CS-C3170 - Web Software Development

Course Project – WeCare Application

Requirement Checklist

Haibi Peng 875552

Epic checklist

- Application structure ***ALL FULFILLED***
  - Application divided into logical folders (akin to the part on Structuring Web Applications) √
  - Dependencies exported from deps.js √
  - Project launched from app.js, which is in the root folder √
  - Configurations in a separate folder (e.g. config) √
    - Test configurations separate from production configurations √
    - Configurations loaded from environmental variables or e.g. dotenv -files √
- Users ***ALL FULFILLED***
  - Email and password stored in the database for each user √
    - Password not stored in plaintext format √
    - Emails must be unique (same email cannot be stored twice in the database) √
  - Users can register to the application √
  - Registration form is accessible at /auth/registration √
    - Registration uses labels to clarify the purpose of the input fields √
    - Registration form is validated on the server √
      - Email must be a valid email (clarified from before, i.e. email must be validated - no need to e.g. send a mail to the address though) √
      - Password must contain at least 4 characters √
      - Validation errors shown on page √
      - In case of validation errors, email field is populated (password is not) √
  - User-specific functionality is structured into logical parts (e.g.

userController.js, userService.js) ✓

- Authentication *ALL FULFILLED*
  - Application uses session-based authentication ✓
  - Login form is accessible at /auth/login ✓
    - Login form asks for email and password ✓
    - Login uses labels to clarify the purpose of the input fields ✓
    - Login form has a link to the registration form ✓
    - If the user types in an invalid email or password, a message "Invalid email or password" is shown on the login page. ✓
      - Form fields are not populated ✓
  - Authentication functionality is structured into logical parts (e.g. authController.js or part of userController.js, ...). ✓
  - Application has a logout button that allows the user to logout (logging out effectively means clearing the session) ✓
    - Logout functionality is at /auth/logout ✓
- Middleware *ALL FULFILLED*
  - The application has middleware that logs all the errors that occurred within the application ✓
  - The application has middleware that logs all requests made to the application ✓
    - Logged information contains current time, request method, requested path, and user id (or anonymous if not authenticated) ✓
  - The application has middleware that controls access to the application ✓
    - Landing page at / is accessible to all ✓
    - Paths starting with /api are accessible to all ✓
    - Paths starting with /auth are accessible to all ✓
    - Other paths require that the user is authenticated ✓
      - Non-authenticated users are redirected to the login form at /auth/login ✓
  - Application has middleware that controls access to static files ✓
    - Static files are placed under /static ✓

    - Middleware functionality is structured into logical parts (e.g. separate middlewares folder). ✓
- Reporting *ALL FULFILLED*
    - Reporting functionality is available under the path /behavior/reporting ✓
    - Reporting cannot be done if the user is not authenticated ✓
    - When accessing /behavior/reporting, user can choose whether morning or evening is being reported ✓
        - User reporting form depends on selection ✓
        - Page at /behavior/reporting shows whether morning and/or evening reporting for today has already been done ✓
    - Morning reporting form contains fields for date, sleep duration, sleep quality, and generic mood ✓
        - Date is populated by default to today, but can be changed ✓
            - Form has a date field for selecting the date ✓
        - Sleep duration is reported in hours (with decimals) ✓
        - Sleep quality and generic mood are reported using a number from 1 to 5, where 1 corresponds to very poor and 5 corresponds to excellent. ✓
            - Form has a slider (e.g. range) or radio buttons for reporting the value ✓
        - Form contains labels that clarify the purpose of the input fields and the accepted values ✓
        - Form fields are validated ✓
            - Sleep duration must be entered, must be a number (can be decimal), and cannot be negative ✓
            - Sleep quality and generic mood must be reported using numbers between 1 and 5 (integers). ✓
            - In case of validation errors, form fields are populated ✓
    - Evening reporting form contains fields for date, time spent on sports and exercise, time spent studying, regularity and quality of eating, and generic mood ✓
        - Date is populated by default to today, but can be changed ✓

            - □ Form has a date field for selecting the date ✓
        - □ Time spent on sports and exercise and time spent studying are reported in hours (with decimals) ✓
        - □ Regularity and quality of eating and generic mood are reported using a number from 1 to 5, where 1 corresponds to very poor and 5 corresponds to excellent. ✓
            - □ Form has a slider (e.g. range) or radio buttons for reporting the value ✓
        - □ Form contains labels that clarify the purpose of the input fields and the accepted values ✓
        - □ Form fields are validated ✓
            - □ Time spent on sports and exercise and time spent studying are reported in hours must be entered, must be a number (can be decimal), and cannot be negative ✓
            - □ Regularity and quality of eating and generic mood must be reported using numbers between 1 and 5 (integers). ✓
            - □ In case of validation errors, form fields are populated ✓
    - o Reported values are stored into the database
        - □ The database schema used for reporting works for the task ✓
        - □ Reporting is user-specific (all reported values are stored under the currently authenticated user) ✓
        - □ If the same report is already given (e.g. morning report for a specific day), then the older report is removed ✓
            - □ If the functionality for handling duplicate reports is something else, the functionality is described in documentation ✓
    - o Reporting functionality structured into logical parts (separate views folder, separate controller for reporting, service(s), ...) ✓
- Summarization ***ALL FULFILLED***
    - o Summary functionality is available under the path /behavior/summary ✓
    - o Main summary page contains the following statistics, by default shown

for the last week and month ✓

        - Weekly average (by default from last week) ✓
            - Average sleep duration ✓
            - Average time spent on sports and exercise ✓
            - Average time spent studying ✓
            - Average sleep quality ✓
            - Average generic mood ✓
        - Monthly average (by default from last month) ✓
            - Average sleep duration ✓
            - Average time spent on sports and exercise ✓
            - Average time spent studying ✓
            - Average sleep quality ✓
            - Average generic mood ✓
    - Summary page has a selector for week and month. Check input type="week" and input type="month". ✓
        - When the week is changed, the weekly average will be shown for the given week. ✓
        - When the month is changed, the monthly average will be shown for the given month. ✓
        - If no data for the given week exists, the weekly summary shows text suggesting that no data for the given week exists. ✓
        - If no data for the given month exists, the monthly summary shows text suggesting that no data for the given month exists. ✓
    - Summary data / averages calculated within the database ✓
        - When doing weekly reporting, the weekly averages are calculated in the database ✓
        - When doing monthly reporting, the monthly averages are calculated in the database ✓
    - Summarization page contains statistics only for the current user. ✓
- Landing page (i.e. page at the root path of the application) ***ALL FULFILLED***
    - Landing page briefly describes the purpose of the application ✓
    - Landing page shows a glimpse at the data and indicates a trend ✓

        - Landing page shows users' average mood for today and and yesterday ✓
        - If the average mood yesterday was better than today, tells that things are looking gloomy today ✓
        - If the average mood yesterday was was worse today, tells that things are looking bright today ✓
    - Landing page has links / buttons for login and register functionality ✓
    - Landing page has links / buttons for reporting functionality ✓
- Testing ***PARTIALLY FULFILLED***
    - The application has at least 5 meaningful automated tests. All tests detect if e.g. tested functionality is changed so that it no longer works as expected. ✓
    - The application has at least 10 meaningful automated tests. All tests detect if e.g. tested functionality is changed so that it no longer works as expected. ✓
    - The application has at least 15 meaningful automated tests. All tests detect if e.g. tested functionality is changed so that it no longer works as expected. ✗
    - The application has at least 20 meaningful automated tests. All tests detect if e.g. tested functionality is changed so that it no longer works as expected. ✗
- Security ***ALL FULFILLED***
    - Passwords are not stored in plaintext ✓
    - Field types in the database match the actual content (i.e., when storing numbers, use numeric types) ✓
    - Database queries done using parameterized queries (i.e., code cannot be injected to SQL queries) ✓
    - Data retrieved from the database are sanitized (i.e., if showing content from database, using <%= ... %> instead of <%- ...%> unless explicitly stated what for). ✓
    - Users cannot access data of other users. ✓
    - Users cannot post reports to other users' accounts. ✓
- Database ***ALL FULFILLED***

  - Expensive calculations such as calculating averages are done in the database ✓
  - Indices are used when joining tables if the queries are such that they are used often ✓
  - Database uses a connection pool ✓
  - Database credentials are not included in the code ✓
- User interface / views ***ALL FULFILLED***
  - Views are stored in a separate folder ✓
  - User interface uses partials for header content ✓
  - User interface uses partials for footer content ✓
  - Recurring parts are separated into own partials (e.g. partial for validation errors) ✓
  - Pages with forms contain functionality that guides the user ✓
    - Labels are shown next to form fields so that the user knows what to enter to the form fields ✓
    - Form fields are validated and user sees validation errors close to the form fields ✓
    - In the case of validation errors, form fields are populated (with the exception of the login page) ✓
  - User interface uses a style library or self-made stylesheets (see e.g. Twitter Bootstrap for a style library) ✓
    - If Twitter Bootstrap or other external style libraries are used, they are used over a content delivery network
  - Different pages of the application follow the same style ✓
  - User sees if the user has logged in (e.g. with a message 'Logged in as my@email.net' shown at the top of the page) ✓
- APIs ***ALL FULFILLED***
  - The application provides an API endpoint for retrieving summary data generated over all users in a JSON format ✓
  - The API is accessible by all ✓
  - The API allows cross-origin requests ✓
  - Endpoint /api/summary provides a JSON document with sleep duration, time spent on sports and exercise, time spent studying, sleep quality, and

generic mood averaged over the last 7 days ✓

  - Endpoint /api/summary/:year/:month/:day provides a JSON document with averages for sleep duration, time spent on sports and exercise, time spent studying, sleep quality, and generic mood for the given day ✓

- Deployment ***ALL FULFILLED***
  - Application is available and working in an online location (e.g. Heroku) at an address provided in the documentation ✓
  - Application can be run locally following the guidelines in documentation ✓
- Documentation ***ALL FULFILLED***
  - Documentation contains necessary CREATE TABLE statements needed to create the database used by the application ✓
  - Documentation contains the address at which the application can currently be accessed ✓
  - Documentation contains guidelines for running the application ✓
  - Documentation contains guidelines for running tests ✓