Specifications

Contents

Backend requirements specifications:

Routes

- ✓ USER
  - a GET method : http://localhost:4000/api/users?email=XXX&username=YYY
  - a GET method : http://localhost:4000/api/users/:userID
  - a POST method : http://localhost:4000/api/users
  - a PUT method : http://localhost:4000/api/users/:userID
  - a DELETE method : http://localhost:4000/api/users/:userID
- ✓ WORKING TIME
  - a GET (ALL) method : http://localhost:4000/api/workingtime/:userID?start=XXX&end=YYY
  - a GET (ONE) method : http://localhost:4000/api/workingtime/:userID/:id
  - a POST method : http://localhost:4000/api/workingtime/:userID
  - a PUT method : http://localhost:4000/api/workingtime/:id
  - a DELETE method : http://localhost:4000/api/workingtime/:id
- ✓ CLOCKING – a GET method : http://localhost:4000/api/clocks/:userID
  - a POST method : http://localhost:4000/api/clocks/:userID

Mandatory schemas for the application:

- users = { " username ": " string - required - can 't be null " , " email ": " string - required - can 't be null - X@X . X " , }
- clocks = { " time ": " datetime - required - can 't be null " , " status ": " boolean - required - true (when clock 'in) - can 't be null " , " user ": " user - required - can 't be null " }
- workingtime = { " start ": " datetime - required - can 't be null - YYYY - MM - DD HH : mm : ss " , " end ": " datetime - required - can 't be null - YYYY - MM - DD HH : mm : ss " , " user ": " user - required - can 't be null " }

Data organisation requirement:

- Date need to be as follows: 'YYYY-MM-DD hh:mm:ss'
- Need 8 hours of rest between two shifts

Role requirement

- Administrator
- Employee
- Manager
- General Manager

Role specification

- All users can:
  - Edit their account information

    - Delete their account
    - Report their departure and arrival times
    - View their dashboards
- Manager and General Manager can:
    - Manage their team(s)
    - View the view the averages of the daily and weekly hours of the team over a given period
    - View the daily and weekly working hours of an employee over a period of time
    - View their employees' dashboards
- Administrator can:
    - Do anything they want on the application

Authentication specification

- Need to use 'JSON Web Tokens'
    - With the 'Joken' library
- Need to be able to handle password

Frontend requirements specifications:

Main application

Only using one view should be used, this view should contain at least those five components:

- User
  - Need to identify the current user
  - Need to identify the current user on all pages
  - Implement those method ->
    - Create User (createUser())
    - Update User (updateUser())
    - Get User (getUser())
    - Delete User (deleteUser())
- WorkingTimes
  - Need to display the working/ed times recorded
  - Need to use this route to display the information “/workingTimes/:userID”
  - Need to implement in the display of WorkingTimes :
    - UserId
    - WorkingTimes data (the table summarizing the offset times)
  - Need to implement the getWorkingTimes() method
- WorkingTime
  - Used for displaying, creating, modifying and deleting a working time
  - Need to be linked to the routes :
    - /workingTime/:userid (for creation
    - /workingTime/:userid/:workingtimeid (for modification and deletion)
  - Need to implement the mehods:
    - createWorkingTime()
    - updateWorkingTime()
    - deleteWorkingTime()
- ClockManager
  - Declare Worked Hours
    - Clock In
    - Clock Out
  - Need to connect to the route “/clock/:userid”
  - Need to access to :
    - startDateTime data (is worth null if no work period is in progress)
    - clockIn (a boolean that is true if a work period is in progress)
    - refresh() and clock() methods (to pass from active to inactive and vice versa)
  - Need to update clock-in/out by a manager
  - Display overworking hours of team users to their manager on given period
- ChartManager
  - For management purposes, it needs to have three graphs displaying information on employee working shifts, hours, working time week organisation, etc...
  - Connected to the /chartManager/:userid route
- Page for User to connect/disconnect
- Page for switching roles/position
- Restriction on specific data depending on Users roles

- Need to be accessible to visual handicaps person
    - Reader
    - Colour blind
- Need a Calendar display with user planed/passed and current week shifts
    - Display hours in overtime
    - Display information on day/night shift
    - Display information on Vacation, sick leave and holidays
- Need a Smartphone display access for the application for clocking remotely.
- Send messages to Management or higher up

Deployment requirements specifications:

CI

- Deployment within DOCKER containers
    - One container for the Front
    - One container for the Back
    - One container for the Database
        - Data need to be persistent
- The application needs to be available from internet

CD

- When modification is pushed, set a pipeline to:
    - Auto compile
    - Test the modification
    - Build docker image
    - Deploy on the server

Deployment

- The application needs to be available on any browser
- The application needs to have a display on a mobile device
- The application needs to have a build to for Android deployment
- The application needs to work offline (but the login shall work only with a connection)

Security

- Injection Attacks:
    - Check for XSS (Cross-Site Scripting).
- Test for SQL and NoSQL injections.
- Configuration Files:
    - Look for exposed configuration files.
- Password Security:
    - Ensure password hashes are strong.
- Access Control:
    - Verify routes and endpoints are secure from unauthorized users.
- Denial of Service (DoS):
    - Test if the app can be overwhelmed or if the database can be filled.
- JWT Security:
    - Check if JWT tokens are HttpOnly.
    - Identify how to recover tokens if they’re not secure.
- Password Transmission:
    - Ensure passwords are not sent in clear text.
- HTTPS:
    - Confirm the application only uses HTTPS to protect data.

- Other Vulnerabilities:
    - Look for additional weaknesses, like CSRF (Cross-Site Request Forgery).
- Team Collaboration:
    - Share findings and ideas with classmates for better results.

Documentation

- Add a view to display the rules, contracts, legal documentation, payment regulation for user to access
- Add a view to display training documentation for user to access or a step-by-step overlay tutorial on the application
- Have Users stories
- Dispose of the Data base schematic