

Postman Mini Project

Getting started with postman:

Installations:

1. First download Postman : <https://postman.com/downloads>
2. Check to see if Node is installed using terminal : node - -version ([Node.js](#) and npm: <https://nodejs.org/>)
3. If not, then install the node.js
4. Once you install [node.js](#) , npm is automatically install
5. Go to terminal and type npm –version to check the version
6. Next, install Newman using terminal: npm install -g newman

```
anumsharma@Anum-MacBook-Pro ~ % node --version  
v22.16.0  
anumsharma@Anum-MacBook-Pro ~ % npm --version  
10.9.2  
anumsharma@Anum-MacBook-Pro ~ %
```

Postman Terms:

1. Collections: all the requests and assertions for an API; the top level folder
2. Folder: a group of requests and assertions that reside in the collection

Open Postman

Create a new workspace - Postman Essential

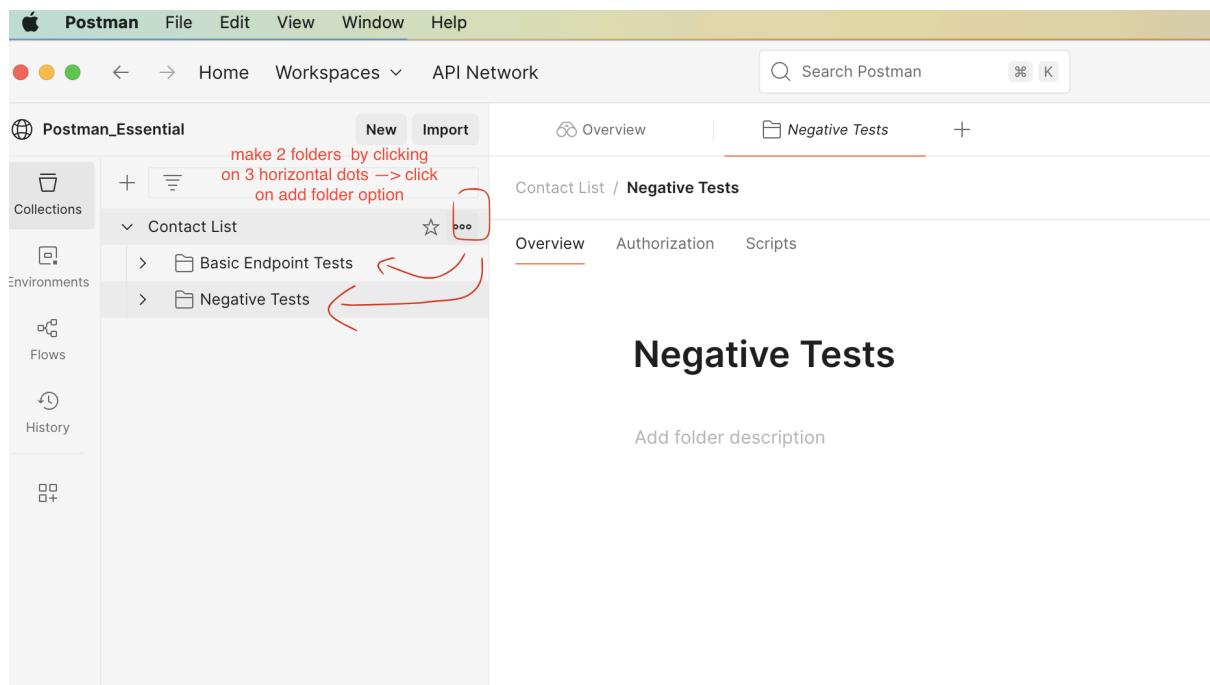
The screenshot shows the Postman application interface. At the top, the menu bar includes Postman, File, Edit, View, Window, Help, and a search bar. On the right, there are icons for Invite, Connect, Unwatch, and Upgrade. The main area displays a workspace named "Postman_Essential". A red arrow points from the "New" button in the top navigation to the "Create Collection" button in the center of the workspace. Another red arrow points to the "workspace name" field, which is currently set to "Postman_Essential". The left sidebar shows sections for Environments, Flows, and History. The central workspace contains a collection titled "My first collection" with two folders: "First folder inside collection" and "Second folder inside collection", each containing several requests.

Create a new collection - Contact List

This screenshot shows a new collection named "Contact List" within the same workspace. A red arrow points from the "Collections" section in the sidebar to the "Contact List" entry in the main navigation bar. The collection itself is currently empty, as indicated by the message "This collection is empty. Add a request to start working." The right side of the screen provides a detailed view of the collection, including tabs for Overview, Authorization, Scripts, Variables, and Runs. It also features a "New collection - Contact List" header, a "Contact List" title, and a note encouraging users to "Make things easier for your teammates with a complete collection description." A "View complete documentation →" link is also present. The collection was created by the user "You".

Under New Collection - add 2 new folders :

- Basic endpoint tests
- Negative tests



How to Create Postman Requests and Assertions?

We will use API called “Contact List API”

We have created Contact List API

URL: <https://thinking-tester-contact-list.herokuapp.com/>

It allows users to add, edit and delete contacts like they would in an address book

This API also has a user Interface, or UI to go with it.

Let do that → follow below steps :

Navigate to <https://thinking-tester-contact-list.herokuapp.com/>

Create an account which we will use for API call

Click on sign up button

Contact List App

Welcome! This application is for testing purposes only. The database will be purged as needed to keep costs down.

The API documentation can be found [here](#).

Log In:

Email

Password

Submit

Not yet a user? Click here to sign up!

Sign up

Add first name , last name , email, password and submit button
NOTE - note down the email and password . this will use in POST method

Add User

Sign up to begin adding your contacts!

anu

S

anu@gmail.com

Submit

Cancel



User land on contact list page

Contact List

[Logout](#)

Click on any contact to view the Contact Details

[Add a New Contact](#)



Name	Birthdate	Email	Phone	Address	City, State/Province, Postal Code	Country
------	-----------	-------	-------	---------	-----------------------------------	---------

Let add new contact in order to see in our GET request

Contact List

[Logout](#)

Click on any contact to view the Contact Details

[Add a New Contact](#)

Name	Birthdate	Email	Phone	Address	City, State/Province, Postal Code	Country
Mary Marshall	1969-04-04	mary@yahoo.com	9094481212	22 Elm St. Apt 2	Newport NH 03773	USA
Anil Shetty	1965-01-01	anil@gmail.com	3434564545	123 Sesame St	New York NY 01234	USA

How JSON web tokens (JWTs)work?

The most common way to secure API's

Generated through a POST request

When a valid username and password are sent in the request, a JWT is generated

The JWT can then be used in all requests to the API

So let's generate JWT

Go back to POSTMAN

1. Under Contact List collection → Basic Endpoints Tests folder → add a new request
2. Give it name - Login
3. Select Method- POST
4. Enter the URL - <https://thinking-tester-contact-list.herokuapp.com/users/login>
5. Under body section → select Raw radio button + JSON format and paste the below JSON :

```
{  
  "Email" : "enter your email address you entered while signup",  
  "Password": "enter the password"  
}
```
6. Click on “SEND”button
7. User can see response with 200 OK in response section
8. Save the token value
9. Save the login request by clicking on the save button on the top right side.

The screenshot shows the Postman application interface with the following annotations:

1. add new request: A red circle highlights the "New" button in the top navigation bar.
2. select the POST method: A red circle highlights the "POST" method dropdown in the request details panel.
3. enter URL: A red circle highlights the URL input field containing "https://thinking-tester-contact-list.herokuapp.com/users/login".
4. Enter raw data in json format: A red circle highlights the JSON raw data input field, which contains the following JSON:

```
1 {  
2   "email": "anuS@gmail.com",  
3   "password": "anu@123"  
4 }
```

5. click on SEND button: A red circle highlights the "Send" button in the top right corner of the request details panel.
6. got the token in response section . save it: A red circle highlights the token value in the response JSON, which is "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOjI2ODUxZDE5ZWl2ZDQjNzAwMTVmYmMyNjAiLCJpYXQiOjE3NTAxOTU1ODh9.96xu10esh92CLqtAdxe5ufYM1Z00NH6K_cwcRC1G_WQ".

Requests to the Contacts List API

1. GET request : returns all contacts
2. GET/{contactID} request: returns one specific contact
3. POST request - adds a contact
4. PUT/{contactId} request - updates a specific contact
5. DELETE/{contactId} request - deletes a specific contact

Postman Documentation:

<https://documenter.getpostman.com/view/4012288/2s8YRiKDbu> (helps in getting URL link)

-----Back to Postman-----

GET method - get all contacts using postman :

1. Create a new request - Get Contact List
2. Select the method - GET
3. Enter the URL-
4. Go to Authorization tab
5. Select the Auth Type - Bearer token
6. Enter the token value you save in above steps
7. Click on "send" button
8. User will see lists in response section
9. Save the request by clicking on Save button on the top right

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'n_Essential' and 'Contact List' sections. Under 'Contact List', 'Basic Endpoint Tests' is expanded, showing 'POST Login' and 'GET Get Contact List'. A red box highlights 'GET Get Contact List' with the label '1. add a new request - Get Contact List'. The main workspace shows a 'Get Contact List' request. Step 2, 'select GET method', points to the 'GET' button. Step 3, 'enter the URL', points to the URL field containing 'https://thinking-tester-contact-list.herokuapp.com/contacts'. Step 4, 'Select Auth Type - Bearer token', points to the 'Bearer Token' dropdown in the Authorization tab. Step 5, 'enter the token value', points to a redacted token value in the token field. Step 6, 'click on "Send" button', points to the blue 'Send' button. Step 7, 'User can see the lists under response section', points to the JSON response body which displays a list of contact details. Step 8, 'Save the request', points to the 'Save' button in the top right corner.

2. select GET method

3. enter the URL

4. Select Auth Type - Bearer token

5. enter the token value

6. click on "Send" button

7. User can see the lists under response section

GET method - get specific contact from list using postman :

1. Create a new request in easy way
2. Hover over Get Contact List request
3. Click on 3 dots menu
4. Select the duplicate options

The screenshot shows the Postman interface with a workspace named 'man_Essential'. In the left sidebar, under 'Contact List / Basic Endpoint Tests', there is a 'Basic Endpoint Tests' folder containing a 'POST Login' request and a 'GET Get Contact List' request. A red arrow points to the three-dot menu next to the 'GET Get Contact List' request, and another red box highlights the 'Duplicate' option in the dropdown menu.

5. New request is added
6. Rename the request by clicking on 3 dots menu and select rename option

The screenshot shows the Postman interface with the same workspace and endpoint structure. The 'GET Get Contact List' request has been renamed to 'Get Contact List Copy'. A red arrow points to the three-dot menu next to the renamed request, and a red box highlights the 'Rename' option in the dropdown menu.

7. Make a request a - Get Contacts
8. Select the method - GET
9. Enter the URL -
<https://thinking-tester-contact-list.herokuapp.com/contacts/6851d260b6d2c70015fbc263>
Where 6851d260b6d2c70015fbc263 is contact ID

10. Get the contacts ID value from Get Contacts List response

The screenshot shows the Postman interface with the 'Get Contact List' request selected. The 'Authorization' tab is open, showing 'Bearer Token' selected. The 'Body' tab displays a JSON response with an array of contacts. A red box highlights the '_id' field of the first contact, which is '6851d260b6d2c70015fbc263'. The response status is 200 OK.

```
[{"_id": "6851d260b6d2c70015fbc263", "firstName": "Mary", "lastName": "Marshall", "birthdate": "1969-04-04", "email": "mary@yahoo.com", "phone": "9994481212", "street1": "22 Elm St. Apt 2", "city": "Newport", "stateProvince": "NH", "postalCode": "03773", "country": "USA", "owner": "6851d19eb6d2c70015fbc260"}]
```

11. Go to Authorization tab

12 Select the Auth Type - Bearer token

13. Enter the token value you save in above steps

14. Click on "send" button

15. User will see lists in response section

16. Save the request by clicking on Save button on the top right

The screenshot shows the Postman interface with the 'Get Contact' request selected. The 'Authorization' tab is open, showing 'Bearer Token' selected. The 'Body' tab displays a JSON response with a single contact. A red box highlights the '_id' field of the contact, which is '6851d260b6d2c70015fbc263'. The response status is 200 OK.

```
{"_id": "6851d260b6d2c70015fbc263", "firstName": "Mary", "lastName": "Marshall", "birthdate": "1969-04-04", "email": "mary@yahoo.com", "phone": "9994481212", "street1": "22 Elm St. Apt 2", "city": "Newport", "stateProvince": "NH", "postalCode": "03773", "country": "USA", "owner": "6851d19eb6d2c70015fbc260"}
```

POST method - add contacts using postman :

1. Create a new request - Add Contact
2. Select the method - POST
3. Enter the URL- <https://thinking-tester-contact-list.herokuapp.com/contacts> and under body tab enter raw data in json format

```
{  
    "firstName": "Prunella",  
    "lastName": "Prunewhip",  
    "birthdate": "1977-07-07",  
    "email": "pprunewhip@fake.com",  
    "phone": "8005551000",  
    "street1": "123 Main St.",  
    "street2": "Apartment Q",  
    "city": "New York",  
    "stateProvince": "NY",  
    "postalCode": "12345",  
    "country": "USA"  
}
```

NOTE: User can get the URL from postman documentation

<https://documenter.getpostman.com/view/4012288/2s8YRiKDbu#3cf734e9-f970-4431-b7d2-2cd04a3614be>

The screenshot shows the Postman interface with the following details:

- ENVIRONMENT:** No Environment
- LAYOUT:** Double Column
- LANGUAGE:** cURL - cURL
- Request Type:** POST
- Request URL:** https://thinking-tester-contact-list.herokuapp.com/contacts
- Headers:** Authorization: Bearer {{token}}
- Body:** raw (json)
- JSON Body Content:**

```
{  
    "firstName": "Prunella",  
    "lastName": "Prunewhip",  
    "birthdate": "1977-07-07",  
    "email": "pprunewhip@fake.com",  
    "phone": "8005551000",  
    "street1": "123 Main St.",  
    "street2": "Apartment Q",  
    "city": "New York",  
    "stateProvince": "NY",  
    "postalCode": "12345",  
    "country": "USA"  
}
```
- Example Request (curl):**

```
curl --location 'www.thinking-tester-contact-list.herokuapp.com/contacts' \  
--header 'Authorization: Bearer {{token}}' \  
--data-raw '{  
    "firstName": "Prunella",  
    "lastName": "Prunewhip",  
    "birthdate": "1977-07-07",  
    "email": "pprunewhip@fake.com",  
    "phone": "8005551000",  
    "street1": "123 Main St.",  
    "street2": "Apartment Q",  
    "city": "New York",  
    "stateProvince": "NY",  
    "postalCode": "12345",  
    "country": "USA"  
}'
```
- Example Response:**

```
{  
    "_id": "6360732107537d0013a7ea1",  
    "firstName": "Prunella",  
    "lastName": "Prunewhip",  
    "birthdate": "1977-07-07",  
    "email": "pprunewhip@fake.com",  
    "phone": "8005551000",  
    "street1": "123 Main St.",  
    "street2": "Apartment Q",  
    "city": "New York",  
    "stateProvince": "NY",  
    "postalCode": "12345",  
    "country": "USA"  
}
```

4. Go to Authorization tab
5. Select the Auth Type - Bearer token
6. Enter the token value you save in above steps
7. Save the request by clicking on Save button on the top right
8. Click on “send” button
9. User will see lists in response section

The screenshot shows the Postman interface with a collection named 'Postman_Essential'. Under 'Contact List', there is a 'Basic Endpoint Tests' folder containing 'POST Login', 'GET Get Contact List', 'GET Get Contact', and 'POST Add Contacts'. The 'POST Add Contacts' request is selected. The URL is set to <https://thinking-tester-contact-list.herokuapp.com/contacts>. The 'Body' tab is selected, showing a raw JSON payload:

```

1 {
2   "_id": "685286abb6d2c70015fbc60d",
3   "firstName": "Prunella",
4   "lastName": "Prunewhip",
5   "birthdate": "1977-07-07",
6   "email": "pprunewhip@fake.com",
7   "phone": "8005551000",
8   "street1": "123 Main St.",
9   "street2": "Apartment Q",
10  "city": "New York",
11  "stateProvince": "NY",
12  "postalCode": "12345",
13  "country": "USA"
14 }

```

The response shows a 201 Created status with 88 ms and 1.05 KB. A red checkmark is placed over the JSON tab in the Body section.

POST method - add contacts using postman :

1. Create a new request - Update Contact
2. Select the method - PUT
3. Enter the URL-

<https://thinking-tester-contact-list.herokuapp.com/contacts/685286abb6d2c70015fbc60d> and under body tab update raw data(first name, last name, email, address , postal code) in json format

```
{
  "firstName": "Neal",
  "lastName": "Smith",
  "birthdate": "1980-05-04",
  "email": "nealsmith@fake.com",
  "phone": "8005551000",
  "street1": "566 Main St.",
  "street2": "Apartment Q",
  "city": "New York",
  "stateProvince": "NY",
  "postalCode": "45545",
  "country": "USA"
}
```

NOTE: User can get the URL from postman documentation

<https://documenter.getpostman.com/view/4012288/2s8YRiKDbu#3cf734e9-f970-4431-b7d2-2cd04a3614be>

4. Go to Authorization tab
5. Select the Auth Type - Bearer token
6. Enter the token value you save in above steps

7. Save the request by clicking on Save button on the top right

8. Click on “send” button

9. User will see lists in response section

The screenshot shows the Postman interface with a collection named "Postman_Essential". Under "Contact List", there is a "Basic Endpoint Tests" folder containing several requests: "POST Login", "GET Get Contact List", "GET Get Contact", "POST Add Contacts", and "PUT Update Contact". The "PUT Update Contact" request is selected. The URL is <https://thinking-tester-contact-list.herokuapp.com/contacts/685286abb6d2c70015fbc60d>. The "Body" tab is selected, showing a JSON payload:

```
1 {  
2   "firstName": "Neal", update the values  
3   "lastName": "Smith",  
4   "birthdate": "1980-05-04",  
5   "email": "nealsmith@fake.com",  
6   "phone": "8005551000",  
7   "street1": "566 Main St.",  
8   "street2": "Apartment Q",  
9   "city": "New York",  
10  "stateProvince": "NY",  
11  "postalCode": "45545",  
12  "country": "USA"  
13 }  
14  
15 }
```

Red annotations highlight the "Body" tab, the URL, the "firstName" field, and the "Send" button. The response section shows a 200 OK status with a response time of 103 ms and a size of 1.04 KB. The response body is identical to the request body, indicating no changes were made.

DELETE_method - delete contacts using postman :

1. Create a new request - Delete Contact

2. Select the method - DELETE

3. Enter the URL-

<https://thinking-tester-contact-list.herokuapp.com/contacts/685286abb6d2c70015fbc60d>

Where 685286abb6d2c70015fbc60d is the ID number of contact

4. Go to Authorization tab

5. Select the Auth Type - Bearer token

6. Enter the token value you save in above steps

7. Save the request by clicking on Save button on the top right

8. Click on “send” button

9. User will see lists in response section

The screenshot shows the Postman interface with the following details:

- Request Type:** DELETE
- URL:** <https://thinking-tester-contact-list.herokuapp.com/contacts/685286abb6d2c70015fb60d>
- Authorization:** Bearer Token (with placeholder {{vault:authorization-secret}})
- Response Status:** 200 OK
- Response Body:** Contact deleted

In order to check that contact is deleted , go to Get Contact request and update the
ID- [685286abb6d2c70015fb60d](#) and when click on send button , user got the response status = 404 NOT FOUND

The screenshot shows the Postman interface with the following details:

- Request Type:** GET
- URL:** <https://thinking-tester-contact-list.herokuapp.com/contacts/685286abb6d2c70015fb60d>
- Authorization:** Bearer Token (with placeholder)
- Response Status:** 404 Not Found

We have 6 saved requests . Each request is successful.

Negative Tests

Let make unsuccessful request in negative Tests folder with 5 different negative testing scenarios

1. A request is sent with a missing authentication token

The screenshot shows the Postman interface with a test named "[CONFLICT] GET Get Contact- Unauthorized". The request method is GET, and the URL is <https://thinking-tester-contact-list.herokuapp.com/contacts>. The "Authorization" tab is selected, showing "Bearer Token" as the type and a placeholder "Token" with the instruction "donot enter anything in token value". The "Test Results" section shows a 401 Unauthorized response with the error message "Please authenticate.".

2. A record is not found - enter the ID whose record is already deleted

The screenshot shows the Postman interface with a test named "[CONFLICT] GET Get Contact-Not Found". The request method is GET, and the URL is <https://thinking-tester-contact-list.herokuapp.com/contacts/685286abb6d2c70015fb60d>. The "Authorization" tab is selected, showing "Bearer Token" and a placeholder "Token" with red annotations: "enter that contact ID where user perform delete" and "Contact request". The "Test Results" section shows a 404 Not Found response with the message "Next: Variables, params, and headers".

3. A request is sent with missing required information - first name , last name value is not entered

The screenshot shows the Postman interface with a POST request to <https://thinking-tester-contact-list.herokuapp.com/contacts>. The request body is JSON:

```

1 {
2   "firstName": "", remove the first name , last name values
3   "lastName": "", 
4   "birthdate": "1977-07-07",
5   "email": "pprunewhip@fake.com",
6   "phone": "8005551000",
7   "street1": "123 Main St.",
8   "street2": "Apartment Q",
9   "city": "New York",
10  "stateProvince": "NY",
11  "postalCode": "12345",
12  "country": "USA"
13

```

The response status is 400 Bad Request, and the error message is:

```

1 {
2   "errors": {
3     "firstName": {
4       "name": "ValidatorError",
5       "message": "Path 'firstName' is required.",
6       "properties": {
7         "message": "Path 'firstName' is required.",
8         "type": "required",
9         "path": "firstName",
10        "value": ""
11      },
12      "kind": "required",
13      "path": "firstName",
14      "value": ""
15    }
16  }
17}

```

4. A value is sent with too many characters - Last name with more than 20 characters

The screenshot shows the Postman interface with a POST request to <https://thinking-tester-contact-list.herokuapp.com/contacts>. The request body is JSON:

```

1 {
2   "firstName": "Patrick",
3   "lastName": "Prunewhipdsafadfdfsf", enter last name value more than 20 characters
4   "birthdate": "1977-07-07",
5   "email": "pprunewhip@fake.com",
6   "phone": "8005551000",
7   "street1": "123 Main St.",
8   "street2": "Apartment Q",
9   "city": "New York",
10  "stateProvince": "NY",
11  "postalCode": "12345",
12  "country": "USA"
13

```

The response status is 400 Bad Request, and the error message is:

```

1 {
2   "errors": {
3     "lastName": {
4       "name": "ValidatorError",
5       "message": "Path 'lastName' ('Prunewhipdsafadfdfsf') is longer than the maximum allowed length (20).",
6       "properties": {
7         "message": "Path 'lastName' ('Prunewhipdsafadfdfsf') is longer than the maximum allowed length (20).",
8         "type": "maxLength",
9         "maxLength": 20,
10        "path": "lastName",
11        "value": "Prunewhipdsafadfdfsf"
12      }
13    }
14  }
15}

```

5. A value is sent that is not in correct format- invalid email format

This scenario needs 2 steps :

First - add a new contact and then save the ID

The screenshot shows the Postman interface with a collection named "t_Essential". A POST request titled "Add Contacts" is selected. The URL is <https://thinking-tester-contact-list.herokuapp.com/contacts>. The request body contains a JSON object representing a new contact:

```

1 {
2   "firstName": "Jimmy",
3   "lastName": "Gill",
4   "birthdate": "1997-04-01",
5   "email": "jgill@fake.com",
6   "phone": "8005331000",
7   "street1": "123 Main St.",
8   "street2": "Apartment Q",
9   "city": "New York",
10  "stateProvince": "NY",
11  "postalCode": "12345",
12  "country": "USA"
13 }

```

The response status is 201 Created, and the response body shows the newly created contact with an assigned ID:

```

1 {
2   "_id": "68529818b6d2c70015fb69b",
3   "firstName": "Jimmy",
4   "lastName": "Gill",
5   "birthdate": "1997-04-01",
6   "email": "jgill@fake.com",

```

A note in the UI says "save this ID, we will used in update the contact request".

Second - update the contact with invalid email format

The screenshot shows the same Postman interface with the "t_Essential" collection. A PUT request titled "Update Contact - Invalid Email" is selected. The URL is <https://thinking-tester-contact-list.herokuapp.com/contacts/68529818b6d2c70015fb69b>. The request body is identical to the previous one but with an invalid email address:

```

1 {
2   "firstName": "Jimmy",
3   "lastName": "Gill",
4   "birthdate": "1997-04-01",
5   "email": "jgill@fake",
6   "phone": "8005331000",
7   "street1": "123 Main St.",
8   "street2": "Apartment Q",
9   "city": "New York",
10  "stateProvince": "NY",
11  "postalCode": "12345",
12  "country": "USA"
13 }

```

The response status is 400 Bad Request, and the response body shows validation errors:

```

1 {
2   "errors": [
3     {
4       "email": {
5         "name": "ValidatorError",
6         "message": "Email is invalid",
7         "properties": [
8           {
9             "message": "Email is invalid",
10            "type": "user defined",
11            "path": "email",
12            "value": "jgill@fake",
13            "reason": {}
14          }
15        ],
16        "kind": "user defined",
17        "path": "email",
18        "value": "jgill@fake",
19        "reason": {}
20      }
21    ]
22 }

```

Environment Variables

Environment - a group of variables in postman

Initial value - the variable's value that is saved in env file

Current value - a temporary variable value

The screenshot shows the Postman interface with the following annotations:

1. Click on environment tab
2. Click on + sign to create an env
3. give Environment name - Contact List
4. Enter the variable name with initial
5. Select the Environment - Contact List
6. save it

Variable	Type	Initial value	Current value
firstname	default	Amy ✓	Amy
lastName	default	Smith ✓	Smith

Set the Env variables - go to Env tab

Set the env variable for collection in POST request

The screenshot shows the Postman interface with the following annotations:

1. Go to collection
2. https://thinking-tester-contact-list.herokuapp.com/contacts
3. Body → enter the variable name with curly
4. Headers (10)
5. Body → raw

```

1 {
2   "firstName": "{{firstName}}",
3   "lastName": "{{lastName}}",
4   "birthdate": "1997-04-01",
5   "email": "jgill@fake.com",
6   "phone": "8005331000",
7   "street1": "123 Main St.",
8   "street2": "Apartment 0",
9   "city": "New York",
10  "stateProvince": "NY",
11  "postalCode": "12345",
12  "country": "USA"
13 }
  
```

Body Cookies Headers (11) Test Results
201 Created - 358 ms - 1.04 KB Save Response

```

1 {
2   "_id": "68529e438aa9610015dc392",
3   "firstName": "Amy",
4   "lastName": "Smith",
5   "birthdate": "1997-04-01",
6   "email": "jgill@fake.com",
7   "phone": "8005331000",
8   "street1": "123 Main St.",
9   "street2": "Apartment 0"
  
```

Variables in request

- E firstName Amy
- E lastName Smith
- All variables
- E Environment
- firstName Amy
- lastName Smith
- C Collection
- No variables defined in this collection. Add
- G Globals
- No global variables in this workspace. Add
- V Vault
- No vault secrets defined. Add secrets

```

1 {
2   "firstName": "{{firstName}}",
3   "lastName": "{{lastName}}",
4   "birthdate": "1997-04-01",
5   "email": "jill@fake.com",
6   "phone": "8005331000",
7   "street1": "123 Main St.",
8   "street2": "Apartment Q",
9   "city": "New York",
10  "stateProvince": "NY",
11  "postalCode": "12345",
12  "country": "USA"
13 }

```

Body

```

1 {
2   "_id": "68529e438aa9610015dcc393",
3   "firstName": "Amy",

```

Saving response data as a variable

Variables can be programmatically saved using the Test tab of the postman request once the variable has been saved, it can be used in all other requests

Save Contact ID for reuse

Save token for reuse

Essential

Contact List

Basic Endpoint Tests

POST Login

GET Get Contact List

GET Get Contact

POST Add Contacts

PUT Update Contact

DEL Delete Contact

Negative Tests

GET Get Contact List - Unauthorized

POST Add Contacts - Last Name too Long

GET Get Contact - Not Found

POST Add Contacts - Missing Required...

PUT Update Contact - Invalid Email

Overview Params Authorization Headers (10) Body Scripts Settings

Pre-request

```

1 var jsonData = pm.response.json();
2 pm.environment.set("contactId",jsonData._id);

```

Post-response

Body Cookies Headers (11) Test Results

```

1 {
2   "_id": "6852a1768aa9610015dcc3a7",
3   "firstName": "Amy",
4   "lastName": "Smith",
5   "birthdate": "1997-04-01",
6   "email": "jill@fake.com",
7   "phone": "8005331000",

```

The screenshot shows the 'Variables in request' section in Postman. It includes environment variables for 'firstName' (Amy) and 'lastName' (Smith), and a specific variable 'contactId' (6852a1768aa9610015dcc3a7) which is highlighted with a red box. The 'contactId' variable is also highlighted with a red box in the main interface above.

Variables in request

E	firstName	Amy
E	lastName	Smith

All variables

E Environment

firstName	Amy
lastName	Smith
contactId	6852a1768aa9610015dcc3a7

C Collection

No variables defined in this collection. [Add](#)

G Globals

Now go to GET Contacts request and change the URL

The screenshot shows a GET request in Postman to the URL `https://thinking-tester-contact-list.herokuapp.com/contacts/{{contactId}}`. The 'Authorization' tab is selected, showing 'Bearer Token' as the auth type and a placeholder for the token. The response body is displayed below, showing a JSON object with contact details. A red box highlights the URL in the request bar and the 'contactId' variable in the environment table.

Contact List / Basic Endpoint Tests / Get Contact

GET https://thinking-tester-contact-list.herokuapp.com/contacts/{{contactId}}

Authorization

Bearer Token

Body

```
{ } JSON
```

```
1 {  
2   "_id": "6852a1768aa9610015dcc3a7",  
3   "firstName": "Amy",  
4   "lastName": "Smith",  
5   "birthdate": "1990-04-01",  
6   "email": "jill@example.com",  
7   "phone": "+0005331000",  
8   "street1": "123 Main St.",  
9   "street2": "Apartment Q",  
10  "city": "New York",  
11  "stateProvince": "NY",  
12  "postalCode": "12345",  
13}
```

200 OK 358 ms 1.04 KB Save Response

Let do it for token value too

Step 1

Write the script for setting the token value

The screenshot shows the Postman interface with a POST request to `https://thinking-tester-contact-list.herokuapp.com/users/login`. In the 'Scripts' tab of the 'Post-response' section, there is a script:

```
1 var jsonData = pm.response.json();
2 pm.environment.set("token",jsonData.token);
```

The response body is displayed in JSON format, showing a user object with a token field. A red box highlights the token value, which is also circled with a red marker.

Step 2 - set token value as variable value

The screenshot shows the Postman interface with a GET request to `https://thinking-tester-contact-list.herokuapp.com/contacts/{contactId}`. In the 'Authorization' tab, the 'Type' is set to 'Bearer Token' and the 'Token' field contains `{{token}}`. A red box highlights the token placeholder, and a red arrow points to it from the previous step's screenshot.

The response body is displayed in JSON format, showing a contact object. A red circle highlights the number 7 at the bottom of the JSON tree.

Assertions :

Status type assertions

200- the request was completed as expected

201 - a new response was created

401 - the user is not authenticated

404- the resource was not found

The screenshot shows the Postman application interface. On the left, the sidebar displays a tree structure of API collections: 'Contact List' is expanded, showing 'Basic Endpoint Tests' which contains several test cases like 'GET Get Contact List', 'POST Add Contacts', etc. A red box highlights 'Basic Endpoint Tests'. In the main workspace, a specific test case 'Get Contact' is selected. The 'Test Results' tab at the bottom shows a green 'PASSED' status with the message 'Status code is 200'. The 'Script' tab in the top navigation bar contains the following JavaScript code:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

The screenshot shows the Postman interface with a test failure. The URL is <https://thinking-tester-contact-list.herokuapp.com/contacts/685286abb6d2c70015fb60d>. The status code is 404 Not Found. The Pre-request script contains:

```

1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });

```

The Test Results tab shows a single failed assertion:

FAILED Status code is 200 | AssertionError: expected response to have status code 200 but got 404

BODY Assertions:

A body assertion verifies that body of response contains the text that we were expecting.

Response body: contains string: contains string: the text is in the body of response, but may not be the entire response

Response body: is equal to string: contains string: the text matches with entire body of the response

Positive

The screenshot shows the Postman interface with a successful test. The URL is <https://thinking-tester-contact-list.herokuapp.com/contacts/{{contactId}}>. The status code is 200 OK. The Pre-request script contains:

```

1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("correct email address is returned", function () {
5   pm.expect(pm.response.text()).to.include("@fake.com");
6 });
7

```

The Test Results tab shows two passed assertions:

PASSED Status code is 200

PASSED correct email address is returned

Negative

The screenshot shows the Postman interface with a 'Contact List' workspace. A 'Basic Endpoint Tests' collection is open, containing a 'POST Add Contacts' test. The 'Post-request' tab has the following code:

```

1 pm.test("missing first name error message is returned", function () {
2   pm.expect(pm.response.text()).to.include("first name is required");
3 });

```

The 'Test Results' section shows a single failed assertion:

FAILED missing first name error message is returned | Assertion: expected '{"errors":{"firstName":{"name":"Valid...'}} to include 'first name is required'

Header and response time assertion

Response header : information passed with an API response that includes additional information about the response, such as the format of the response or any security controls

The screenshot shows the Postman interface with a 'Contact List' workspace. A 'Basic Endpoint Tests' collection is open, containing a 'GET Get Contact' test. The 'Post-response' tab has the following code:

```

1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("correct email address is returned", function () {
6   pm.expect(pm.response.text()).to.include("@fake.com");
7 });
8
9 pm.test("Content-Type is Application/json", function () {
10   pm.response.to.have.header("Content-Type","application/json; charset=utf-8");
11 });

```

The 'Test Results' section shows three passed assertions:

- PASSED Status code is 200
- PASSED correct email address is returned
- PASSED Content-Type is Application/json

Response time : the time it takes for a request to reach the server and return a response

Debugging with postman console:

Check that your HTTP verb is correct

Check that authentication token is sent

Check that URL of the request is correct

Check that all variables have populated correctly in the request

Postman interface showing a successful POST request to add a contact. The response body is highlighted with a red box.

```

User-Agent: "PostmanRuntime/7.44.0"
Accept: "*/*"
Cache-Control: "no-cache"
Postman-Token: "5cd204ee-935e-4e7e-b880-8a305d14d36f"
Host: "thinking-tester-contact-list.herokuapp.com"
Accept-Encoding: "gzip, deflate, br"
Connection: "keep-alive"
Content-Length: "302"
Cookie: "token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2D0UxZ0E5ZW12ZD3jNzAwMTVnYmMyNjAiLCJpYXQiOjE3NTAyNDYNT39.hX-jgxHmrcT0p2Sh2EHtxKEwkJAXnxzUjD09gs00"
Request Body >
{
  "firstName": "Amy",
  "lastName": "Smith",
  "birthdate": "1997-04-01",
  "email": "jill@fake.com",
  "phone": "8005331000",
  "street1": "123 Main St."
}

```

JSON assertions

JSON- javascript object notation

API requests and responses are often in JSON format

Postman interface showing a successful GET request to get a contact. The response body is highlighted with a red box.

```

pm.test("Content-Type is Application/json", function () {
  pm.response.to.have.header("Content-Type", "application/json; charset=utf-8");
});

pm.test("correct first name is returned", function () {
  var jsonData = pm.response.json();
  console.log(jsonData);
  pm.expect(jsonData.firstName).to.eql(pm.environment.firstName);
});

pm.test("correct last name is returned", function () {
  var jsonData = pm.response.json();
  pm.expect(jsonData.lastName).to.eql(pm.environment.lastName);
});

```

The screenshot shows the Postman interface with a workspace named "Postman_Essential". A GET request is defined to "https://thinking-tester-contact-list.herokuapp.com/contacts/{contactId}". The "Post-response" section contains the following code:

```

3 });
4
5 pm.test("correct email address is returned", function () {
6     pm.expect(pm.response.text()).to.include("#fake.com");
7 });
8
9 pm.test("Content-Type is Application/json", function () {
10    pm.response.to.have.header("Content-Type","application/json; charset=utf-8");
11 });
12
13 var jsonData = pm.response.json();
14
15 pm.test("correct first name is returned", function () {
16     pm.expect(jsonData.firstName).to.eql("Amy");
17 });
18
19 pm.test("correct last name is returned", function () {
20     pm.expect(jsonData.lastName).to.eql("Smith");
21 });

```

The "Test Results" section shows six green "PASSED" status indicators, each corresponding to one of the assertions in the post-response script.

Body Assertion	JSON assertion
We are looking for value somewhere in the body	We are looking for specific field value and validating what you expected

Nested JSON Assertions

Nested JSON: JSON that has values nested inside another object

For example :

```

"Address": {
  "Street": "123 Main St",
  "City": "belmont",
  "State": "MA",
  "postalcode": "23243"
}

```

}

The screenshot shows the Postman interface with the following details:

- Workspace:** stman_Essential
- Collection:** Address API
- Request:** GET Get Address
- URL:** https://25536437-5c33-435a-a284-efd3f231bc7d.mock.pstmn.io/getAddress/1
- Test Script (Pre-request):**

```
pm.test("correct city is returned", function () {  
    var jsonData = pm.response.json();  
    pm.expect(jsonData.address.city).to.eql("Belmont");  
});
```
- Test Results (Body):**

```
1 {  
2   "businessId": 1,  
3   "name": "Amalgamated Steel",  
4   "address": {  
5     "street": "123 Main St.",  
6     "city": "Belmont",  
7     "state": "MA",  
8     "postalCode": "01734"  
9   }  
10 }
```
- Status:** 200 OK
- Time:** 488 ms
- Size:** 938 B

The screenshot shows the Postman interface with the following details:

- Workspace:** stman_Essential
- Collection:** Address API
- Request:** GET Get Address
- URL:** https://25536437-5c33-435a-a284-efd3f231bc7d.mock.pstmn.io/getAddress/1
- Test Script (Pre-request):**

```
pm.test("correct city is returned", function () {  
    var jsonData = pm.response.json();  
    pm.expect(jsonData.address.city).to.eql("Belmont");  
});
```
- Test Results (Body):**

```
PASSED | correct city is returned
```
- Status:** 200 OK
- Time:** 488 ms
- Size:** 938 B

Asserting on Complicated JSON

Postman interface showing a test script for a GET request to https://98f2c9ce-fa98-4bd1-ac26-3da85ea67799.mock.pstmn.io/owners/1.

Test Script (Scripts tab):

```

1 pm.test("Amy's second is a Siamese", function () {
2     var jsonData = pm.response.json();
3     pm.expect(jsonData.pets.cats[1].breed).to.eql("Siamese");
4 });

```

Test Results (Test Results tab):

PASSED Amy's second is a Siamese

2nd example 

The screenshot shows the Postman interface with the 'Get Pet Owners' request selected. The 'Body' tab is open, displaying a JSON response. A red box highlights the 'owners' key in the JSON structure.

```

1 {
2   "owners": [
3     {
4       "id": 1,
5       "name": "Amy",
6       "pets": [
7         {
8           "cat": [
9             {
10              "name": "Fluffy",
11              "age": 2,
12              "breed": "Persian"
13            },
14            {
15              "name": "Mister Whiskers",
16              "age": 4,
17              "breed": "Siamese"
18            }
19          ],
20          "dog": [
21            {
22              "name": "Spot",
23              "age": 3,
24              "breed": "Labrador Retriever"
25            }
26          ]
27        },
28        {
29          "id": 2,
30          "name": "Bob",
31          "pets": [
32            {
33              "cat": [
34                {
35                  "name": "Galadriel",
36                  "age": 2,
37                  "breed": "Hagdoll"
38                },
39                {
40                  "name": "Gandalf",
41                  "age": 3,
42                  "breed": "Hagdoll"
43                }
44              ],
45              "dog": [
46                {
47                  "id": 3,
48                  "name": "Carol",
49                  "pets": [
50                    {
51                      "cat": []
52                    }
53                  ]
54                }
55              ]
56            }
57          ]
58        }
59      ],
60      "id": 3,
61      "name": "Carol",
62      "pets": [
63        {
64          "cat": []
65        }
66      ]
67    }
68  ]
69}

```

The screenshot shows the Postman interface with the 'Get Pet Owners' request selected. The 'Tests' tab is highlighted with a red box. The test script is as follows:

```

1 pm.test("Carol's second dog is named Fred", function () {
2   var jsonData = pm.response.json();
3   pm.expect(jsonData.owners[2].pets.dogs[1].name).to.eql("Fred");
4 });
5 pm.test("Bob's first cat is 2 years old", function () {
6   var jsonData = pm.response.json();
7   pm.expect(jsonData.owners[1].pets.cats[0].age).to.eql(2);
8 });

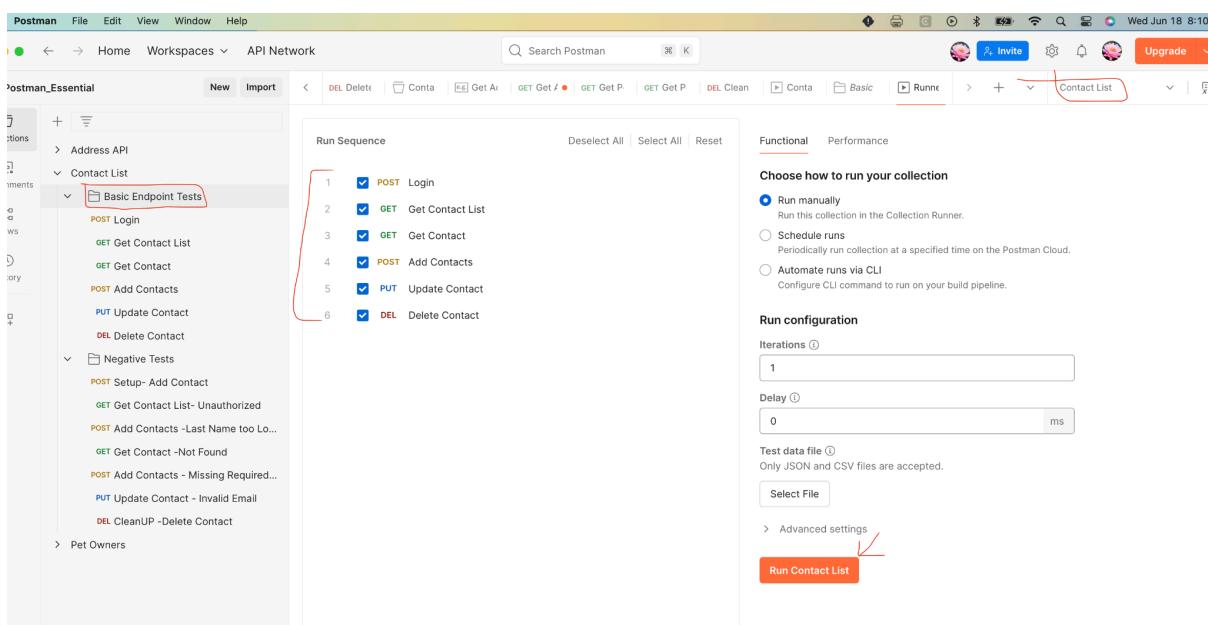
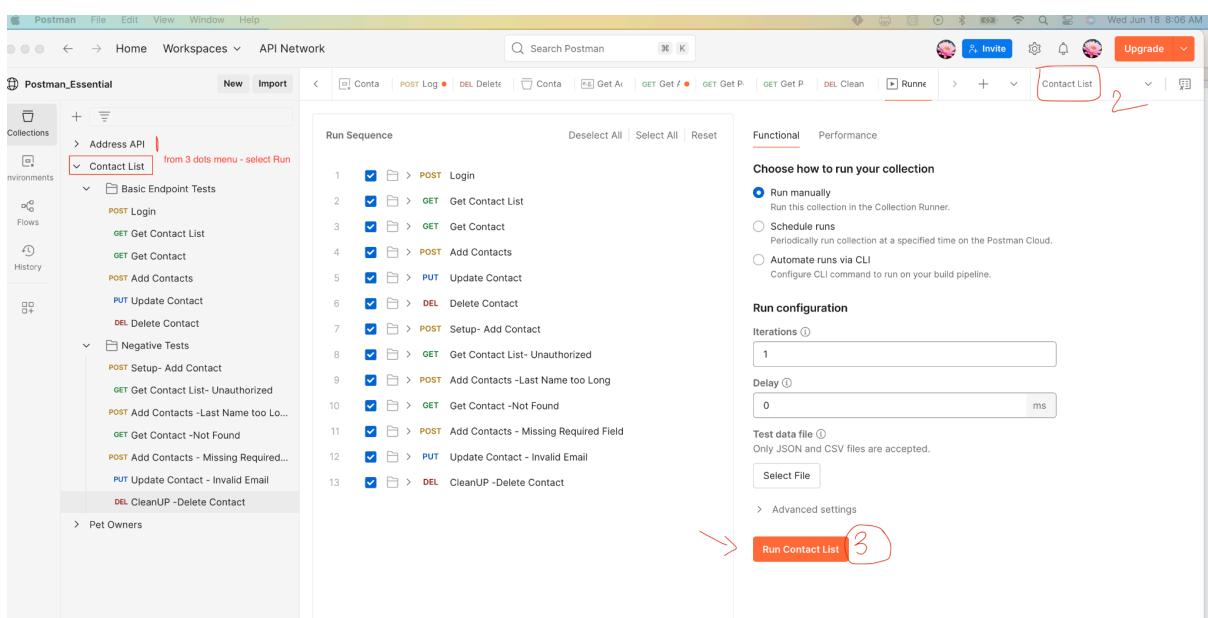
```

Tips for navigating complicated assertions :

1. Objects are in curly brackets {}
2. Arrays are in square brackets []
3. Arrays will always need a bracketed number, even if there's only one item in the array; then the number will be [0]
4. Work from left to right. Using the indentations to determine the order of navigation.
5. Anytime there is an indentation, you need a new dot in your navigation

Running the Test Collections

The postman collection runner : A feature of postman that will send in each request of your collection automatically.



Postman File Edit View Window Help

Home Workspaces API Network Search Postman Wed Jun 18 8:40 AM

Contact List - Run results

Ran today at 08:40:12 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	Contact List	1	1s 502ms	12	139 ms

All Tests Passed (12) Failed (0) Skipped (0)

Iteration 1

POST Login
https://thinking-tester-contact-list.herokuapp.com/users/login
PASS Response time is less than 3000ms

GET Get Contact List
https://thinking-tester-contact-list.herokuapp.com/contacts
PASS Response time is less than 3000ms

POST Add Contacts
https://thinking-tester-contact-list.herokuapp.com/contacts
PASS Response time is less than 3000ms
PASS Status code is 201

GET Get Contact
https://thinking-tester-contact-list.herokuapp.com/contacts/6852dde11ae8a400154ee62d
PASS Response time is less than 3000ms
PASS Status code is 200
PASS correct email address is returned
PASS Content-Type is Application/json
PASS correct first name is returned
PASS correct last name is returned

Run Again Automate Run New Run Export Results

+ Import

Contact List - Basic Endpoint Tests

- POST Login
- GET Get Contact List
- POST Add Contacts
- GET Get Contact
- PUT Update Contact
- DEL Delete Contact
- Negative Tests
 - POST Setup- Add Contact
 - GET Get Contact List- Unauthorized
 - GET Get Contact -Not Found
 - POST Add Contacts - Missing Required...
 - POST Add Contacts -Last Name too Lo...
 - PUT Update Contact - Invalid Email
 - DEL CleanUP -Delete Contact
- Pet Owners

GET Get Contact List | GET Get Contact | Contact List | POST Add Contacts +

Contact List - Run results

Ran today at 08:40:12 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	Contact List	1	1s 502ms	12	139 ms

RUN SUMMARY

Action	Count
POST Login	1 0
GET Get Contact List	1 0
POST Add Contacts	2 0
GET Get Contact	6 0
PUT Update Contact	1 0
DELETE Delete Contact	1 0

Run Again Automate Run

Running the collection in newman :

Install [node.js](#)

Check with command if nodeJS is installed or not

node -v

Check with if npm is installed or not

npm -v

Install newman using command:

npm install -g newman

Check with command if newman is install or not

newman —version

Export the collection and environment variables from postman by exporting in desktop

Go to terminal and run the following command :

```
newman run ContactList.postman_collection.json -e ContactList.postman_environment.json
```

To see other commands options, type

newman run -h

```
Anusharma@Anum-MacBook-Pro Desktop % newman run ContactList.postman_collection.json -e ContactList.postman_environment.json
newman
Contact List
  □ Basic Endpoint Tests
  4 Login
    ✓ Post https://thinking-tester-contact-list.herokuapp.com/users/login [200 OK, 1.25kB, 499ms]
      ✓ Response time is less than 3000ms

  4 Get Contact List
    GET https://thinking-tester-contact-list.herokuapp.com/contacts [200 OK, 2.9kB, 87ms]
      ✓ Response time is less than 3000ms

  4 Add Contacts
    POST https://thinking-tester-contact-list.herokuapp.com/contacts [201 Created, 1.07kB, 93ms]
      ✓ Response time is less than 3000ms
      ✓ Status code is 201

  4 Get Contact
    GET https://thinking-tester-contact-list.herokuapp.com/contacts/6852e7781a12590015ee4c89 [200 OK, 1.07kB, 86ms]
      ✓ Response time is less than 3000ms
      ✓ Status code is 200
      ✓ correct email address is returned
      ✓ Content-Type is application/json
      ✓ correct first name is returned
      ✓ correct last name is returned

  4 Update Contact
    PUT https://thinking-tester-contact-list.herokuapp.com/contacts/685286abb6d2c70015fbc60d [401 Unauthorized, 787B, 82ms]
      ✓ Response time is less than 3000ms

  4 Delete Contact
    DELETE https://thinking-tester-contact-list.herokuapp.com/contacts/685286abb6d2c70015fbc60d [401 Unauthorized, 787B, 83ms]
      ✓ Response time is less than 3000ms

  □ Negative Tests
  4 Setup- Add Contact
    POST https://thinking-tester-contact-list.herokuapp.com/contacts [401 Unauthorized, 787B, 80ms]
      ✓ Response time is less than 3000ms
      1. Status code is 201

  4 Get Contact List- Unauthorized
    GET https://thinking-tester-contact-list.herokuapp.com/contacts [401 Unauthorized, 787B, 81ms]
      ✓ Response time is less than 3000ms

  4 Get Contact -Not Found
    GET https://thinking-tester-contact-list.herokuapp.com/contacts/685286abb6d2c70015fbc60d [404 Not Found, 662B, 87ms]
      ✓ Response time is less than 3000ms
      2. Status code is 200

  4 Add Contacts - Missing Required Field
    POST https://thinking-tester-contact-list.herokuapp.com/contacts [400 Bad Request, 1.39kB, 87ms]
      ✓ Response time is less than 3000ms
      3. missing first name error message is returned

  4 Add Contacts -Last Name too Long
    POST https://thinking-tester-contact-list.herokuapp.com/contacts [401 Unauthorized, 787B, 80ms]
      ✓ Response time is less than 3000ms

  4 Update Contact - Invalid Email
    PUT https://thinking-tester-contact-list.herokuapp.com/contacts/null [401 Unauthorized, 787B, 82ms]
      ✓ Response time is less than 3000ms

  4 CleanUP -Delete Contact
    DELETE https://thinking-tester-contact-list.herokuapp.com/contacts/null [400 Bad Request, 765B, 82ms]
      ✓ Response time is less than 3000ms

  ┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐
  │              │ ┌── executed ─┐ ┌── failed ─┐
  │ iterations   │ └── 1 ─┘ └── 0 ─┘
  └────────────────┘ ┌─────────────────┐ ┌─────────────────┐
  requests       └── 13 ─┘ └── 0 ─┘
```

```

↳ Update Contact
  ✓ POST https://thinkin-tester-contact-list.herokuapp.com/contacts/48528eab04dc70015fbccbd (401 Unauthorized, 787B, 82ms)
    ✓ Response time is less than 3000ms

↳ Delete Contact
  DELETE https://thinkin-tester-contact-list.herokuapp.com/contacts/48528eab04dc70015fbccbd (401 Unauthorized, 787B, 81ms)
    ✓ Response time is less than 3000ms

↳ Negative Tests
  ↳ Setup- Add Contact
    ✓ POST https://thinkin-tester-contact-list.herokuapp.com/contacts (401 Unauthorized, 787B, 80ms)
      ✓ Response time is less than 3000ms
    1. Status code is 201

  ↳ Get Contact List- Unauthorized
    ✓ GET https://thinkin-tester-contact-list.herokuapp.com/contacts (401 Unauthorized, 787B, 81ms)
    ✓ Response time is less than 3000ms

  ↳ Get Contact -Not Found
    ✓ GET https://thinkin-tester-contact-list.herokuapp.com/contacts/48528eab04dc70015fbccbd (404 Not Found, 642B, 81ms)
    ✓ Response time is less than 3000ms
    2. Status code is 200

  ↳ Add Contacts - Missing Required Field
    POST https://thinkin-tester-contact-list.herokuapp.com/contacts (400 Bad Request, 1,199B, 87ms)
    ✓ missing first name error message is returned

  ↳ Add Contacts -Last Name too Long
    POST https://thinkin-tester-contact-list.herokuapp.com/contacts (401 Unauthorized, 787B, 80ms)
    ✓ Response time is less than 3000ms

  ↳ Update Contact - Invalid Email
    PUT https://thinkin-tester-contact-list.herokuapp.com/contacts/null (401 Unauthorized, 787B, 81ms)
    ✓ Response time is less than 3000ms

  ↳ cleanUP -Delete Contact
    ✓ DELETE https://thinkin-tester-contact-list.herokuapp.com/contacts/null (400 Bad Request, 761B, 82ms)
    ✓ Response time is less than 3000ms



|                   | executed | failed |
|-------------------|----------|--------|
| iterations        | 1        | 0      |
| requests          | 13       | 0      |
| test-scripts      | 19       | 0      |
| perequest-scripts | 13       | 0      |
| assertions        | 22       | 1      |



total run duration: 1865ms  

  total data received: 3.87kB (approx)  

  average response time: 116ms [min: 80ms, max: 499ms, s.d.: 110ms]



# failures      detail



- AssertionErr. Status code is 201
      expected response to have status code 201 but got 401
      at AssertionErr in test-script
      include "Negative Tests / Setup- Add Contact"
- AssertionErr. Status code is 200
      expected response to have status code 200 but got 404
      at AssertionErr in test-script
      include "Negative Tests / Get Contact -Not Found"
- AssertionErr. missing first name error message is returned
      expected response to have status code 400 but got 401
      to include "first name is required"
      at AssertionErr in test-script
      include "Negative Tests / Add Contacts - Missing Required Field"

```

If your files are in a different location - for example Projects/Newman/ContactList-

You can navigate to that location :

Newman run **Projects/Newman/ContactList**
contactList.collection.json -e
Projects/Newman/ContactList/contactList.environment.json

Practise API:

Contact List app:

<https://thinking-tester-contact-list.herokuapp.com>

Pet Store API: <https://petstore.swagger.io>

Poke API: <https://pokeapi.co>

Integrate the postman tests into your workplace CI/CD platform

Jenkins documentation:

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

GitLab documentation:

<https://learning.postman.com/docs/integrations/available-integrations/ci-integrations/gitlab-ci>

Travis CI documentation:

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

Circle CI documentation:

<https://learning.postman.com/docs/integrations/available-integrations/ci-integrations/circleci>

Postman CLI tool:

<https://learning.postman.com/docs/postman-cli/postman-cli-overview>