Name: **Fasih Khalil**

Umair Ahmad

Roll No: **FA23-BCS-282**

FA23-BCS-244

Database Systems Project Report

Project Report for Corporate Vendor and Contract Management System

1. Overview

The Corporate Vendor and Contract Management System is designed to streamline vendor management, contract handling, purchase order tracking, and budget monitoring. The system incorporates web-based modules with database-backed functionalities for enhanced efficiency and compliance.

2. Functionalities Implemented

Based on the files and code analysed, the system implements the following core functionalities:

1. **Vendor Management**:
    a. Vendor registration, login, and profile management.
    b. Performance reviews and analytics.
2. **Contract Management**:
    a. Initiating contracts with vendors.
    b. Tracking contract renewals and expirations.
3. **Purchase Order Management**:
    a. Creating and tracking purchase orders.
    b. Ensuring compliance with departmental budgets.

4. **Budget Monitoring**:
    a. Allocating, tracking, and validating budgets.
    b. Automatic notifications for budget overspending.
5. **Reports and Dashboards**:
    a. Generating reports for procurement, vendor performance, and budget tracking.

3. Database Structure

The database, defined in the `vendordb.sql` file, includes the following main components:

1. **Tables**:
    a. **Budgets**: Tracks financial allocations and expenditures.
    b. **Departments**: Links departmental activities with budgets.
    c. **Vendors**: Stores vendor details and performance metrics.
    d. Additional tables for contracts, purchase orders, and user management (assumed based on the system requirements).
2. **Constraints**:
    a. Foreign key relationships for referential integrity.
    b. Generated fields for computed values (e.g., remaining budget).
    c. Validation constraints like CHECK for performance ratings.

4. Backend Logic

The backend, implemented using **Node.js**, includes:

- API routes for CRUD operations on vendors, contracts, and budgets.
- Connection to a MySQL database using the **MySQL2** library.
- Middleware for handling JSON data and ensuring crossorigin compatibility.
- Assumed features:
    - Routes for notification triggers (e.g., contract renewals).
    - Error handling and logging mechanisms.

5. Frontend Implementation

The frontend uses **HTML** for form submissions and data presentation. Observed functionalities include:

- Vendor registration and login pages.
- Role-based dashboards (e.g., manager, team, vendor).
- Forms for budget allocation, purchase order tracking, and vendor evaluation.
- Additional assumed features:
    - Interactive navigation using JavaScript or frameworks. ○ Use of CSS or libraries like Bootstrap for styling and responsiveness.

6. Observations

1. **Integration**:
    a. The database schema aligns well with backend functionality.
    b. Expected use of AJAX or fetch requests for dynamic updates on the web interface.
2. **Notifications**:

    a. Features like contract renewal alerts and budget overspending warnings are likely implemented via backend triggers or scheduled jobs.
3. **Scope for Optimization**:
    a. Query optimization in MySQL.
    b. Enhancements to frontend interactivity and aesthetics using Bootstrap or React.

7. Steps to be Completed

1. **Frontend Integration**:
    a. Connect the HTML forms to backend routes for dynamic data updates.
    b. Test responsiveness and usability across devices.
2. **Database Enhancements**:
    a. Verify constraints and relationships for all tables.
    b. Test triggers and stored procedures to ensure correct functionality.
3. **Backend Completion**:
    a. Add routes for report generation and analytics.
    b. Implement detailed error logging and testing.
4. **Testing and Deployment**:
    a. Conduct end-to-end testing to identify bugs.
    b. Prepare deployment scripts for hosting on a server (e.g., Heroku, AWS).
5. **Additional Features**:
    a. Include data visualization for dashboards (e.g., charts for vendor performance).

8. Assumptions

- Additional use cases like team management and procurement analytics may require separate backend routes and frontend forms.
- Certain complex triggers (e.g., notifications) might use server-side scripts or MySQL events.

9. Deliverables

- **ERD**: A graphical representation of entities and their relationships (e.g., in `erd.drawio`).
- **Relational Schema**: The schema design visualized in MySQL Workbench (`err.mwb`).
- **Functional Web Interface**: HTML forms and Node.js backend linked to the MySQL database.
- **Detailed Word Report**: Includes all findings and technical documentation.