Course Project

Overview

In the course project, you will build a web application for self-monitoring purposes. The application provides users an opportunity to report behavior daily.

Reported content

The behavior reported to the application are as follows:

- Sleep duration and sleep quality
- Time spent on sports and exercise
- Time spent studying
- Regularity and quality of eating
- Generic mood

When the application is opened (and the user has logged in), the application provides two options for reporting: morning and evening.

- When choosing morning, the application asks for the sleep duration and sleep quality, as well as generic mood. Sleep duration is given as a decimal indicating hours slept, and sleep quality and generic mood are given using a scale from 1 to 5, where 1 indicates very poor and 5 indicates excellent.

- When choosing evening, the application asks for time spent on sports and exercise, time spent studying, regularity and quality of eating, and generic mood. Time spent on sports and exercise, and time spent studying are given as a decimal number indicating the number of hours. Regularity and quality of eating as well as the generic mood are given using a scale from 1 to 5, where 1 indicates very poor and 5 indicates excellent.

In both options, by default, the application assumes that the reporting is done for the current day (or, in the case of the morning, for the previous night). Both reporting options should also provide an input field through which a different day can be chosen -- this way, if a user skips reporting, the data can be filled in at a later point.

Summarization

The application provides functionality for summarization of responses. Each user can view statistics of their reports on a weekly and monthly level. These statistics are as follows.

- Average sleep duration
- Average time spent on sports and exercise
- Average time spent studying
- Average sleep quality
- Average generic mood

High-level summaries generated from all the users of the application are shown both on the landing page of the application and provided through an API.

Schedule

The deadline for the project is on 11th of December. After the project has been submitted, it will be reviewed by other course participants. Reviews will begin on the 12th of December and must be completed by 18th of December.

Returning the project

Returning the project is done by returning a zip file that contains the project source code. The zip file should contain also instructions for running the project, necessary CREATE TABLE statements, required configurations, an internet address where the application can be tried out, and any other documentation laid out in the more detailed checklist below.

When returning the project, you will use a version of the checklist to specify which requirements were completed.

Reviewing the projects

Details announced later.

Steps to get started

Here is a possible list of steps that can be taken to get started with the application. Consider writing tests throughout the project and consider using a version control system (e.g. a private repository on GitHub for the project). For specific requirements, refer to the (epic) checklist below.

1. Create a folder structure for the project (following the one outlined in "Structuring Web Applications"). Add files 'app.js' and 'deps.js' to the project.
2. Create a database for the project and add necessary files (configuration, database query functionality) to the project. Based on your interpretation of the requirements, create a database schema. Do not fixate on this schema, and assume that you might change it later on. Also, consider the possibility of feeding in data to the database from the command line / from an online browser for testing purposes.
3. Add some middlewares to the project, at least an error reporting middleware.
4. Add functionality for reporting (morning) behavior. When adding reports, use a fixed user id (e.g. 1).
5. Add functionality for summarization of individual responses on a weekly level, now focusing on a single (last?) week. When building reports, use a fixed user id (e.g. 1).
6. Create the landing page that shows the average mood from today and yesterday. Add a note on trend (i.e. going up / down).
7. Add functionality for reporting (evening) behavior. When adding reports, use a fixed user id (e.g. 1).
8. Adjust summarization of individual responses to include the reported evening behavior. Continue using a fixed user id (e.g. 1).
9. Add monthly summarization functionality and implement the possibility to select a week and / or a month.
10. Implement registration and authentication functionality. Change the fixed user id to that of the user in the session.

11. Take a style library into use.
12. Clean up and document.
13. Implement APIs
14. Continue working on missing content.

Epic checklist

A more detailed checklist is shown below. We suggest storing intermediate versions of the project in a private GitHub repository or similar. When checking for completed requirements in grading, each of the following bullet counts as one requirement. Deviation from individual requirements is possible, given that they are documented.

- Application structure
    - 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
    - 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 an authentic email
            - 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

    - 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

    - 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
        - 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
    - 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

- 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

    - Reporting functionality structured into logical parts (separate views folder, separate controller for reporting, service(s), ...)

- Summarization

    - Summary functionality is available under the path /behavior/summary
    - 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)

    - 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

    - 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 20 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 30 meaningful automated tests. All tests detect if e.g. tested functionality is changed so that it no longer works as expected.

- Security

    - 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

    - 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

    - 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

    - 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 averages for sleep duration, time spent on sports and exercise, time spent studying, sleep quality, and generic mood for each day 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

    - 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

    - 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

Frequently asked questions

- none so far

Previous part