



API Testing

I. Case studies

Use the APIs listed here (<https://regres.in/>) and create API automation test suite as well as scripting it.

II. Requirements for solution

- ☒ ~~Write your code in any programming language of your choice~~
- ☐ Create a local git repository like you would in a team setting and push your code to GitHub or any other publicly available repository service.
- ☒ ~~Include instructions for setting up and running your code..~~
- ☒ ~~Write an API automation to cover all or some cases from the above list~~
- ☒ ~~Tests to be runnable from the command line~~
- ☒ ~~Tests to be configurable to run on various OSes (Mac, Linux etc)~~
- ☒ ~~Tests to be able to run in parallel~~
- ☒ ~~A README for the following information to explain the problem and solution in a concise manner.~~
- ☒ ~~Steps to run the test suite/case~~
- ☒ ~~Information on any architectural decisions eg. choice of tech stack, code structure and any trade offs made~~

III. Solutions



In this test assignment, I choose Postman since the following benefits

- Easy to set up. You do not need to set up an environment variable or something like this.
- Can run tests on any machine and any time.
- Quick returns to the result
- Easy to learn and use. Maybe just one hour for a walkthrough then all team members can do it. Reduce risk for resource management, you will not pressure any changes in the resources such as it's difficult to find the candidate that fits with these skills or it takes a long time to find a new one as hiring processes or so on.
- Cost for maintenance is low

IV. API Automation Test Suite

According to the API document at <https://regres.in/> then the test suite is divided as follows

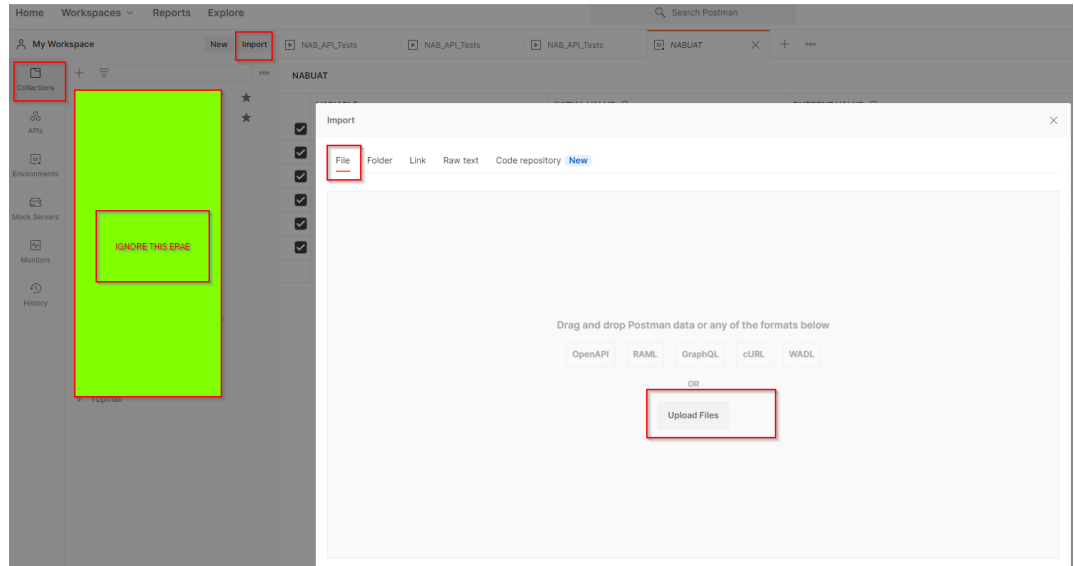
- Create user
 - Create user unsuccessfully if missing name
 - Create user unsuccessfully if using the existing name
 - Create user successfully
- Find user
 - Find user successfully - valid user id
 - Find user unsuccessfully - invalid user id
 - Find user unsuccessfully - wrong format user id (should be integer only)
- Update user
 - Update user successfully with put method
 - Update user unsuccessfully with put method - missing name

- Update user unsuccessfully with put method - missing job
- Update user unsuccessfully with put method - missing both job & name - bug
- Update user unsuccessfully with put method - user id not found
- Update user successfully with patch method - only name
- Update user successfully with patch method - only job
- Update user successfully with patch method - both name & job
- Update user unsuccessfully with patch method - missing both job & name
- Register user
 - Register user successfully
 - Register user unsuccessfully - missing email
 - Register user unsuccessfully - missing password
 - Register user unsuccessfully - missing both email & password
 - Register user unsuccessfully - have password, but wrong email
 - Check password rule
- Login with user
 - Login successfully - correct email & password
 - Login unsuccessfully - missing email
 - Login unsuccessfully - missing password
 - Login unsuccessfully - missing both email & password
 - Login unsuccessfully - correct email, but wrong password
 - Login unsuccessfully - wrong email, correct password
 - Login unsuccessfully - wrong both email & password
- Find single resource
 - Find a resource successfully - user id found
 - Find a resource unsuccessfully - user id not found
 - Find a resource unsuccessfully - user id not correct format
- Find list of resources
 - There is no resource
 - There are 2 resources
 - There are 6 resources
 - There are 7 resources
 - There are 12 resources
- Delete user:
 - Delete user successfully

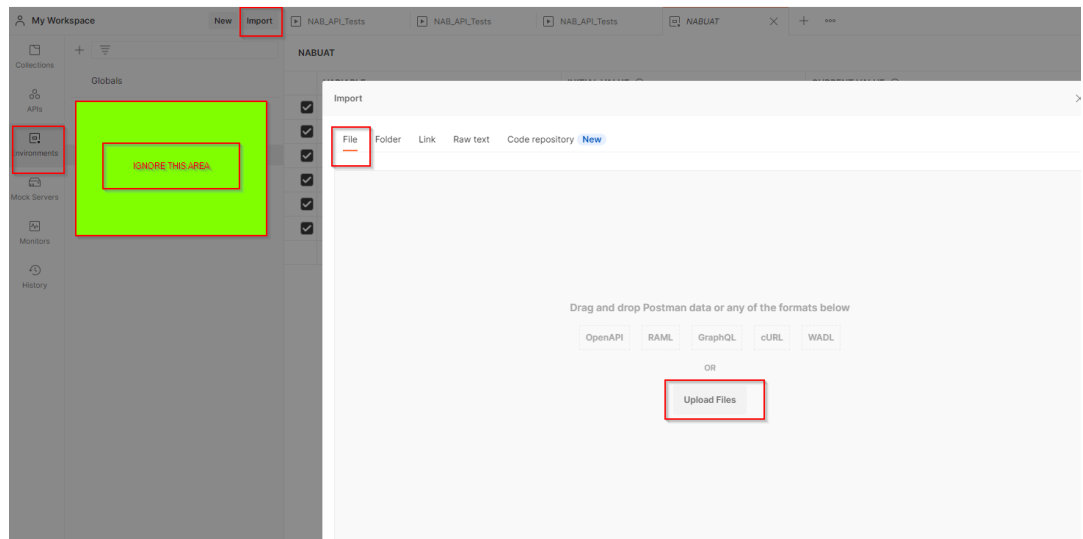
V. Guideline (readme file)

- ☐ Download and install postman according to your OS version at <https://www.postman.com/downloads/>

☐ Import the collection

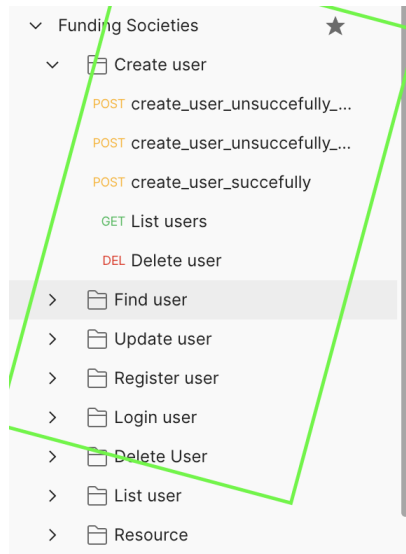


☐ Import the environment



☐ Checking the script

After imported then you should see the code looks like this



This authorization method will be used for every request in this folder. You can over one in the request.

Type

Inherit auth from parent

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#) ↗

This folder is using No Auth from collection [Funding Societies](#).

CONGRATULATIONS!! NOW YOUR CODE IS READY TO RUN

VI. Run tests

Two ways to run the test

1. Run for the specific endpoint in the collections folder
 - a. Just choose your request then click send button



- b. Click on the 'Test' tab to view the script for each endpoint as below screenshot

POST ▼ {{url}}/api/users

Params Authorization Headers (9) Body ● Pre-request Script **Tests ●** Settings

```
1 pm.test("Status code is 403", function () {
2   pm.response.to.have.status(403); //The correct status base on the req
3 });
```

Body Cookies Headers (13) **Test Results (0/1)** 🌐 Status: 201 Created Time: 5

All Passed Skipped Failed

FAIL Status code is 403 | AssertionError: expected response to have status code 403 but got 201

c. View test result

The screenshot displays the Postman interface for a POST request to `{{url}}/api/users`. The 'Tests' tab is active, showing a test script with three lines of code:

```
1 pm.test("Status code is 403", function () {  
2   pm.response.to.have.status(403); //The correct status base on the req  
3 });
```

Below the script, the 'Test Results (0/1)' tab is selected, showing a single failed test result. The result is marked with a red 'FAIL' button and the message: 'Status code is 403 | AssertionError: expected response to have status code 403 but got 201'. The status bar at the bottom indicates 'Status: 201 Created' and 'Time: 5'.

d. You can view the details for the response by clicking on the Body tab as the screenshot

POST {{url}}/api/users

Params Authorization Headers (9) Body ● Pre-request Script Tests ● Settings

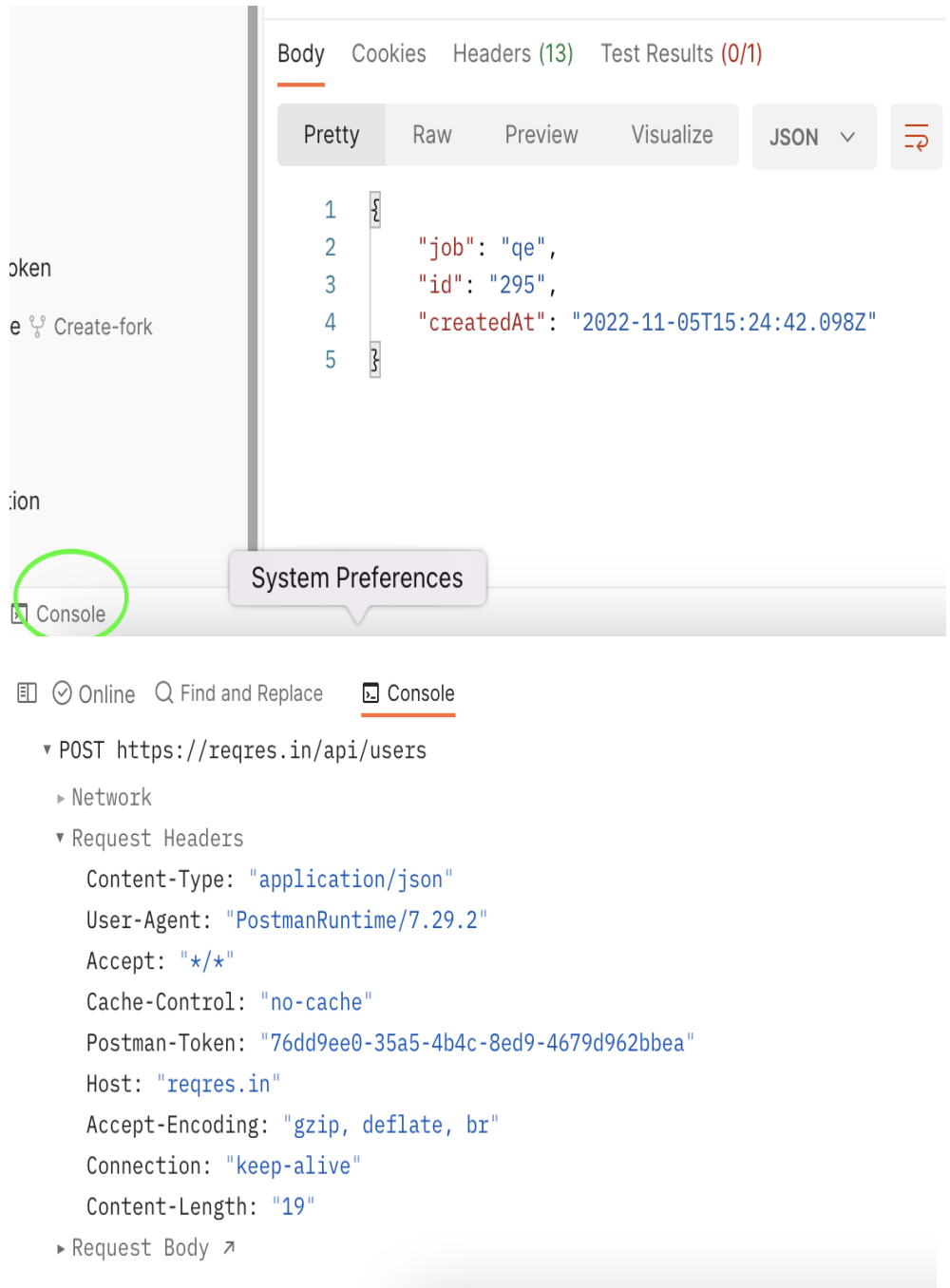
```
1 pm.test("Status code is 403", function () {
2   pm.response.to.have.status(403); //The correct status base on the req
3 });
```

Body Cookies Headers (13) Test Results (0/1) Status: 201 Created

Pretty Raw Preview Visualize JSON

```
1 {
2   "job": "qe",
3   "id": "295",
4   "createdAt": "2022-11-05T15:24:42.098Z"
5 }
```

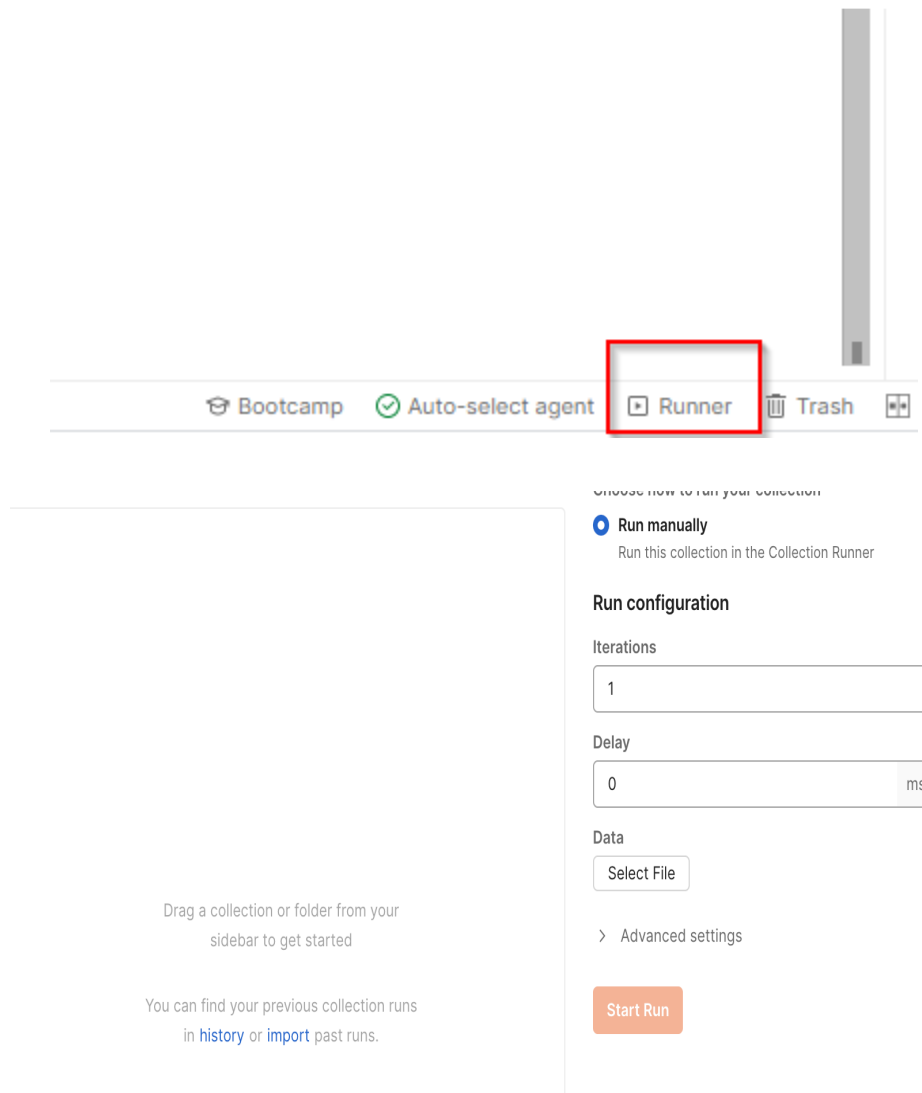
- e. View the log file by clicking on the Console tab at the bottom left corner



2. Run one more tests via Collection Runner

Another way to run your test is by using the Collection Runner where you can run all your test suites as well as define the flow as you want.





















- a. First, click on the Runner on the bottom right corner



- b. Second, drag a collection or folder into the RUNNER ORDER section. And you should see something like this

RUN ORDER

Deselect All | Select All | Reset

- ☒  > **POST** create_user_unsuccessfully_missing_name
- ☒  > **POST** create_user_unsuccessfully_duplicate_name
- ☒  > **POST** create_user_successfully
- ☒  > **GET** List users
- ☒  > **DEL** Delete user
- ☒  > **GET** Find existing user - successful
- ☒  > **GET** Find a new creation user - successful
- ☒  > **GET** Find a user with wrong id format - fail
- ☒  ☒  > **GET** Find a user with id not found - fail
- ☒  > **PUT** Update user successfully with put method
- ☒  > **PUT** Update user unsuccessfully with put method - missing name
- ☒  > **PUT** Update user unsuccessfully with put method - missing job
- ☒  > **PUT** Update user unsuccessfully with put method - missing job C
- ☒  > **PUT** Update user unsuccessfully with put method - missing both
- ☒  > **PATCH** Update user successfully with patch method - only name
- ☒  > **PATCH** Update user successfully with patch method - only job
- ☒  > **PATCH** Update user successfully with patch method - include both
- ☒  > **PATCH** Update user unsuccessfully with patch method - missing both
- ☒  > **POST** Register user successfully by using valid email

- c. You can deselect tests that you don't want to run by unchecking the checkbox.

- d. Set up and run your test

Choose how to run your collection

☒ **Run manually**

Run this collection in the Collection Runner

Run configuration

Iterations

Delay

 ms

Data

✓ Advanced settings

☐ Save responses ⓘ

☒ Keep variable values ⓘ

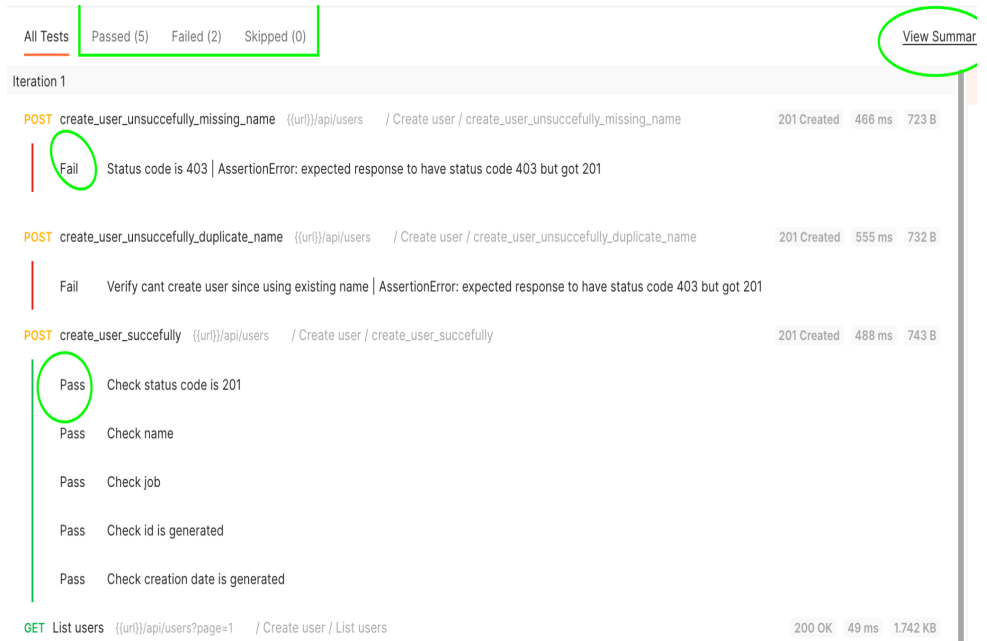
☐ Run collection without using stored cookies

☒ Save cookies after collection run ⓘ

Run Funding Societies

e. View test result

Test reports provide the status Pass/Fail/Skipped as well as details for each test. You also can view the summary by clicking on the hyperlink on the right corner as below screenshot.



3. Run test via command line

More details can find at

<https://learning.postman.com/docs/running-collections/using-newman-cli/installing-running-newman/>

Here is the basic steps need to do

- Install Node js <https://nodejs.org/en/download/package-manager/>
- Install Newman

Install Newman from npm globally on your system, which allows you to run it from anywhere:

```
$ npm install -g newman
```

c. Run your tests

```
Xuyens-MacBook-Pro:~ XuyenTran$ newman run /Users/xuyenauto/Documents/Funding_Societies.postman_collection.json -e /Users/xuyenauto/Documents/funding.postman_environment.json
newman
newman run your_collection -e your_env

Funding Societies

□ Create user
↳ create_user_unsuccessfully_missing_name
  POST https://reqres.in/api/users [201 Created, 718B, 743ms]
  1. Status code is 403

↳ create_user_unsuccessfully_duplicate_name
  POST https://reqres.in/api/users [201 Created, 723B, 544ms]
  2. Verify cant create user since using existing name

↳ create_user_succefully
  POST https://reqres.in/api/users [201 Created, 735B, 490ms]
  ✓ Check status code is 201
  ✓ Check name
  ✓ Check job
  ✓ Check id is generated
  ✓ Check creation date is generated

↳ List users
  GET https://reqres.in/api/users?page=1 [200 OK, 1.74kB, 65ms]

↳ Delete user
```

d. Test result after finished

	executed	failed
iterations	1	0
requests	38	0
test-scripts	35	0
prerequest-scripts	0	0
assertions	59	13
total run duration: 16.7s		
total data received: 4.99kB (approx)		
average response time: 420ms [min: 43ms, max: 743ms, s.d.: 181ms]		

e. You can export the result by using reporters option

```
Xuyens-MacBook-Pro:~ XuyenTran$ newman run /Users/xuyenauto/Documents/Funding_Societies.postman_collection.json -e /Users/xuyenauto/Documents/funding.postman_environment.json --reporters cli,json --reporter-json-export outputfile.json
newman

Funding Societies

□ Create user
↳ create_user_unsuccefully_missing_name
  POST https://reqres.in/api/users [201 Created, 720B, 710ms]
    1. Status code is 403

Xuyens-MacBook-Pro:~ XuyenTran$ ls
#.profile#      Downloads      Postman        cache.ga      skype-export
Applications    Library        Projects       firmware
Creative Cloud  Movies        Public         node_modules
Desktop          Music          Ruby           outputfile.json
Documents       Pictures      VirtualBox    VMs          package-lock.json
Xuyens-MacBook-Pro:~ XuyenTran$
```

f. For more options, please visit

<https://learning.postman.com/docs/running-collections/using-newman-cli/newman-options/>

VII. Test Report

Test suite	Scenario	Status	Note
Create user	Create user unsuccessfully if missing name	FAILED	It should NOT allow to create user without name
	Create user unsuccessfully if using the existing name	FAILED	It should NOT allow to create user existing name
	Create user successfully	PASSED	
Find user	Find a new creation user - successful	FAILED	Unable to find a new user id when successful to create a new one
	Find existing user - successful	PASSED	
	Find a user with id not found - fail	PASSED	

Update user	Update user successfully with put method	PASSED	
	Update user unsuccessfully with put method - missing name	FAILED	
	Update user unsuccessfully with put method - missing job	FAILED	
	Update user unsuccessfully with put method - missing both name and job	FAILED	
	Update user successfully with patch method - only name	PASSED	
	Update user successfully with patch method - only job	PASSED	
	Update user successfully with patch method - include both name & job	PASSED	
	Update user unsuccessfully with patch method - missing both name & job	FAILED	
Register user	Update user unsuccessfully with patch method - missing both name & job	PASSED	
	Update user unsuccessfully with patch method - missing both name & job	FAILED	
	Register user unsuccessfully - missing email	PASSED	
	Register user unsuccessfully - missing username	PASSED	
	Register user unsuccessfully - missing username	PASSED	
	Register user unsuccessfully - missing both username & password	PASSED	
	Register user unsuccessfully - have password, but wrong email	PASSED	
Login user	Login successfully - correct email & password	PASSED	
	Login unsuccessfully - missing email	PASSED	
	Login unsuccessfully - missing password	PASSED	
	Login unsuccessfully - missing both email & password	PASSED	

	Login unsuccessfully - correct email, but wrong password	FAILED	
	Login unsuccessfully - wrong email, correct password	PASSED	
	Login unsuccessfully - wrong both email & password	PASSED	
Delete user	Delete user successfully	PASSED	
	Find deleted user	FAILED	
List user	List user of page 1	PASSED	
Resource	Single resource - Found	PASSED	
	Single resource - Not found	PASSED	
	List resource	PASSED	