Lab5 - Express server and mongoose

Extend Lab4:

Complete Lab 4 as per the original requirements. In addition, implement the following features:

Please read the whole file before starting to code.

GREENS ARE BONUS

We need to create an Employee system with the following 3 models:

1- Employee model

employee should have:

- username: String, required, unique, no spaces, min 8
- firstName: String,required, min length 3, capitalized, max length 15
- lastName: String,required, min length 3, capitalized, max length 15
- dob: Date, required
- password: String, required, min length 8
- createdAt: Date, timeStamp,
- updatedAt: Date, timeStamp

2- Profiles model

profile should have:

- empId: required, unique
- title: String, required
- description: String, required
- yearOfExperience: default 0
- department: String, required
- phone: String, required
- email: required, valid email

3- Leaves model

leave should have:

empId: required

type: String, required and one of [annual, casual, sick]

duration: +ve number, required

createdAt: Date, timeStamp,

updatedAt: Date, timeStamp,

status: String and one of [inprogress, cancelled, ended], default Inprogress

Employee can only change it to cancelled if it's not ended

empBukupId: (id of colleague who will cover work during the leave)

Shouldn't be the same as empId

Use autoIncremental id instead of mongo id

Each employee must have one profile.
Each employee can have many leaves.

Think about models relations and if there is any enhancement, do it

A User can login and add, edit, view or delete his own todos.

2- Implement the following end points:

HTTP Method	route	Description
post	/employees	- Add an employee with the following **required** attributes username, password, firstName, lastName Return registered employee with token if success Notes: - Handle validation errors returned from mongo
Post	/employees/login	Return (employee with token }) Don't return password If the the authentication failed Return error with 401 status code
GET	/employees	Return id Return the firstName of employees Return the age of the employees (check mongoose feature)
DELETE	/employees/:id	Delete the employee with selected id
PATCH	/employees/:id	- Edit employee with the selected id - Return ({employee: theEmployeeAfterEdit}) if success - Handle validation errors returned from mongo
POST	/leave	Employee submit a leave Return the new leave to the employee
PATCH	/leave/:id	Edit leave, only can edit type, duration and status
GET	/employees/:id/leaves	Return the leaves of specific employee
GET	/leaves?limit=10&skip=#no&status=#status Value	- Return the leaves with specific required filters (status, defaults are limit 10 skip 0) (don't reinvent the wheel) - Return user name and user id only with each leave

- Profile routes are the same as employee routes

3- Protect all endpoints with a proper authentication layer. (except for registration and login of course!)

ACL (Authorization)

1- Each employee can only get/edit/delete his data, profile and leaves.
2- Each employee can only get/edit himself.

3- Create a client side app with single page that display the first 10 leaves of specific employee id and status filter
(handle CORS)

requirement:

- .env (port and mongo url)
- Debugger
- Linter

READ THE DOCUMENTATION
https://mongoosejs.com/docs/

Useful reads:

JWT **(must read)**
Handle errors in express **(IMPORTANT)**
CORS
How bcryptjs works