Task List

1. Tasks 1. Build a database.
    a. Sub-tasks 1. The users will need a database to store all the employees and their demographical, job history, and evaluation class.
        i. Create a Demographic class that includes:
            1. A name variable and add method
            2. Address variable and add method
            3. Phone variable and add method
            4. Emails variable and add method
            5. Emergency Contact/Insurance provider variable and add method
        ii. Create a array list JobHistory that contains the Job class info which includes
            1. Company name variable and add method
            2. Supervisor variable and add method
            3. Teammate names (if applicable) variable and add method
            4. Length on the job variable and add method
            5. Job title variable and add method
            6. Team roles variable and add method
        iii.
            1. Critical skills used/learned (preferably ones related to the applied job.)
            2. Level of each skill employee has. (beginner-expert) peer-reviewed.
            3. Soft skills developed at each job. (preferably ones team-orientated)
            4. Gifts the employee has been found to have. (Self-evaluated personal skills)
        iv. Create a Evaluation class that includes
            1. Evaluating supervisor
            2. Date of Eval
            3. The mental state of the employee (Commonly used emotional adjectives)
            4. Recommendations/notes on current job satisfaction, proficiency, and efficiency.
    b. Sub-tasks 2. The users will need a GUI interface to collect the data. Something the employees can interact with and put their information into.
        i.
    c. Sub-tasks 3. The users will need a way to save this on the interface.
        i. When saving data the information is first checked for any SQL injections to prevent attacks on vulnerable information. (Small businesses are often seen as vulnerable/easy targets so this is a good safety measure).
        ii. After checking the data it will be saved into the database and create a new record for that employee.

    d. Sub-tasks 4. Saved data will need to send an alert to the user that it has been correctly created and saved appropriately.
    e. Sub-tasks 5. Employees will need a button to clock in and clock out
2. Tasks 2. The user will need to be able to read a user from the database.
    a. Sub-tasks 1. A query at the data layer will be written to return specific employee demographics, job history, and evaluation tracking.
        i. Information to be returned.
    b. Sub-tasks 2. The user will need a search function in the GUI interface the user can use to select a specific user.
        i. The search function should be able to have a subfunction of which specific database they wish to access. Also, a user authentication level to make sure normal employees and anyone under a certain access level may not access employee information.
    c. Sub-tasks 2.1 The user should be able to filter employees and information using a specific data query type
    d. Sub-tasks 3. The search mechanism that is triggered to access the function (A search bar similar to Canvas/google should be sufficient)
        i. After a search is triggered it will ask if you wish to access a specific database or all information. Then that data will be displayed on the GUI
    e. Sub-tasks 4. In the case of no data or wrong access level then the user should receive a message explaining that the user does not exist or they do not have such access.
3. Task 3. The user must be able to edit the employee's information in the databases.
    a. Sub-task1. Need an edit button for the three databases on the GUI interface.
        i. When the button is clicked it will determine if you have the access level to make edits or if you're the account holder of the account you're trying to edit.
        ii. After it is checked the information is then sent through the database security check and then the database will be updated and edit the existing information in the database.
        iii. After any edits, it will autosave the new information and alert the user that the information has been edited and the changes have been saved.
        iv. All fields need to be filled in
4. Task 4. Deleting an employee from the database
    a. Sub-tasks 1. A delete button must be added to the GUI interface for each database.
    b. Sub-tasks 2. If the button is clicked it will check if you have the access to commit such changes.
    c. Sub-tasks 3. If the check goes through then a message will pop up asking for confirmation of the deletion.
    d. Sub-tasks 4. Once confirmed it will send a delete query to the specified or all databases to delete the record.

    e. Sub-tasks 5. If there was no problem in the deletion an alert will be sent to the user to inform them of its completion and remind them what exactly was deleted.
5. Task 5. The users will need to be able to log in/out and clock in/out to keep track of their time for timecard capabilities
    a. Sub-tasks 1. Creating an abstract Person class
        i. Create abstract methods and variables for logging in and out.
        ii. Need abstract methods and variables for punching in, taking a break, and capturing time for employees to create a reliable time card. Which when clicked will automatically share that info with the database.
6. Task 6. The users need a way to validate their new and old employees data going in and out
    a. Sub-tasks 1. Creating a private class for helper/data validation methods
        i. Create a validate phone number method and variable. That checks if it is a real phone number.
        ii. Sub-tasks 2. Create a validate email address method and variable. Make sure the email is from the company and follows guidelines
        iii. Sub-tasks 3. Create a validate password method and variable. Makes sure all password requirements are met and followed.
        iv. Sub-tasks 4. Create a save changes method and variable. Saves changes made to employee information such as changed password, etc.
        v. Sub-tasks 5. Create a SQL injection check method and variable. Checks the info to make sure it is not an injection before it gets sent anywhere else.
        vi. Sub-tasks 6. Create a check access level method and variable. Checks access level of employee when trying to perform tasks that require higher access.
        vii. Sub-tasks 7. Create a validate login method. Checks password, and email to make sure it belongs to a real employee and logs them in.
7. Task 7. The users will need a higher access level employee
    a. Sub-tasks 1. Create a manager class. This manager will have access level to the CRUD methods regarding evaluations.

My Tasks:

8. Tasks 1. Build a database.
    a. Sub-tasks 1. The users will need a database to store all the employees and their demographical, job history, and evaluation class.
        i. Create a Demographic class that includes:
            1. A name variable and add method
            2. Address variable and add method
            3. Phone variable and add method
            4. Emails variable and add method

5. Emergency Contact/Insurance provider variable and add method

ii. Create a array list JobHistory that contains the Job class info which includes
   1. Company name variable and add method
   2. Supervisor variable and add method
   3. Teammate names (if applicable) variable and add method
   4. Length on the job variable and add method
   5. Job title variable and add method
   6. Team roles variable and add method

iii.
   1. Critical skills used/learned (preferably ones related to the applied job.)
   2. Level of each skill employee has. (beginner-expert) peer-reviewed.
   3. Soft skills developed at each job. (preferably ones team-orientated)
   4. Gifts the employee has been found to have. (Self-evaluated personal skills)

iv. Create a Evaluation class that includes
   1. Evaluating supervisor
   2. Date of Eval
   3. The mental state of the employee (Commonly used emotional adjectives)
   4. Recommendations/notes on current job satisfaction, proficiency, and efficiency.

b. Sub-tasks 2. The users will need a GUI interface to collect the data. Something the employees can interact with and put their information into.
   i.

c. Sub-tasks 3. The users will need a way to save this on the interface.
   i. When saving data the information is first checked for any SQL injections to prevent attacks on vulnerable information. (Small businesses are often seen as vulnerable/easy targets so this is a good safety measure).
   ii. After checking the data it will be saved into the database and create a new record for that employee.

d. Sub-tasks 4. Saved data will need to send an alert to the user that it has been correctly created and saved appropriately.

e. Sub-tasks 5. Employees will need a button to clock in and clock out

9. Tasks 2. The user will need to be able to read a user from the database.
   a. Sub-tasks 1. A query at the data layer will be written to return specific employee demographics, job history, and evaluation tracking.
      i. Information to be returned.
   b. Sub-tasks 2. The user will need a search function in the GUI interface the user can use to select a specific user.

    i. The search function should be able to have a subfunction of which specific database they wish to access. Also, a user authentication level to make sure normal employees and anyone under a certain access level may not access employee information.

c. Sub-tasks 2.1 The user should be able to filter employees and information using a specific data query type
d. Sub-tasks 3. The search mechanism that is triggered to access the function (A search bar similar to Canvas/google should be sufficient)
    i. After a search is triggered it will ask if you wish to access a specific database or all information. Then that data will be displayed on the GUI
e. Sub-tasks 4. In the case of no data or wrong access level then the user should receive a message explaining that the user does not exist or they do not have such access.

Task 6.

    i. Sub-tasks 5. Create a SQL injection check method and variable. Checks the info to make sure it is not an injection before it gets sent anywhere else.