**Software Engineering (CSC 4350)**
**Fall 2020 Semester**
**The Techonauts**
Bonnie Atelsek
Kimberly Luviano-Garcia
Matthew Kabat
Terrence Gaines
Nathan Murzello
09/24/2020

**Section 1:**

| Assignee Name | Email | Task | Duration (Hours) | Dependency | Due Date | Evaluation |
|---|---|---|---|---|---|---|
| Bonnie Atelsek | batelsek1@student.gsu.edu | Revising the problem statement & Use case diagrams | 2 hours | None | 9/23/2020 | 100%, completed work on time |
| Terrence Gaines | tgaines6@student.gsu.edu | Database Specification & Analysis | 5 1/2 hours | None | 9/23/2020 | 100%,completed work on time |
| Nathan Murzello | nmurzello1@student.gsu.edu | Database Specification & Analysis | 4 hours | None | 9/23/2020 | 100%,completed work on time |
| Matthew Kabat | mkabat1@student.gsu.edu | Defining the use cases and requirements | 2 hours | Use cases need to be defined before diagrams | 9/23/2020 | 100%,completed work on time |
| Kimberly Luviano-Garcia (coordinator) | kluvianogarcia1@student.gsu.edu | Making the assignment table & writing the report | 1.5 hours | None | 9/23/2020 | 100%,completed work on time |

**Section 2: Problem Statement**

1. What is your product, on a high level?

    a. Our product provides an easy, streamlined way to create and store employee evaluations and automatically use employees' evaluation statistics to calculate standardized salary bonuses.

2. Who is it for?

    a. Our product is designed for companies with multiple employees and a regular evaluating system, for example, quarterly or biannually. Since the system can store a range of employees and automatically calculate their bonuses, this makes it ideal for growing companies who might not have preexisting digitized evaluation systems who are looking to standardize their bonus system and streamline the evaluation process.

3. What problem does it solve?

    a. Our product addresses the problem of standardizing bonuses. The system evaluates all employees on the same criteria and automatically generates their bonus amount based on their cumulative evaluation statistics, thus achieving the goal of a truly equitable compensation system, removed from mistakes or discrimination. The simplicity not only makes it easier for managers to monitor and evaluate their employees, but assures that all employees will receive fair treatment.

4. What alternatives are available?

    a. Small companies often opt for hard copies of employee files and evaluations, but this management system is not efficiently scalable. Outsourcing employee evaluations to external HR management companies is another common solution, but is often costly and lacks the responsiveness that today's companies require. There are also online HR software vendors, like BambooHR, Lattice, or ClearCompany, but these systems can be confusing and non-transparent to the wide variety of companies that need this technology.

5. Why is this project compelling and worth developing?

    a. Workplace culture surrounding salary and bonuses is often fraught with social stigma and the potential for discrimination. Our system removes those roadblocks and allows for a workplace that can operate on transparency and equality. It simplifies the

experience for all users involved, and will thus allow for greater productivity and communication.

6. Describe the top-level objectives, differentiators, target customers, and scope of your product.

    a. Our top level objectives are to create a system which will allow employers to create, store, and view employee evaluations for each registered employee, and which will also automatically calculate and display employee bonus amounts on their profiles based on their past evaluation statistics.

    b. Our product is differentiated by its aim and its design. Unlike many other available systems, we seek to provide a streamlined and entirely fair evaluation experience. By automatically calculating bonuses, we both take that responsibility off of the shoulders of the managers we are marketing towards, and assure impartiality for the associated employees. Our system, unlike most alternatives, is a guarantee of equality and ease of use.

    c. Our target customers are mid-size and growing companies looking to standardize and digitize their evaluation system.

    d. The scope of our project outlines a database to store employee profiles and their past evaluations, a website to both allow the entry of new evaluations (giving prompted numerical rankings on aspects of job performance, ex. punctuality, work ethic, etc.) and to calculate and display salary bonuses based on the aggregated employee evaluation statistics.

7. What are the competitors and what is novel in your approach?

    a. While our system does have competitors, such as Lattice and ClearCompany, we are concerned largely with the idea of workplace equity. By taking the actual calculation of the bonuses out of the hands of the user, we assure that all employees will receive the same treatment. This makes our system easier for the employer and fairer to the employees. We allow access to all past evaluations, meaning that our system is entirely transparent, which removes the potential for discrimination accusations, and even more importantly, drastically reduces the likelihood of monetary discrimination.

8. Make it clear that the system can be built, making good use of the available resources and technology.

    a. Free database technology is easily available and usable, as are free website development tools and web hosting, which means that our website is feasibly

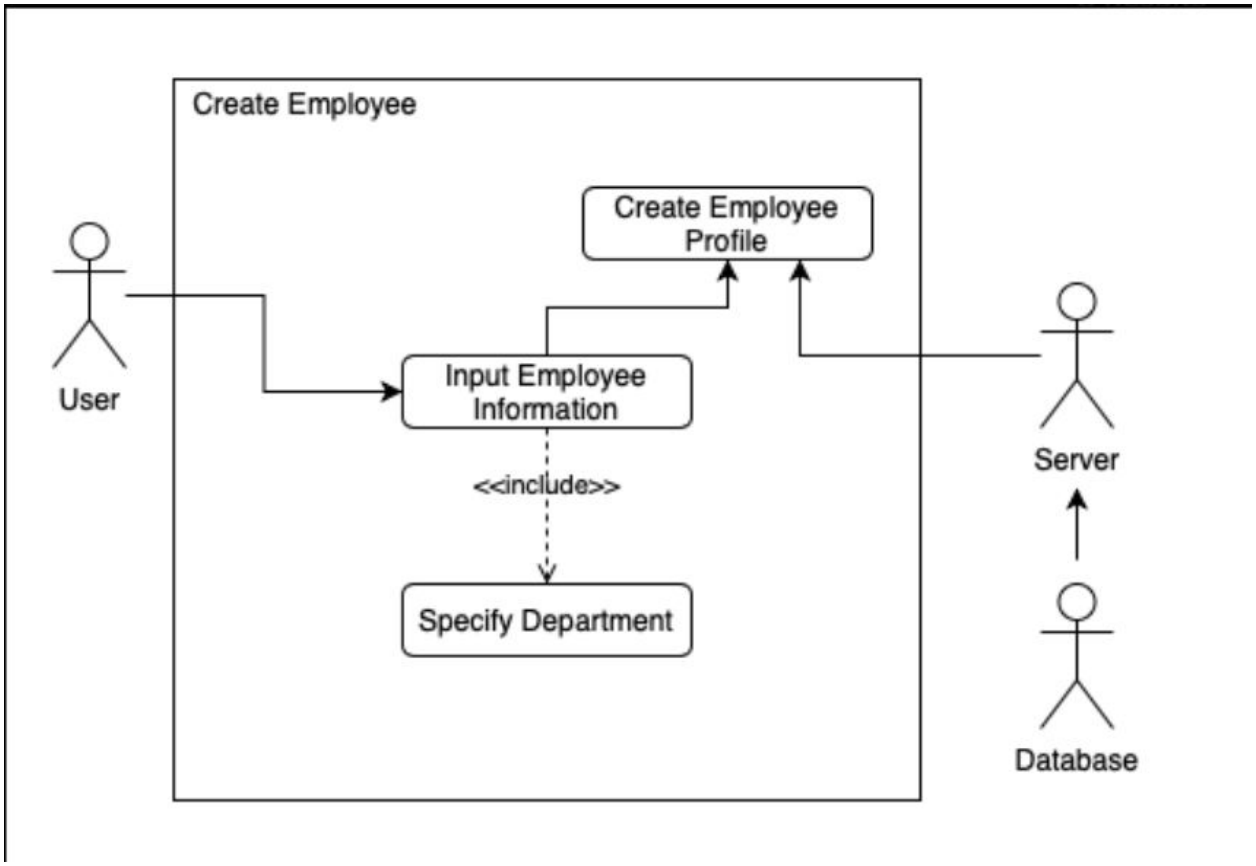
buildable for a team of five. We plan to use MS SQL for our database. Our system needs to be built for the reasons discussed in Items 7 and 5.

9. What is interesting about this project from a technical point of view?

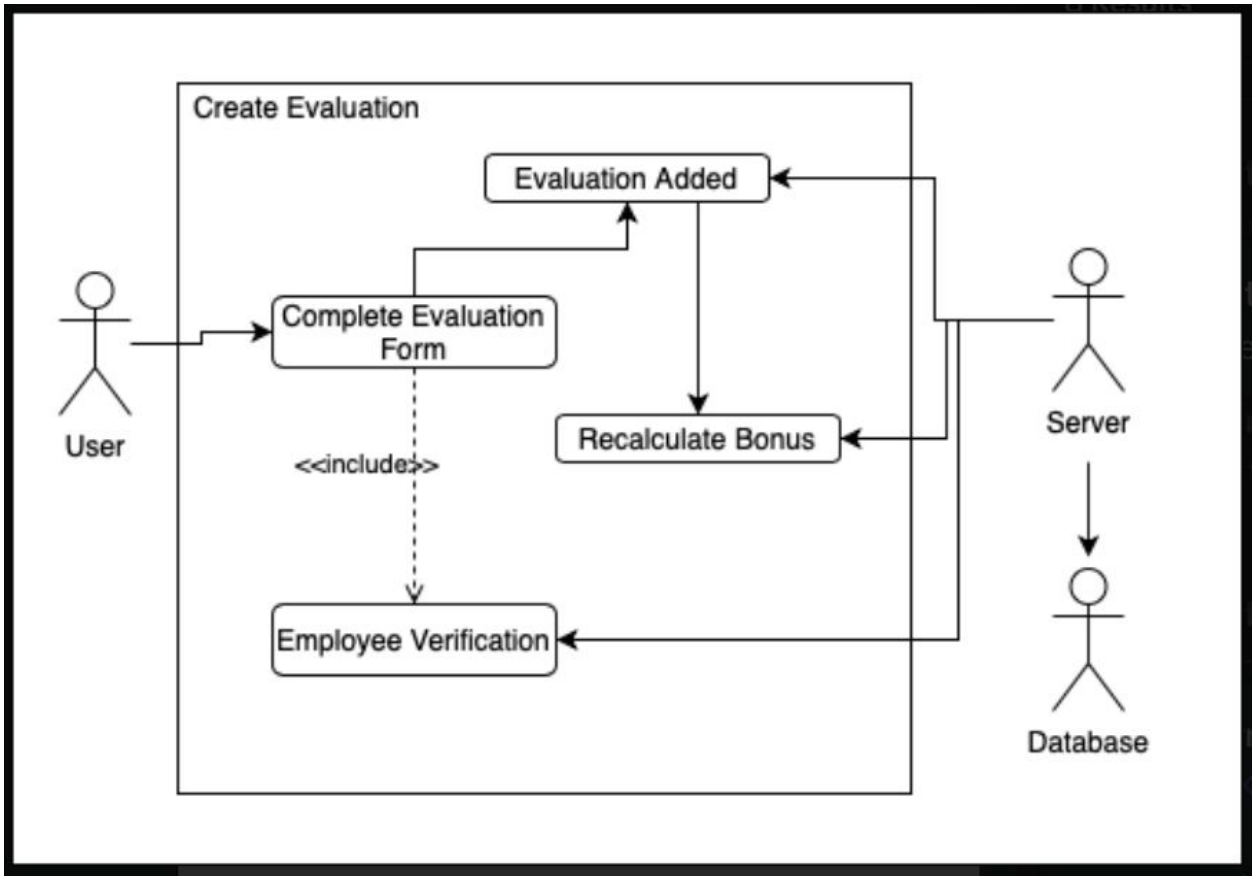
   a. We will need to design an algorithm weighing the different aspects of the evaluation, and make sure that it can be effectively applied universally, regardless of employee differences such as the length of their tenure at the company. We'll also need to assure that our system is (above all) secure, in order to maintain our mission statement of providing discrimination/tamper free evaluations, which will require some outside research and some creative problem solving to accommodate multiple users.

**Section 3: System Requirements**

Use Case #4 Diagram



Use Case 8 Diagram



# Use Cases

**Use Case no.:** 1

**Use Case Name:** Create Department

**Actors:** User, Database, Server

**Description:** A user can create a department which has the following properties: department number, department name, and location. Employees will be added to one of these departments when first made.

**Exception Path:** If the user tries to create a database with a duplicate number or name, they will be told to select a different number/name. They will also be told to fill all fields should they attempt to submit a unfinished form.

**Alternate Path: None.**

**Pre-condition: A department with the same number or name cannot already exist.**

**Post-condition: A new department has been inserted into the database after pre-conditions were met.**

**Requirement number: 1**

**Use Case number: 1**

**Introduction: Create a department.**

**Inputs: A department name, number, and location.**

**Requirements Description: The provided name and number must be unique.**

**Outputs: A department with the provided attributes.**

**Use Case no.: 2**

**Use Case Name: Edit Department**

**Actors: User, Database, Server**

**Description: A user can edit the properties of a department.**

**Exception Path: If the user tries to modify a database by giving it a duplicate number or name, they will be told to select a different number/name. They will also be told to fill all fields should they attempt to submit a unfinished form.**

**Alternate Path: None.**

**Pre-condition: A department must first exist for this feature to be used. A department cannot be given the same name or number as another department.**

**Post-condition: The chosen department in the database was modified as per the user's specification after pre-conditions were met.**

**Requirement number: 2**

**Use Case number: 2**

**Introduction: Edit an existing department's attributes.**

**Inputs: An existing department.**

**Requirements Description: A department must exist, and the name and location must not be the same as another department's after being edited.**

**Outputs: A department with its attributes modified to what the user specified.**

**Use Case no.: 3**

**Use Case Name: Delete Department**

**Actors: User, Database, Server**

**Description: A user can remove a department.**

**Exception Path: If a user tries to delete a department that has employees assigned to it, then they will be told to first reassign all employees in the department before deleting it.**

**Alternate Path: None**

**Pre-condition: A department must first exist for this feature to be used. There cannot be any employees registered to the department when it is deleted.**

**Post-condition: The chosen department has been removed from the database after pre-conditions were met.**

**Requirement number: 3**

**Use Case number: 3**

**Introduction: Delete an existing department.**

**Inputs: An existing department.**

**Requirements Description: The selected department cannot be deleted if it contains any employees.**

**Outputs: Nothing.**

**Use Case no.: 4**

**Use Case Name: Create Employee**

**Actors: User, Database, Server**

**Description: A supervisor can input general information about an employee. This information is stored within a database specific to the application. The info takes the form of biographical and work-related information such as their department.**

**Exception Path: If the user fails to fill in all mandatory fields such as name and department, then they will be required to do so before the new employee is accepted.**

**Alternate Path: None.**

**Pre-condition: A department must exist for the employee to be assigned to.**

**Post-condition: A new employee has been inserted into the database after pre-conditions were met.**

**Requirement number: 4**

**Use Case number: 4**

**Introduction: Create an employee.**

**Inputs: A department and various bits of work-related and biographical information about an employee.**

**Requirements Description: A department must exist for the employee to be assigned to.**

**Outputs: An employee with the specified attributes.**

**Use Case no.: 5**

**Use Case Name: Edit Employee**

**Actors: User, Database, Server**

**Description: A user can modify an employee's file, changing things such as their biographical info or moving them to another department.**

**Exception Path: If a user attempts to submit a modified form that is missing mandatory fields, then they will be required to fill those fields before the form will be accepted.**

**Alternate Path: None**

**Pre-condition: An employee must first exist for this feature to be used.**

**Post-condition: The chosen employee in the database was modified as per the user's specification after pre-conditions were met.**

**Requirement number: 5**

**Use Case number: 5**

**Introduction: Modifies an employee to take on the provided attributes.**

**Inputs: A selected employee and any changed attributes**

**Requirements Description: An employee must exist to be modified.**

**Outputs: A modified employee.**

**Use Case no.: 6**

**Use Case Name: Delete Employee**

**Actors: User, Database, Server**

**Description: A user can remove an employee from the database. This will also remove the employee from their department records as well as removing any associated evaluations.**

**Exception Path: None.**

**Alternate Path: None.**

**Pre-condition: An employee must first exist for this feature to be used.**

**Post-condition: The chosen employee has been removed from the database after pre-conditions were met.**

**Requirement number: 6**

**Use Case number: 6**

**Introduction: Deletes the selected employee.**

**Inputs: A selected employee.**

**Requirements Description: An employee must exist to be deleted.**

**Outputs: Nothing.**

**Use Case no.: 7**

**Use Case Name: Search for Employee**

**Actors: User, Database, Server**

**Description: A user can search for an employee by Employee ID or department name.**

**Exception Path: None.**

**Alternate Path: None.**

**Pre-condition: None.**

**Post-condition: Search results were displayed.**

**Requirement number: 7**

**Use Case number: 7**

**Introduction: Searches for an employee using Employee ID or department name.**

**Inputs: Employee ID or department name**

**Requirements Description: None.**

**Outputs: A list of employees who match the search criteria.**

**Use Case no.: 8**

**Use Case Name: Input Evaluation**

**Actors: User, Database, Server**

**Description: A user can input performance information for an employee in the form of an evaluation. This information is stored within a database specific to the application. The info takes the form of several numerical values that each represent how well the employee has met a certain standard.**

**Exception Path: If the user fails to fill in all fields, then they will be required to do so before the new evaluation is accepted.**

**Alternate Path: None.**

**Pre-condition: An employee must be selected. The employee must have been created beforehand.**

**Post-condition: A new evaluation tied to the chosen employee has been added to the database after pre-conditions were met.**

**Requirement number: 8**

**Use Case number: 8**

**Introduction: Submit an evaluation for a selected employee.**

**Inputs: A selected employee and an evaluation.**

**Requirements Description: An employee must exist.**

**Outputs: An evaluation attached to the selected employee.**

**Use Case no.: 9**

**Use Case Name: View Evaluations**

**Actors: User, Database, Server**

**Description: A user can view the evaluations registered to an employee.**

**Exception Path: None.**

**Alternate Path: None.**

**Pre-condition: An employee must first exist and then must be selected.**

**Post-condition: The list of evaluations for the chosen employee was displayed after pre-conditions were met.**

**Requirement number: 9**

**Use Case number: 9**

**Introduction: Displays the evaluations for a selected employee.**

**Inputs: An employee.**

**Requirements Description: An employee must exist to be selected.**

**Outputs: The selected employee's evaluations.**

**Use Case no.: 10**

**Use Case Name: View Bonus**

**Actors: User, Database, Server**

**Description: A user can view the generated bonus for an employee. This number will have been calculated from that employee's stored evaluations.**

**Exception Path: None.**

**Alternate Path: None.**

**Pre-condition: An employee must first exist and then must be selected.**

**Post-condition: The bonus for the chosen employee was displayed after pre-conditions were met.**

**Requirement number: 10**

**Use Case number: 10**

**Introduction: Displays an employee's bonus.**

**Inputs: An employee.**

**Requirements Description: An employee must exist to be selected.**

**Outputs: The selected employee's bonus.**

**Use Case no.: 11**

**Use Case Name: Create Complaint**

**Actors: User, Database, Server**

**Description: A user can register a complaint to an employee.**

**Exception Path: If the user fails to fill in all mandatory fields, then they will be required to do so before the new complaint is accepted.**

**Alternate Path: None.**

**Pre-condition: An employee must be selected. The employee must have been created beforehand.**

**Post-condition: A new complaint attached to the selected employee was added to the database after pre-conditions were met.**

**Requirement number: 11**

**Use Case number: 11**

**Introduction: Creates a complaint that is registered to the selected employee.**

**Inputs: An employee and a complaint.**

**Requirements Description: An employee must exist to be selected.**

**Outputs: A complaint that is registered to a selected employee.**

**Use Case no.: 12**

**Use Case Name: Edit Complaint**

**Actors: User, Database, Server**

**Description: A user can modify a complaint that has been registered to an employee.**

**Exception Path: If a user attempts to submit a modified form that is missing mandatory fields, then they will be required to fill those fields before the form will be accepted.**

**Alternate Path: None.**

**Pre-condition: An employee and a complaint must be selected. Both the employee and the complaint must have been created beforehand.**

**Post-condition: The chosen complaint in the database was modified as per the user's specification after pre-conditions were met.**

**Requirement number: 12**

**Use Case number: 12**

**Introduction: Modifies an existing complaint that has been registered to an employee.**

**Inputs: A complaint, and by extension, a user.**

**Requirements Description: Both a complaint and a user must exist to be selected.**

**Outputs: A complaint that has been modified as per the user's specifications.**

**Use Case no.: 13**

**Use Case Name: Delete Complaint**

**Actors: User, Database, Server**

**Description: A user can delete a complaint that has been registered to an employee.**

**Exception Path: None.**

**Alternate Path: None.**

**Pre-condition: An employee and a complaint must be selected. Both the employee and the complaint must have been created beforehand.**

**Post-condition: The specified complaint was removed from the database after pre-conditions were met.**

**Requirement number: 13**

**Use Case number: 13**

**Introduction: Deletes an existing complaint that has been registered to an employee.**

**Inputs: A complaint, and by extension, a user.**

**Requirements Description: Both a complaint and a user must exist to be selected.**

**Outputs: Nothing.**

## SECTION 4:

Database Specification

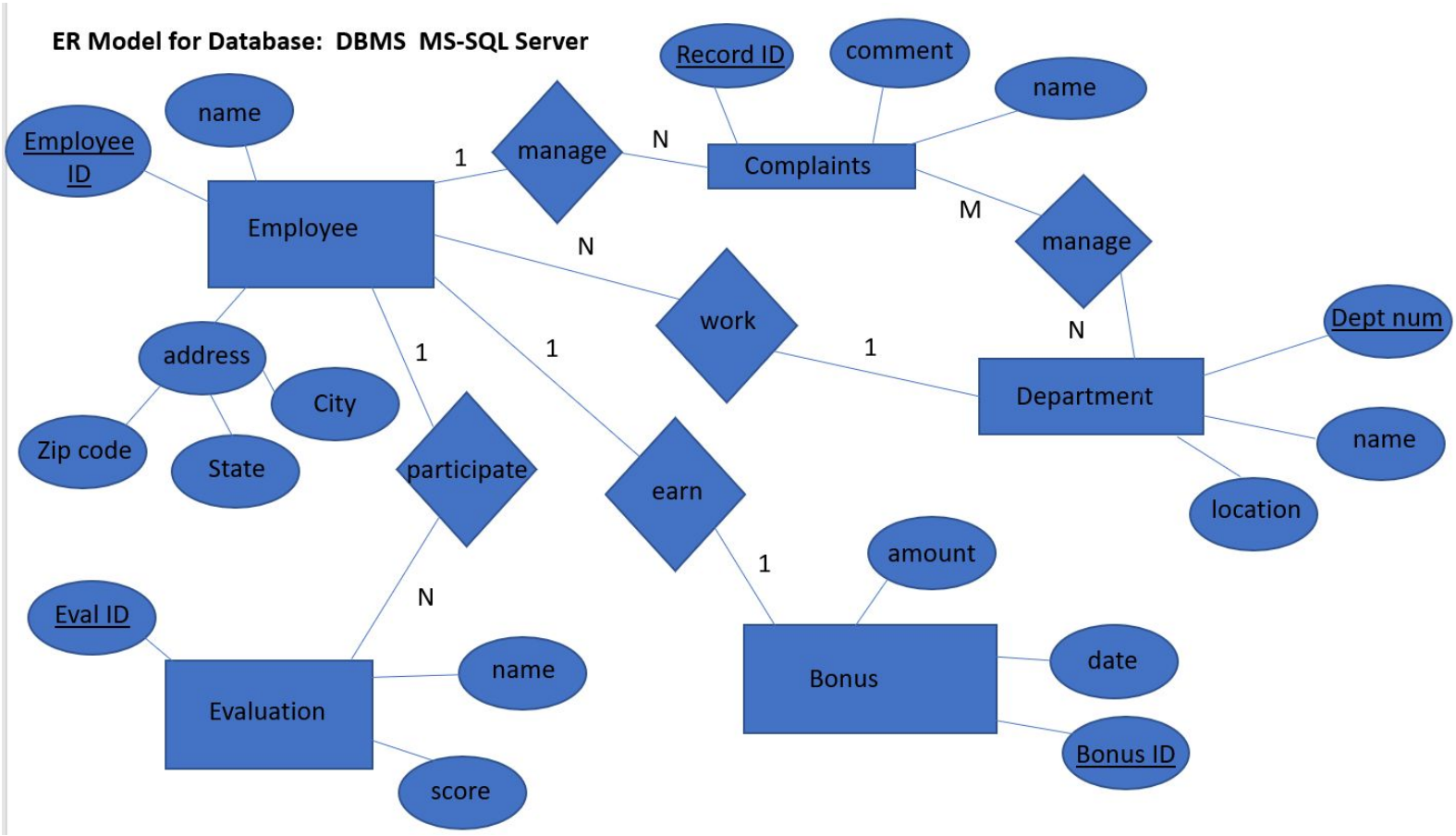
Database Management System: MS-SQL Server

**Parent Tables:**

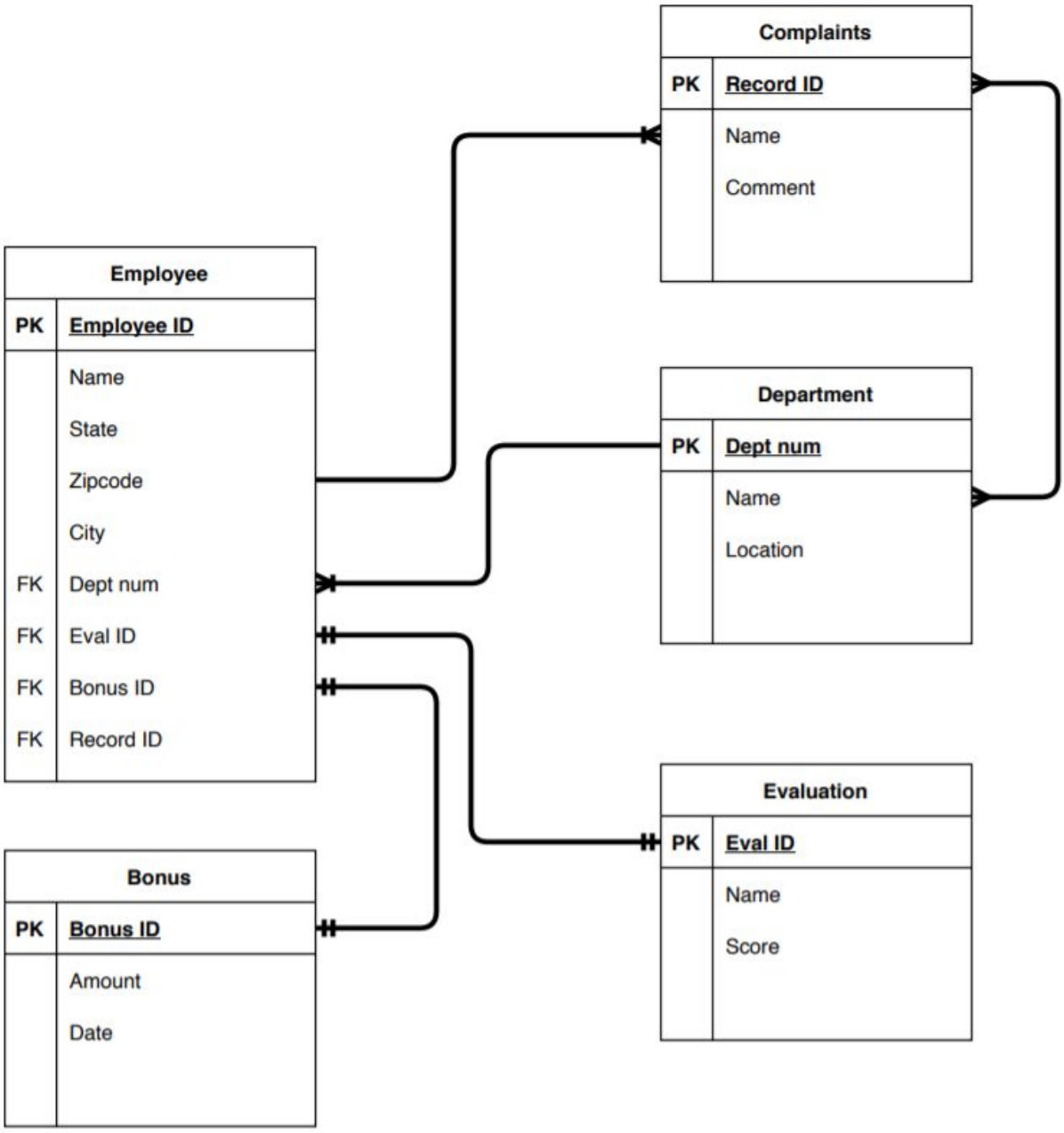
1. Employee: Employee ID(PK), Name, State, Zip Code, Dept Num (FK), Eval ID (FK), Bonus ID (FK)

**Child Tables:**

1. Department: Dept Num (PK), Name, Location
2. Evaluation: Eval ID(PK), Name, Score
3. Bonus: Bonus ID(PK), Amount, Date
4. Complaints: Record ID (PK), comment, name


ER Model for Database: DBMS MS-SQL Server
Record ID
comment
name
name
Employee ID
manage
1
N
Complaints
Employee
M
manage
N
work
N
1
1
1
Dept num
address
City
Zip code
State
participate
earn
Department
name
location
amount
1
Eval ID
N
name
Evaluation
Bonus
date
Bonus ID
score

# Relations Schema



# Data Dictionary

| Table | Attribute | Data Type | Primary Key | Foreign Key | Constraints |
|---|---|---|---|---|---|
| EMPLOYEE | Employee ID | VARCHAR(25) | YES | | Unique |
| EMPLOYEE | Fname | VARCHAR(25) | | | |
| EMPLOYEE | Lname | VARCHAR(25) | | | |
| EMPLOYEE | Address | VARCHAR(100) | | | |
| EMPLOYEE | City | VARCHAR(100) | | | |
| EMPLOYEE | State | VARCHAR(100) | | | |
| EMPLOYEE | Zip | VARCHAR(100) | | | |
| EMPLOYEE | Salary | INT | | | >0 |
| EMPLOYEE | Dept_Name | VARCHAR(100) | | | |
| DEPARTMENT | Dname | VARCHAR(25) | | | Unique |

| | | | | | |
|---|---|---|---|---|---|
| DEPARTMENT | Dnumber | INT | YES | | Positive |
| DEPARTMENT | Location | INT | | | |
| BONUS | Bonus ID | INT | YES | | <current date |
| BONUS | Amount | INT | | | |
| BONUS | Date | DATE | | | |
| EVALUATION | Eval_ID | INT | YES | | |
| EVALUATION | Name | VARCHAR(100) | | | |
| EVALUATION | Score | INT | | | |
| COMPLAINTS | Record_ID | INT | YES | | |
| COMPLAINTS | Name | VARCHAR(100) | | | |
| COMPLAINTS | Comment | VARCHAR(100) | | | |

Appendix: GitHub Screenshots

Link:https://github.com/nathanMurzello/Techonauts