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 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: The employee will be a waged employee. They could fulfill many positions such as a manager, supervisor, and CEO. They will also have other attributes that are useful to keep track off, such as, name, their position, amount of PTO left, and pay increase. Pay Increase is the amount over the base the employee receives. 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.

Position: This entity tells what position the employee has. Depending on the position, this contains the title of the position, their hourly wage and amount of PTO they can claim annually. The position can be vacant on occasions such as someone gets fired, leaves or the position has just been created.

PTO (Paid time off): This entity works out as a record and basically tells us the hours of PTO the employee has taken. Although, not all employees can get a PTO like employees who are interns and volunteers or depending on the owner's decision.

Shift: A shift will contain multiple aspects such as date of shift, punch-in time, punch-out time, and breakout time. A shift belongs to any employee. 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:

1. Employees must receive payment(wrong)
2. An employee must be designated to only one position.
3. Multiple employees can be assigned to several shifts.
4. Each employee can claim only one PTO.

4. Initial list of functional requirements

Entity: Employee (Weak)
Relations: Designated to Position, Receives Payment, Claims PTO, Assigned to Shift
Attributes:

- Employee_ID (PK)
- Name
- position_ID
- Address
- Wage_Increase
- PTO Left

Entity: Position (Strong)
Relations: Designated to Employee
Attributes:

- Position_ID (PK)
- Position_Title
- Hourly Wage
- Hours of PTOs per year

Entity: PTO (Weak)
Relations: Claimed by Employee
Attributes:

- PTO_ID (PK)
- StartDate
- EndDate
- Reason

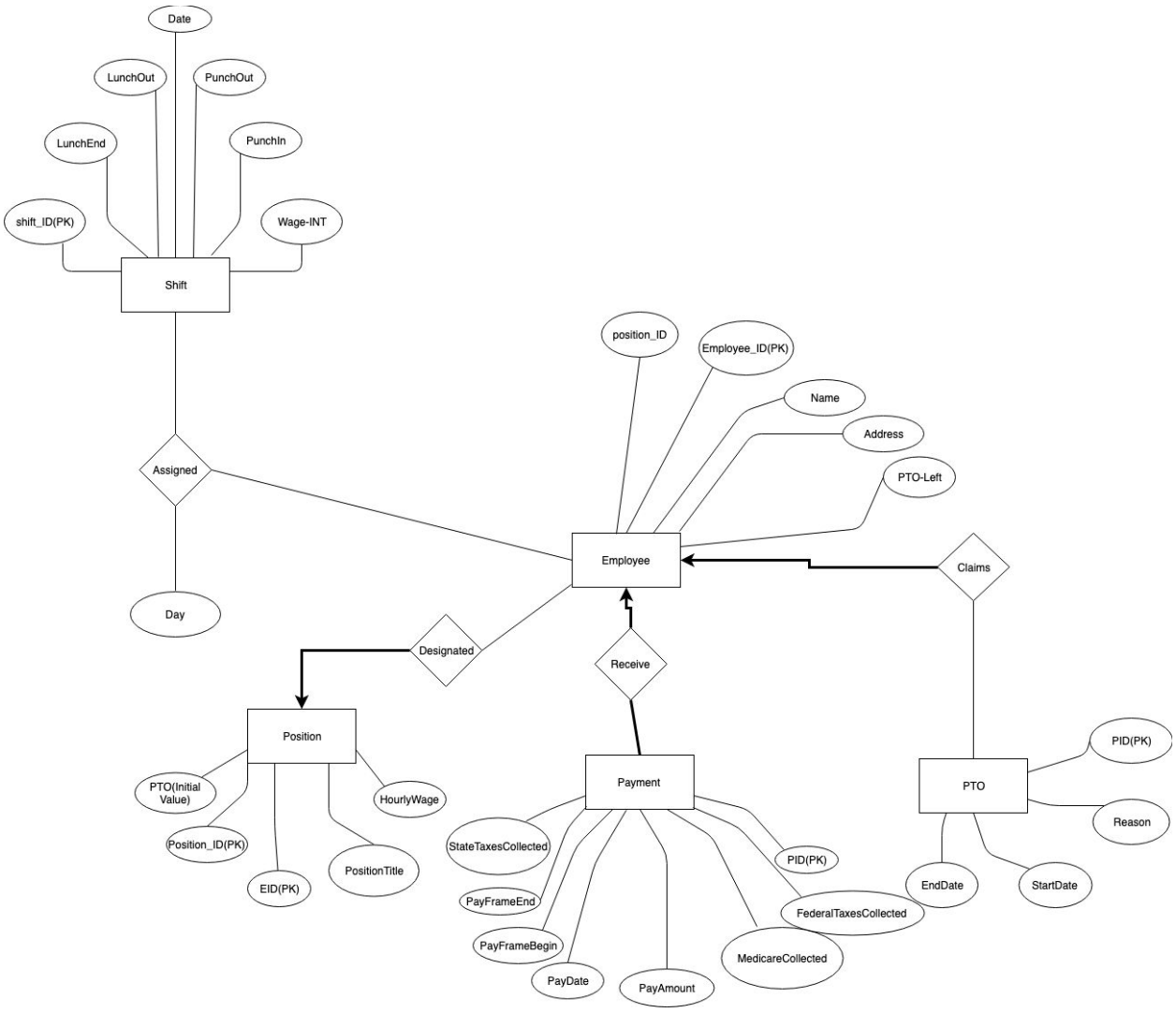
Entity: Shift (Strong)
Relations: Assigned to Employee
Attributes:

- shift_ID (PK)
- Employee_ID
- Date
- PunchInTime
- Punch Out Time
- Lunch BeginTime
- LunchEndTime

Entity: Payment (Weak)
Relations: Received by Employee
Attributes:

- PID(PK)
- PayDate
- PayAmount
- FederalTaxesCollected
- StateTaxesCollected
- MedicareCollected
- PaymentFrameBegin
- PaymentFrameEnd

5. Entity Relationship Diagrams (ERD):



6. ERDs Test:

Business Rule	Entity 1	Relationship	Type Rel	Entity 2	Pass/Fail	Modify
1	Employee	Receives	1 to M	Payment	Pass	
2	Payment	Received by	M to 1	Employee	Fail	Add constraint
3	Shift	Assigned to	M to M	Employee	Pass	
4	Employee	Designate to	M to 1	Position	Fail	Add

						constraint : an employee must have a position.
5	**Employee**	**Claims**	**1 to 1**	**PTO**	**Pass**	
6	**Employee**	**Assign**	**M to M**	**Shift**	**Pass**	
7	**Position**	**Designate to**	**1 to M**	**Employee**	**Pass**	
8	**PTO**	**Claimed by**	**1 to 1**	**Employee**	**Fail**	**Add constraint**

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.
- MySQL 8.0 is going to be used as the DBMS.
- 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.
- The front-end portion of the database should be easy 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.
- We will aim the application to run on desktop computers.
- It will be programmed in Java 8.

8. Enumerate the work done by each team member

1. Preyansh Kotecha -8
2. Timothy Chan (git master) - 9
3. Anne Lanaza - 9
4. Jesus Garnica (Team Lead) - 9

5. Michael Gilbert - 9
6. Divyaraj Bakrola - 8