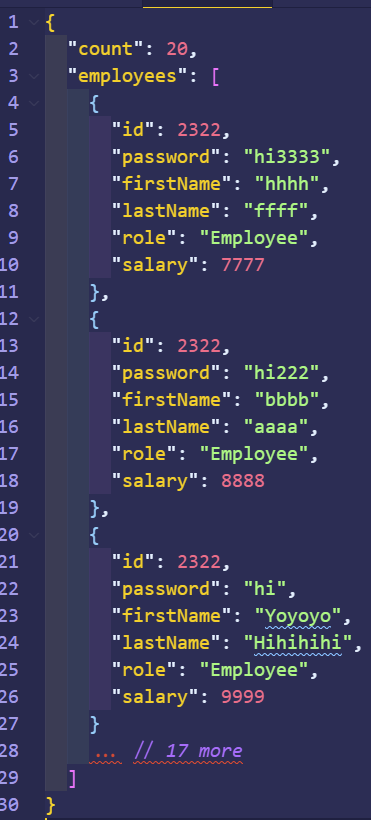
1. Create a new employee.  
   Description: Creates a new employee. Requires request body with firstName, lastName, role, and salary fields in JSON format. Returns the sent info except the salary and the id field.  
   Path: /create\_employee   
   Headers: Authorization: Bearer <token> // must be a person who can craete  
   Method: POST  
   Request body: JSON  
    {  
    “firstName”: “Hihihi”,  
    “lastName”: “Hey”,  
    “password”: “123”,  
    “role”: “Employee”,  
    “salary”: 4000,  
    }   
   Responses:  
    1. Successfully created  
    Request Body: JSON  
    { “firstName”: “Hihihi”, “lastName”: “Hey”, role”: “Employee”,  
    “salary”: 4000, “id”: 34}  
    Status Code: 200 OK  
    2. Incorrect Request Body (one example)  
    Request Body: JSON {“message”: “Salary is too big”}  
    Status Code: 400 Bad Request
2. Read all users’ info  
   Description: Returns all users in JSON format  
   Path: /users  
   Headers: Authorization: Bearer <token> needed to gain access to users’ passwords and/or salary  
   Method: GET  
   Requets Query: limit=20 (Shows only the first 20 employees),   
    sort\_by=salary(Shows the most salary first),  
    reverse=true (Reverses the order),  
    password=true (See passwords),  
    salary=true (Shows salaries),  
    default values: limit=none, sort\_by=id, reverse=false,   
    password=false, salary=false,   
    firstName=none, lastName=none, role=all  
   Responses  
    1. Successful request  
    Request Body: JSON  
      
    Status Code: 200 OK  
    2. Incorrect Request Body (one example)  
    Request Body: JSON {“message”: “Limit must be > 0”}  
    Status Code: 400 Bad Request
3. Update employee info  
   Description: Updates existing user info. Returns the updated user info if successful else returns error message in JSON.  
   Path: /users/:id/update  
   Method: PUT  
   Request Body: JSON  
    {“password”: “45454545”} // attempting to change password  
   Request Param: id (the id of the user whose info is going to be changed)  
   Response Body:  
    1. Successful  
    Request Body: JSON  
    { … all fields of this user … }  
    Status code: 200 OK  
    2. Incorrect Request Body  
    Request Body: JSON  
    {“message”: “New password must contain at least 1 lower   
    character”}  
    Status code: 400 Bad Request
4. Delete an employee  
    Description: Deletes an employee in the database. Returns deleted user info if successful else returns error message  
   Path: /users/:id/delete  
   Method: DELETE  
   Headers: Authorization: Bearer <token> // token of a person who can delete this Request Param: id (the id of the user who is being deleted)  
   Response Body:  
    1. Successful  
    Request Body: JSON  
    { … all fields of this recently deleted user … }  
    Status code: 200 OK  
    2. Incorrect Request Body  
    Request Body: JSON  
    {“message”: “Not valid token to delete this user”}  
    Status code: 401 Unauthorized
5. Display a specific employee info  
   Description: Specifying the first name, last name, or role, show the user’s info whose info matches the given fields.  
   **Use the same implementation as that of requirement #3** with specifying the queries, firstName=<firstName>&lastName=<lastName>&role=<role>