Practice tasks

**Title:**Implementing REST API

**Purpose:**

The purpose of this task is to build a REST API with help of Express.

**Main Task: Implementing REST API**

**Estimated time: 4h – 8h**

Now, when you have some experience with Express and you learned its details, is time to build your own RESTful API with help of Express. We will keep working on Todo application. For this task you won’t need to build a front-end part.

Our application will consist only of REST API, so you may need Postman or Insomnia to test your application during the development, feel free to pick application of your choice.

Also, though you were using MongoDB during the course materials, we won’t use it in this home task because that is not the main goal of this task and we have not taught you databases yet.

So, let’s get our hands dirty with RESTful API. Let’s define business requirements first.

* As a user, I want to be able to register an account.
* A user has email, password, first name and last name.
* Email should be unique in our application.
* As a user, I want to be able to login into an account.
* Notice, we expect you to use JWT tokens instead of sessions and cookies. We will give you more information on that later.
* As a user, I want to be able to add a task to my Todo list.
* A task should contain title and description.
* Every task of a user should have unique title in the application.
* We should allow to access these endpoints only for sign-up users. A user should provide its JWT token in Authorization header.
* As a user, I want to be able to fetch the list of my tasks.
* A user should not see done tasks that do not belong to the use
* We should allow to access these endpoints only for sign-up users. A user should provide its JWT token in Authorization header.
* As a user, I want to be able to mark task as done.
* Done tasks should not be presented in the Todo list.
* A user should not be able to mark task as done if it does not belong to the user.
* We should allow to access these endpoints only for sign-up users. A user should provide its JWT token in Authorization header.
* As a user, I want to be able to fetch the list of my done tasks.
* A user should not see done tasks that do not belong to the user.
* We should allow to access these endpoints only for sign-up users. A user should provide its JWT token in Authorization header.
* As a user, I want to be able to delete a task.
* It does not matter is it task done or not.
* We should allow to access these endpoints only for sign-up users. A user should provide its JWT token in Authorization header.
* As a user, I want to be able to change a task content.
* We should allow to access these endpoints only for sign-up users. A user should provide its JWT token in Authorization header.

As promised, you can read about implementing JWT tokens with help of Express and JWT here <https://medium.com/front-end-weekly/learn-using-jwt-with-passport-authentication-9761539c4314> or here <https://www.digitalocean.com/community/tutorials/api-authentication-with-json-web-tokensjwt-and-passport>. **Important** don’t forget to use bcrypt module to hash user's passwords when saving them to your “database”.

Speaking of “database” we will get to databases soon, but for now you should store all the data in memory and providing access to the data via services. Each function of a service should be async to behave in the analogous way the real database API behaves, so that in future home tasks we could easily replace the implementation with calls to real database. You will have two services, UsersService that will be controlling Users and all operations related to them, TasksService that will be controlling all operations related to tasks. You may also extract all the Auth logic into AuthService, but that is up to you.

Also, a user should be able to access only her or his tasks, it affects all endpoints related to tasks.

Because we need a way to provided automated tests for home task, we will provide you a specification of desired API that you should implement.

|  |  |
| --- | --- |
| **POST**/api/auth/register | |
| Can Access: | Anyone |
| Request body: | |
| {      email: string;      password: string;  } | |
| Responses: | |
| 200 Ok | Successfully registered a user. |
| 400 Bad Request | Email field is not valid email or password is length is less than 6 characters. Also when any of the fields are missing. |
| 505 Internal Server Error | Something went wrong on the server |

|  |  |
| --- | --- |
| **POST**/api/auth/login | |
| Can Access: | Anyone |
| Request body: | |
| {      email: string;      password: string;  } | |
| Responses: | |
| 200 Ok | Successfully login a user. Returns JWT token and user object without password. |
| 400 Bad Request | Email field is not valid email, or any fields are missing, or user with provided credentials does not exists. |
| 505 Internal Server Error | Something went wrong on the server |
| Response body on success: | |
| {      user: {            email: string;      };      token: string;  } | |

|  |  |
| --- | --- |
| **POST**/api/tasks | |
| Can Access: | Registered User |
| Request Headers | |
| Authorization | Bearer YOUR\_TOKEN\_HERE |
| Request body: | |
| {      title: string;      description: string;  } | |
| Responses: | |
| 201 Created | Successfully created a to-do task for a user. |
| 400 Bad Request | Schema validation of body failed or a user already has a task with provided title. |
| 401 Unauthorized | Authorization validation failed. E.g., a request without a token. |
| 505 Internal Server Error | Something went wrong on the server |

|  |  |
| --- | --- |
| **GET**/api/tasks | |
| Can Access: | Registered User |
| Request Headers | |
| Authorization | Bearer YOUR\_TOKEN\_HERE |
| Responses: | |
| 200 Ok | Successfully returned tasks. Returns empty array if no tasks. |
| 401 Unauthorized | Authorization validation failed. E.g., a request without a token. |
| 505 Internal Server Error | Something went wrong on the server |
| Response body on success: | |
| Array of {      done: boolean;      title: string;      description: string;  } | |

|  |  |
| --- | --- |
| **POST**/api/tasks/done | |
| Can Access: | Registered User |
| Request Headers | |
| Authorization | Bearer YOUR\_TOKEN\_HERE |
| Request body: | |
| {      title: string;  } | |
| Responses: | |
| 200 Ok | Successfully marked a user task as done. |
| 400 Bad Request | Schema validation of body failed. |
| 401 Unauthorized | Authorization validation failed. E.g., a request without a token. |
| 404 Not Found | A task with provided title does not exists in user to-do list. |
| 505 Internal Server Error | Something went wrong on the server |

|  |  |
| --- | --- |
| **GET**/api/tasks/done | |
| Can Access: | Registered User |
| Request Headers | |
| Authorization | Bearer YOUR\_TOKEN\_HERE |
| Responses: | |
| 200 Ok | Successfully returned done tasks. Returns empty array if no tasks. |
| 401 Unauthorized | Authorization validation failed. E.g., a request without a token. |
| 505 Internal Server Error | Something went wrong on the server |
| Response body on success: | |
| Array of {      done: boolean;      title: string;      description: string;  } | |

|  |  |
| --- | --- |
| **PUT**/api/tasks/:taskTitle | |
| Can Access: | Registered User |
| Request Headers | |
| Authorization | Bearer YOUR\_TOKEN\_HERE |
| Request body: | |
| {      title: string;      description: string;  } | |
| Responses: | |
| 200 Ok | Successfully updated a task. |
| 400 Bad Request | Schema validation failed. Or user tries to set a new title that already exists in user tasks list. |
| 401 Unauthorized | Authorization validation failed. E.g., a request without a token. |
| 404 Not Found | A task with such taskTitle is not found in user tasks. |
| 505 Internal Server Error | Something went wrong on the server |

|  |  |
| --- | --- |
| **DELETE**/api/tasks/ | |
| Can Access: | Registered User |
| Request Headers | |
| Authorization | Bearer YOUR\_TOKEN\_HERE |
| Request body: | |
| {      title: string;  } | |
| Responses: | |
| 200 Ok | Successfully deleted a task. Also, if task with provided title does not exists |
| 400 Bad Request | Schema validation failed. |
| 401 Unauthorized | Authorization validation failed. E.g., a request without a token. |
| 505 Internal Server Error | Something went wrong on the server |

**Assessment checklist:**

1. Validate that provided API tests succeeds.
2. Validate that user created services that are exposing only async function
3. Validate that user correctly implemented JWT auth.
4. Validate that user handles errors and logs them.