

REQRES.IN API TESTING

Test Cases

List of Test Cases

| | |
|--|----|
| INTRODUCTION | 6 |
| Endpoints | 6 |
| /api/users | 6 |
| /api/unknown | 6 |
| /api/registration | 7 |
| /api/login | 7 |
| 1. Endpoint: /api/users | 7 |
| 1.1 Test Scenario: Get All Users | 7 |
| 1.1.1 Test Case: Get users with no parameters | 7 |
| 1.1.2 Test Case: Get All users with empty id | 8 |
| 1.1.3 Test Case: Get All Users for all users on a Page | 8 |
| 1.1.3 Test Case: Get All Users for the First Page | 8 |
| 1.1.4 Test Case: Get All Users for the Last Page | 9 |
| 1.1.5 Test Case: Get All Users with empty page number | 9 |
| 1.1.6 Test Case: Get All Users with valid page and valid per_page | 9 |
| 1.1.7 Test Case: Get Users for invalid page number (Beyond Last Page) | 10 |
| 1.1.8 Test Case: Get All Users with negative page number | 11 |
| 1.1.9 Test Case: Get All Users with Exceedingly Large Page Number | 11 |
| 1.1.10 Test Case: Get All Users with Exceedingly Large Per_Page Number | 12 |
| 1.1.11 Test Case: Get All Users with unknown query params | 13 |
| 1.1.12 Test Case: Get All Users with valid page and per_page but Invalid POST method | 13 |
| 1.1.13 Test Case: Get user with minimum valid id | 14 |
| 1.1.14 Test Case: Get user with maximum valid id | 14 |
| 1.1.15 Test Case: Get User with non-existent id (beyond maximum) | 14 |
| 1.1.16 Test Case: Get User with non-existent id (less than minimum) | 15 |
| 1.1.17 Test Case: Get User with negative id | 15 |
| 1.1.18 Test Case: Get User with id (string format) | 16 |
| 1.1.19 Test Case: Get User with id (float) | 16 |
| 1.2 Test Scenario: Create User | 17 |
| 1.2.1 Test Case: Create User with all valid parameters | 17 |
| 1.2.2 Test Case: Create User with valid email | 18 |
| 1.2.3 Test Case: Create user with all empty body | 19 |
| 1.2.4 Test Case: Create an existing user | 19 |
| 1.2.5 Test Case: Create User with invalid email | 20 |
| 1.2.6 Test Case: Create user with invalid email format | 21 |

| | |
|--|----|
| 1.2.7 Test Case: Create user with empty request body..... | 22 |
| 1.2.8 Test Case: Create user with an invalid field..... | 23 |
| 1.2.9 Test Case: Create user without email..... | 23 |
| 1.2.10 Test Case: Create user with only firstName. | 24 |
| The API responds with an invalid user schema but still returns a "201 Created" status code. | 24 |
| 1.2.10 Test Case: Create user with only lastName. | 24 |
| 1.2.11 Test Case: Create user with large string in firstname..... | 25 |
| 1.2.12 Test Case: Create user with large string in lastName. | 25 |
| 1.2.13 Test Case: Create user with digit as input in first name. | 26 |
| 1.2.14 Test Case: Create user with digit as input in first name. | 26 |
| 1.2.15 Test Case: Create user with special character as input in first name. | 26 |
| 1.2.16 Test Case: Create user with special character as input in last name..... | 27 |
| 1.2.17 Test Case: Create user with only avatar..... | 27 |
| 1.2.18 Test Case: Create user with invalid avatar format..... | 27 |
| 1.2.19 Test Case: Create user with id. | 28 |
| 1.2.20 Test Case: Create user with existing id..... | 28 |
| 1.2.21 Test Case: Create user with invalid id (string as input). | 28 |
| 1.2.22 Test Case: Create user with empty id..... | 29 |
| 1.2.23 Test Case: Create User with all valid parameters along with id. | 29 |
| 1.3 Test Scenario: Update User | 30 |
| 1.3.1 Test Case: Update User of existing id with all correct body parameters..... | 30 |
| 1.3.2 Test Case: Update User of existing id with empty firstname. | 30 |
| 1.3.3 Test Case: Update User of existing id with empty lastname. | 31 |
| 1.3.4 Test Case: Update user with empty avatar..... | 31 |
| 1.3.5 Test Case: Update user of non-existing id with all correct body..... | 32 |
| 1.3.6 Test Case: Update User of existing id with all empty body parameters..... | 32 |
| 1.3.7 Test Case: Update user of existing id with empty email | 32 |
| 1.3.8 Test Case: Update user of existing id with invalid email format | 33 |
| 1.3.9 Test Case: Update user of existing id with invalid first name format..... | 33 |
| 1.3.10 Test Case: Update user of existing id with invalid first name format..... | 34 |
| 1.3.11 Test Case: Update User of existing id with large string in first name..... | 34 |
| 1.3.12 Test Case: Update User of existing id with large string in last name | 35 |
| 1.3.13 Test Case: Update User of existing id with large string in avatar | 35 |
| 1.3.14 Test Case: Update user of existing id with invalid avatar format | 36 |
| 1.3.15 Test Case: Update User using non-existent field in schema. | 36 |
| 1.4 Test Scenario: Delete User | 36 |

| | |
|--|-----------|
| 1.4.1 Test Case: Delete User of existing id. | 36 |
| 1.4.2 Test Case: Verify Delete User of existing id. | 36 |
| 1.4.3 Test Case: Delete User of non-existing id. | 37 |
| 2. Endpoint: /api/unknown | 37 |
| 2.1 Test Scenario: Request resources..... | 37 |
| 2.1.1 Test Case: Request with Default Parameters | 37 |
| 2.1.2 Test Case: Request with Valid Page and Per Page Parameters | 37 |
| 2.1.3 Test Case: Request for the Last Page..... | 38 |
| 2.1.4 Test Case: Request for a Single Item on a Page..... | 38 |
| 2.1.5 Test Case: Request with Default Parameters using invalid method Post. | 38 |
| 2.1.6 Test Case: Request for a Page Beyond Total Pages..... | 38 |
| 2.1.7 Test Case: Request with Negative Page Number..... | 39 |
| 2.1.8 Test Case: Request with Negative Items Per Page..... | 39 |
| 2.1.9 Test Case: Request with Non-Integer Page Number (String) | 40 |
| 2.1.10 Test Case: Request with Exceedingly Large Page Number | 40 |
| 2.2 Test Scenario: Create Resource | 40 |
| 2.2.1 Test Case: Create Resource with all valid data. | 40 |
| 2.2.2 Test Case: Create Resource with only name. | 41 |
| 2.2.3 Test Case: Create Resource with only color..... | 41 |
| 2.3 Test Scenario: Update Resource..... | 41 |
| 2.3.1 Test Case: Update Resource of existing id with all valid data. | 41 |
| 2.3.2 Test Case: Update Resource of existing id with empty name. | 42 |
| 2.3.3 Test Case: Update Resource of existing id with empty year..... | 42 |
| 2.3.4 Test Case: Update Resource of existing id with empty color..... | 42 |
| 2.3.5 Test Case: Update Resource of existing id with empty pantone value. | 43 |
| 2.3.6 Test Case: Partial Update Resource of existing id with only name..... | 43 |
| 2.3.7 Test Case: Partial Update Resource with only year. | 44 |
| 2. Test Scenario: Delete Resource | 44 |
| 2.4.1 Test Case: Delete Resource of existing id. | 44 |
| 2.4.2 Test Case: Verify Delete Resource of existing id..... | 44 |
| 2.4.3 Test Case: Delete Resource of non-existing id..... | 44 |
| 3. Endpoint: /api/register | 45 |
| 3.1 Test Scenario: Register User | 45 |
| 3.1.1 Test Case: Registration of defined user with all valid parameter (username, email and password)..... | 45 |
| 3.1.2 Test Case: Registration of defined user with email and password only | 45 |

| | |
|--|-----------|
| 3.1.3 Test Case: Registration of undefined user | 45 |
| 3.1.4 Test Case: Registration with existing email and password | 46 |
| 3.1.5 Test Case: Registration with empty email and empty password..... | 46 |
| 3.1.6 Test Case: Registration with empty email and valid password | 46 |
| 3.1.7 Test Case: Registration with valid email and empty password | 47 |
| 3.1.8 Test Case: Registration of defined user with email only | 47 |
| 3.1.9 Test Case: Registration of defined user with invalid email only | 47 |
| 3.1.10 Test Case: Registration of defined user with password only..... | 48 |
| 3.1.11 Test Case: Registration of defined user with long password..... | 48 |
| 3.1.12 Test Case: Registration of defined user with long email | 49 |
| 3.1.13 Test Case: Registration of defined user with invalid email and password | 49 |
| 3.1.14 Test Case: Registration of defined user with valid email and invalid password | 49 |
| 3.1.15 Test Case: Registration with Invalid Request Body Media Type..... | 50 |
| 3.1.16 Test Case: Registration with wrong method (GET) but correct URL..... | 50 |
| 3.1.17 Test Case: Incorrect HTTP Method (PUT) and URL for Registration..... | 51 |
| 3.1.18 Test Case: Registration of a new user..... | 51 |
| 4. Endpoint: /api/login | 51 |
| 4.1 Test Scenario: Post User | 51 |
| 4.1.1 Test Case: Login of defined user with all valid parameter (email and password)..... | 51 |
| 4.1.2 Test Case: Login of defined user with all valid parameters: Verify case insensitivity of email address..... | 52 |
| 4.1.3 Test Case: Login with Valid Credentials and Trailing Whitespace in Email..... | 52 |
| 4.1.4 Test Case: Login of unregistered user..... | 53 |
| 4.1.5 Test Case: Login of registered user with correct email and incorrect password..... | 53 |
| 4.1.6 Test Case: Login of registered user with correct email and empty password..... | 54 |
| 4.1.7 Test Case: Login of defined user with all correct email and password: Verify case sensitivity of password..... | 54 |
| 4.1.8 Test Case: Login with empty email and empty password. | 54 |
| 4.1.9 Test Case: Login with wrong method (GET) but correct URL. | 55 |
| 4.1.9 Test Case: Incorrect HTTP Method (PUT) and URL for Login. | 55 |
| 4.1.10 Test Case: Login with unknown request body..... | 55 |

INTRODUCTION

This document is consisting of all the possible positive and negative test cases based on the provided site reqres.in. For each test, we have a brief description, test data, the expected result, actual result and Pass/Fail status. Test cases are covering various scenarios for the following endpoints:

1. /api/users
2. /api/unknown
3. /api/register
4. /api/login

By executing these test cases, the main aim is to validate the functionality and behavior of the reqres.in site across various scenarios, ensuring its reliability and robustness in different conditions.

Endpoints

/api/users

Get Users

- Endpoint for retrieving a list of users.
- Query Parameters: **page**, **per_page**.
- Response: List of users with fields **id**, **email**, **first_name**, **last_name**, **avatar**.

GET User

- Endpoint for fetching existing user.
- Path Parameter: **id**.
- Response: A user with fields **id**, **email**, **first_name**, **last_name**, **avatar**.

Create User

- Endpoint for creating a new user.
- Request Body Parameters: **email**, **first_name**, **last_name**, **avatar**.
- Response: Created user object with generated **id** and **createdAt**.

Update User

- Endpoint for updating existing user.
- Supports full and partial update.
- Request Body Parameters: **email**, **first_name**, **last_name**, **avatar**.
- Response: Updated user object with generated **id** and **updatedAt**.

Delete User

- Endpoint for deleting existing user.
- Path Parameter: **id**.
- Response: No Content confirming successful deletion.

/api/unknown

Get Resources

- Endpoint for retrieving a list of resources.

- Query Parameters: **page, per_page**.
- Response: List of resources with fields **id,name,year,color,pantagone_value**.

GET Resource

- Endpoint for fetching existing resource.
- Path Parameter: **id**.
- Response: A resource with fields **id,name,year,color,pantagone_value**.

Create Resource

- Endpoint for creating a new resource.
- Request Body Parameters: **name,year,color,pantagone_value..**
- Response: Created resource object with generated **id** and **createdAt**.

Update Resource

- Endpoint for updating existing resource.
- Supports full and partial update.
- Request Body Parameters: **email, first_name, last_name, avatar**.
- Response: Updated resource object with generated **id** and **updatedAt**.

Delete Resource

- Endpoint for deleting existing resource.
- Path Parameter: **id**.
- Response: No Content confirming successful deletion.

/api/registration

Registration

- Endpoint for user registration.
- Parameters: **username,email, password**.
- Response: id and token.

/api/login

Login

- Endpoint for user authentication.
- Parameters: **email, password**.
- Response: Authentication token.

1. Endpoint: /api/users

1.1 Test Scenario: Get All Users

1.1.1 Test Case: Get users with no parameters.

Description: This test case validates the "Get All users" API endpoint with default parameters. It confirms that the API returns a successful response (status code 200) containing user data structured according to default pagination settings: 6 items per page, with a total of 12 items across 2 pages. The test ensures the presence of essential fields ('page', 'per_page', 'total', 'total_pages') and verifies the data integrity of user attributes such as ID, email, first_name, last_name, and avatar.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|-----------------------------------|----------------------------|-----------|----------|
| | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Response body should not be empty | Response body is not empty | Pass | |

| | | | | |
|--|--|---|------|--|
| | Response body should contain 'page', 'per_page', 'total', 'total_pages' fields | response body contains 'page', 'per_page', 'total', 'total_pages' fields | Pass | |
| | Response body should have an array of users data | Response body has an array of users data | Pass | |
| | Each user of users should have valid schema | Each user of users has valid schema | Pass | |
| | The default parameters specify that the default page number should be 1, with 6 items per page, total 12 items across 2 pages. | For default parameters, the default page number is set to 1, with 6 items displayed per page, resulting in a total of 12 items spread across 2 pages. | Pass | |

1.1.2 Test Case: Get All users with empty id.

Description: This test case evaluates the default behavior of the "Get All users" API endpoint when accessed without specifying an ID parameter. The objective is to confirm that the API interprets the absence of an ID as a request for all users and responds accordingly.

By initiating a request to the "/api/users/" endpoint without providing an ID parameter, the test aims to validate that the API returns a successful response (status code 200) containing user data structured according to default pagination settings. The verification process includes ensuring the presence of essential fields ('page', 'per_page', 'total', 'total_pages') in the response body and verifying the integrity of user attributes such as ID, email, first_name, last_name, and avatar.

This test case aims to ascertain that the API's default behavior aligns with expectations, providing comprehensive user data when accessed without specifying an ID parameter.

Assertions and Severity: Same as "Get All users with no parameters" (1.1.1)

1.1.3 Test Case: Get All Users for all users on a Page.

Description: This test case evaluates the behavior of the "Get All Users" API endpoint when requesting all users on a single page with a custom pagination setting of 12 users per page. The objective is to verify that the API accurately returns all available users within a single page based on the specified pagination parameter.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|--|---|-----------|----------|
| | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Page number should be 1. | Page number is 1 | Pass | |
| | There should be 12 users in the total 1 page as total users 12 | 12 users in total 1-page total users 12 | Pass | |

1.1.3 Test Case: Get All Users for the First Page.

Description: This test case evaluates the behavior of the "Get All Users" API endpoint when accessing the first page of user data. It verifies that the API returns the expected subset of users based on default

pagination settings. The test ensures that the response includes essential pagination information and accurately represents user attributes for the initial page of data retrieval.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|--|---|-----------|----------|
| | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Page number should be 1. | Page number is 1 | Pass | |
| | There should be 6 users in the first page. | 6 users in the first page. | Pass | |
| | information of each user should match the information of the first 6 users from the "allUsers" environment variable. | information of each user matches the information of the first 6 users from the "allUsers" environment variable. | Pass | |

1.1.4 Test Case: Get All Users for the Last Page.

Description: This test case evaluates the "Get Users" API endpoint's pagination functionality by verifying the number of users displayed on the last page. By navigating to the last page using the page parameter, the test ensures that the API returns the correct count of users and accurately represents the final page according to the pagination settings.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---------------------|---|---------------------------|-----------|----------|
| Params: Page = 2 | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Page number should be 2. | Page number is 2 | Pass | |
| | There should be 6 users in the last page. | 6 users in the last page. | Pass | |

1.1.5 Test Case: Get All Users with empty page number.

Description: This test case verifies whether the API returns the first page of user data when accessed without specifying a page number in the request to the "api/users?page" endpoint. It aims to ensure that the API responds with the initial page of user information, as per default behavior.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|--|----------------------------|-----------|----------|
| | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Page number should be 1. | Page number is 1. | Pass | |
| | There should be 6 users in the first page. | 6 users in the first page. | Pass | |

1.1.6 Test Case: Get All Users with valid page and valid per_page.

Description: This test case aims to validate the functionality of the "api/users?page=1&per_page=4" endpoint. It verifies whether the API accurately displays the specified number of users per page and navigates to the correct page as per the provided parameters.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--------------------------|--|----------------------------|-----------|----------|
| Page = 1 Per_page = 4 | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Page number should be 1. | Page number is 1 | Pass | |
| | There should be 4 users in the first page. | 4 users in the first page. | Pass | |

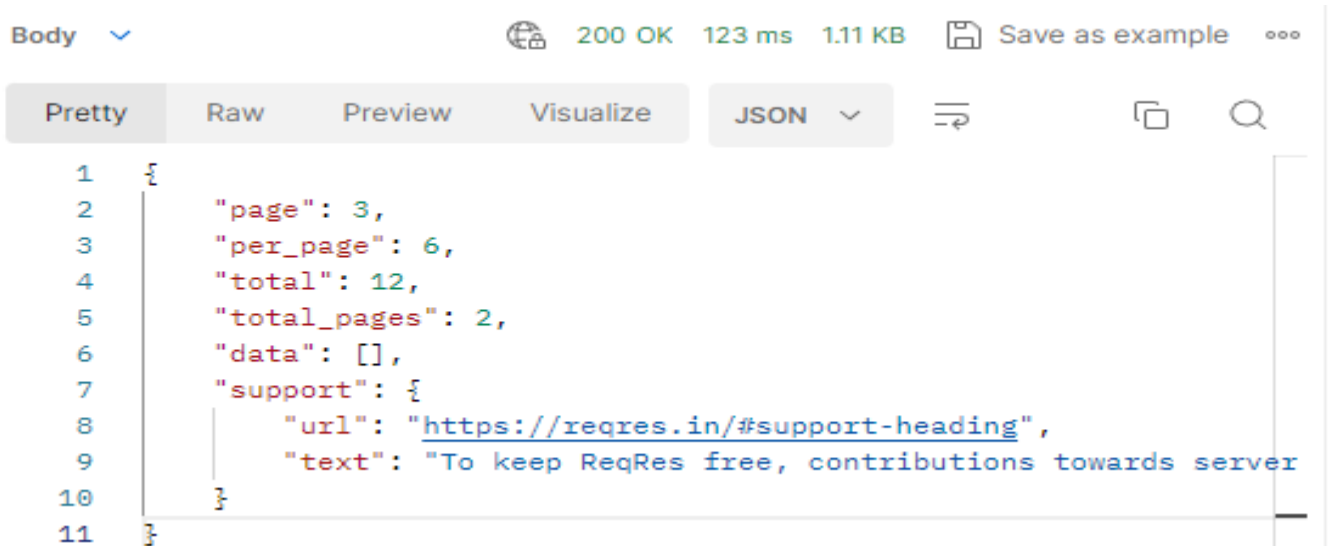
1.1.7 Test Case: Get Users for invalid page number (Beyond Last Page).

Description: This test case aims to assess the behavior of the "api/users?page=3" endpoint when accessed with an invalid page number beyond the last available page. It verifies whether the API responds with an appropriate error message in the response body, indicating the invalidity of the requested page.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---|---|-----------|----------|
| page=3 | Status Code should be 404 (Not Found) | Status Code: 200 (OK) | Fail | High |
| | Response body error message: Page not found | Response body did not provide error message | Fail | |

Visualization:



```
Body 200 OK 123 ms 1.11 KB Save as example ...
Pretty Raw Preview Visualize JSON
1 {
2   "page": 3,
3   "per_page": 6,
4   "total": 12,
5   "total_pages": 2,
6   "data": [],
7   "support": {
8     "url": "https://reqres.in/#support-heading",
9     "text": "To keep ReqRes free, contributions towards server"
10  }
11 }
```

The screenshot indicates that the API response does not contain any data and also does not provide an error message indicating that the page was not found.

1.1.8 Test Case: Get All Users with negative page number.

Description: This test case evaluates the behavior of the "api/users?page=-3" endpoint when accessed with a negative page number. The objective is to verify whether the API responds with an error message in the response body, indicating the invalidity of the requested page due to the negative page number.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---|---|-----------|----------|
| page=-3 | Status Code should be 404 (Not Found) | Status Code: 200 (OK) | Fail | High |
| | Response body error message: Page not found | Response body did not provide error message | Fail | |

Visualization:



Since a negative page scenario isn't possible, the API should ideally respond with a "400 Bad Request" error instead of a "200 OK" response.

1.1.9 Test Case: Get All Users with Exceedingly Large Page Number.

Description: This test case will check if the api/users?page= 1265436789 endpoint with large page number show error message in the response body.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|------------------|---|---|-----------|----------|
| page= 1265436789 | Status Code should be 404 (Not Found) | Status Code: 200 (OK) | Fail | Medium |
| | Response body error message: Page not found | Response body did not provide error message | Fail | |

Vizualization:



```
Body 200 OK 111 ms 1.1 KB Save as example
Pretty Raw Preview Visualize JSON
{
  "page": 1265436789,
  "per_page": 6,
  "total": 12,
  "total_pages": 2,
  "data": [],
  "support": {
    "url": "https://reqres.in/#support-heading",
    "text": "To keep ReqRes free, contributions towards server"
  }
}
```

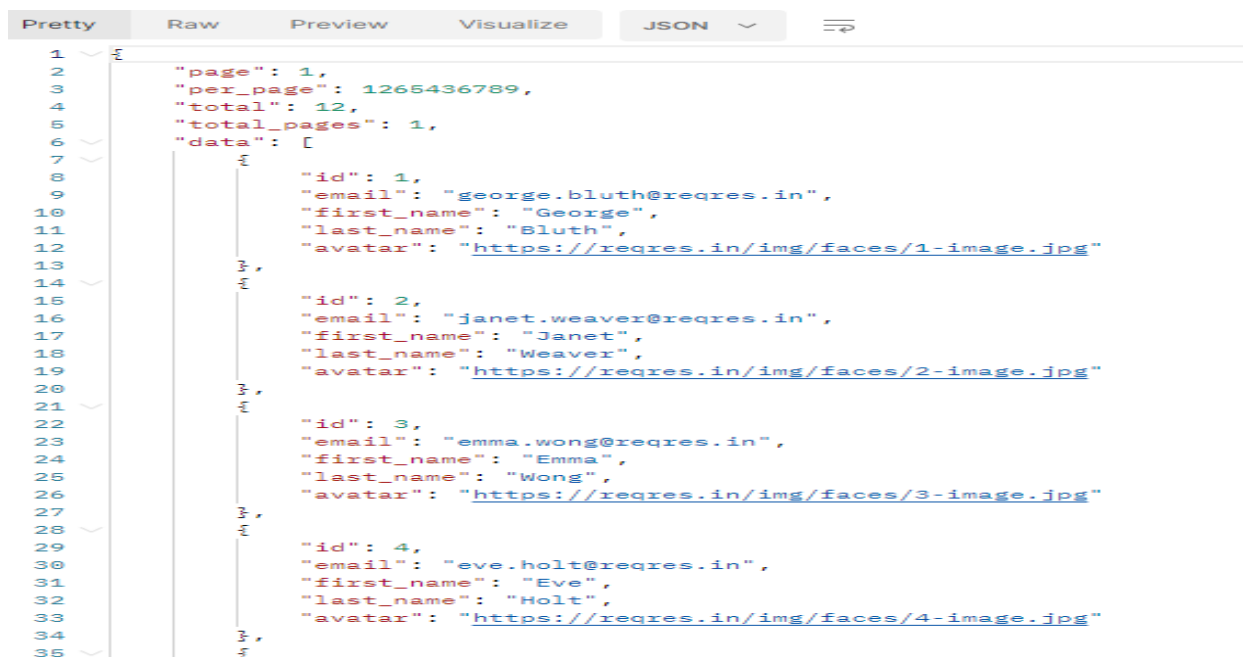
As an exceedingly large page scenario is not feasible, the API should return an error instead of a "200 OK" response.

1.1.10 Test Case: Get All Users with Exceedingly Large Per_Page Number.

Description: This test case will check if the `api/users?per_page= 1265436789` endpoint with large page number show error message in the response body.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|----------------------|---|---|-----------|----------|
| Per_page= 1265436789 | Status Code should be 400 (Bad Request) | Status Code: 200 (OK) | Fail | Medium |
| | Response body error message: Page not found | Response body did not provide error message | Fail | |



```
Pretty Raw Preview Visualize JSON
{
  "page": 1,
  "per_page": 1265436789,
  "total": 12,
  "total_pages": 1,
  "data": [
    {
      "id": 1,
      "email": "george.bluth@reqres.in",
      "first_name": "George",
      "last_name": "Bluth",
      "avatar": "https://reqres.in/img/faces/1-image.jpg"
    },
    {
      "id": 2,
      "email": "janet.weaver@reqres.in",
      "first_name": "Janet",
      "last_name": "Weaver",
      "avatar": "https://reqres.in/img/faces/2-image.jpg"
    },
    {
      "id": 3,
      "email": "emma.wong@reqres.in",
      "first_name": "Emma",
      "last_name": "Wong",
      "avatar": "https://reqres.in/img/faces/3-image.jpg"
    },
    {
      "id": 4,
      "email": "eve.holt@reqres.in",
      "first_name": "Eve",
      "last_name": "Holt",
      "avatar": "https://reqres.in/img/faces/4-image.jpg"
    }
  ]
}
```

For an extremely large "per_page" parameter, the API should ideally return a "400 Bad Request" error instead of a "200 OK" response. Currently, it returns a "200 OK" response with all users on 1 page.

1.1.11 Test Case: Get All Users with unknown query params.

Description: This test case aims to verify whether the "api/users" endpoint accepts unknown query parameters. Ideally, instead of returning any data, the API should respond with a "Bad Request" error when encountering unknown query parameters.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|----------------------------|---|---|-----------|----------|
| job="something", age=12 | Status Code should be 400 (Bad Request) | Status Code: 200 (OK) | Fail | High |
| | Response body error message: unknown query params | Response body did not provide error message | Fail | |

GET {{base_url}}/{{user_endpoint}}?job="something"&age=12

Params • Authorization Headers (6) Body Pre-request Script Tests • Settings

Query Params

| Key | Value |
|-----|-------------|
| job | "something" |
| age | 12 |

body Cookies Headers (17) Test Results (2/6)

Pretty Raw Preview Visualize JSON

```
1 {
2   "page": 1,
3   "per_page": 6,
4   "total": 12,
5   "total_pages": 2,
6   "data": [
7     {
8       "id": 1,
9       "email": "george.bluth@reqres.in",
10      "first_name": "George",
11      "last_name": "Bluth",
12      "avatar": "https://reqres.in/img/faces/1-image.jpg"
13    },
14    {
15      "id": 2,
16      "email": "janet.weaver@reqres.in",
17      "first_name": "Janet",
18      "last_name": "Weaver",
19      "avatar": "https://reqres.in/img/faces/2-image.jpg"
20    }
21  ]
22 }
```

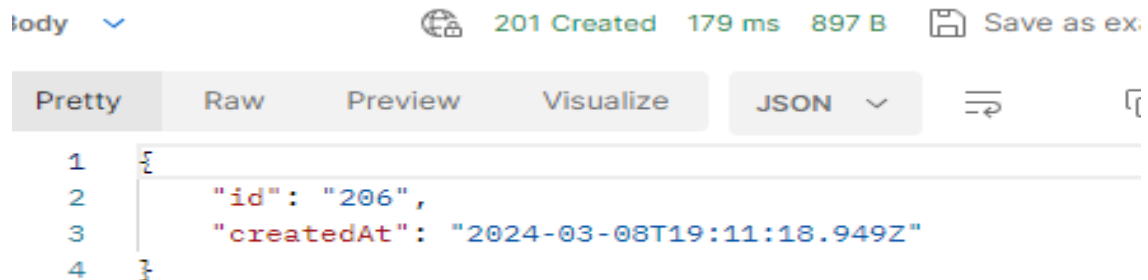
Even in the case of unknown query parameters, the API currently responds with a "200 OK" status code and returns all user data.

1.1.12 Test Case: Get All Users with valid page and per_page but Invalid POST method

Description: The request attempts to retrieve all users using /api/users endpoint with valid pagination parameters (page and per_page), but employs an invalid POST method, resulting in a method not allowed error.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-------------------------|---------------------------------------|----------------------|-----------|----------|
| Page = 1, per_page=4 | Status Code should be 405 | Status Code: 200 | Fail | High |
| | Response phrase: "Method Not Allowed" | Response phrase "OK" | Fail | |



The screenshot shows a REST client interface. At the top, it displays the status '201 Created' with a response time of '179 ms' and a size of '897 B'. Below this, there are tabs for 'Pretty', 'Raw', 'Preview', and 'Visualize', with 'Pretty' selected. The JSON payload is displayed as follows:

```
1 {  
2   "id": "206",  
3   "createdAt": "2024-03-08T19:11:18.949Z"  
4 }
```

Instead of responding with a 405 Method Not Allowed error as expected, the API erroneously returns a 201 Created status code, providing an ID (which is a string) and a "createdAt" field in the response payload.

1.1.13 Test Case: Get user with minimum valid id.

Description: This test case will check if the api/users/1 endpoint will show the correct user in the response body.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---------------------------|------------------|-----------|----------|
| Id = 1 | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Id number should be 1. | Id number is 1. | Pass | |

1.1.14 Test Case: Get user with maximum valid id.

Description: This test case will check if the api/users/12 endpoint will show the correct user in the response body.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---------------------------|------------------|-----------|----------|
| Id = 12 | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Id number should be 12. | Id number is 12. | Pass | |

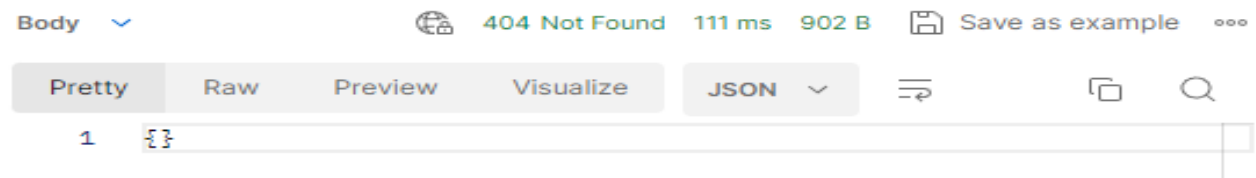
1.1.15 Test Case: Get User with non-existent id (beyond maximum).

Description: This test case will check if the api/users/13 endpoint show correct response code with an error message in the response body.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|--|------------------------|-----------|----------|
| Id = 13 | Status Code should be 404 | Status Code: 404 | Pass | Low |
| | Response body error message: User not found | Response body is empty | Fail | |

Visualization:



The error message for a non-existent ID is correct; however, it would be preferable if the API provided a more descriptive error message.

1.1.16 Test Case: Get User with non-existent id (less than minimum).

Description: This test case will check if the api/users/0 endpoint show correct response code with an error message in the response body.

Assertions and Severity:

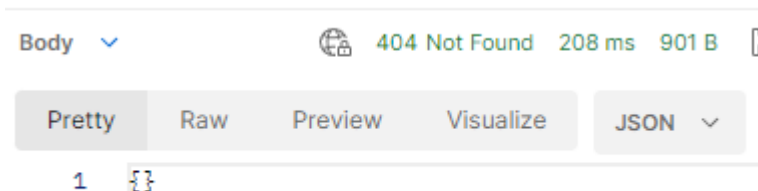
| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|--|------------------------|-----------|----------|
| | Status Code should be 404 | Status Code: 404 | Pass | Low |
| | Response body error message: User not found | Response body is empty | Fail | |

1.1.17 Test Case: Get User with negative id.

Description: This test case will check if the api/users/-2 endpoint show correct response code with an error message in the response body.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|--|------------------------|-----------|----------|
| | Status Code should be 400 (Bad Request) | Status Code: 404 | Pass | Low |
| | Response body error message: User not found | Response body is empty | Fail | |



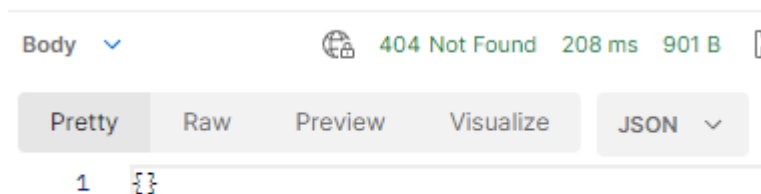
For a negative ID, the API should ideally respond with a "400 Bad Request" status code instead of a "404 Not Found" status code. Additionally, it would be preferable if the API provided an error message instead of an empty body in such cases.

1.1.18 Test Case: Get User with id (string format).

Description: This test case will check if the api/users/abc endpoint show correct response code with an error message in the response body.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---|------------------------|-----------|----------|
| | Status Code should be 400 (Bad Request) | Status Code: 404 | Pass | Low |
| | Response body error message: User not found | Response body is empty | Fail | |



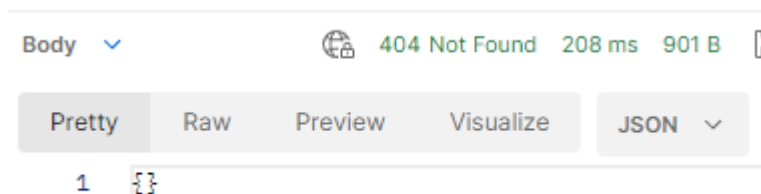
Id should be integer not string so it should have given 400 (Bad Request) error not 404 error.

1.1.19 Test Case: Get User with id (float).

Description: This test case will check if the api/users/5.6abc endpoint show correct response code with an error message in the response body.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---|------------------------|-----------|----------|
| | Status Code should be 400 (Bad Request) | Status Code: 404 | Pass | Low |
| | Response body error message: User not found | Response body is empty | Fail | |



Id should be integer. Not float, so it should have given 400 (Bad Request) error not 404 error.

1.2 Test Scenario: Create User

1.2.1 Test Case: Create User with all valid parameters.

Description: This test case will check if the api/users endpoint will create user and do Schema validation.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|--|-----------|----------|
| <pre>"email": "surraiya522@gmail.com", "first_name": "surraiya", "last_name": "tonni", "avatar": "https://reqres.in/img/faces/2-image.jpg"</pre> | Status Code should be 201 | Status Code: 201 | Pass | High |
| | Response body should contain id, first_name, last_name, email and createdAt fields. | Response body contains id, first_name, last_name, email and createdAt fields. | Pass | |
| | Schema should be valid | Schema is not valid | Fail | |
| | Verify user creation by retrieving the created user using GET req: Confirm user data: Email, first name, last name and avatar match the expected values : status should be 200 | Returns id as string in response and when using GET req to retrieve created user, returns 404 error. | Fail | |

Visualization:

| | | | | |
|--------|--|---------|-----------|-----------------|
| Body | 201 Created | 100 ms | 1.01 KB | Save as example |
| Pretty | Raw | Preview | Visualize | JSON |
| 1 | { | | | |
| 2 | "email": "surraiya522@gmail.com", | | | |
| 3 | "first_name": "surraiya", | | | |
| 4 | "last_name": "tonni", | | | |
| 5 | "avatar": "https://reqres.in/img/faces/2-image.jpg", | | | |
| 6 | "id": "344", | | | |
| 7 | "createdAt": "2024-03-08T08:33:49.827Z" | | | |
| 8 | } | | | |

The API returns the ID as a string in the response. However, when attempting to retrieve the created user using a GET request, it returns a "404 Not Found" error.

1.2.2 Test Case: Create User with valid email.

Description: This test case will check if the api/users endpoint will create user and do Schema validation.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|---|---|-----------|----------|
| <pre>"email": "surraiya123@gmail.com",</pre> | Status Code should be 201 | Status Code: 201 | Pass | High |
| | Response body should contain createdAt fields. | Response body contains createdAt fields. | Pass | |
| | Response body should contain first_name and last_name field. | Response body doesn't contain first_name and last_name field. | Fail | |
| | Schema should be valid | Schema is not valid | Fail | |
| | Should be possible to retrieve created user with GET request. (200) | Not possible to retrieve created user with GET request. (404) | Fail | |

```
Pretty  Raw  Preview  Visualize  JSON  ⌵  ⌵  
1  {  
2    "email": "surraiyaIslamtonni@gmail.com",  
3    "id": "56",  
4    "createdAt": "2024-03-08T19:39:43.444Z"  
5  }
```

The Json Schema for the response is not valid and the id here is string instead of integer.

GET

⌵

https://reqres.in/api/users/56

Params Authorization Headers (8) Body ● Pre-req

ody Cookies Headers (16) Test Results (2/15)

Pretty Raw Preview Visualize JSON ⌵ ⌵

1 {}

Retrieving the created user using the returned ID results in a "404 Not Found" error along with an empty body, indicating that the user was not successfully created.

1.2.3 Test Case: Create user with all empty body.

Description: This test case will check if the api/users endpoint will create user and do Schema validation.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|---|-----------|----------|
| <pre>"email": "", "first_name": "", "last_name": "", "avatar": ""</pre> | Status Code should be 400 | Status Code: 201 | Fail | Medium |
| | Response body should contain error message: Bad request | Response body doesn't contain error message | Fail | |
| | Response body should contain first_name and last_name field. | Response body doesn't contain first_name and last_name field. | Fail | |
| | Schema should be valid | Schema is not valid | Fail | |

Visualization:



```
Body 201 Created 116 ms 956 B Save as example ...  
Pretty Raw Preview Visualize JSON  
1 {  
2   "email": "",  
3   "first_name": "",  
4   "last_name": "",  
5   "avatar": "",  
6   "id": "272",  
7   "createdAt": "2024-03-08T08:36:03.220Z"  
8 }
```

The API incorrectly responds with a "201 Created" status code even for an entirely empty body of the JSON schema, which should warrant a "400 Bad Request" status code instead.

1.2.4 Test Case: Create an existing user.

Description: This test case will check if the api/users endpoint will create an user with existing user data and do Schema validation.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|------------------------------------|------------------|-----------|----------|
| <pre>"email": "janet.weaver@reqres.in", "first_name": "Janet",</pre> | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error | Response body | Fail | |

| | | | | |
|--|------------------------|---|------|--|
| <pre>"last_name": "Weaver", "avatar": "https://reqres.in/img/faces/2- image.jpg" }</pre> | message: Bad request | contains existing user data with different id | | |
| | Schema should be valid | Schema is not valid | Fail | |

Visualization:

Body 201 Created 102 ms 1020 B Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "email": "janet.weaver@reqres.in",
3   "first_name": "Janet",
4   "last_name": "Weaver",
5   "avatar": "https://reqres.in/img/faces/2-image.jpg",
6   "id": "423",
7   "createdAt": "2024-03-08T08:37:42.040Z"
8 }
```

Even though the email, first_name, last_name, and avatar fields already exist in the API, it accepts duplicate creation requests with a "201 Created" response code. Ideally, it should provide an error message indicating that the user already exists instead of allowing duplicate creations.

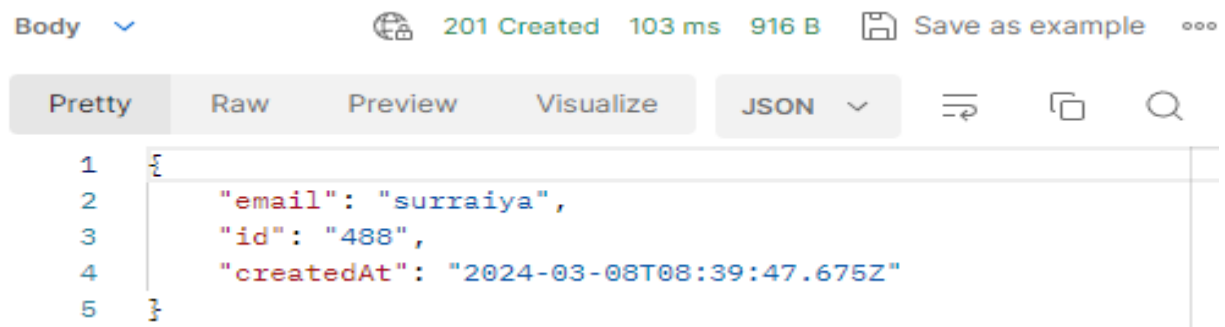
1.2.5 Test Case: Create User with invalid email.

Description: This test case will check if the api/users endpoint will create user with an invalid email and do Schema validation.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---------------------|---|---|-----------|----------|
| "email": "surrैया", | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body contains invalid email and id. | Fail | |
| | Response body should contain "emailInvalid" message. | Response body doesn't contain "emailInvalid" message. | Fail | |
| | Schema should be valid | Schema is not valid | Fail | |

Visualization:



```
1 {
2   "email": "surraiya",
3   "id": "488",
4   "createdAt": "2024-03-08T08:39:47.675Z"
5 }
```

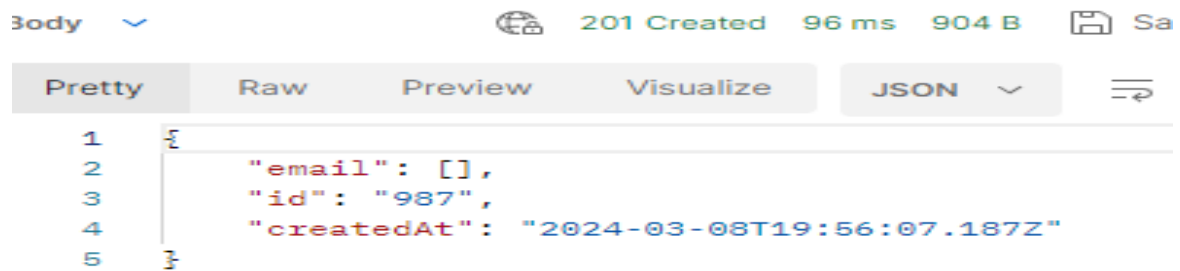
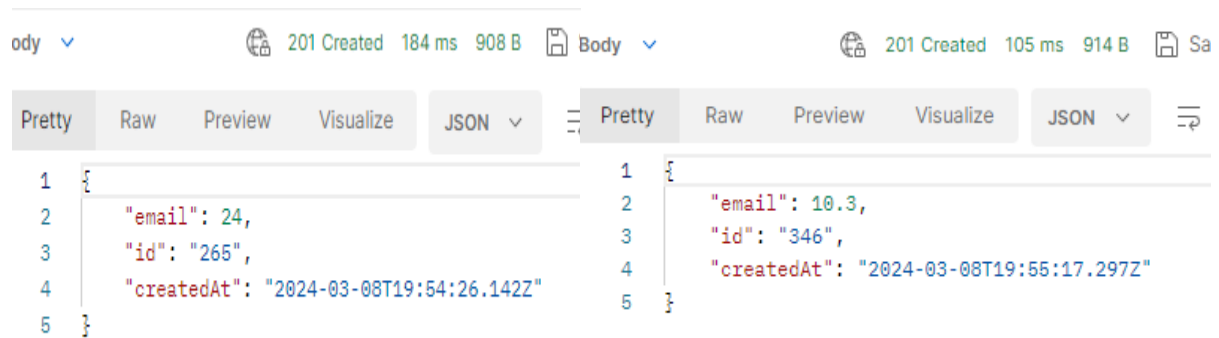
The API does not validate email addresses; it accepts any string as an email address without enforcing proper email format validation.

1.2.6 Test Case: Create user with invalid email format.

Description: This test case will check if the api/users endpoint will create user with an invalid email format.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|---|--|-----------|----------|
| <pre>"email": 24, "email": 10.3, "email": [],</pre> | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body contains invalid email and id. | Fail | |
| | Response body should contain "emailsInvalid" message. | Response body doesn't contain "emailsInvalid" message. | Fail | |



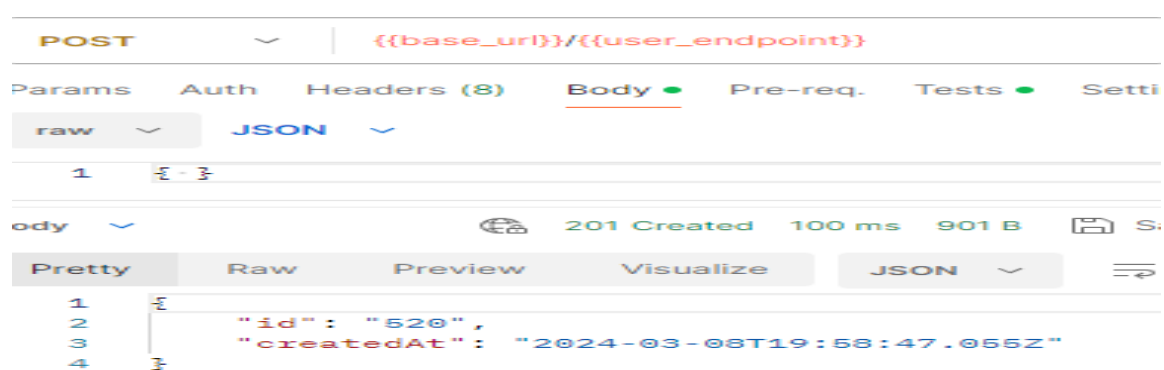
Based on the provided screenshots, it's evident that the API is accepting any data as an email without any form of validation.

1.2.7 Test Case: Create user with empty request body.

Description: This test case will check if the api/users endpoint will create user with an invalid email format.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|--|---|-----------|----------|
| {} | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain "requestBodyCannotBeEmpty" message. | Response body doesn't contain "requestBodyCannotBeEmpty" message. | Fail | |



Even with empty body, the api is responding with 201 Created. The response schema is invalid.

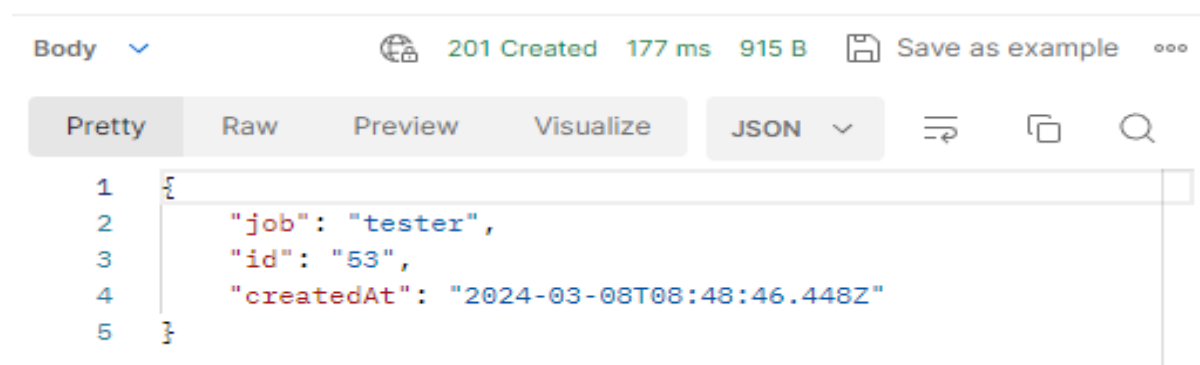
1.2.8 Test Case: Create user with an invalid field.

Description: This test case will check if the api/users endpoint will create user with an invalid field.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------------|---|--|-----------|----------|
| "job": "tester" | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain "invalidField" message. | Response body doesn't contain "invalidField" message. | Fail | |

Visualization:



The API allows requests with a "Job" field, even though it is not a valid field in the user schema. Additionally, it responds with an "id" (which is not even a string) and "createdAt." This behavior indicates a lack of proper schema validation and raises concerns about the consistency of the API's data handling.

1.2.9 Test Case: Create user without email.

Description: This test case will check if the api/users endpoint will create user without email. The best practice is to make email a required and unique field so that distinct user can be created. The main purpose of this test is to check if email is required field or not.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|---|--|-----------|----------|
| { "first_name": "surraiya", "last_name": "tonni", "avatar": "https://reqres.in/img/faces/2-image.jpg" } | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain | Response body doesn't contain | Fail | |

| | | | | |
|--|-------------------------------|-------------------------------|--|--|
| | "emailIsRequired" message. | "emailIsRequired" message. | | |
|--|-------------------------------|-------------------------------|--|--|

Body ▾ 201 Created 181 ms 993 B Save as exam

Pretty Raw Preview Visualize JSON ▾

```

1 {
2   "first_name": "surraiya",
3   "last_name": "tonni",
4   "avatar": "https://reqres.in/img/faces/2-image.jpg",
5   "id": "194",
6   "createdAt": "2024-03-08T20:12:10.031Z"
7 }

```

The API's response with a "201 Created" status indicates that it allows user creation without requiring an email field. This deviates from best practices, as making the email field required is typically considered good practice for user creation.

1.2.10 Test Case: Create user with only firstName.

Description: This test case will check if the api/users endpoint will create user with only firstName.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|---|--|-----------|----------|
| <pre>{ "first_name": "surraiya" }</pre> | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain "emailIsRequired" message. | Response body doesn't contain "emailIsRequired" message. | Fail | |

Body ▾ 201 Created 117 ms 925 B

Pretty Raw Preview Visualize JSON ▾

```

1 {
2   "first_name": "surraiya",
3   "id": "183",
4   "createdAt": "2024-03-08T20:16:12.876Z"
5 }

```

The API responds with an invalid user schema but still returns a "201 Created" status code.

1.2.10 Test Case: Create user with only lastName.

Description: This test case will check if the api/users endpoint will create user with only lastName.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|----------------------------------|---|--|-----------|----------|
| <pre>"first_name": "tonni"</pre> | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain "invalidField" message. | Response body doesn't contain "invalidField" message. | Fail | |

1.2.11 Test Case: Create user with large string in firstname.

Description: This test case will check if the api/users endpoint will create user with large string in firstname.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|---|-----------|----------|
| <pre>"first_name": "surraiya surraiyasurraiyasu rraiyasurraiyasurr aiyasurraiyasurrai yasurraiyasurraiya surraiyasurraiyasu rraiyasurraiyasurr aiyasurraiyasurrai yasurraiyasurraiya surraiyasurraiyasu rraiyasurraiyasurr aiya " "last_name": "Islam"</pre> | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain 'firstnameInvalid' message. | Response body doesn't contain 'firstnameInvalid' message. | Fail | |

1.2.12 Test Case: Create user with large string in lastName.

Description: This test case will check if the api/users endpoint will create user with large string in lastName.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|---|--|-----------|----------|
| <pre>"first_name": "surraiya", "last_name": "Islam idhooshogfihjosi"</pre> | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |

| | | | | |
|---------------------|---|--|------|--|
| 1fdgjsporeuuhuser " | Response body should contain 'lastNameInvalid' message. | Response body doesn't contain 'lastNameInvalid' message. | Fail | |
|---------------------|---|--|------|--|

1.2.13 Test Case: Create user with digit as input in first name.

Description: This test case will check if the api/users endpoint will create user with digit as input in first name.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-------------------|--|---|-----------|----------|
| "first_name" : 20 | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain 'firstNameInvalid' message. | Response body doesn't contain 'firstNameInvalid' message. | Fail | |

1.2.14 Test Case: Create user with digit as input in first name.

Description: This test case will check if the api/users endpoint will create user with digit as input in last name.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|----------------|---|--|-----------|----------|
| "last_name": 4 | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain 'lastNameInvalid' message. | Response body doesn't contain 'lastNameInvalid' message. | Fail | |

1.2.15 Test Case: Create user with special character as input in first name.

Description: This test case will check if the api/users endpoint will create user with special character as input in first name.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--------------------------------|--|---|-----------|----------|
| "first_name" : "#\${}%^##^" | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain 'firstNameInvalid' message. | Response body doesn't contain 'firstNameInvalid' message. | Fail | |

1.2.16 Test Case: Create user with special character as input in last name.

Description: This test case will check if the api/users endpoint will create user with special character as input in last name.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|------------------------------|---|--|-----------|----------|
| "last_name": "#\${}%^##^" | Status Code should be 400 | Status Code: 201 | Fail | Medium |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain 'lastNameInvalid' message. | Response body doesn't contain 'lastNameInvalid' message. | Fail | |

1.2.17 Test Case: Create user with only avatar.

Description: This test case will check if the api/users endpoint will create user with only avatar.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|---|--|-----------|----------|
| "avatar": "https://reqres.in/img/faces/2-image.jpg" | Status Code should be 400 | Status Code: 201 | Fail | Medium |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |

1.2.18 Test Case: Create user with invalid avatar format.

Description: This test case will check if the api/users endpoint will create user with invalid avatar format.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---------------------------|------------------|-----------|----------|
| | Status Code should be 400 | Status Code: 201 | Fail | Medium |

| | | | | |
|----------------|---|--|------|--|
| "avatar" : 123 | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain 'invalidAvatarFormat' message. | Response body doesn't contain 'invalidAvatarFormat' message. | Fail | |

1.2.19 Test Case: Create user with id.

Description: This test case will check if the api/users endpoint will create user with id.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|--|---|-----------|----------|
| "id" : 23 | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain 'ID field cannot be provided manually' error message. | Response body doesn't contain 'ID field cannot be provided manually' error message. | Fail | |

1.2.20 Test Case: Create user with existing id.

Description: This test case will check if the api/users endpoint will create user with existing id.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|--|---|-----------|----------|
| "id" : 1 | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain 'ID field cannot be provided manually' error message. | Response body doesn't contain 'ID field cannot be provided manually' error message. | Fail | |

1.2.21 Test Case: Create user with invalid id (string as input).

Description: This test case will check if the api/users endpoint will create user with invalid id format.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---------------------------|------------------|-----------|----------|
| | Status Code should be 400 | Status Code: 201 | Fail | High |

| | | | | |
|--------------------------|--|---|------|--|
| <code>"id": "abc"</code> | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain 'ID field cannot be provided manually' error message. | Response body doesn't contain 'ID field cannot be provided manually' error message. | Fail | |

1.2.22 Test Case: Create user with empty id.

Description: This test case will check if the api/users endpoint will create user with empty id.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|------------------------|--|---|-----------|----------|
| <code>"id": " "</code> | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain 'ID field cannot be provided manually' error message. | Response body doesn't contain 'ID field cannot be provided manually' error message. | Fail | |

1.2.23 Test Case: Create User with all valid parameters along with id.

Description: This test case will check if the api/users endpoint will create user with invalid id format.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|---|-----------|----------|
| <code>"id": 123,</code> <code>"email":</code> <code>"123surraiya@gmail.com",</code> <code>"first_name":</code> <code>"surraiya",</code> <code>"last_name":</code> <code>"tonni",</code> <code>"avatar":</code> <code>"https://reqres.in/img/faces/2-image.jpg"</code> | Status Code should be 400 | Status Code: 201 | Fail | High |
| | Response body should contain error message: Bad request | Response body doesn't contain error message: Bad request | Fail | |
| | Response body should contain 'ID field cannot be provided manually' error message. | Response body doesn't contain 'ID field cannot be provided manually' error message. | Fail | |

Visualization:

```
1 {
2   "id": 123,
3   "email": "surraiya65@gmail.com",
4   "first_name": "surraiya",
5   "last_name": "tonni",
6   "avatar": "https://reqres.in/img/faces/2-image.jpg",
7   "createdAt": "2024-03-08T08:55:24.347Z"
8 }
```

1.3 Test Scenario: Update User

1.3.1 Test Case: Update User of existing id with all correct body parameters.

Description: This test case will check if the api/users/1 endpoint will update the existing users' data.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|---|---|-----------|----------|
| <code>"email": "surraiya123@gmail.com", "first_name": "surraiya", "last_name": "tonni", "avatar": "https://reqres.in/img/faces/2-image.jpg"</code> | Status Code should be 200 | Status Code: 200 | Pass | High |
| | Response body should be updated with the new data. | Response body is updated with new data. | Pass | |
| | Schema should be valid | Schema is valid | Pass | |
| | Should be possible to retrieve updated user with GET request. | Not possible to retrieve created user with GET request. | Fail | |

1.3.2 Test Case: Update User of existing id with empty firstname.

Description: This test case will check if the api/users/1 endpoint will update the existing users' data with empty first name.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|---|-----------|----------|
| <code>"email": "surraiya123@gmail.com", "first_name": " ", "last_name": "tonni",</code> | Status Code should be 200 | Status Code: 200 | Pass | High |
| | Response body should be updated with the new data. | Response body is updated with new data. | Pass | |

| | | | | |
|---|---|---|------|--|
| <pre>"avatar": "https://reqres.in/img/faces/2- image.jpg"</pre> | Schema should be valid | Schema is valid | Pass | |
| | Should be possible to retrieve updated user with GET request. | Not possible to retrieve created user with GET request. | Fail | |

1.3.3 Test Case: Update User of existing id with empty lastname.

Description: This test case will check if the api/users/1 endpoint will update the existing users' data with empty last name.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|---|---|-----------|----------|
| <pre>"email": "surraiya123@gmail.com", "first_name": "Surraiya", "last_name": "", "avatar": "https://reqres.in/img/faces/2- image.jpg"</pre> | Status Code should be 200 | Status Code: 200 | Pass | High |
| | Response body should be updated with the new data. | Response body is updated with new data. | Pass | |
| | Schema should be valid | Schema is valid | Pass | |
| | Should be possible to retrieve updated user with GET request. | Not possible to retrieve created user with GET request. | Fail | |

1.3.4 Test Case: Update user with empty avatar.

Description: This test case will check if the api/users/1 endpoint will update the existing users' data with empty avatar.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|---|-----------|----------|
| <pre>"email": "surraiya123@gmail.com", "first_name": "surraiya", "last_name": "tonni", "avatar":</pre> | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Response body should be updated with the new data. | Response body is updated with new data. | Pass | |
| | Schema should be valid | Schema is valid | Pass | |

| | | | | |
|--|---|---|------|--|
| | Should be possible to retrieve updated user with GET request. | Not possible to retrieve created user with GET request. | Fail | |
|--|---|---|------|--|

1.3.5 Test Case: Update user of non-existing id with all correct body

Description: This test case will check if the api/users/123 endpoint will update the existing users' data with empty avatar.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|---|--|-----------|----------|
| <pre>"email": "surraiya123@gmail.com", "first_name": "surraiya", "last_name": "tonni", "avatar": "https://reqres.in/img/faces/2-image.jpg"</pre> | Status Code should be 405 | Status Code: 200 | Fail | Medium |
| | Response body should contain 'idNotExist' message | Response body doesn't contain 'idNotExist' message | Fail | |
| | Status phrase should be: Method Not Allowed | Status phrase: OK | Fail | |

1.3.6 Test Case: Update User of existing id with all empty body parameters.

Description: This test case will check if the api/users/1 endpoint will update the existing users' data with all empty body parameters.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|---|--|-----------|----------|
| <pre>"email": "", "first_name": "", "last_name": "", "avatar": ""</pre> | Status Code should be 400 | Status Code: 200 | Fail | Medium |
| | Response phrase should contain error message: Bad request | Response phrase doesn't contain error message: Bad request | Fail | |

1.3.7 Test Case: Update user of existing id with empty email

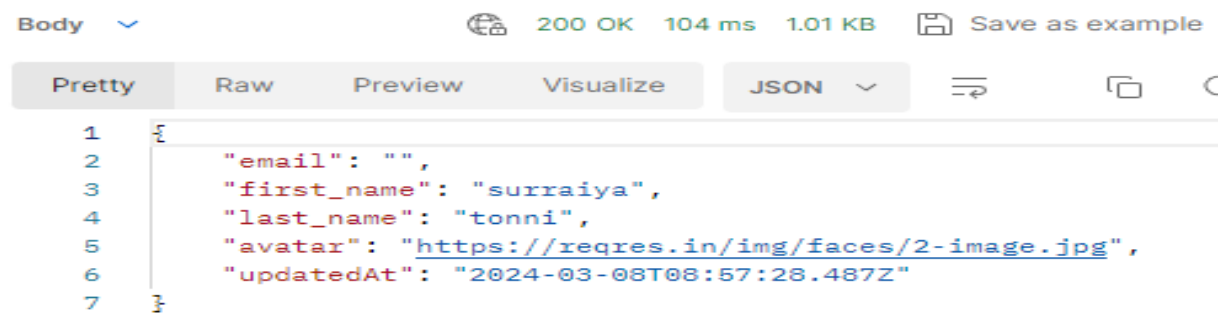
Description: This test case will check if the api/users/1 endpoint will update the existing users' data with empty email.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---------------------------|------------------|-----------|----------|
| | Status Code should be 400 | Status Code: 200 | Fail | High |

| | | | | |
|---|--|---|------|--|
| <pre>"email": "", "first_name": "surraiya", "last_name": "tonni", "avatar": "https://reqres.in/img/faces/2-image.jpg"</pre> | Response body should contain 'emailsMissing' message | Response body doesn't contain 'emailsMissing' message | Fail | |
| | Status phrase should be: Bad request | Status phrase: OK | Fail | |

Visualization:



1.3.8 Test Case: Update user of existing id with invalid email format

Description: This test case will check if the api/users/1 endpoint will update the existing users' data with invalid email format.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|---|--|-----------|----------|
| <pre>"email": "surraiya", "first_name": "surraiya", "last_name": "tonni", "avatar": "https://reqres.in/img/faces/2-image.jpg"</pre> | Status Code should be 400 | Status Code: 200 | Fail | High |
| | Response body should contain 'invalidEmail' message | Response body doesn't contain 'invalidEmail' message | Fail | |
| | Status phrase should be: Bad request | Status phrase: OK | Fail | |

1.3.9 Test Case: Update user of existing id with invalid first name format

Description: This test case will check if the api/users/1 endpoint will update the existing users' data with invalid first name format.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|------------------------------|-----------------------|-----------|----------|
| <pre>"email": "surraiya@gmail.com", "first_name": 123,</pre> | Status Code should be 400 | Status Code: 200 | Fail | High |
| | Response body should contain | Response body doesn't | Fail | |

| | | | | |
|---|--------------------------------------|-------------------------------------|------|--|
| <pre>"last_name": "tonni", "avatar": "https://reqres.in/img/faces /2-image.jpg"</pre> | 'invalidFirstName' message | contain 'invalidFirstNa me' message | | |
| | Status phrase should be: Bad request | Status phrase: OK | Fail | |

1.3.10 Test Case: Update user of existing id with invalid first name format

Description: This test case will check if the api/users/1 endpoint will update the existing users' data with invalid last name format.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|--|-----------|----------|
| <pre>"email": "surrैया@gmail.com", "first_name": "surrैया", "last_name": 123, "avatar": "https://reqres.in/img/faces /2-image.jpg"</pre> | Status Code should be 400 | Status Code: 200 | Fail | High |
| | Response body should contain 'invalidLastName' message | Response body doesn't contain 'invalidLastNa me' message | Fail | |
| | Status phrase should be: Bad request | Status phrase: OK | Fail | |

1.3.11 Test Case: Update User of existing id with large string in first name

Description: This test case will check if the api/users/1 endpoint will update the existing users' data with large string in first name.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|---|---|-----------|----------|
| <pre>"email": "surrैया@gmail.com", "first_name": "surrैयाsurrैया surrैयाsurrैयाsurrैयाsurr surrैयाsurrैयाsurrैयाsurr surrैयाsurrैयाsurrैयाsurr surrैयाsurrैयाsurrैयाsurr surrैयाsurrैयाsurrैयाsurr surrैयाsurrैया", "last_name": "tonni", "avatar": "https://reqres.in/img/faces /2-image.jpg"</pre> | Status Code should be 400 | Status Code: 200 | Fail | Low |
| | Response body should contain 'invalidFirstName' message | Response body doesn't contain 'invalidFirstNa me' message | Fail | |
| | Status phrase should be: Bad request | Status phrase: OK | Fail | |

1.3.12 Test Case: Update User of existing id with large string in last name

Description: This test case will check if the api/users/1 endpoint will update the existing users' data with large string in last name.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|---|---|-----------|----------|
| <pre>"email": "surraiya@gmail.com", "first_name": "tonni", "last_name": "surraiyasurraiya surraiyasurraiyasurraiyasurr aiyasurraiyasurraiyasurraiya surraiyasurraiyasurraiyasurr aiyasurraiyasurraiyasurraiya surraiyasurraiyasurraiyasurr aiyasurraiyasurraiyasurraiya surraiyasurraiya", "avatar": "https://reqres.in/img/faces /2-image.jpg"</pre> | Status Code should be 400 | Status Code: 200 | Fail | Low |
| | Response body should contain 'invalidLastName' message | Response body doesn't contain 'invalidFirstName' message | Fail | |
| | Status phrase should be: Bad request | Status phrase: OK | Fail | |

1.3.13 Test Case: Update User of existing id with large string in avatar

Description: This test case will check if the api/users/1 endpoint will update the existing users' data with large string in avatar.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|---|---|-----------|----------|
| <pre>"email": "\${randomEmail}", "first_name": "tonni", "last_name": "tonni", "avatar": "https://surraiyasurraiya surraiyasurraiyasurraiyasurr aiyasurraiyasurraiyasurraiya surraiyasurraiyasurraiyasurr aiyasurraiyasurraiyasurraiya surraiyasurraiyasurraiyasurr aiyasurraiyasurraiyasurraiya surraiyasurraiya"</pre> | Status Code should be 400 | Status Code: 200 | Fail | Medium |
| | Response body should contain 'invalidAvatar' message | Response body doesn't contain 'invalidFirstName' message | Fail | |
| | Status phrase should be: Bad request | Status phrase: OK | Fail | |

1.3.14 Test Case: Update user of existing id with invalid avatar format

Description: This test case will check if the api/users/1 endpoint will update the existing users' data with invalid avatar.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|--|-----------|----------|
| <code>"email": "{\$randomEmail}",</code> <code>"first_name": "surraiya",</code> <code>"last_name": "tonni",</code> <code>"avatar": "avatar"</code> | Status Code should be 400 | Status Code: 200 | Fail | Medium |
| | Response body should contain 'invalidAvatarFormat' message | Response body doesn't contain 'invalidFirstName' message | Fail | |
| | Status phrase should be: Bad request | Status phrase: OK | Fail | |

1.3.15 Test Case: Update User using non-existent field in schema.

Description: This test case will check if the api/users/1 endpoint will update the existing users' data with non-existent field in schema.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|---|--|-----------|----------|
| <code>"job": "software engineer",</code> <code>"age": 20</code> | Status Code should be 400 | Status Code: 200 | Fail | High |
| | Response body should contain 'invalidField' message | Response body doesn't contain 'invalidField' message | Fail | |
| | Status phrase should be: Bad request | Status phrase: OK | Fail | |

1.4 Test Scenario: Delete User

1.4.1 Test Case: Delete User of existing id.

Description: This test case will check if the api/users/2 endpoint will delete user data.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|-------------------------------------|---------------------------|-----------|----------|
| | Status Code should be 204 | Status Code: 204 | Pass | High |
| | Status phrase should be: No Content | Status phrase: No Content | Pass | |

1.4.2 Test Case: Verify Delete User of existing id.

Description: This test case will check if the api/users/2 endpoint deleted the user.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---------------------------|------------------|-----------|----------|
| | Status Code should be 404 | Status Code: 200 | Fail | High |

| | | | | |
|--|------------------------------------|-------------------|------|--|
| | Status phrase should be: Not found | Status phrase: Ok | Fail | |
|--|------------------------------------|-------------------|------|--|

1.4.3 Test Case: Delete User of non-existing id.

Description: This test case will check if the api/users/20 endpoint delete user with non-existing id.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|------------------------------------|---------------------------|-----------|----------|
| | Status Code should be 404 | Status Code: 204 | Fail | Medium |
| | Status phrase should be: Not Found | Status phrase: No Content | Fail | |

2. Endpoint: /api/unknown

2.1 Test Scenario: Request resources

2.1.1 Test Case: Request with Default Parameters

Description: This test case will check if the api/unknown endpoint will request for resources with default parameters.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|--|---------------------------------------|-----------|----------|
| | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Response body should not be empty | Response body is not empty | Pass | |
| | Response body should contain users list | Response body contains users list | Pass | |
| | Schema should be valid | Schema is valid | Pass | |
| | Total number of resources should be correct. | Total number of resources is correct. | Pass | |

2.1.2 Test Case: Request with Valid Page and Per Page Parameters

Description: This test case will check if the api/unknown?page=4&per_page=3 endpoint will request for resources with Valid Page and Per Page Parameters.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------------------------------|--|--|-----------|----------|
| Params Page = 4 Per page= 3 | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Response body should not be empty | Response body is not empty | Pass | |
| | Response body should contain 3 user per page | Response body contains 3 user per page | Pass | |
| | Schema should be valid | Schema is valid | Pass | |
| | Total number of pages should be 4 | Total number of pages is 4 | Pass | |

2.1.3 Test Case: Request for the Last Page

Description: This test case will check if the api/unknown?page=2 endpoint will request for the last page.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--------------------|--|--|-----------|----------|
| Params Page = 2 | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Response body should not be empty | Response body is not empty | Pass | |
| | Response body should contain 6 user per page | Response body contains 6 user per page | Pass | |
| | Schema should be valid | Schema is valid | Pass | |
| | Page should be 2 | Page is 2 | Pass | |

2.1.4 Test Case: Request for a Single Item on a Page.

Description: This test case will check if the api/unknown?per_page=1 endpoint will request for the last page.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|------------------------|--|--|-----------|----------|
| Params Per_page = 1 | Status Code should be 200 | Status Code: 200 | Pass | Medium |
| | Response body should not be empty | Response body is not empty | Pass | |
| | Response body should contain 1 user per page | Response body contains 1 user per page | Pass | |
| | Schema should be valid | Schema is valid | Pass | |
| | Total Page should be 12 | Total Page is 12 | Pass | |

2.1.5 Test Case: Request with Default Parameters using invalid method Post.

Description: This test case will check if the api/unknown endpoint will request for resources with default parameters using invalid method POST.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---|--------------------------|-----------|----------|
| | Status Code should be 405 | Status Code: 201 | Fail | Low |
| | Response body should contain error message | Response body contain id | Fail | |
| | Status phrase should be: Method Not Allowed | Status phrase: Create | Fail | |

2.1.6 Test Case: Request for a Page Beyond Total Pages.

Description: This test case will check if the api/unknown?page=20 endpoint will request for resources with page number which is beyond total pages.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---------------------------|------------------|-----------|----------|
| Params | Status Code should be 404 | Status Code: 200 | Fail | Low |

| | | | | |
|---------|--|-------------------------------------|------|--|
| Page=20 | Response body should contain error message | Response body contain error message | Fail | |
| | Status phrase should be: Not Found | Status phrase: OK | Fail | |

Visualization:

Body 200 OK 182 ms 1.09 KB Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "page": 20,
3   "per_page": 6,
4   "total": 12,
5   "total_pages": 2,
6   "data": [],
7   "support": {
8     "url": "https://reqres.in/#support-heading",
9     "text": "To keep ReqRes free, contributions towards serv
10 }
11 }
```

Body Cookies Headers (17) Test Results (3/6) Status: 200 OK

All Passed Skipped Failed

- PASS** Response is a JSON object
- PASS** Response body is not empty
- FAIL** For retrieving page beyond total pages, response status code should be 404 | AssertionError: expected response to have status code 404 but got 200
- FAIL** For retrieving page beyond total pages, response phrase should be: Not Found | AssertionError: expected response to have status reason 'Not Found' but got 'OK'
- PASS** For retrieving page beyond total pages, response should contain: []
- FAIL** For retrieving page beyond total pages, response should contain: page doesn't exist | AssertionError: expected '{"page":20,"per_page":6,"total":12,"t..."' to include 'page doesn't exist'

2.1.7 Test Case: Request with Negative Page Number

Description: This test case will check if the `api/unknown?page=-2` endpoint will request for resources with negative page number.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--------------------|--|-------------------------------------|-----------|----------|
| Params Page= -2 | Status Code should be 404 | Status Code: 200 | Fail | Low |
| | Response body should contain error message | Response body contain error message | Fail | |
| | Status phrase should be: Not Found | Status phrase: OK | Fail | |

2.1.8 Test Case: Request with Negative Items Per Page

Description: This test case will check if the `api/unknown?page=-2` endpoint will request for resources with negative item per page.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|----------|--------|-----------|----------|
|-----------|----------|--------|-----------|----------|

| | | | | |
|----------------------------|--|-------------------------------------|------|-----|
| Params Per_page= - 2 | Status Code should be 400 | Status Code: 200 | Fail | Low |
| | Response body should contain error message | Response body contain error message | Fail | |
| | Status phrase should be: Bad Request | Status phrase: OK | Fail | |

2.1.9 Test Case: Request with Non-Integer Page Number (String)

Description: This test case will check if the api/unknown?page=-abc endpoint will request for resources with Non-Integer Page Number (String).

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|----------------------|--|-------------------------------------|-----------|----------|
| Params Page = abc | Status Code should be 400 | Status Code: 200 | Fail | Low |
| | Response body should contain error message | Response body contain error message | Fail | |
| | Status phrase should be: Bad Request | Status phrase: OK | Fail | |

2.1.10 Test Case: Request with Exceedingly Large Page Number

Description: This test case will check if the api/unknown?page=-1265436789 endpoint will request for resources with Exceedingly Large Page Number.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--------------------------------|--|-------------------------------------|-----------|----------|
| Params Page = 1265436789 | Status Code should be 404 | Status Code: 200 | Fail | Medium |
| | Response body should contain error message | Response body contain error message | Fail | |
| | Status phrase should be: Not Found | Status phrase: OK | Fail | |

2.2 Test Scenario: Create Resource

2.2.1 Test Case: Create Resource with all valid data.

Description: This test case will check if the api/unknown endpoint will create resource with all valid data.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|--|-----------|----------|
| "id": 30, "name": "surruiya", "year": 2010, "color": "blue", "pantone_value": "value" | Status Code should be 201 | Status Code: 201 | Pass | Medium |
| | Response body should contain newly created user data | Response body contains contain newly created user data | Pass | |
| | Status phrase should be: Created | Status phrase: Created | Pass | |
| | Schema should be valid | Schema invalid | Fail | |

2.2.2 Test Case: Create Resource with only name.

Description: This test case will check if the api/unknown endpoint will create resource with only name.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------------------|--|--|-----------|----------|
| "name": "surraiya" | Status Code should be 201 | Status Code: 201 | Pass | Medium |
| | Response body should contain newly created resource data | Response body contains contain newly created resource data | Pass | |
| | Status phrase should be: Created | Status phrase: Created | Pass | |
| | Schema should be valid | Schema invalid | Fail | |

2.2.3 Test Case: Create Resource with only color.

Description: This test case will check if the api/unknown endpoint will create resource with only color.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|----------------------|--|--|-----------|----------|
| "color": "yellow" | Status Code should be 201 | Status Code: 201 | Pass | Medium |
| | Response body should contain newly created resource data | Response body contains contain newly created resource data | Pass | |
| | Status phrase should be: Created | Status phrase: Created | Pass | |
| | Schema should be valid | Schema invalid | Fail | |

2.3 Test Scenario: Update Resource

2.3.1 Test Case: Update Resource of existing id with all valid data.

Description: This test case will check if the api/unknown/1 endpoint will update resource of existing id with all valid data.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|--|-----------|----------|
| "name": "surraiya", "year": 2010, "color": "blue", "pantone_value": "value" | Status Code should be 200 | Status Code: 200 | Pass | High |
| | Response body should contain newly updated resource data | Response body contains contain newly updated user data | Pass | |
| | Status phrase should be: OK | Status phrase: OK | Pass | |

| | | | | |
|--|--|---------------------------|------|--|
| | Should be possible to request updated data with GET method | GET method shows old data | Fail | |
|--|--|---------------------------|------|--|

2.3.2 Test Case: Update Resource of existing id with empty name.

Description: This test case will check if the api/unknown/1 endpoint will update resource of existing id with empty name.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|--|-----------|----------|
| <pre>"name": "", "year": 2010, "color": "blue", "pantone_value": "value"</pre> | Status Code should be 200 | Status Code: 200 | Pass | High |
| | Response body should contain newly updated resource data | Response body contains contain newly updated user data | Pass | |
| | Status phrase should be: OK | Status phrase: OK | Pass | |
| | Should be possible to request updated data with GET method | GET method shows old data | Fail | |

2.3.3 Test Case: Update Resource of existing id with empty year.

Description: This test case will check if the api/unknown/1 endpoint will update resource of existing id with empty year.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|--|-----------|----------|
| <pre>"name": "", "year": 0, "color": "blue", "pantone_value": "value"</pre> | Status Code should be 200 | Status Code: 200 | Pass | High |
| | Response body should contain newly updated resource data | Response body contains contain newly updated user data | Pass | |
| | Status phrase should be: OK | Status phrase: OK | Pass | |
| | Should be possible to request updated data with GET method | GET method shows old data | Fail | |

2.3.4 Test Case: Update Resource of existing id with empty color.

Description: This test case will check if the api/unknown/1 endpoint will update resource of existing id with empty color.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|------------------------|---------------------------|------------------|-----------|----------|
| <pre>"name": "",</pre> | Status Code should be 200 | Status Code: 200 | Pass | High |

| | | | | |
|--|--|--|------|--|
| <pre>"year": 2010, "color": "", "pantone_value": "value"</pre> | Response body should contain newly updated resource data | Response body contains contain newly updated user data | Pass | |
| | Status phrase should be: OK | Status phrase: OK | Pass | |
| | Should be possible to request updated data with GET method | GET method shows old data | Fail | |

2.3.5 Test Case: Update Resource of existing id with empty pantone value.

Description: This test case will check if the api/unknown/1 endpoint will update resource of existing id with empty color.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|--|-----------|----------|
| <pre>"name": "", "year": 2010, "color": "", "pantone_value": ""</pre> | Status Code should be 200 | Status Code: 200 | Pass | High |
| | Response body should contain newly updated resource data | Response body contains contain newly updated user data | Pass | |
| | Status phrase should be: OK | Status phrase: OK | Pass | |
| | Should be possible to request updated data with GET method | GET method shows old data | Fail | |

2.3.6 Test Case: Partial Update Resource of existing id with only name.

Description: This test case will check if the api/unknown/1 endpoint will partially update resource of existing id with only name.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-------------------------------|--|--|-----------|----------|
| <pre>"name": "surraiya"</pre> | Status Code should be 200 | Status Code: 200 | Pass | High |
| | Response body should contain newly updated resource data | Response body contains contain newly updated user data | Pass | |
| | Status phrase should be: OK | Status phrase: OK | Pass | |

| | | | | |
|--|--|---------------------------|------|--|
| | Should be possible to request updated data with GET method | GET method shows old data | Fail | |
|--|--|---------------------------|------|--|

2.3.7 Test Case: Partial Update Resource with only year.

Description: This test case will check if the api/unknown/1 endpoint will partially update resource of existing id with only year.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--------------|--|--|-----------|----------|
| "year": 2010 | Status Code should be 200 | Status Code: 200 | Pass | High |
| | Response body should contain newly updated resource data | Response body contains contain newly updated user data | Pass | |
| | Status phrase should be: OK | Status phrase: OK | Pass | |
| | Should be possible to request updated data with GET method | GET method shows old data | Fail | |

2. Test Scenario: Delete Resource

2.4.1 Test Case: Delete Resource of existing id.

Description: This test case will check if the api/unknown/2 endpoint will delete resource data.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|-------------------------------------|---------------------------|-----------|----------|
| | Status Code should be 204 | Status Code: 204 | Pass | High |
| | Status phrase should be: No Content | Status phrase: No Content | Pass | |

2.4.2 Test Case: Verify Delete Resource of existing id.

Description: This test case will check if the api/unknown/2 endpoint will verify that resource data is deleted.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|------------------------------------|-------------------|-----------|----------|
| | Status Code should be 404 | Status Code: 200 | Fail | High |
| | Status phrase should be: Not Found | Status phrase: OK | Fail | |

2.4.3 Test Case: Delete Resource of non-existing id.

Description: This test case will check if the api/unknown/20 endpoint will verify that resource data is deleted.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|------------------------------------|---------------------------|-----------|----------|
| | Status Code should be 404 | Status Code: 204 | Fail | Medium |
| | Status phrase should be: Not Found | Status phrase: No Content | Fail | |

3. Endpoint: /api/register

3.1 Test Scenario: Register User

3.1.1 Test Case: Registration of defined user with all valid parameter (username, email and password)

Description: This test case will check if the api/register endpoint will register the existing users with all valid parameter (username, email, and password).

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|---|-----------|----------|
| <code>"username": "eve", "email": "eve.holt@reqres.in", "password": "pistol"</code> | Status Code should be 200 | Status Code: 400 | Fail | Medium |
| | Response body should contain registered users' id and token. | Response body contain "Note: Only defined users succeed registration" message | Fail | |
| | Status phrase should be: OK | Status phrase: Bad Request | Fail | |

3.1.2 Test Case: Registration of defined user with email and password only

Description: This test case will check if the api/register endpoint will register the existing users with email and password only.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|--|-----------|----------|
| <code>"email": "eve.holt@reqres.in", "password": "pistol"</code> | Status Code should be 200 | Status Code: 400 | Pass | Medium |
| | Response body should contain registered users' id and token. | Response body contains registered users' id and token. | Pass | |
| | Status phrase should be: OK | Status phrase: OK | Pass | |

3.1.3 Test Case: Registration of undefined user

Description: This test case will check if the api/register endpoint will register the undefined user.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|---|-----------|----------|
| <pre>"username": "surraiya", "email": "surraiya123@gmail.com", "password": "abc12"</pre> | Status Code should be 400 | Status Code: 400 | Pass | Medium |
| | Response body should contain "Note: Only defined users succeed registration" message | Response body contain "Note: Only defined users succeed registration" message | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |

3.1.4 Test Case: Registration with existing email and password

Description: This test case will check if the api/register endpoint will register with existing email and password.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|--|-----------|----------|
| <pre>"email": "janet.weaver@reqres.in", "password": "123"</pre> | Status Code should be 400 | Status Code: 200 | Fail | High |
| | Response body should contain 'User already registered' | Response body contains registered users' id and token. | Fail | |
| | Status phrase should be: Bad Request | Status phrase: OK | Fail | |

3.1.5 Test Case: Registration with empty email and empty password

Description: This test case will check if the api/register endpoint will register with empty email and empty password.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|---|---|-----------|----------|
| <pre>"email": "", "password": ""</pre> | Status Code should be 400 | Status Code: 400 | Pass | High |
| | Response body should contain "error": "Missing email or username" | Response body contains "error": "Missing email or username" | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |

3.1.6 Test Case: Registration with empty email and valid password

Description: This test case will check if the api/register endpoint will register with empty email and valid password.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|---|---|-----------|----------|
| <pre>"email": "", "password": "pistol"</pre> | Status Code should be 400 | Status Code: 400 | Pass | High |
| | Response body should contain "error": "Missing email or username" | Response body contains "error": "Missing email or username" | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |

3.1.7 Test Case: Registration with valid email and empty password

Description: This test case will check if the api/register endpoint will register with valid email and empty password.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|--|-----------|----------|
| <pre>"email": "eve.holt@reqres.in", "password": ""</pre> | Status Code should be 400 | Status Code: 400 | Pass | Medium |
| | Response body should contain "error": "Missing password" | Response body contains "error": "Missing password" | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |

3.1.8 Test Case: Registration of defined user with email only

Description: This test case will check if the api/register endpoint will register with valid email only.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|--|-----------|----------|
| <pre>"email": "eve.holt@reqres.in"</pre> | Status Code should be 400 | Status Code: 400 | Pass | Medium |
| | Response body should contain "error": "Missing password" | Response body contains "error": "Missing password" | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |

3.1.9 Test Case: Registration of defined user with invalid email only

Description: This test case will check if the api/register endpoint will register with invalid email only.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---------------------------------------|--|--|-----------|----------|
| <pre>"email": "eveholtreqresin"</pre> | Status Code should be 400 | Status Code: 400 | Pass | Medium |
| | Response body should contain "error": "Missing password" | Response body contains "error": "Missing password" | Pass | |

| | | | | |
|--|---|----------------------------|------|--|
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |
|--|---|----------------------------|------|--|

3.1.10 Test Case: Registration of defined user with password only

Description: This test case will check if the api/register endpoint will register with password only.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-------------------------------|---|---|-----------|----------|
| "email": "eveholtreqresin" | Status Code should be 400 | Status Code: 400 | Pass | Medium |
| | Response body should contain "error": "Missing email or username" | Response body contains "error": "Missing email or username" | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |

3.1.11 Test Case: Registration of defined user with long password

Description: This test case will check if the api/register endpoint will register with long password.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|--|-----------|----------|
| "email": "eve.holt@reqres.in", "password": "aaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaa" | Status Code should be 400 | Status Code: 200 | Fail | Medium |
| | Response body should contain "error": "invalid password" | Response body contains registered users' id and token. | Fail | |
| | Status phrase should be: Bad Request | Status phrase: OK | Fail | |

3.1.12 Test Case: Registration of defined user with long email

Description: This test case will check if the api/register endpoint will register with long email.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|---|-----------|----------|
| <code>"email": "aaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaa aa@regres.in", "password": "123"</code> | Status Code should be 400 | Status Code: 400 | Pass | High |
| | Response body should contain "Note: Only defined users succeed registration" message | Response body contain "Note: Only defined users succeed registration" message | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |
| | Response body should contain "error": "Invalid email" | Response body doesn't contain "error": "invalid password" | Fail | |

3.1.13 Test Case: Registration of defined user with invalid email and password

Description: This test case will check if the api/register endpoint will register the defined user with invalid email and password

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|---|-----------|----------|
| <code>"email": "eveholtregresin", "password": "pistol"</code> | Status Code should be 400 | Status Code: 400 | Pass | Medium |
| | Response body should contain "Note: Only defined users succeed registration" message | Response body contain "Note: Only defined users succeed registration" message | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |

3.1.14 Test Case: Registration of defined user with valid email and invalid password

Description: This test case will check if the api/register endpoint will register the existing users with valid email and invalid password.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|---------------------------|------------------|-----------|----------|
| | Status Code should be 400 | Status Code: 200 | Fail | Medium |

| | | | | |
|---|--|--|------|--|
| <pre>"email": "eve.holt@reqres.in", "password": 123</pre> | Response body should contain "error": "invalid password" | Response body contains registered users' id and token. | Fail | |
| | Status phrase should be: Bad Request | Status phrase: OK | Fail | |

3.1.15 Test Case: Registration with Invalid Request Body Media Type.

Description: This test case will check if the api/register endpoint will register with Request Body Media Type.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|---|---|-----------|----------|
| <pre>"email": "eveholt@reqres.in"</pre> | Status Code should be 400 | Status Code: 400 | Pass | Medium |
| | Response body should contain "error": "Missing email or username" | Response body contains "error": "Missing email or username" | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |

3.1.16 Test Case: Registration with wrong method (GET) but correct URL.

Description: This test case will check if the api/register endpoint will work with wrong method (GET) but correct URL.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|-----------------------------------|-----------|----------|
| <pre>"email": "janet.weaver@reqres.in", "password": "123"</pre> | Status Code should be 405 | Status Code: 200 | Fail | High |
| | Response body should contain "error in request method" | Response body contains user list. | Fail | |
| | Status phrase should be: Method Not Allowed | Status phrase: OK | Fail | |

3.1.17 Test Case: Incorrect HTTP Method (PUT) and URL for Registration

Description: This test case will check if the api/register endpoint will work with wrong method (PUT) but correct URL.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|-----------------------------------|-----------|----------|
| <code>"email": "janet.weaver@reqres. in", "password": "123"</code> | Status Code should be 405 | Status Code: 200 | Fail | High |
| | Response body should contain "error in request method" | Response body contains user list. | Fail | |
| | Status phrase should be: Method Not Allowed | Status phrase: OK | Fail | |

3.1.18 Test Case: Registration of a new user.

Description: This test case will check if the api/register endpoint will register with valid parameters (username, email, and password).

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|---|-----------|----------|
| <code>"username": "surraiya", "email": "surraiya123@gmail .com", "password": "abc123"</code> | Status Code should be 200 | Status Code: 400 | Fail | Medium |
| | Response body should contain registered users' id and token. | Response body contain "Note: Only defined users succeed registration" message | Fail | |
| | Status phrase should be: OK | Status phrase: Bad Request | Fail | |

Visualization:

| | | | | |
|--------|--|--------------|--------------------|-------------------------|
| Body | Cookies | Headers (14) | Test Results (2/4) | Status: 400 Bad Request |
| Pretty | Raw | Preview | Visualize | JSON |
| 1 | { | | | |
| 2 | "error": "Note: Only defined users succeed registration" | | | |
| 3 | } | | | |

A new user should able to register him/her self with the API.

4. Endpoint: /api/login

4.1 Test Scenario: Post User

4.1.1 Test Case: Login of defined user with all valid parameter (email and password).

Description: This test case will check if the api/login endpoint will work for defined user with all valid parameter (email and password).

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|--|-----------|----------|
| <pre>"email": eve.holt@reqres.in ", "password": "pistol"</pre> | Status Code should be 200 | Status Code: 200 | Pass | High |
| | Response body should contain token "QpwL5tke4Pnpja7X4" | Response body contains token "QpwL5tke4Pnpja7X4" | Pass | |
| | Status phrase should be: OK | Status phrase: OK | Pass | |

4.1.2 Test Case: Login of defined user with all valid parameters: Verify case insensitivity of email address.

Description: This test case will check if the api/login endpoint will work for defined user with all valid parameter and case insensitivity of email address.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|--|-----------|----------|
| <pre>"email": "EVE.holt@reqres.in ", "password": "pistol"</pre> | Status Code should be 200 | Status Code: 400 | Fail | High |
| | Response body should contain token "QpwL5tke4Pnpja7X4" | Response body contains "error": "user not found" | Fail | |
| | Status phrase should be: OK | Status phrase: Bad Request | Fail | |

4.1.3 Test Case: Login with Valid Credentials and Trailing Whitespace in Email.

Description: This test case will check if the api/login endpoint will work for defined user with Valid Credentials and Trailing Whitespace in Email.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|--|-----------|----------|
| <pre>"email": " eve.holt@reqres.in ", "password": "pistol"</pre> | Status Code should be 400 | Status Code: 400 | Pass | High |
| | Response body should contain "error": "user not found" | Response body contains "error": "user not found" | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |

4.1.4 Test Case: Login of unregistered user.

Description: This test case will check if the api/login endpoint will work for defined user with Valid Credentials and Trailing Whitespace in Email.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|--|-----------|----------|
| <code>"email": "surraiya123@gmail. com", "password": "abc123"</code> | Status Code should be 400 | Status Code: 400 | Pass | High |
| | Response body should contain "error": "user not found" | Response body contains "error": "user not found" | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |

4.1.5 Test Case: Login of registered user with correct email and incorrect password.

Description: This test case will check if the api/login endpoint will work for defined user with correct email and incorrect password.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|--|-----------|----------|
| <code>"email": "eve.holt@reqres.in", "password": "abc123"</code> | Status Code should be 401 | Status Code: 200 | Fail | High |
| | Response body should contain error: wrong password | Response body contains token "QpwL5tke4Pnpja7X4" | Fail | |
| | Status phrase should be: Unauthorized | Status phrase: OK | Fail | |

4.1.6 Test Case: Login of registered user with correct email and empty password.

Description: This test case will check if the api/login endpoint will work for defined user with correct email and empty password.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|--|-----------|----------|
| <code>"email": "surraiya123@gmail.com", "password": ""</code> | Status Code should be 400 | Status Code: 400 | Pass | High |
| | Response body should contain "error": "Missing password" | Response body contains "error": "Missing password" | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |

4.1.7 Test Case: Login of defined user with all correct email and password: Verify case sensitivity of password.

Description: This test case will check if the api/login endpoint will work for defined user with correct email and Verify case sensitivity of password.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|--|--|-----------|----------|
| <code>"email": "eve.holt@reqres.in", "password": "PIsTol"</code> | Status Code should be 401 | Status Code: 200 | Fail | High |
| | Response body should contain error: wrong password | Response body contains token "QpwL5tke4Pnpja7X4" | Fail | |
| | Status phrase should be: Unauthorized | Status phrase: OK | Fail | |

4.1.8 Test Case: Login with empty email and empty password.

Description: This test case will check if the api/login endpoint will work for defined user with correct email and empty password.

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|--|---|---|-----------|----------|
| <code>"email": "", "password": ""</code> | Status Code should be 400 | Status Code: 400 | Pass | High |
| | Response body should contain "error": "Missing email or username" | Response body contains "error": "Missing email or username" | Pass | |
| | Status phrase should be: Bad Request | Status phrase: Bad Request | Pass | |

4.1.9 Test Case: Login with wrong method (GET) but correct URL.

Description: This test case will check if the api/login endpoint will work for invalid method (GET).

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|--|-------------------|-----------|----------|
| | Status Code should be 405 | Status Code: 200 | Fail | Medium |
| | Status phrase should be: Invalid Method | Status phrase: OK | Fail | |

4.1.9 Test Case: Incorrect HTTP Method (PUT) and URL for Login.

Description: This test case will check if the api/login endpoint will work for invalid method (PUT).

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|-----------|--|-------------------|-----------|----------|
| | Status Code should be 405 | Status Code: 200 | Fail | Medium |
| | Status phrase should be: Invalid Method | Status phrase: OK | Fail | |

4.1.10 Test Case: Login with unknown request body.

Description: This test case will check if the api/login endpoint will work for invalid method (PUT).

Assertions and Severity:

| Test Data | Expected | Actual | Pass/Fail | Severity |
|---|--|--|-----------|----------|
| <code>"job": "leader",</code> <code>"age": 20</code> | Status Code should be 400 | Status Code: 400 | Pass | Medium |
| | Response body should show invalid request body. | Response body show "error": "Missing email or username" | Fail | |

Visualization:

Params Auth Headers (8) Body ● Pre-req. Tests ● Settings

raw JSON

```
1 {  
2   |   "job": "leader",  
3   |   "age": 20  
4 }
```

Body 400 Bad Request 125 ms 887 B Save as

Pretty Raw Preview Visualize JSON

```
1 {  
2   |   "error": "Missing email or username"  
3 }
```