Milestone 1
Group 1

Team Members:
Preyansh Kotecha, Divyaraj Bakrola, Jesus Garnica
, Michael Gilbert, Timothy Chan, Anne Lanaza

1. Executive Summary:

Managing payroll has always been a time-consuming job. Companies are spending valuable time adding, editing, and retrieving employee payroll information on massive spreadsheets. Massive spreadsheets are inefficient and an eyesore. Imagine, what if your company can save those valuable time for other things, for great things. Now your company can. Save valuable time with our “Advanced Payroll Management System”. APMS is a seamless system that will handle the boring job of shift managing for you.

Our application will allow a company to manage the wages given out to the salary paid employees. The APMS is based on the scope of the features we plan to have for the database. It will allow us to get a more in-depth understanding of the relationship between items.

There will be a main page intended for employees in order for them to be able to punch in and punch out when they enter and leave work. It will also allow them to leave for lunch and punch back in after lunch. There will also have to a sysadmin page so managers can log in, create employees, and set wages. The system will also allow managers to delete employees. It is much easier to keep track of the record of punch in and out times with an online database rather than manually writing down the times.

The database will handy for keeping track of all the numerous employees of a company or business. It will be able to readily handle updates such as the inclusion of new employees as well as the removal of former employees without creating issues for the rest of the data that is stored. For example, we will have a list of current employees but also of former employees and keep a list logged of their previous shifts. Our database has a sort of unique problem of not necessarily deleting all data when an employee is removed because that information might be useful to keep for a certain amount of time.

The importance of this project is to demonstrate and utilize the concepts learned in the class, and apply it to our database project that has real-life application. It will also help us enhance our abilities to work with others, wherein through the process we can learn from each other.

2. Entities:

Employee: An employee can be salaried or waged. They could fulfill many roles such as a manager, supervisor, and CEO. They will also have other attributes that are useful to keep track off, such as, total pay, class they belong to, amount of PTO, hours worked, and overtime pay. It also includes if the employee is still part of the company.

The database will save information about them even if they have quit for archiving purposes.

Class: This entity works as which class an employee belongs to like how a department functions. Depending on which class, it has details like the base pay, bonus pay, required amount hours of work that is expected from its employee.

Salary: This entity shows all the critical information about the salary paid to the employee. It already indicates the total amount of wage paid. It holds data such as bank account, routing number, tax rate deductions, and the name of the employee.

PTO (Paid time off): This has all the details about the PTO that an employee has. It works as a record for every PTO an employee claims from the company. It contains attributes, like start date, end date, and the reason for leave.

Shift: A shift will contain multiple aspects such as punch in time, punch out time, lunch punch out and lunch punch in. A shift belongs to any employee. Salaried employees do not need to keep track of their time but hourly employees must. It depends on the business owner on how they would like to keep track of their employees. An employee may have thousands of shifts attached to them but they do not need one immediately at the creation of the employee.

Payment: A payment is given to an employee on a certain schedule. It includes date issued, shift time frame, total payment, federal taxes withheld, state taxes withheld, and Medicare taxes withheld.

3. Business Rules:

Employee:

1. An employee must belong to only one class and all class must have at least one employee.
2. An employee can claim zero to many PTO's.
3. An employee can be paid by salary.

Shift:

1. Shift must belong to only one employee.

Payment:

1. Payment is given to every employee.

4. Initial list of functional requirements

Entity: Employee
Attributes:

- EID (PK)
- CID (FK)
- SID (FK)
- PID(FK)
- Name (composite) - VARCHAR
- Address (multi-value) - VARCHAR
- PTO Left - INT
- Salaried - Boolean

Entity: Class
Attributes:

- CID (PK)
- EID (FK)
- Insurance Type - VARCHAR
- Work Time Cap - INT
- Base Pay - INT
- PTO Initial Value - INT

Entity: Salary
Attributes:

- SID (PK)
- EID (FK)
- Bank Account - CHAR
- Over time - INT
- Amount - INT
- Bonus Pay - INT
- Hours Worked - INT

Entity: PTO
Attributes:

- PID (PK)
- EID (FK)
- Reason - VARCHAR
- Start date - VARCHAR
- End date - VARCHAR

- Value - INT

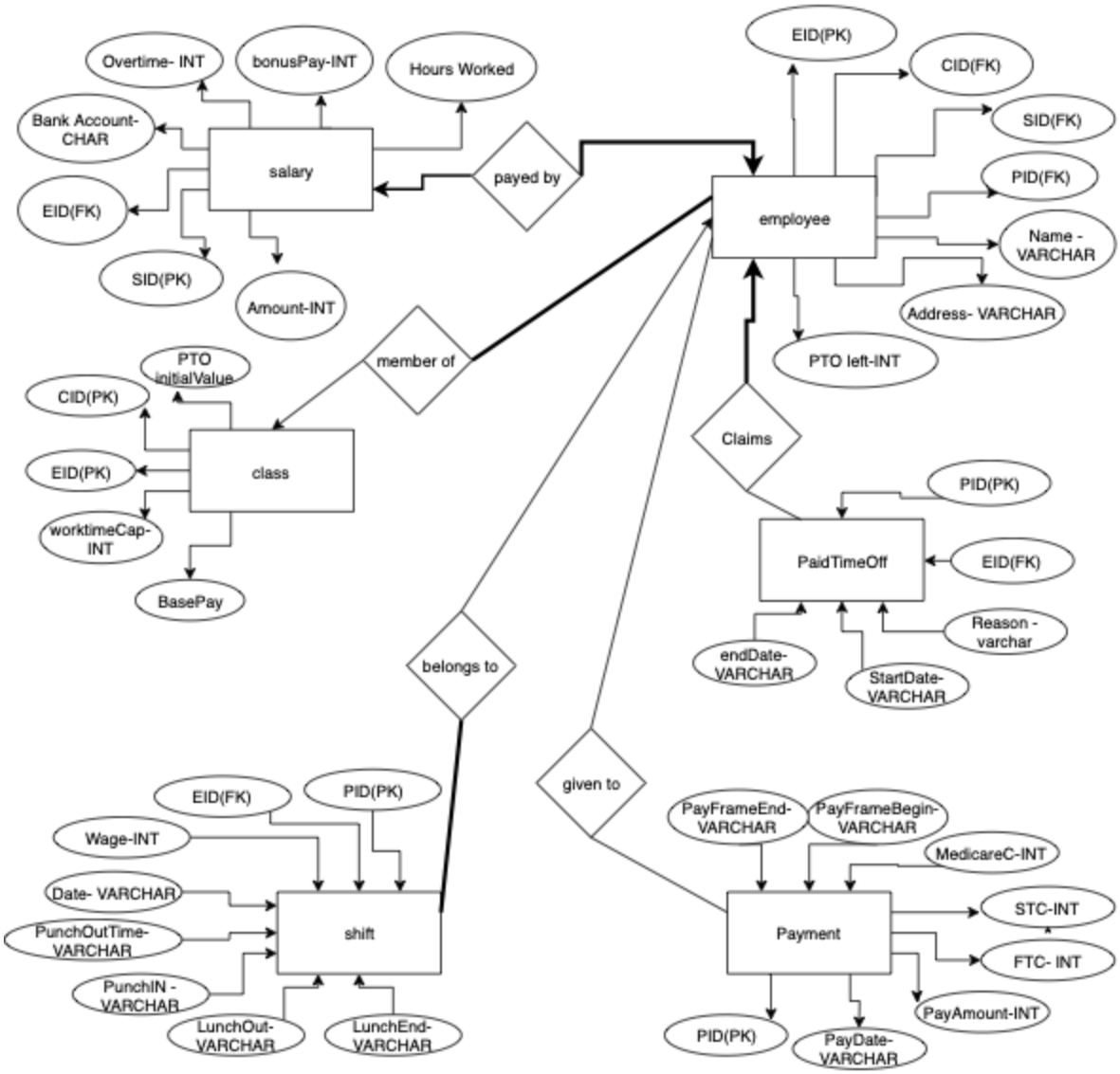
Entity: Shift
Attributes:

- PID (PK)
- EID (FK)
- Wage - INT
- date - VARCHAR
- PunchInTime - VARCHAR
- PunchOutTime - VARCHAR
- LunchBeginTime - VARCHAR
- LunchEndTime - VARCHAR

Entity: Payment
Attributes:

- PID(PK)
- PaymentDate - VARCHAR
- PaymentAmount - Int
- FederalTaxesCollected - Int
- StateTaxesCollected - Int
- MedicareCollected - Int
- PaymentFrameBegin - VARCHAR
- PaymentFrameEnd - VARCHAR

5. Entity Relationship Diagrams (ERD):



6. ERDs Test:

Business Rule	Entity 1	Relationship	Type Rel	Entity 2	Pass/Fail	Modify
1	Employee	Member of	M to 1	Class	Fail	Add Participation constraint
2	Employee	Claims	1 to M	PTO	Pass	
3	Employee	Paid by	1 to 1	Salary	Fail	Add Participation constraint
4	Shift	Belong to	M to 1	Employee	Pass	
5	Payment	Given to	1 to 1	Employee	Fail	Add Participation

7. Initial list of NON-functional requirements:

- The database should be able to load any shift within 5 seconds or less.
- It should be able to handle adding hundreds of employees for each instance.
- The database should be able to handle a nearly unlimited amount of shifts that are cleared on a yearly basis for the sake of space.
- It should able to scale from just a few users to hundreds.
- Data should not be overridden ever while another function is accessing that resource.
- The data should be absolutely clearly organized based on the role it plays in the database.
- Data should be easy to modify without breaking the rest of the database.
- There should not be a limit on entities which do not require them.
- Feedback on the recording of the data should be quick to the user.
- The front-end portion of the database should be easy to use to use and read although it can be minimal.

- The database should not go down more than 3 times a week for users.
- A database error should not bring down the entire database and cause issues for the other users.

8. Enumerate the work done by each team member

1. Preyansh Kotecha (team lead) -8
2. Timothy Chan (git master) - 9
3. Anne Lanaza - 9
4. Jesus Garnica - 10
5. Michael Gilbert - 9
6. Divyaraj Bakrola - 6