Project A

Step 1 : Set up the Express application

You will first create your Express application.

There are tools that allow you to automatically generate the structure of your application: leave them aside for now. Here you will put your application in a simple `index.js` file.

We will not use a database right away. Instead, you will use the “hard-coded” data in the attached `data.js` file.

You'll soon realize that by working this way, all changes made to the `users` and `schedules` data are erased every time you restart your application... yes, that's why we'll later use a database! But for now, having some “hard-coded" data will let you learn how to manipulate Express routes and return the right information, without mixing everything with databases. First things first, patience!

Step 2 : Create the first routes to return all the information

You will create the following routes :

- ‘/', which returns the message "Welcome to our schedule website".
- ‘/users’, which returns the list of users
- ‘/schedules’, which returns the list of schedules

You can use the `curl` command to make your http queries from the command line. Of course, you can also make your requests in the browser, but doing it from the command line can help you understand how HTTP requests work. ☺

For example, here is how to make requests (here, Express is configured to listen on port 3000) :

```
$> curl localhost:3000/users
[{"firstname":"James","lastname":"Bond","email":"james.bond@gmail.com","password":"b6b7fb4cad4bc020f76e16889a8e9065cb708d0f8c304a8a3db609b644da9536"},
{"firstname":"Tony","lastname":"Stark","email":"starkrulz@gmail.com","password":"a836ebba36776b21dd0f5cdca497bff65c5bdfc8411cfbfe0111f27bde1c1894"},
{"firstname":"Ali","lastname":"G","email":"nameisnotborat@gmail.com","password":"3b5fe14857124335bb8832cc602f8edcfa12db42be36b135bef5bca47e3f2b9c”}]
$>
```

By default, `curl` makes GET requests, but we can change this behavior via options (see Step 4).

Step 3 : Create parameterized routes

You will then create routes to return the information for a given user.

Thus:

- the URL '/users/2' will return the information of user n°2
- the URL '/users/2/schedules' will return a list of all schedules for user n°2

Of course you don't have to create a route for every user. Imagine if we have 200 users, we are not going to create 200 routes!

Instead, look at URL parameters to see how to handle this effectively.

```
$> curl localhost:3000/users/2
{"firstname":"Ali","lastname":"G","email":"nameisnotborat@gmail.com","password":"3b5fe14857124335bb8832cc602f8edcfa12db42be36b135bef5bca47e3f2b9c”}
$> curl localhost:3000/users/1
{"firstname":"Tony","lastname":"Stark","email":"starkrulz@gmail.com","password":"a836ebba36776b21dd0f5cdca497bff65c5bdfc8411cfbfe0111f27bde1c1894”}
$> curl localhost:3000/users/0/schedules
[{"user_id":0,"day":1,"start_at":"2PM","end_at":"4PM"},{"user_id":0,"day":2,"start_at":"2PM","end_at":"4PM"},{"user_id":0,"day":3,"start_at":"2PM","end_at":"4PM"}]
$>
```

Step 4 : Create routes to update data

Finally, you will create routes to add new users and new schedules.

For this, you will have to create the following routes in POST :

- ‘/schedules’ to add a new schedule. It will return the newly created schedule.
- ‘/users’ (this time in POST!) to add a new user. It will return the newly created user. The user's password must be encrypted in SHA256.

To test these routes, you will need to make POST requests and send data as if there was a form.

This is done using the -X POST option (to specify that the request is in POST), combined with the -d option to specify the data attached to the request.

Here is an example of a command line session:

```
$> curl localhost:3000/users
[{"firstname":"James","lastname":"Bond","email":"james.bond@gmail.com","password":"b6b7fb4cad4bc020f76e16889a8e9065cb708d0f8c304a8a3db609b644da9536"},
{"firstname":"Tony","lastname":"Stark","email":"starkrulz@gmail.com","password":"a836ebba36776b21dd0f5cdca497bff65c5bdfc8411cfbfe0111f27bde1c1894"},
{"firstname":"Ali","lastname":"G","email":"nameisnotborat@gmail.com","password":"3b5fe14857124335bb8832cc602f8edcfa12db42be36b135bef5bca47e3f2b9c”}]

$> curl localhost:3000/users/3

$> curl -d "firstname=Donald&lastname=Duck&email=coincoin@gmail.com&password=daisy" -X POST localhost:3000/users
{"firstname":"Donald","lastname":"Duck","email":"coincoin@gmail.com","password":"QgKe8hUlb4+p/ttTVC7mVT7vdgJ7EW+PrFNGIRseRzw="}Mezards-MacBook-
Pro:dmaiga laurie$

$> curl localhost:3000/users
[{"firstname":"James","lastname":"Bond","email":"james.bond@gmail.com","password":"b6b7fb4cad4bc020f76e16889a8e9065cb708d0f8c304a8a3db609b644da9536"},
{"firstname":"Tony","lastname":"Stark","email":"starkrulz@gmail.com","password":"a836ebba36776b21dd0f5cdca497bff65c5bdfc8411cfbfe0111f27bde1c1894"},
{"firstname":"Ali","lastname":"G","email":"nameisnotborat@gmail.com","password":"3b5fe14857124335bb8832cc602f8edcfa12db42be36b135bef5bca47e3f2b9c"},
{"firstname":"Donald","lastname":"Duck","email":"coincoin@gmail.com","password":"QgKe8hUlb4+p/ttTVC7mVT7vdgJ7EW+PrFNGIRseRzw="}]

$> curl localhost:3000/users/3
{"firstname":"Donald","lastname":"Duck","email":"coincoin@gmail.com","password":"QgKe8hUlb4+p/ttTVC7mVT7vdgJ7EW+PrFNGIRseRzw=“}
```

Terminal session walkthrough

First, we query all users, and we see that there are only 3 of them.
We check that user 3 does not exist -> when we query /users/3, nothing is returned.

We then add a user by making a POST request with the data we want to add. A user is returned to us.
Then, when we query all users again, we see that our 4th user has been added to the list of users.
And this time, when we request /users/3, our new user is returned.

Beware, we remind you that these changes on the data are not persistent! If you restart your Express application, you will only have the 3 initial users again.