Company Software Management and Authorization Systems

Team Data SOX

Professor: Chaiyaporn Mutsalklisana 12/6/14 Database Management

Team Members

Rahul Kale
Jignash Reddy
Priyabrata Purohit
Leroy Pinto

Table of Contents

List of Figures

1. Problem Description

ManageSoft is a software organization which is based in Boston, MA. It was founded in 1999. Basically the organization has many departments which has various projects and it is handled by Project managers. Software's needed by software engineers, other employees and for the project requirement is supplied by many vendors to the company. But the request of software made by the employees has to be approved by the Project manager.

So our main objective of this project is to assist an employee of the organization in installing the required software which is needed for the project and it will help to increase the project efficiency .Our system will help the team members of the project to keep in par of the software's available with the minimum process time using Database Management system.

2. Project Goals:

We realize the Importance of Softwares in growing IT Industries. Software Engineers now a days need to work with up-to-date softwares all the time in order to meet the deadlines declared by their respective Customers. And an IT Industry that supports many versatile Software Engineers has to cope with the new and different software requests. So it is very pivotal that the Softwares be installed on time and the Vendors providing the softwares be paid. All this is in a very organized and short time.

Our goal thus includes to implement the following in order to encompass the IT industry's installation of softwares.

A. Records of Software Installation on an Employee Machine with the name of the Software and the date when installed.
B. Records of the most recent Software Updates and the Vendors providing it. So if any Software Engineer wants to work with the most recent one, he can do so.
C. Records of the requests for software initiated by the Software Engineer and authorized by the Project Manager and finally approved by the IT Help Desk.
D. Records of the Vendors providing the softwares and their previous payments. This way, the IT Help Desk would have track of all the Vendors and the Payment would be smooth enough.
E. Records of the Employee's Payment History. So an Employee can check his Salary and whatever increment he had received previously.
F. Records of Employees' general information.

G. Records of Employee Workstation. So if the IT Help Desk needs to know the compatibility of a particular software with a particular Workstation, he could check it here.

3. Actors

A. Roles ad Task

The Company Software Management System will require the integral part of various Users in the system. The application main aim is to allow the company to track the request and the installation of the various software in the Workstation of the Employee.

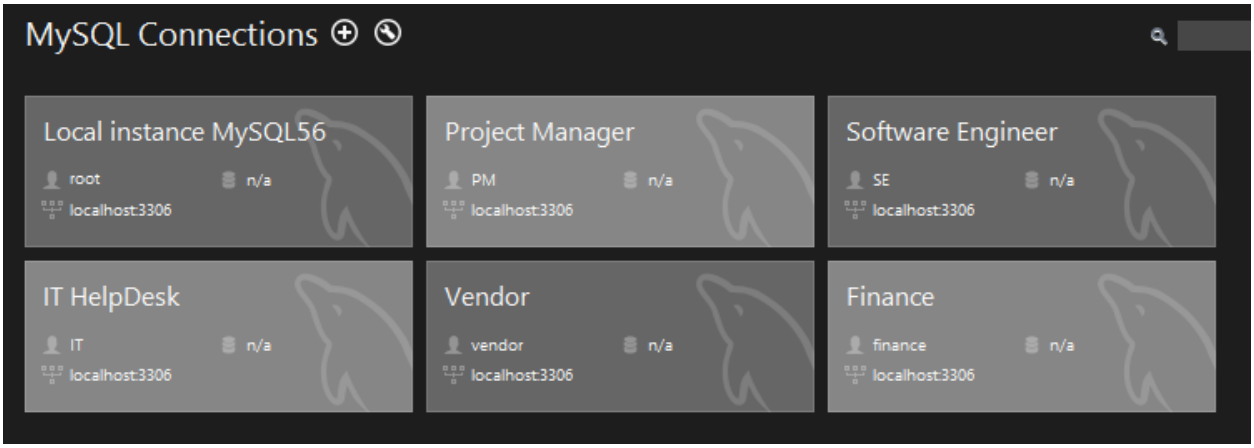

Figure 1MySQL Users

Software Engineer: The Application User

The company employee who will use this software for software installation request and installation of the required software for the project.

Project Manager: The Application User

The user generally has multiple projects and employees under him. He will check the software request, requested by the employee and approve or reject based on the understanding and approvals.

IT Help Desk

The user is the one who controls the employee's software installation on the systems. Keeps a tab of the request which are open or close. Checks the softwares available with the Vendors.

Vendors

Supplies the softwares needed in the organization and are linked with the IT help department for the supply and installation.

Finance

Keeps a tab of the software price and payments to the Vendors.

B. Privileges

Software Engineer

Grant select,Insert,Update on request to 'SoftwareEngineer';

Grant select,Update on software to 'SoftwareEngineer';

Grant select on employeepaymenthistory to 'SoftwareEngineer';

Grant select on shift to 'SoftwareEngineer';

Grant select on project to 'SoftwareEngineer';

Grant select,Update on employeaddress to 'SoftwareEngineer';

GRANT SELECT ON data.SoftwaredetailReport TO 'SE'@'localhost' → View

Project Manager

Grant select on softwareengineer to 'Project Manager';

Grant select,Update on request to 'Project Manager';

Grant select on project to 'Project Manager';

Grant select on software to 'Project Manager';

GRANT SELECT ON data.EmployeewholeDetail TO 'PM'@'localhost' → View

IT Help Desk

Grant select,Insert,Update on request to 'ItHelpDesk';

Grant select,Update on software to 'ItHelpDesk';

Grant select on softwarecosthistory to 'ItHelpDesk';

Grant select on vendor to 'ItHelpDesk';

GRANT SELECT ON data.VendoralllDetail TO 'IT'@'localhost' → View

Vendor

Grant select,Insert,Update on software to 'vendor';

Grant select,Insert,Update on softwarecosthistory to 'vendor';

Grant select,Insert,Update on vendoraddress to 'vendor';

Grant select,Insert,Update on vendorcontact to 'vendor';

Grant select on payments to 'vendor';

GRANT SELECT ON data.VendoralllDetail TO 'Vendor'@'localhost' → View

Finance

Grant select,Insert,Update on payments to 'Finance';

Grant select on vendor to 'Finance';

4. ER Diagram

This ER diagram represents basic ER diagrams with attributes and the relationships between the tables.

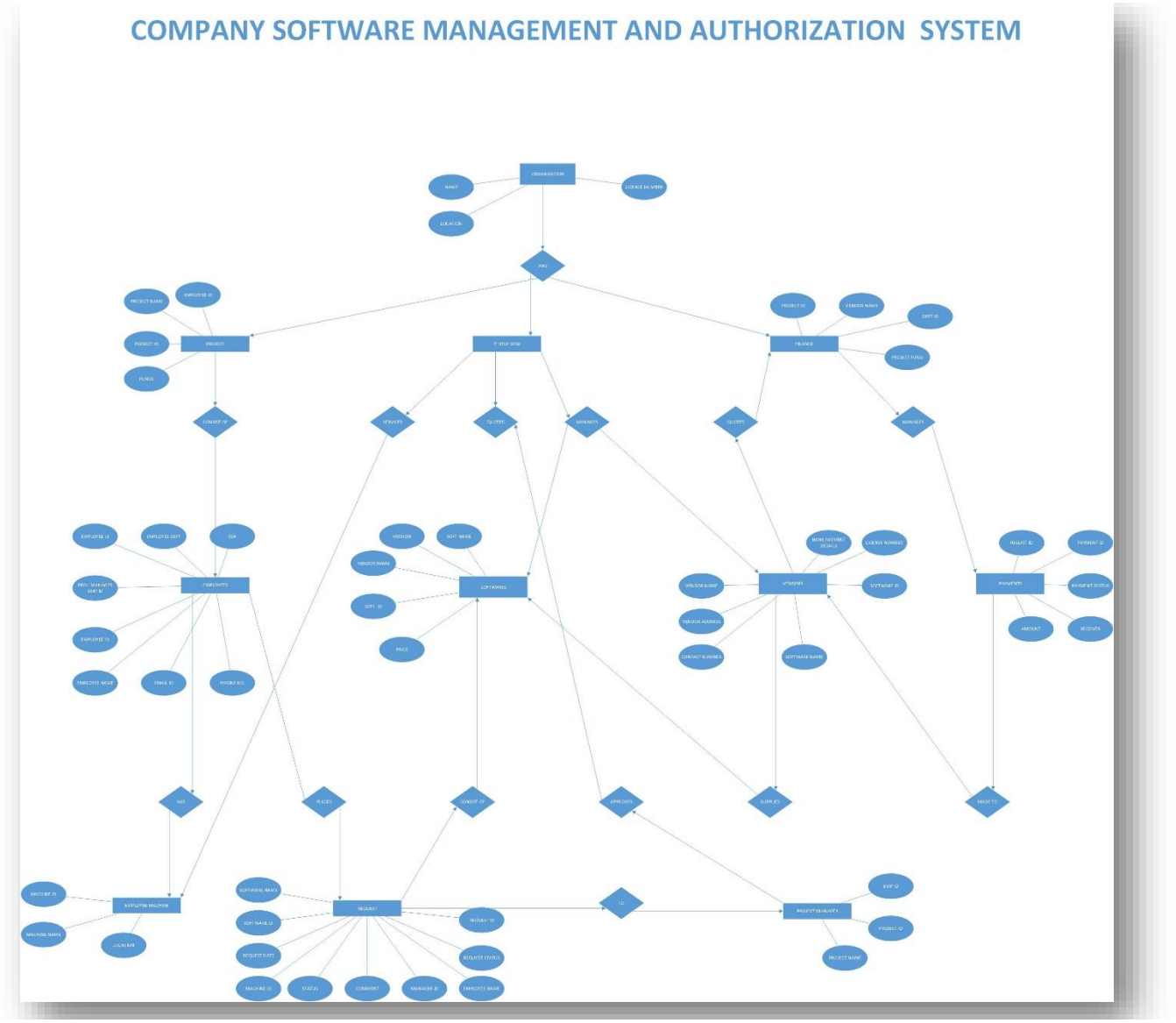

Figure 2: ER Diagram

5. EER Diagram

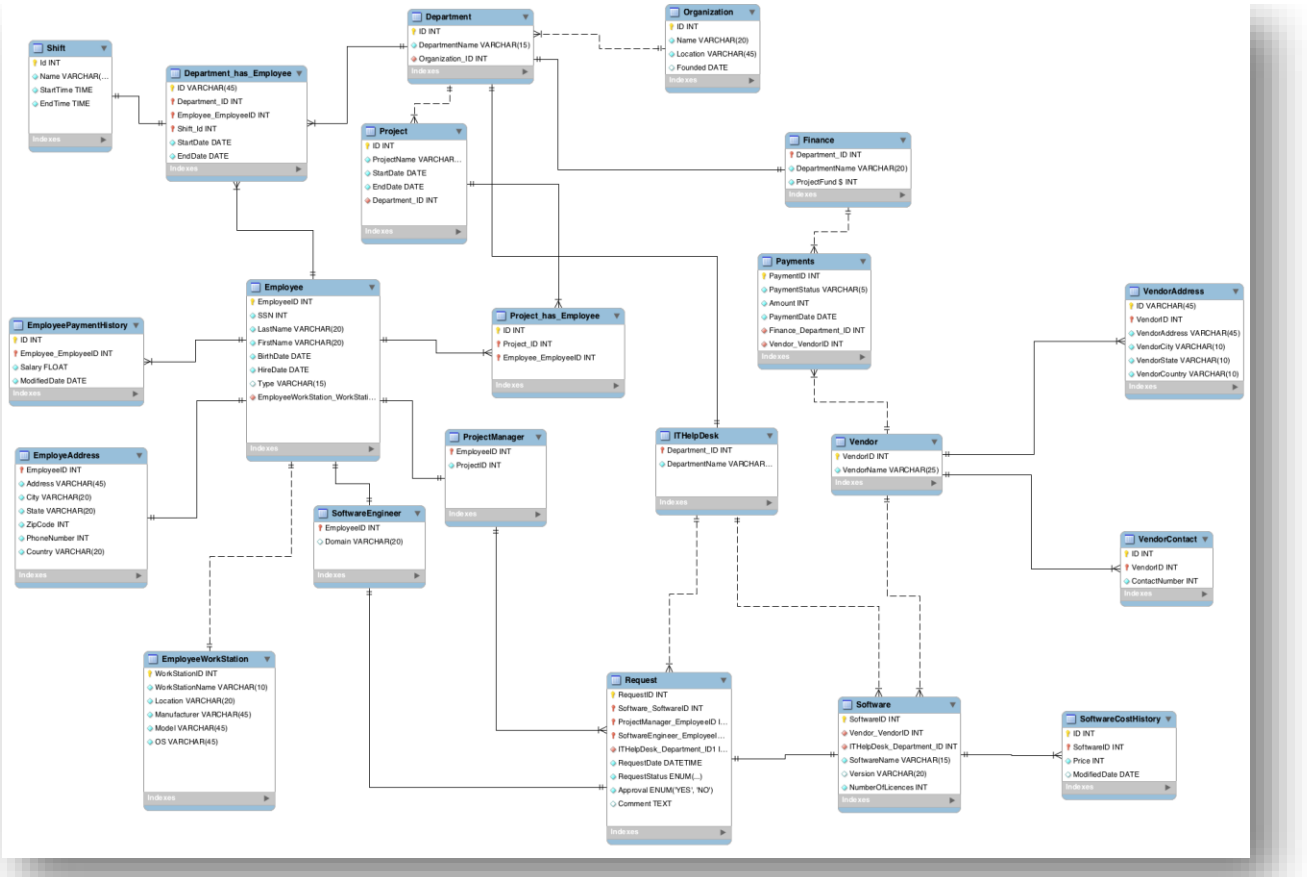

Figure 3: EER Diagram

Our team identified the basic elements and attribute required to build the software request management system. We Identified 21 basic entities in the ER diagram.

Organization, department, shift, department_has_employee, employee, employee_update, employeepaymenthistory, employeeworkstation, employeeaddress, project, projectmanager, softwareengineer, request, software, softwarecosthistory, vendor, vendorcontact, vendoraddress, ithelpdesk, finance, payments.

With respect to database designing, determining the relationships with entities was important, and at the modeling stage, we implemented the database rules. We tried to improve entity relationship as we went through the project implementation and also adjust the relationship accordingly. Business process understanding is much important during the database design stage. We tried our best to improve the process flow of the request management system.

6. Relationships and Cardinalities

(a) **Organization** and **Department** — One to Many

An Organization must have at least one Department
A Department must belong to exactly one Organization

(b) **Department** and **Employee** — Many to Many

A Department must have at least one Employee
An Employee must belong to at least one Department

(c) **Department** and **Project** — One to Many

A Department must have at least one Project
A Project must belong to exactly one Department

(d) **Department** and **Finance** — One to One

A Department must have exactly one Finance Division
A Finance Division must belong to exactly one Department

(e) **Department** and **ITHelpDesk** — One to One

A Department must have exactly one IT Help Desk
An IT help Desk must belong to exactly one Department

(f) **Project** and **Employee** — Many to Many

A Project must consist of at least one Employee
An Employee must belong to at least one Project

(g) **Employee** and **EmployeePaymentHistory** — One to Many

An Employee must have at least one Employee Payment History
Each Employee Payment History must belong to exactly one Employee

(h) **Employee** and **EmployeeAddress** — One to One

An Employee must have exactly one Employee Address
An Employee Address must belong to exactly one Employee

(i) **Employee** and **EmployeeWorkStation** — One to One

An Employee must have exactly one Employee Work Station
Each Employee Workstation must belong to exactly one Employee

(j) **SoftwareEngineer** and **Request** — One to One

Each Software Engineer must send exactly one Request
A Request can be sent by only one Software Engineer

(k) **ProjectManager** and **Request** — One to Many

A Project Manager must approve at least one Request
Each Request must be authorized by only one Project Manager

(l) **ITHelpDesk** and **Request** — One to Many

An IT Help Desk must manage at least one Request
Each Request must be managed by only one IT Help Desk

(m) **ITHelpDesk** and **Software** — One to Many

An IT Help Desk must manage at least one Software
Each Software must be managed by only one IT Help Desk

(n) **Finance** and **Vendors** — Many to Many

A Finance division must make payments to many Vendors
Each Vendor must be compensated by many Finance divisions

(o) **Vendor** and **Software** — One to Many

A Vendor must provide at least one Software.
Each Software may be provided by exactly one Vendor

(p) **Software** and **Request** — One to One

Each Software must be referenced in exactly one Request
A Request must consist of exactly one Software

(q) **Software** and **SoftwareCostHistory** — One to Many

Each Software must have at least one Software Cost History
A Software Cost History must be associated with exactly one Software

(r) **Vendor** and **VendorContact** One to Many

A Vendor must have at least one Vendor Contact Number
A Vendor Contact Number must belong to exactly only one Vendor

(s) **Vendor** and **VendorAddress** One to Many

A Vendor must have at least one Vendor Address
A Vendor Address must belong to exactly only one Vendor

7. USE CASE Diagram

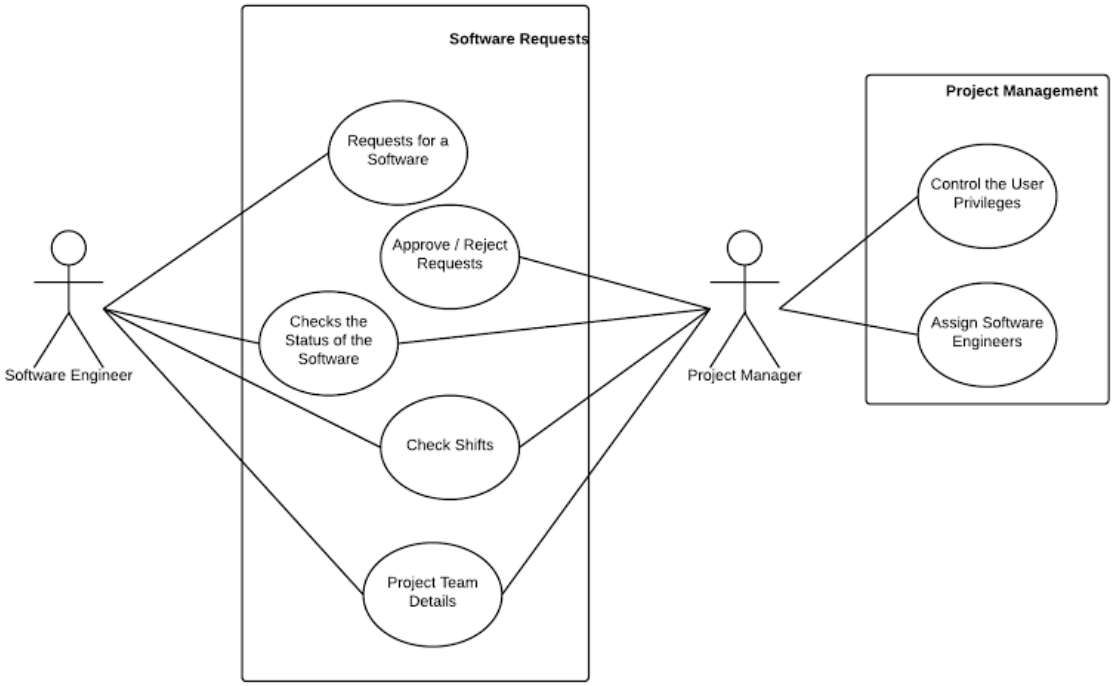

Figure 4: Use Case Diagram 1

1. **Software Engineer**- Software Engineer requests for the software needed and checks the shifts of the employees and the status of the software.
2. **Project Manager**- Assigns the projects to software engineers and controls and managers the projects. He can also approve/reject the software requests and can check the shifts as well project team details.

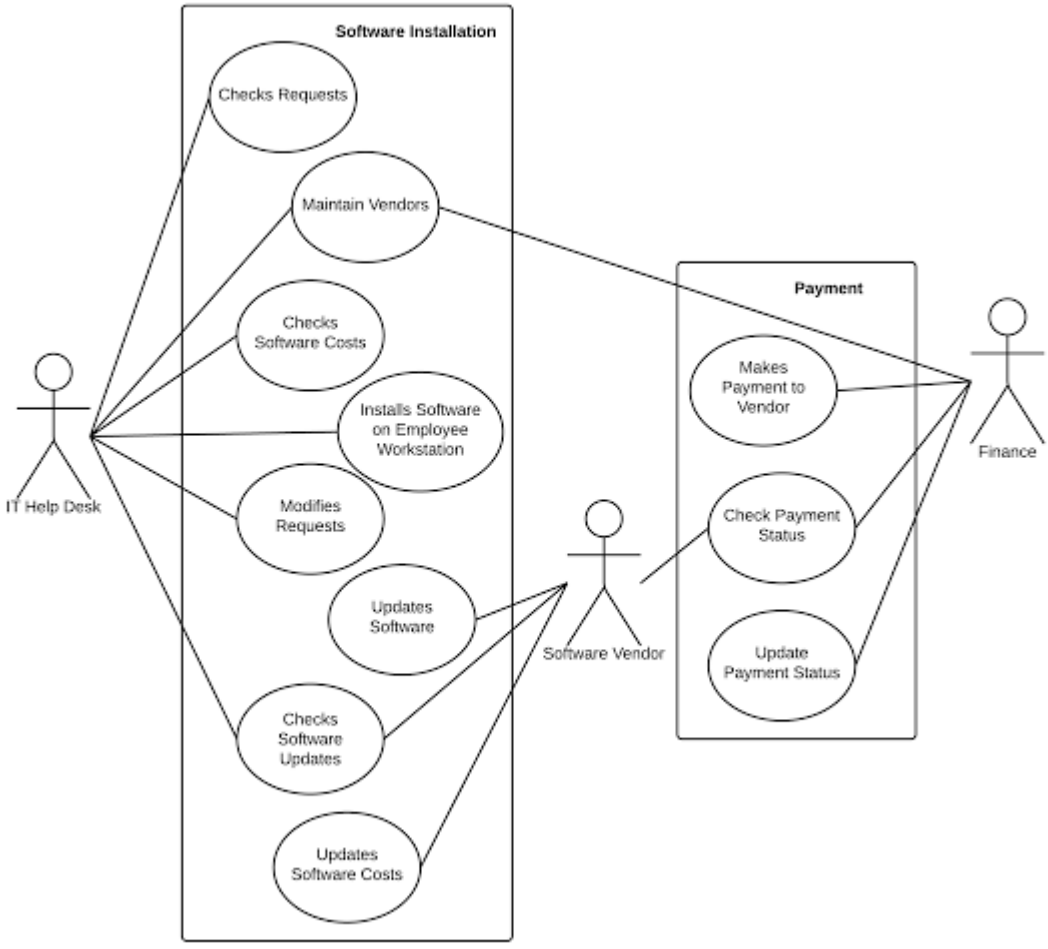

Figure 5 Use Case Diagram 2

3. **IT helpdesk**- It checks the request of the software and maintains the vendors and checks the software cost. It helps in Installing the software on employee workstation and can modify the software request .It can check the software updates.
4. **Vendor-** Vendor supplies the softwares to the organization and updates the software and checks the software updates .It updates the software costs.
5. **Finance-**Finance department maintains the vendor and makes the payment to vendor .It checks the payment status and update payment status.

8. Tables with attributes

1. **Organization** (OrganisationID, Name, Location, Founded)
2. **Department** (ID, DepartmentName, Organization_ID)
3. **Project** (ID, ProjectName, StartName, EndDate, Department_ID)
4. **Employee** (EmployeeID, SSN, LastName, FirstName, BirthDate, HireDate, Type, EmployeeWorkstation_WorkStationID)
5. **EmployeeAddress** (EmployeeID, Address, City, State, ZipCode, PhoneNumber, Country)
6. **EmployeePayementHistory** (ID,salary, ModifiedDate, Employee_EmployeeID)
7. **Department_has_Employee** (ID, Depatment_ID, Employee_EmployeeID, Shift_ID, StartDate, EndDate)
8. **Shift** (ID, Name, StartDate, EndDate)
9. **Project_has_Employee** (ID, Project_ID, Employee_EmployeeID)
10. **ProjectManger** (EmployeeID, ProjectID)
11. **SoftwareEngineer** (EmployeeID, Domain)
12. **ITHelp Desk** (Department_ID, DepartmentName)
13. **Finance** (DepartmentID, DepartmentName, ProjectName)
14. **Vendor** (VendorID, VendorName)
15. **VendorAddress** (ID, VendorID, VendorAddress, VendorCity, VendorState, VendorCountry)
16. **VendorContact** (ID, VendorID, ContactNumber)
17. **EmployeeWorkstation** (WorkstationID, WorkstationName, Location, Manufacturer, Model, OS)
18. **Request** (RequestID, Software_SoftwareID, ProjectManager_EmployeeID, ITHelpdesk_Department, RequestDate, RequestStatus, Approval, Comment)
19. **Software** (SoftwareID, SoftwareName, Version, NumberOfLicenses, Vendor_ID, ITHelpDesk_Department_ID)
20. **SoftwareCostHistory** (ID, SoftwareID, Price, ModifiedDate)
21. **Payments** (PayementID, Paymentstatus, Amount, PaymentDate, Finance_Department_ID, Vendor_VendorID)

9. Security

Security is the main issues of any system. This system is secured using all possible secure code methods to avoid any breach of data. This system deals with views, which are used to control the data that is accessed by the user. There are three views in the system involving the Finance view, It Helpdesk View, Project Manager View.

10. Views

Views used in this application:

A. Finance view

```
CREATE VIEW Departmentdetails  AS

SELECT Name,DepartmentName,ProjectName

FROM Department

Inner Join Organization

On Department.Organization_Id = Organization.Id

Inner Join project

ON department.Id = project.department_ID;
```

B. It Helpdesk View:

```
CREATE VIEW VendoralllDetail AS

SELECT VendorID,SoftwareName,Version,Paymentstatus,Amount,PaymentDate

FROM Vendor

Inner Join software

ON Vendor.VendorId = Software.Vendor_VendorID

Inner Join Payments

ON Vendor.VendorId= payments.Vendor_VendorID;
```

C. Project Manager View:

```
CREATE VIEW EmployeewholeDetail AS

SELECT Firstname,lastname ,SSN,Type,Address,City,State,
ZipCode,phoneNumber,Country,salary,ModifiedDate

FROM employee

Inner Join EmployeAddress

On EmployeAddress.employeeId = employee.employeeId

Inner Join Employeepaymenthistory

ON employee.employeeId= Employeepaymenthistory.employee_employeeId;
```

11. Procedures

A. Employee Shifts

```
CREATE PROCEDURE spSHIFTOFEMPLY()


SELECT Name, count(Employee_EmployeeID)

FROM department_has_employee

inner join shift
On shift.Id= department_has_employee.shift_ID

Group BY Name;
```

B. Employee Requests

```
Create Procedure spRequestEmployee()

Select RequestID, SoftwareID, SoftwareName, EmployeeID,
        SoftwareEngineer_EmployeeID, LastName,FirstName,
       EmployeeWorkStation_WorkStationID, RequestStatus

from request
inner join software on (request.Software_SoftwareID = software.SoftwareID )
inner join employee on (request.SoftwareEngineer_EmployeeID = employee.EmployeeID )

Where request.RequestStatus = 'Open';
```

C. Software Request count

```
Create Procedure spSoftwareRequestCount()

Select  VendorID, Software_SoftwareID, SoftwareName,NumberOfLicences,
       count(Software_SoftwareID) as 'Used Licenses'
from software
inner join vendor on (software.Vendor_VendorID = vendor.VendorID)
inner join request on (software.SoftwareID = request.Software_SoftwareID)
Group By Software_SoftwareID;
```

12. Triggers:

A. Delete Trigger

```
CREATE TABLE deletedVendorAddress
(
    VendorID INT,
    Vendoraddress Varchar(45) NOT NULL,
    VendorCity Varchar(10) NOT NULL,
    VendorState Varchar(10) NOT NULL,
    VendorCountry Varchar(10) NOT NULL,
    changedon datetime DEFAULT NULL
);

USE Data;
Delimiter $$

CREATE TRIGGER After_delete_Address_Details
AFTER DELETE ON VendorAddress
FOR EACH ROW
BEGIN
INSERT INTO deletedVendorAddress VALUES
(OLD.VendorID,
OLD.Vendoraddress,
OLD.VendorCity,
OLD.VendorState,
OLD.Vendorcountry,NOW());
END$$
Delimiter ;
```

B. Update Trigger

```
Use Data;

Create Table Employee_update
(
  id int(10) Not Null auto_increment,
  EmployeeID  int not null,
  ssn int not null,
  lastName varchar(30) not null,
  firstName varchar(30) not null,
  birthdate date not null,
  hiredate date not null,
  changedon datetime Default null,
  action varchar(10) Default null,
  Primary Key (id)
);

Drop table employee_update;

Use Data;
Delimiter $$

Create trigger before_Employee_update
Before Update on employee
For each row begin
Insert into employee_update
      Set action = 'update',
      EmployeeID = old.EmployeeID,
      ssn = old.ssn,
      lastName = old.LastName,
      firstName = old.FirstName,
      birthdate = old.birthdate,
      hireDate = old.hireDate,
      changedon = Now();
END$$
Delimiter ;
```

C. Insert Trigger

```
SELECT * from data.employee;

USE data;

CREATE TABLE Sala12
(
  employee_employeeID int(11) NOT NULL,
  Salary INT
);

SELECT * FROM Sala12;
DELIMITER $$

CREATE TRIGGER acts_INSERT_min_salary
BEFORE INSERT ON EmployeePaymentHistory
FOR EACH ROW
BEGIN
DECLARE msg VARCHAR(255);
IF NEW.salary <
(   SELECT salary
    FROM  EmployeePaymentHistory
    WHERE employee_employeeID = NEW.employee_employeeID
)
THEN
SET msg = 'Violation of Minimum Actor Salary.' ;
SIGNAL SQLSTATE '45000' SET message_text = msg ;
END IF;

END$$
DELIMITER ;
```

Thank You