# **DataBase Final Project**

# **Report on Corporate Vendor and Contract Management System**

## Prepared By: Umer Siddiqui (22I-2038) Abdullah Bin Irshad (22I-1912)

## Section-B

## Submitted To: Miss Sidra Fayyaz

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 

## **Introduction**

The **Corporate Vendor and Contract Management System (CVCMS)** is designed to streamline and improve the management of vendor relationships, contracts, and performance evaluations within a corporate environment. The system ensures efficient tracking, evaluation, and management of vendors, while automating key tasks such as contract renewals, performance assessments, and vendor selection for future engagements.

This report provides an overview of the project, its goals, methodology, schema design, and how the system was built and tested.

## **Background**

In any corporate setup, vendor relationships play a crucial role in the operational efficiency of the company. The need for effective vendor management is essential to reduce costs, mitigate risks, and improve supplier performance. Additionally, keeping track of contract expiration dates, terms, and performance metrics is critical for business continuity and future planning.

The CVCMS aims to address these issues by automating the management of contracts, monitoring vendor performance, and enabling transparent decision-making.

## **Problem Statement**

The existing system lacks centralized management, and information about vendors and contracts is scattered across multiple platforms. This results in inefficiencies, manual tracking, and missed renewal deadlines, which ultimately affects supplier relationships and operational performance.

## **Needs and Scope**

The system will:

* Automate vendor registration, contract renewal processes, and performance reviews.
* Maintain detailed records of vendor evaluations, contract details, and history.
* Provide reporting tools for decision-makers to assess vendor performance and contract statuses.
* Allow for easy filtering and searching of vendor data, contracts, and other related information.

The project focuses on creating a robust system with the following components:

* **Web Interface**: Allows the user to interact with the system, fill out forms, submit data, and view results.
* **Database**: Stores vendor details, contract information, and vendor performance data.
* **Backend**: Handles the business logic and database interaction.

## **Methodology**

The project was divided into several key stages:

1. **Requirements Gathering**: Identifying the core needs of the system and gathering requirements for database design, user interfaces, and functionalities.
2. **Database Design**: Creating the relational schema for storing the necessary data, including vendor details, contracts, and vendor performance reviews.
3. **Backend Development**: Implementing logic for vendor registration, contract renewal, and performance tracking using Node.js and MySQL.
4. **Frontend Development**: Designing a user-friendly interface using HTML, CSS, and JavaScript to allow users to interact with the system.
5. **Testing**: Ensuring the system functions properly by performing unit tests and integration tests.

## **Relational Schema**

The relational schema for the **Corporate Vendor and Contract Management System** is designed to store essential details related to vendors, contracts, and evaluations. Below is a summary of the key entities and their relationships.

**Tables in the Database:**

1. **Vendors:**
   * vendor\_id (Primary Key)
   * Vendorname
   * contactinfo
   * CertificationStatus
   * ServiceCategory
2. **Contracts:**
   * contract\_id (Primary Key)
   * vendor\_id (Foreign Key to Vendors)
   * start\_date
   * end\_date
   * Terms & Conditions
   * status (Active, Expired, Pending Renewal)
3. **Vendor Reviews:**
   * review\_id (Primary Key)
   * vendor\_id (Foreign Key to Vendors)
   * review\_date
   * rating (1-5 scale)
   * ReviewText
4. **Purchase Orders:**
   * PO\_id (Primary Key)
   * Vendor\_id (Foreign Key to Vendors)
   * Budget\_id(Foreign Key to Budgets)
   * TotalCost
   * ItemDetails
   * status (Pending, Shipped, Delivered, etc.)
5. **Vendor Certifications:**
   * certification\_id (Primary Key)
   * vendor\_id (Foreign Key to Vendors)
   * certification\_name
   * certification\_date
   * certification\_status
   * certification\_type
   * issue\_date
   * expiry\_date
6. **Departments:**
   * dept\_id (Primary Key)
   * department\_name
   * BudgetSpent
   * BudgetAllocated
7. **Users:**
   * user\_id (Primary Key)
   * username
   * role (Admin, User, Manager)
   * pass (hashed password)
8. **Contract Renewals:**
   * renewal\_id (Primary Key)
   * contract\_id (Foreign Key to Contracts)
   * renewal\_date
   * renewal\_status
9. **Budget:**
   * budget\_id (Primary Key)
   * department\_id (Foreign Key to Departments)
   * spent\_amount
   * Allocated\_amount
10. **Notifications:**

* notification\_id (Primary Key)
* user\_id (Foreign Key to Users)
* message
* notification\_date
* status (Read, Unread)

## **Relationships:**

* **Vendors ↔ Contracts**: One-to-Many relationship. A vendor can have multiple contracts (One-to-Many between Vendors and Contracts).
* **Vendors ↔ PurchaseOrder**: One-to-Many relationship. A vendor can have multiple purchase orders (One-to-Many between Vendors and purchase orders).
* **Vendors ↔ Vendor Reviews**: One-to-Many relationship. A vendor can have multiple reviews (One-to-Many between Vendors and Vendor Reviews).
* **Contracts ↔ Purchase Orders**: One-to-Many relationship. A contract can have multiple purchase orders (One-to-Many between Contracts and Purchase Orders).
* **Vendors ↔ Vendor Certifications**: One-to-Many relationship. A vendor can have multiple certifications (One-to-Many between Vendors and Vendor Certifications).
* **Departments ↔ Budget**: One-to-Many relationship. A department can have a single budget per year, but multiple years (One-to-Many between Departments and Budget).
* **Departments ↔ User**: One-to-Many relationship. A department can have multiple users (One-to-Many between Departments and users).
* **Contracts ↔ Contract Renewals**: One-to-Many relationship. A contract can have multiple renewals (One-to-Many between Contracts and Contract Renewals).
* **Users ↔ Notifications**: One-to-Many relationship. A user can receive multiple notifications (One-to-Many between Users and Notifications).

## **System Implementation**

The implementation process involved the following steps:

1. **Backend Development (Node.js and MySQL)**:
   * **Express.js** was used to build the RESTful API to handle requests for vendor registrations, contract renewals, and performance reviews.
   * **MySQL** was chosen as the database to store and retrieve vendor data, contracts, and performance metrics.
   * **API Endpoints** were created for adding vendors, submitting contract details, performing evaluations, and handling contract renewals.
2. **Frontend Development (HTML, CSS, and JavaScript)**:
   * A simple and clean **user interface** was built using HTML and CSS for easy navigation.
   * **JavaScript** was used to handle user interactions, validate form inputs (such as email format), and display dynamic content on the page.
   * **Buttons** and forms were styled to improve the user experience (e.g., form submissions and contract renewal buttons).
   * **Charts** were integrated to visualize vendor performance and contract renewal status.
3. **Form Handling and Validation**:
   * Form validation ensured that the inputs were correct and complete before submission.
   * Email addresses were validated for proper formatting, and required fields were checked.
4. **Error Handling and Alerts**:
   * The system provided clear feedback messages, including error messages if something went wrong during form submission or data entry.

## **Challenges Faced**

* **Data Integrity**: Ensuring that the relationships between vendors, contracts, and performance evaluations were maintained without data anomalies.
* **User Interface Design**: Making sure the user interface was both functional and aesthetically pleasing.
* **Performance Optimization**: Optimizing database queries for quick retrieval of contract and vendor data, especially when scaling up.

## **Results**

The system successfully:

* Allowed users to register vendors and manage contracts.
* Supported automatic renewal processing and contract expiration tracking.
* Provided real-time performance evaluation data.
* Displayed results and performance summaries using interactive charts.

After testing, the system performed well and met the requirements set out at the beginning of the project.

**Conclusion**

The **Corporate Vendor and Contract Management System** is a comprehensive solution to manage vendor relationships, contracts, and performance evaluations in a corporate environment. The system automates tedious tasks, reduces human errors, and helps management make informed decisions. Future improvements can include adding more advanced features such as email notifications for contract renewals or integrating more analytics for better decision-making.

**Glossary**

* **Vendor**: A company or individual that provides goods or services to another company.
* **Contract**: A formal agreement between two parties regarding terms of service, prices, and conditions.
* **Renewal**: The process of extending an existing contract for another period.
* **Performance Review**: Evaluation of a vendor’s performance based on various metrics such as quality, delivery time, etc.

# **Deliverable #1 The ERD**

# 

# **Deliverable #2: Description of All Tables Using DESCRIBE Command**

# 

# **OUTPUTS OF THE PROJECT**

