.html)

The Audit Logs page is a tool for tracking user actions. Here's an overview:

**Design and Layout:**

* + **Framework**: Bootstrap 5.3.0.
  + **Visual Elements**:
    - Tabular format with columns for Log ID, Action, Timestamp, and Description.
  + **Features**:
    - Dynamic table that populates with audit logs.
    - Filter options for specific users or actions.

**Scripts**:

* + Includes main.js to fetch and display logs dynamically.

A screenshot of a computer

Description automatically generated

### Backend Logic (server.js)

1. **Fetch Logs (GET /api/audit-logs)**:
   * **Purpose**: Retrieves all user actions logged in the database.
2. **Logging Actions**:
   * Integrated into all major backend functions (e.g., vendor creation, purchase order updates) via logAction.

**Purpose of This Section**

* **Accountability**: Ensures actions are traceable.
* **Security**: Tracks unauthorized access attempts or unexpected actions.

## 6. Budget Tracking Page Overview

### Front-End (HTML - budget-tracking.html)

The Budget Tracking page allows users to monitor departmental budgets. Here's an overview:

**Design and Layout:**

* + **Framework**: Bootstrap 5.3.0.
  + **Visual Elements**:
    - Tables to display budget allocation, expenses, and remaining amounts.
  + **Sections**:
    - Department budget table.
    - Filter for viewing specific departments.

1. **Scripts**:
   * Includes main.js for filtering and exporting budget data.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### Backend Logic (server.js and main.js)

1. **Fetch Budgets (GET /api/department)**:
   * **Purpose**: Retrieves departmental budget data.
2. **Export Budgets (GET /api/report/department-budgets)**:
   * **Purpose**: Exports budgets as a CSV file.

**Purpose of This Section**

* **Financial Tracking**: Provides real-time insights into budget usage.
* **Compliance**: Ensures departments operate within allocated budgets.

## 7. Contract Management Page Overview

### Front-End (HTML - contract-management.html)

The Contract Management page helps manage vendor contracts. Here's an overview:

1. **Design and Layout**:
   * **Framework**: Bootstrap 5.3.0.
   * **Sections**:
     + Contract creation/editing form.
     + Contract list with columns for ID, Vendor, Start/End Dates, and Terms.
2. **Scripts**:
   * Includes main.js for managing contract data dynamically.

A close-up of a contract

Description automatically generated

A screenshot of a computer

Description automatically generated

### Backend Logic (server.js and main.js)

1. **Fetch Contracts (GET /api/contract)**:
   * **Purpose**: Retrieves all contracts.
2. **Add/Update Contract (POST /api/contract or PUT /api/contract/:id)**:
   * **Purpose**: Adds new contracts or updates existing ones.

**Purpose of This Section**

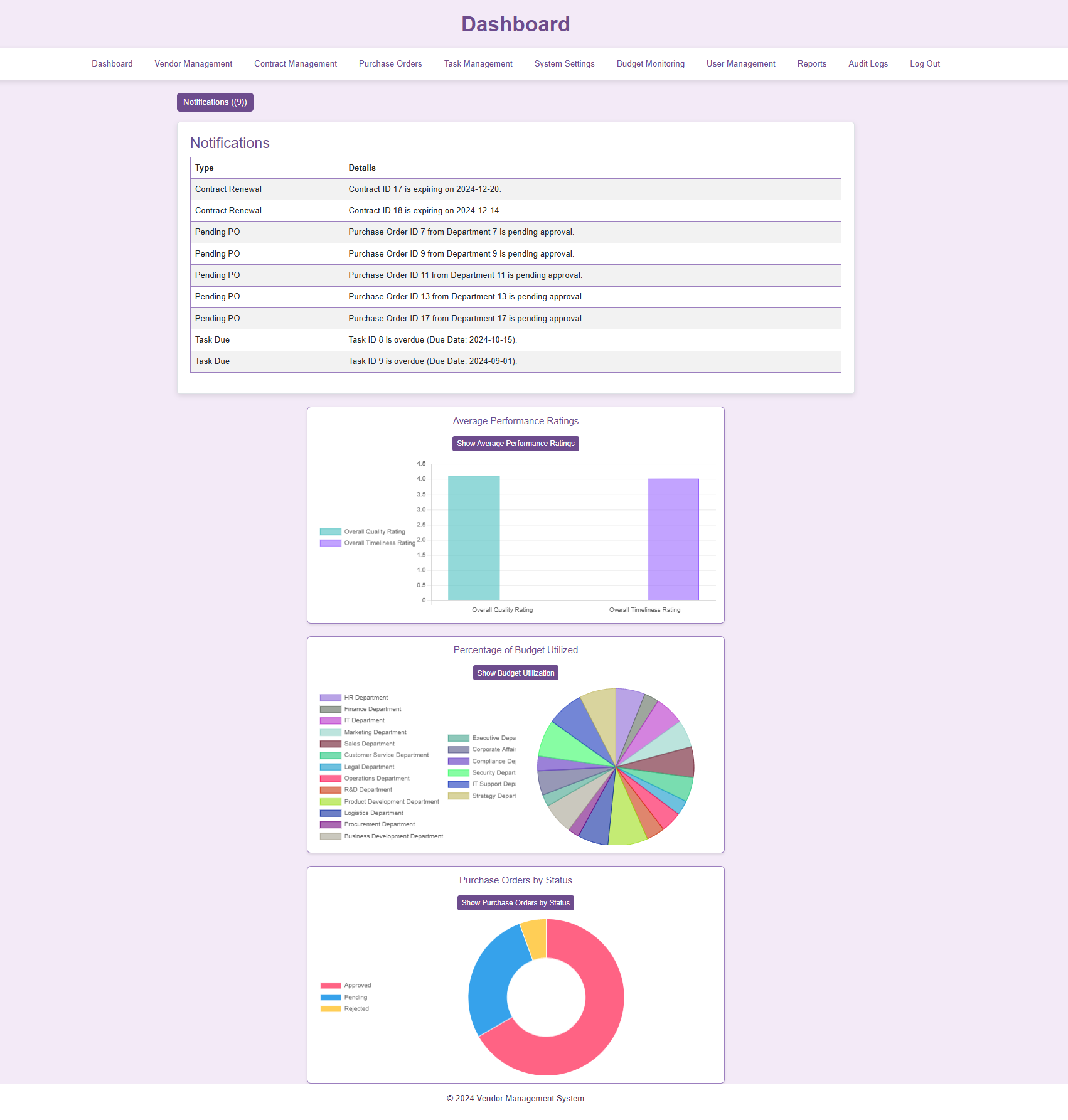
* **Contract Management**: Ensures timely updates and renewals.
* **Vendor Relations**: Tracks contractual obligations.

## 8. Dashboard Page Overview

### Front-End (HTML - dashboard.html)

The Dashboard page provides an overview of system notifications and performance. Here's an overview:

1. **Design and Layout**:
   * **Framework**: Bootstrap 5.3.0.
   * **Features**:
     + Notifications section for contracts, tasks, and purchase orders.
     + Vendor performance visualizations using charts.
     + Includes percentage budget utilization chart.
     + Also includes insights on purchase orders statues
2. **Scripts**:
   * Includes main.js for fetching notifications and performance data.



### Backend Logic (server.js)

1. **Fetch Notifications (GET /api/notifications)**:
   * **Purpose**: Retrieves notifications for tasks, contracts, and purchase orders.
2. **Vendor Performance (GET /reports/vendor-overall-performance)**:
   * **Purpose**: Fetches data for charts.

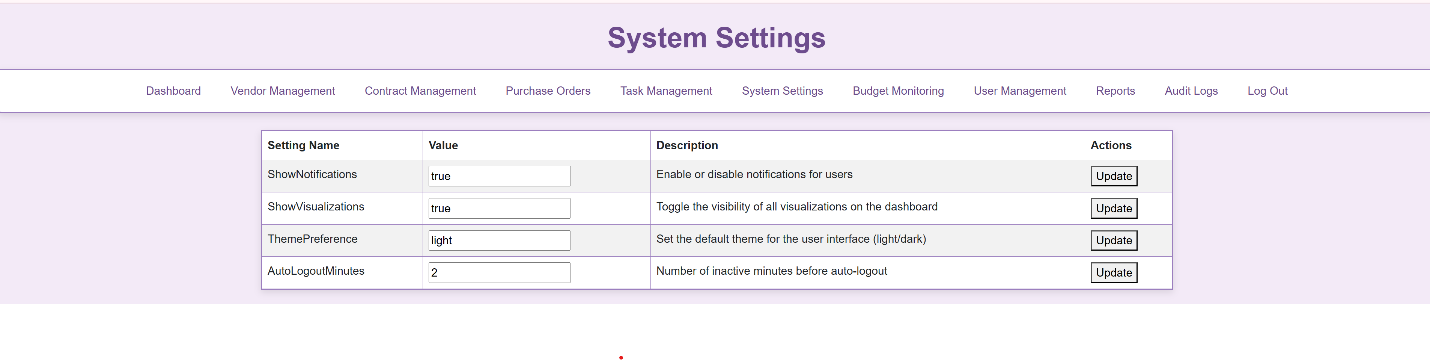
**Purpose of This Section**

* **Summary**: Provides a central hub for key updates.
* **Efficiency**: Improves user workflow with actionable insights.

## System Settings Page Overview

**Front-End (system-settings.html)**  
The System Settings page allows Admins to update configurations dynamically:

1. **Design and Layout:**
   * **Framework:** Bootstrap 5.3.0.
   * **Features:** Editable settings table with real-time updates.
2. **Scripts:**
   * **Includes:** main.js for fetching and updating system settings dynamically.
   * **Dynamic Features:**
     + Fetch settings and render them in the table.
     + Update settings with instant feedback and audit log recording.



**Implemented System Settings:**

1. **ShowNotifications:** Toggles notifications visibility.
2. **ShowVisualizations:** Controls visibility of visualization buttons and legends.
3. **ThemePreference:** Switches between light and dark themes.
4. **AutoLogoutMinutes:** Logs out users after specified idle time.

**Backend Logic (server.js)**

1. **Fetch Settings (GET /api/system-settings):** Retrieves settings from the database.
2. **Update Setting (PUT /api/system-settings/:id):** Updates a setting value and logs changes.

**Purpose**

* **Customization:** Tailors system settings dynamically.
* **Control:** Centralized configuration management.
* **Efficiency:** Ensures seamless updates without disruption.

## 10. Task Management Page Overview

### Front-End (HTML - task-management.html)

The Task Management page facilitates task assignments and updates. Here's an overview:

1. **Design and Layout**:
   * **Framework**: Bootstrap 5.3.0.
   * **Features**:
     + Task creation form.
     + Task list with status updates.
2. **Scripts**:
   * Includes main.js for managing tasks dynamically.

A screenshot of a task management

Description automatically generated

A screenshot of a computer

Description automatically generated

### Backend Logic (server.js)

1. **Fetch Tasks (GET /api/task)**:
   * **Purpose**: Retrieves tasks.
2. **Add/Update Task (POST /api/task or PUT /api/task/:id)**:
   * **Purpose**: Adds or updates tasks.

**Purpose of This Section**

* **Task Tracking**: Provides a clear view of task progress.
* **Collaboration**: Facilitates task delegation.

## 11. User Management Page Overview

### Front-End (HTML - user-management.html)

The User Management page allows Admins to manage system users. Here's an overview:

1. **Design and Layout**:
   * **Framework**: Bootstrap 5.3.0.
   * **Features**:
     + User registration/editing form.
     + User list with edit/delete options.
2. **Scripts**:
   * Includes main.js for handling user data dynamically.

A screenshot of a user management

Description automatically generated

A screenshot of a computer

Description automatically generated

### Backend Logic (server.js)

1. **Fetch Users (GET /api/user)**:
   * **Purpose**: Retrieves all users.
2. **Add/Update User (POST /api/user or PUT /api/user/:id)**:
   * **Purpose**: Adds or updates user details.

**Purpose of This Section**

* **User Management**: Enables admin control over user accounts.
* **Security**: Ensures role-based access

# Detailed Role Functions

1. **Admin**:
   * **Key Abilities**:
     + Manage all system data, users, and settings.
     + Access all reports and audit logs.
     + Full control over vendors, contracts, tasks, and purchase orders.
   * **Purpose**:
     + Ensures oversight and system control.
2. **Procurement Manager**:
   * **Key Abilities**:
     + Manage vendors, contracts, and purchase orders.
     + Limited reporting and budgeting access.
   * **Purpose**:
     + Handles procurement processes and vendor relations.
3. **Department Head**:
   * **Key Abilities**:
     + View department-specific budgets and procurement reports.
     + Filter and track purchase orders.
   * **Purpose**:
     + Focuses on budget adherence and department-level procurement tracking.
4. **General User**:
   * **Key Abilities**:
     + Limited to viewing general reports and dashboards.
   * **Purpose**:
     + Provides basic system access for users requiring information but no editing privileges.

## Table showing Roles

|  |  |  |  |
| --- | --- | --- | --- |
| **Role** | **Accessible Pages** | **Allowed Actions** | **Restricted Actions** |
| **Admin** | -Dashboard -Vendor Management - Contract Management - Purchase Order - Reports - Audit Logs - Budget Tracking - System Settings - User Management - Task Management | - Full access to manage vendors, contracts, and users. - Can approve/reject purchase orders. - View and export all reports. - Access all audit logs. - Manage system settings. - Assign, view, and update tasks. - View and manage budgets. | - None (Admins have full access to all pages and actions). |
| **Procurement Manager** | - Dashboard - Vendor Management - Contract Management - Purchase Order - Reports - Task Management | - Add, edit, delete, and view vendors. - Create, approve/reject purchase orders. - Add and edit contracts. - View and filter procurement and vendor performance reports. - Assign and update tasks. - View department budgets. | - Cannot access user management. - Cannot change system settings. - Limited access to audit logs (only procurement-related). |
| **Department Head** | - Dashboard - Purchase Order - Reports - Budget Tracking | - View and filter purchase orders. - View procurement reports and department budgets. - Export department budget reports. | - Cannot add or edit vendors, contracts, or users. - Cannot approve/reject purchase orders. - No access to audit logs or system settings. |

# Charts using D3 showing meaningful insights on Dashboard:

## Average Performance Ratings Graph Type:

**Bar Chart Objective:** To provide insights into the overall performance of vendors in terms of quality and timeliness of their services. Description: This bar chart displays the average ratings for "Overall Quality" and "Overall Timeliness" of vendor services.

**Each bar represents a distinct metric:**

**Overall Quality Rating:** This measures the vendors' quality of work based on client feedback and performance evaluations.

**Overall Timeliness Rating:** This evaluates how timely vendors are in delivering their services.

**Key Features:** Ratings are averaged across all vendors. The color-coded bars (blue for quality and purple for timeliness) make it easy to distinguish between the two metrics. Legends are positioned on the left for clear identification. Insights: Vendors generally score higher on quality than timeliness, suggesting a need for improvement in meeting deadlines. This graph allows administrators to prioritize areas for improvement when interacting with vendors.

A screen shot of a graph

Description automatically generated

## Percentage of Budget Utilized Graph Type:

**Doughnut Chart Objective:** To visualize the percentage of allocated budget utilized by various departments.

**Description:** The doughnut chart provides a breakdown of budget utilization across departments. Each slice represents a department, and its size correlates to the percentage of the department's allocated budget that has been spent. The legend displays department names with corresponding colors.

**Key Features:**

**Interactive:** Hovering over a slice reveals the exact percentage utilized by the department. The use of distinct colors for each department ensures clarity and quick identification.

**Insights:** Departments with high budget utilization may need closer monitoring to prevent overspending. Departments with low utilization might indicate opportunities for cost-saving or reallocation of resources. Helps in decision-making regarding future budget planning and allocation.

A pie chart with text and a circle

Description automatically generated with medium confidence

## Purchase Orders by Status Graph Type:

**Doughnut Chart Objective:** To categorize purchase orders based on their current statuses (e.g., Pending, Approved, Rejected).

**Description:** This chart visually represents the distribution of purchase orders by their status. Each slice corresponds to a status, and the size reflects the number of purchase orders in that category. The legend provides a color-coded key for interpreting the chart.

**Key Features:** Different colors for each status ensure easy identification. Interactive: Hovering over slices displays the count of purchase orders for each status.

**Insights:** A high proportion of "Pending" orders may indicate delays in processing. The number of "Rejected" orders can highlight issues in vendor compliance or the procurement process. Tracking the "Approved" orders helps assess workflow efficiency and the pace of approvals.

A screen shot of a graph

Description automatically generated