* ADITYA BHATT – 23SCSE1410086
* BHASKAR RAJ – 23SCSE1410019
* BHAVISHAYYA KAUSHIK – 23SCSE1410002

# **Employee Management System**

**1. Introduction**

This document outlines the development of an SQL Employee Management System, a command-line application designed to efficiently manage a company's employee database. Developed using Node.js, Inquirer, and MySQL, this system enables users to perform various operations, including viewing, adding, and updating employee information.

**2. Objectives**

* Develop a user-friendly command-line interface (CLI) application for managing employee data.
* Implement functionalities to view, add, and modify departments, roles, and employees within the system.
* Ensure data integrity and consistency within the MySQL database.

**3. Features and Functionalities**

* **View Operations:**
  + View all departments with their names and IDs.
  + View all roles, including job titles, role IDs, associated departments, and salaries.
  + View all employees with details such as IDs, names, job titles, departments, salaries, and managers.
* **Add Operations:**
  + Add a new department by entering its name.
  + Add a new role by specifying its name, salary, and department.
  + Add a new employee by providing their first name, last name, role, and manager.
* **Update Operations:**
  + Update an existing employee's role by selecting the employee and assigning a new role.

**4. Technology Stack**

* **Backend:**
  + Node.js: Utilized for server-side scripting and application logic.
  + Inquirer: Employed for creating interactive command-line user interfaces.
  + MySQL: Serves as the relational database management system.

**5. Database Schema**

The database comprises three primary tables:

* **Department:**
  + id: INT PRIMARY KEY (Unique identifier for the department)
  + name: VARCHAR(30) (Name of the department)
* **Role:**
  + id: INT PRIMARY KEY (Unique identifier for the role)
  + title: VARCHAR(30) (Title of the role)
  + salary: DECIMAL (Salary associated with the role)
  + department\_id: INT (Foreign key referencing the Department table)
* **Employee:**
  + id: INT PRIMARY KEY (Unique identifier for the employee)
  + first\_name: VARCHAR(30) (First name of the employee)
  + last\_name: VARCHAR(30) (Last name of the employee)
  + role\_id: INT (Foreign key referencing the Role table)
  + manager\_id: INT (Self-referencing foreign key)

**6. Implementation Details**

* **User Interface:**
  + The application operates through a command-line interface, prompting users with options to perform various operations.
* **Data Management:**
  + MySQL is used to store and manage data, ensuring relational integrity between departments, roles, and employees.
* **Application Logic:**
  + Node.js handles the application logic, processing user inputs and interacting with the MySQL database.

**7. Installation and Usage**

1. **Navigate to the Project Directory:**

Bash

cd sql-employee-management-system

1. **Install Dependencies:**

Bash

npm install

1. **Set Up the Database:**
   * Log in to the MySQL shell:

Bash

mysql -u root -p

* + Source the provided schema file to create the database structure:

SQL

source path/to/schema.sql;

* + Exit the MySQL shell:

SQL

quit;

1. **Seed the Database (Optional):**

Bash

npm run seed

(This step populates the database with sample data)

1. **Start the Application:**

Bash

npm run start

**8. Testing**

* **Functionality Testing:**
  + Verify each operation (view, add, update) to ensure they perform as expected.
* **Data Integrity Testing:**
  + Ensure that relationships between tables are maintained, and foreign keys are correctly referenced.

**9. Challenges and Solutions**

* **Data Consistency:**
  + **Challenge:** Maintaining consistent data across related tables.
  + **Solution:** Implemented foreign key constraints and cascades in MySQL to ensure referential integrity.
* **User Input Validation:**
  + **Challenge:** Handling invalid or unexpected user inputs.
  + **Solution:** Utilized Inquirer's validation features to ensure proper input before processing.

**10. Future Enhancements**

* **Additional Features:**
  + Implement functionalities to delete departments, roles, and employees.
  + Add more complex queries, such as viewing employees by manager or department budgets.
* **User Interface:**
  + Develop a graphical user interface (GUI) to enhance user experience.

This report provides a comprehensive overview of the SQL Employee Management System, including its objectives, features, implementation, and future enhancements.