Biometric Time Clock documentation - Yassir technical test

The Biometric Time Clock application is a new tool used by the ABC school to manage its employees. This project is the technical test of the Full-Stack Software Engineer position at Yassir.

The application has a client-server architecture where the server contains the requested API and the database connection built with NodeJS, while the client contains the front-end components with integration to the API using ReactJS.
The database used is postgresql for the need of a relational database only.

1. Database:

The database is called 'biometricTimeClock' and has 2 tables: <u>employee</u> to store the employee information and <u>activity</u> to store the checkin and checkout information, as shown in the following:
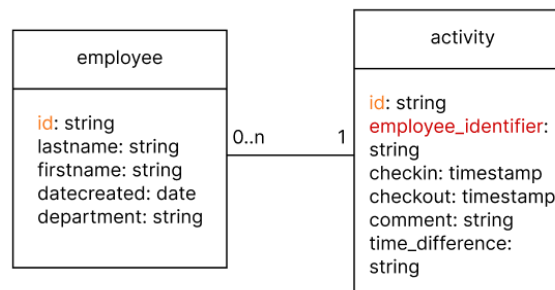
2. Server:

The server contains three files: server.js for the API endpoints, pg.js to initiate a connection with the database and the db.sql for the used sql queries to help create the database and the 2 tables.

The API routes are:

`'/newemployee/:lastname/:firstname/:department'`	To create a new employee by giving the lastname, firstname and department. The datecreated will be the date of the system given by javascript.
`'/getlist'`	Get a list of all employees with their information stored in the employee table.
`'/filterbydate/:creationdate'`	Filter by the date picked by the user on the client side.
`'/check-in/:employeeId/:comment'`	Checkin using the employeeId and save a comment.
`'/check-out/:employeeId/:comment'`	Checout using the employeeId and save a new comment in the comment column of the activity table.

The API endpoints are documented in the code to explain how they work.

3. Client:

The client side has the app.js for the routes, the index.js to render the application, index.css to style the components, the components folder and the images folder.

The client is built in order to integrate and test the API endpoints using the fetch method. For example when getting the list of all employees in the list.js components:

```
const getEmployees = async() =>{
    try{
        fetch(`http://localhost:8000/getlist`).then(
        response => response.json()
        ).then(
            data => {
            setlistemployee(data)
        }
        )
    }catch (err) {
        console.error(err)
    }
}
```

The list component shows the list of all employees that are in the dataset (rendering a list of Employee components). While the new_employee component is used to create a new employee account with 3 inputs: the last name input, the first name input and the department input.

4. Test units:
A set of tests are conducted as follows:

    4.1. Create a new employee:
Testing to create a new employee account with 'John' as last name, 'Lamversky' as first name and department as 'Department B'.

NB: If the user tries to create a new employee without setting input values, an error message will be displayed.

4.2. Get the list of all employees:

Here the created employees shows in the home page. However other employees were created to display a longer list.

Here is the new list of employees:

4.3. Filter by date of creation:

The user can filter employees by date of creation using a date picker.

If the date is not set and the filter button is pressed then the filter will take the system date to filter the employee data.

Example: Filter the employee accounts that were created on November 1st 2023:

NB: As you can see, the time shows a difference of 23h (T23:00:00) because of the different timezone used in the PgAdmin. However, the Filter API route gives a correct result, as it is only the rendering in the front end side that is not making time zone conversions.

4.4. Perform a check-in and a check-out:

This is a test for the checkin and checkout done in the pgAdmin directly since this part is not integrated in the client side. That is because the 2 API routes need the employeeId parameter which is generally returned after authentication when login in.

For the check-in: The checkin route automatically inserts a new row in the activity table with the current system date in the checkin column.

```
insert into activity (id, employee_identifier, checkin, comment)
values ('dfcdfvdfo', '1546546cdssssssssdf', now(), 'Checkin comment');
select * from activity;
```

id [PK] character varying (255)	employee_identifier character varying (255)	checkin timestamp without time zone	checkout timestamp without time zone	comment character varying (255)	time_difference character varying (255)
dfcdfvdfo	1546546cdssssssssdf	2023-11-02 09:21:31.982268	[null]	Checkin comment	[null]

The checkout route updates the existing row data of the employee on the same day.

```
Query   Query History                                                    ⤢    Scratch Pad  ×    ⤢
  1  update activity set checkout = now(), comment = 'Checkout comment' where employee_identifier =
  2  select * from activity;
```

Data Output   Messages   Notifications   ⤢

	id [PK] character varying (255)	employee_identifier character varying (255)	checkin timestamp without time zone	checkout timestamp without time zone	comment character varying (255)	time_difference character varying (255)
1	dfcdfvdfo	1546546cdsssssssdf	2023-11-02 09:21:31.982268	2023-11-02 09:23:24.056277	Checkout comment	[null]

4.5. Calculate the time difference between the checkin and checkout:

The time difference is then calculated and stored in the time_difference column.

```
Query   Query History                                                    ⤢    Scratch Pad  ×    ⤢
  1  update activity set time_difference = (now() - (select checkin from activity
  2  where employee_identifier = '1546546cdsssssssdf' and
  3  EXTRACT('MONTH' FROM checkin) = EXTRACT('MONTH' FROM now()) and
  4  EXTRACT('DAY' FROM checkin) = EXTRACT('DAY' FROM now()) and
  5  EXTRACT('YEAR' FROM checkin) = EXTRACT('YEAR' FROM now())));
  6  select * from activity;
```

Data Output   Messages   Notifications   ⤢

	id [PK] character varying (255)	employee_identifier character varying (255)	checkin timestamp without time zone	checkout timestamp without time zone	comment character varying (255)	time_difference character varying (255)
1	dfcdfvdfo	1546546cdsssssssdf	2023-11-02 09:21:31.982268	2023-11-02 09:23:24.056277	Checkout comment	00:04:38.945327

This query is in the checkout route as the time difference needs to be saved when checking out.

5. Deployment:

The project is available on GitHub.