**Documentation**

This documentation provides an overview of the Python project for managing employee information, including a database and functionalities for adding, retrieving, updating, and calculating salaries.

**Project Structure:**

* src/models.py**:** Contains Python classes defining the database models for Employees, Teams, and Bonuses.
* src/database\_manager.py**:** Implements the DatabaseManager class responsible for interacting with the database using SQLAlchemy.
* tests/database\_manager.py**:** Contains unit tests for the DatabaseManager class.

**Database Models (src/models.py):**

Employees**:**

* Attributes:
* id (Integer, primary key)
* first\_name (String)
* last\_name (String)
* base\_salary (Integer)
* birth\_date (Date)
* hire\_date (Date)

Teams**:**

* Attributes:
* id (Integer, primary key)
* leader\_id (Integer, ForeignKey referencing Employees.id)
* employee\_id (Integer, ForeignKey referencing Employees.id)

Bonuses**:**

* Attributes:
* id (Integer, primary key)
* type (String)
* value (Integer)

**Database Manager (src/database\_manager.py):**

The DatabaseManager class provides methods for interacting with the database:

Initialization:

Establishes a connection to the database using the provided URL (defaults to sqlite:///employees.db).

Creates the database schema if it doesn't exist.

Adding Data:

Methods to add new employee, team, and bonus entries to the database.

Retrieving Data:

Methods to retrieve data from the database, including:

All employees

Specific employees based on various criteria (e.g., name, birthdate)

Team members of a specific leader

Information about bonuses (yearly bonus, leader bonus per member)

Methods to check if an employee is a team leader.

Salary Calculation:

Calculates the total salary for an employee considering:

Base salary

Years with the company (based on hire date)

Yearly bonus

Potential leader bonus (if the employee is a leader)

Provides options to:

Simply calculate the salary

Calculate the salary and print a simulated email message

Calculate the salary and return a tuple containing salary, employee name, and email message

Other Methods:

Removing and updating employee data

Populating the database with sample data (for testing purposes)

**Unit Tests (database\_manager\_test.py):**

These tests verify the functionalities of the DatabaseManager class, focusing on employee and team management aspects in an employee management system. They utilize the Employees, Teams, and DatabaseManager classes from the src.models and src.database\_manager modules, respectively.

Individual Test Descriptions:

1. test\_john\_doe\_is\_team\_leader**:**

*Purpose***:** Checks if an employee named John Doe with a specific birthdate exists and is a team leader.

*Steps***:**

Retrieves all employees using db\_manager.get\_all\_employees().

Asserts that the retrieved employees list is not None.

Filters the employees list to find John Doe based on first and last name.

Asserts that exactly one John Doe is found.

Verifies that the John Doe's birthdate matches the expected date.

Calls db\_manager.is\_leader to check if John Doe is a team leader and asserts the result.

1. test\_john\_doe\_team\_members**:**

*Purpose***:** Validates that John Doe's team members are the expected individuals.

*Steps***:**

Follows similar steps as test\_john\_doe\_is\_team\_leader to retrieve and filter employees for John Doe.

Retrieves John Doe's team members using db\_manager.get\_team\_members.

Asserts that the team member list has two members.

Converts team members' names to a list of formatted strings (first and last name combined).

Asserts that both expected team member names ("Myrta Torkelson" and "Jettie Lynch") are present in the list.

1. test\_tomas\_andre\_not\_in\_john\_doe\_team**:**

*Purpose***:** Confirms that Tomas Andre is not part of John Doe's team.

*Steps***:**

Mirrors the initial steps of test\_john\_doe\_team\_members to retrieve and filter employees for John Doe.

Retrieves John Doe's team members using db\_manager.get\_team\_members.

Converts team members' names to a list of formatted strings.

Asserts that "Tomas Andre" is not present in the list of team member names.

1. test\_gretchen\_watford\_base\_salary**:**

*Purpose***:** Verifies that Gretchen Walford's base salary matches the expected value.

*Steps***:**

Retrieves all employees using db\_manager.get\_all\_employees().

Asserts that the retrieved employees list is not None.

Filters the employee list to find Gretchen Walford based on first and last name.

Asserts that exactly one Gretchen Walford is found.

Asserts that Gretchen Walford's base salary is equal to 4000.

1. test\_tomas\_andre\_not\_team\_leader**:**

*Purpose***:** Ensures that Tomas Andre is not a team leader and attempts to retrieve his non-existent team members (should not raise errors).

*Steps***:**

Retrieves all employees using db\_manager.get\_all\_employees().

Asserts that the retrieved employees list is not None.

Filters the employee list to find Tomas Andre based on first and last name.

Asserts that exactly one Tomas Andre is found.

Calls db\_manager.is\_leader to check if Tomas Andre is a team leader and asserts that he is not.

Attempts to retrieve Tomas Andre's team members using db\_manager.get\_team\_members (expected to return an empty list, not an error).

1. test\_jude\_overcash\_not\_in\_database**:**

*Purpose***:** Confirms that Jude Overcash is not present in the employee database.

*Steps***:**

Retrieves all employees using db\_manager.get\_all\_employees().

Asserts that the retrieved employees list is not None.

Filters the employee list to find Jude Overcash based on first and last name.

Asserts that the filtered list is empty (no Jude Overcash found).

1. test\_employee\_salary\_without\_leadership\_bonus(failing due to incorrect date):

*Purpose***:** Calculates the salary for an employee who is not a team leader and validates the calculated value

**Additional Notes:**

The code utilizes SQLAlchemy for database interactions.

The provided database URL (sqlite:///employees.db) can be replaced with your specific database configuration.

Email functionality is currently simulated in the salary calculation methods and requires further development for actual email sending.

**Future Considerations:**

Implement email sending functionality using an appropriate email service provider.

Add functionalities for managing other aspects of employee information (e.g., contact information, job titles, departments).

Consider implementing user authentication and authorization mechanisms for secure access to the system.

This documentation provides a basic understanding of the project's functionalities and serves as a starting point for further exploration and development.