Practice tasks

**Title:**Simple TODO application

**Purpose:**

The purpose of this task is to implement a simple todo application with help of Express, EJS and Node.js.

**Main Task: Implementing Simple TODO Application**

**Estimated time: 8h - 12h**

To begin with, we provide you with site assets in folder assets. It contains very simple pages, styles and JS code. First, your goal to migrate these static assets to dynamic website that uses Express and EJS as a template engine.

1. You need to set-up EJS and views folder, then move HTML code into views. Inside views folder create layout folder and then pick out shared parts of layout between index page and done page into a layout template.
2. Then pick out unique parts from pages and create pages folder in the views folder and put the unique parts here. E.g., ./views/pages/done.ejs, you should render these pages as a partial inside your layout template based on the template variable name page.
3. Don’t forget to change links to assets, pages etc., because now you are using dynamic website, whereas assets were initially developed for static site. You will also need to set-up your Express application to serve static assets.
4. Afterward you should move task list into partial.
5. When a list (TODO or DONE) is empty the app should render <h3 class="no-tasks">There is no tasks.</h3>  instead of a list.
6. When we have tasks then render every of them in the order they were added. The latest should be presented on the top of the list. If a task is done, then the checkbox inside the task should have checkbox—checked class and svg rendered inside it.
7. Now you should handle form submission, when a user submits a form on the index page.
8. Validate that task length is greater or equal 3. If it is not then render <div class="alert alert--error"> Minimal length for task name is 3 letter! </div> right underneath the form.
9. Validate that task does not exists in our list of tasks (DONE and not DONE). If it is exists then render <div class="alert alert--error"> Task ${taskName} already exists! </div> . We expect the message to contain the task name that was send.
10. If the task passed the validation, then add it to your tasks list. You should implement TasksService that will store all the tasks in memory. All method of TasksService should be async, though they can access data synchronously we just want them to always return a promise, so that we will simulate access to a database.
11. IMPORTANT! Don’t forget to always redirect user back to the page when you processed the form data.
12. Finally, we should implement REST endpoint /api/tasks/:taskName/done that will mark task with name that equals received taskName as done. (Our JS script will send this request to the back-end and will remove the element from the page on success).
13. Make sure that you render done tasks only on the DONE page.
14. Make sure that your server is running on localhost:3000.
15. Make sure to not include this script on DONE page.
16. Don’t forget to clean up your solution, split routing into different files and etc.

Important note, we have some set-up in place for you e.g. ./src/main.js exports a function that starts/stop a server. It is needed for our tests. Please do not touch it. Also don’t change class names and text labels, otherwise our tests can fail because they won’t be able to find expected elements on the page. Use the layout we provide for you, in the end the overall structure of the page should not change, for instance tasks and form should be rendered inside main tag that has class main.

Have fun!

**Assessment checklist:**

1. Check that E2E tests passing. (Requires manual run on your machine)
2. Check that routing is split into multiple modules and located in ./src/routes folder
3. Check that views folder contains.
4. Layout folder that contains generic layout templates.
5. Partials folder that contains re-usable template parts among pages.
6. Pages folder that contains pages templates.
7. Please make a manual pass of a golden path of the application. Add some tasks, mark task as done.
8. Check that TasksService’s methods are all async functions. They can store data in memory, but they should return a Promise to simulate access to the database.
9. Check that styles/fonts etc., loaded correctly and page looks in the same way original site assets look.