Graduation Project Presentation

Team Names:

- Yossef Magdy Shebl (API testing)
- Ahmed Adel (Manual testing)
- George Magdy (Manual testing)
- Yossef Samy (Manual testing)
- Osama (Manual testing)

Track: software development – software testing

Supervisor name: George Nafady

Date:17/10/2024

Introduction

- Overview of the project
- Problem statement
- Objectives
- Motivation for the project

API Testing

- Overview:

This project aims to implement the technical topics learned in this course; it specifically targets the topic of API testing to a hosted website called Natours implemented as part of a popular large back-end course made by Jonas Schmedtmann.

- Problem statement:

As this is a highly popular course accessed by thousands (maybe more) and it is about building API for a website, it is very beneficial to provide a decent test suite to those APIs to validate the importance of this course, also it will be used by every student who implements this project to validate their work.

- Objectives:

The objectives are to ensure proper functionality of the APIs, check integrity of both request and response data, ensure high performance, also to act as a documentation of how to use those APIs and what to expect of them.

- Motivation for the project:

As pointed earlier, many students interested in back-end development using node.js this is their go to course, so it would be of great help to them to validate their work and improve course validity, also help the creator of the course to improve his content and fix any issues.

Manual Testing

- Overview:

This project aims to apply the manual testing techniques covered in this course. Specifically, it focuses on creating detailed test scenarios and test cases for a hosted website called Natours

- Problem statement:

Given the widespread popularity of this course, which focuses on building APIs for a website, there is a clear need for a comprehensive manual test suite. With thousands of students participating, a robust test suite would not only validate the importance of the course content but also provide students with a practical tool to assess their own API implementations. By developing a structured manual test suite, students will be able to confidently verify the functionality, reliability, and quality of their APIs, ensuring they meet industry standards and reinforcing their understanding of manual testing practices in backend development.

- Objectives:

The objective is to validate the functionality, usability, and performance of various features on the site through rigorous manual testing method

- Motivation for the project:

This project targets the many students who rely on this popular Node.js back-end course, focusing specifically on creating manual test scenarios and test cases for API validation. Providing a detailed manual test suite will enable students to thoroughly validate their implementations, ensuring functionality and helping them build confidence in their testing skills. Moreover, this suite will aid the course creator in refining content and addressing any identified issues, ultimately improving the course’s quality and reliability for future students.

Testing process

- Definition of API endpoints
- Types of testing
- Tools used in testing
- Test cases design

API Testing

- API endpoints:

Natours ideas is to view different tours to a variety of places and with many activities, so the endpoints focuses on providing data about:

- Tours.
- Users.
- Reviews.
- Bookings.
- Authentication (login, etc.…).

Endpoints focus on providing way to do CRUD operations on this data.

Specific Endpoints can not be tested because the owner of the website blocked this functionality on demo users also you need to get an admin user which is blocked, some of these endpoints like create tour, delete tour, delete user and many more.

APIs provided are as follows: (official documentation)

1. Tours:

GET¤ All Tours (With Query Parameters), Tour By Id, Top 5 Cheap Tours, Monthly Plan, Tour Stats, Tours Within Radius, Distances To Tours From A Certain Location.

POST¤ Create New Tour.

DELETE¤ Delete Tour.

PATCH¤ Update Tour.

2. Users:

GET¤ All Users, Single User, Current User.

POST¤ Included In Auth APIs.

DELETE¤ User, Current User.

PATCH¤ Update User, Update Current User.

3. Authentication:

POST Sign Up, Login, Forgot Password.

PATCH Reset Password, Update Current User Password.

4. Reviews:

GET All Reviews, Single Review.

POST Create New Review.

DELETE Delete Review.

PATCH Update Review.

5. Write a review on a specific tour

GET All Reviews on a Tour.

POST Create New Review on a Tour.

6. Bookings:

GET All Bookings, Single Booking.

- Types of testing:

Testing will be split mainly into 2 types:

1. Functional testing: testing the functionality if the API includes:
   - Verify response and request body structure and schema.
   - Check URL structure is as expected.
   - Check request method.
2. Security testing:
   - Test cookies.
   - Test heading “authorization”.
   - Test “JWT” token.

- Tools used in testing:

Main tool used in API testing is Postman.

Postman is an API platform which helps with building APIs and testing them.

It provides an environment for running JavaScript code to write and execute test cases.

- Test cases design:

Test cases will focus on functional testing and security testing, testing the positive test cases only (valid input, valid output).

Some endpoints were tested against negative cases.

Each test case includes:

- URL call (API call).
- Expected result.
- Actual result.
- Testing function (pm.test).

Features that will be the focus of our test cases design are outlined in the first point.

Manual Testing

- Types of testing:

Testing will be split mainly into 1 types:

1. Functional testing: testing the functionality if the API includes:
   - Verify response and request body structure and schema.
   - Check URL structure is as expected.
   - Check request method.

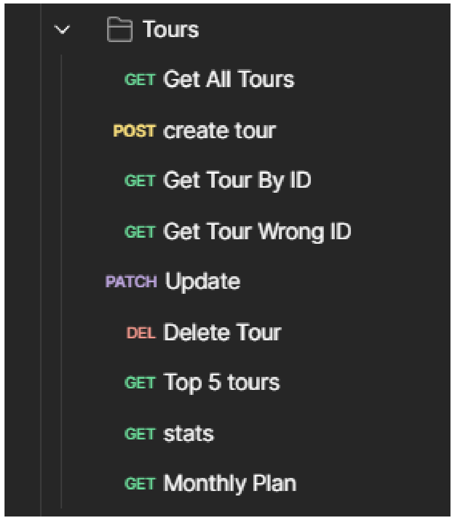
Test execution

- Demonstration of the system
- Code snippets (with output)
- Bugs identified

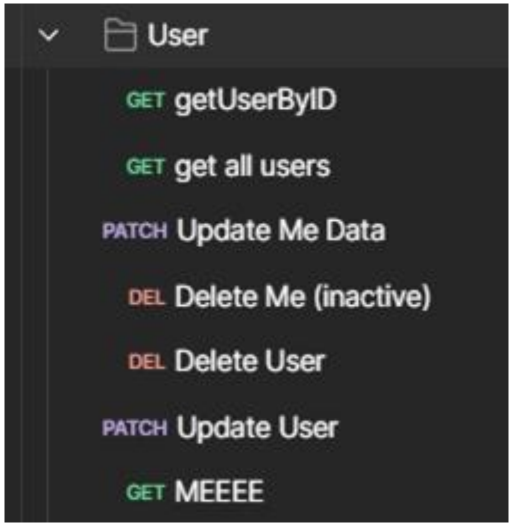
API Testing

- Demonstration of the system:

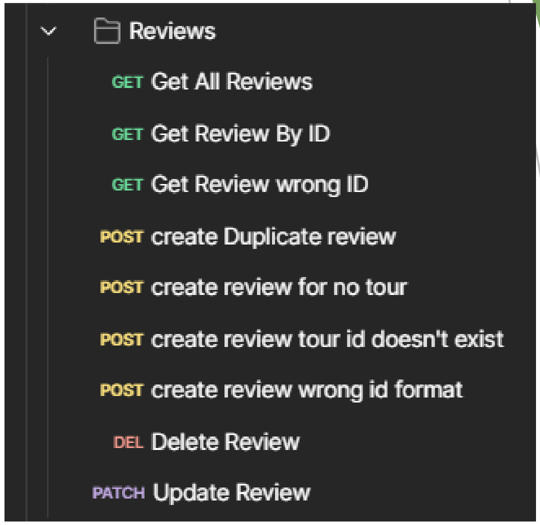
The system consists of a set of APIs to manipulate various data within the system, and they are listed as follows:



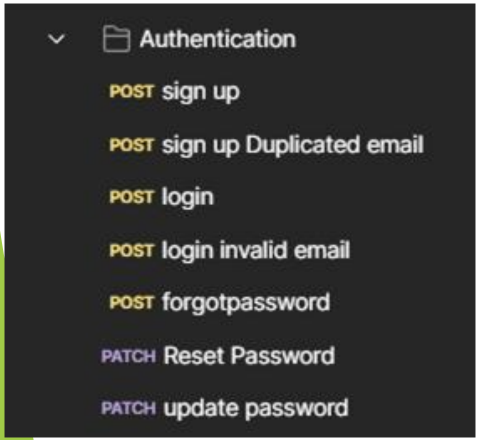
Tours
- GET Get All Tours
- POST create tour
- GET Get Tour By ID
- GET Get Tour Wrong ID
- PATCH Update
- DEL Delete Tour
- GET Top 5 tours
- GET stats
- GET Monthly Plan



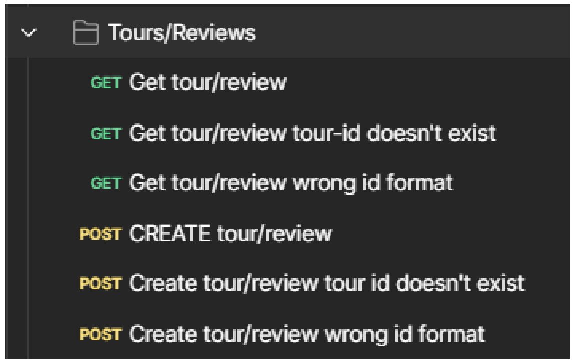
User
- GET getUserByID
- GET get all users
- PATCH Update Me Data
- DEL Delete Me (inactive)
- DEL Delete User
- PATCH Update User
- GET MEEEE



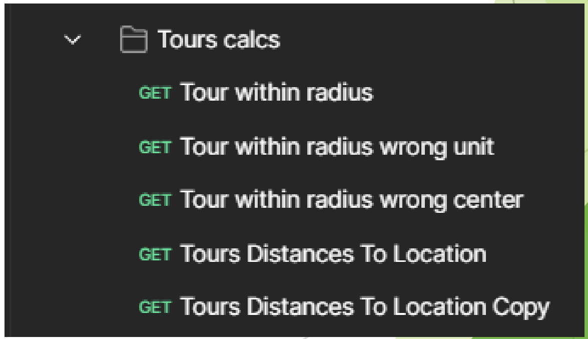
Reviews
- GET Get All Reviews
- GET Get Review By ID
- GET Get Review wrong ID
- POST create Duplicate review
- POST create review for no tour
- POST create review tour id doesn't exist
- POST create review wrong id format
- DEL Delete Review
- PATCH Update Review



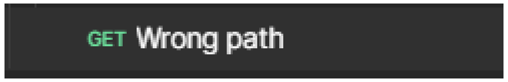
Authentication
- POST sign up
- POST sign up Duplicated email
- POST login
- POST login invalid email
- POST forgotpassword
- PATCH Reset Password
- PATCH update password



Tours/Reviews
- GET Get tour/review
- GET Get tour/review tour-id doesn't exist
- GET Get tour/review wrong id format
- POST CREATE tour/review
- POST Create tour/review tour id doesn't exist
- POST Create tour/review wrong id format



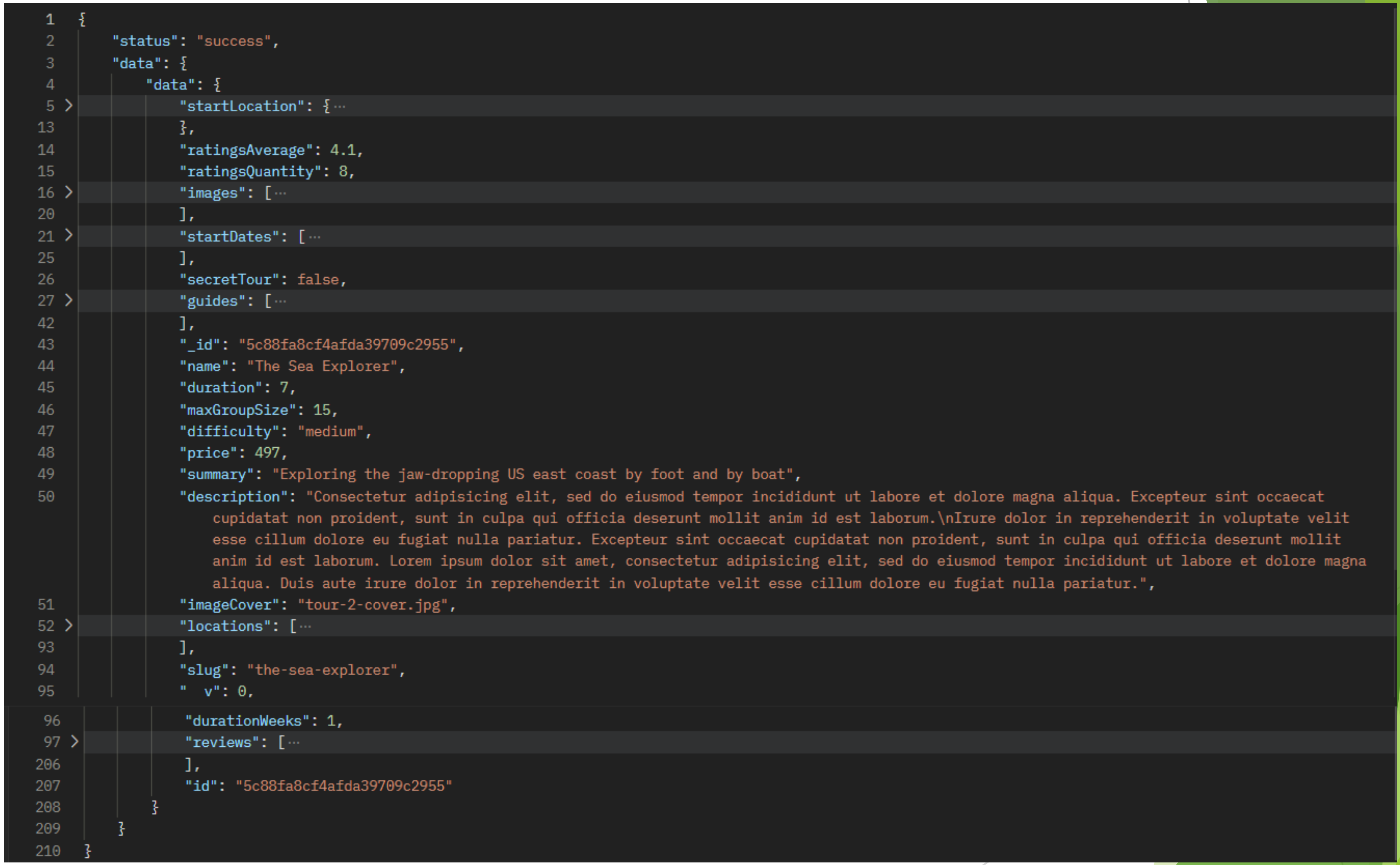
Tours calcs
- GET Tour within radius
- GET Tour within radius wrong unit
- GET Tour within radius wrong center
- GET Tours Distances To Location
- GET Tours Distances To Location Copy



- GET Wrong path

Examples of APIs response:

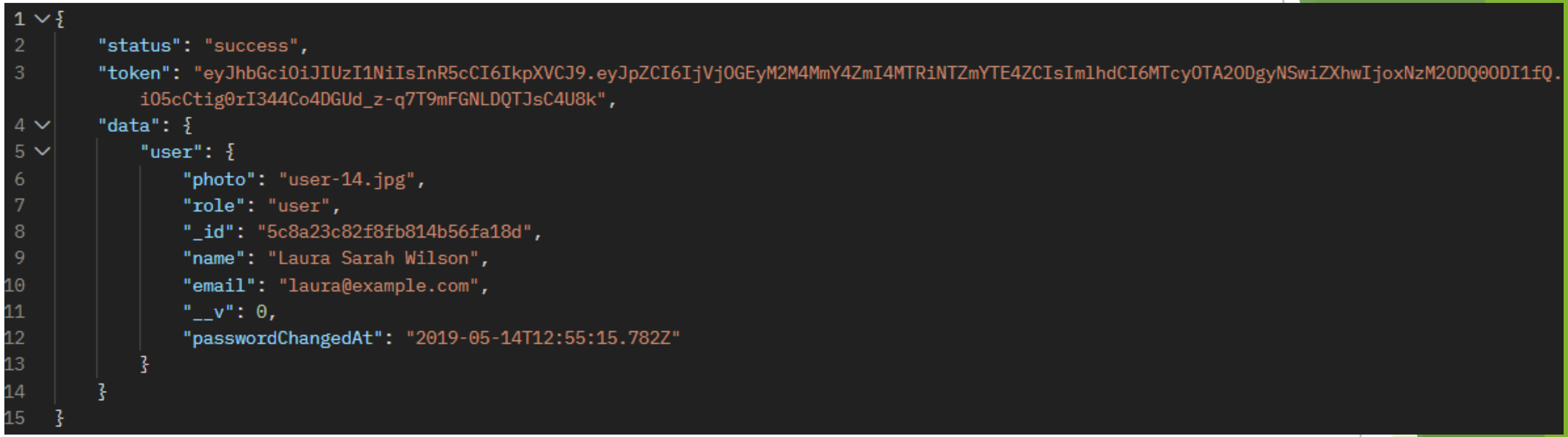
- Get single tour



```
{
    "status": "success",
    "data": {
        "data": {
            "startLocation": {…
            },
            "ratingsAverage": 4.1,
            "ratingsQuantity": 8,
            "images": […
            ],
            "startDates": […
            ],
            "secretTour": false,
            "guides": […
            ],
            "_id": "5c88fa8cf4afda39709c2955",
            "name": "The Sea Explorer",
            "duration": 7,
            "maxGroupSize": 15,
            "difficulty": "medium",
            "price": 497,
            "summary": "Exploring the jaw-dropping US east coast by foot and by boat",
            "description": "Consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Excepteur sint occaecat
                cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.\nIrure dolor in reprehenderit in voluptate velit
                esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit
                anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna
                aliqua. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.",
            "imageCover": "tour-2-cover.jpg",
            "locations": […
            ],
            "slug": "the-sea-explorer",
            "__v": 0,
            "durationWeeks": 1,
            "reviews": […
            ],
            "id": "5c88fa8cf4afda39709c2955"
        }
    }
}
```

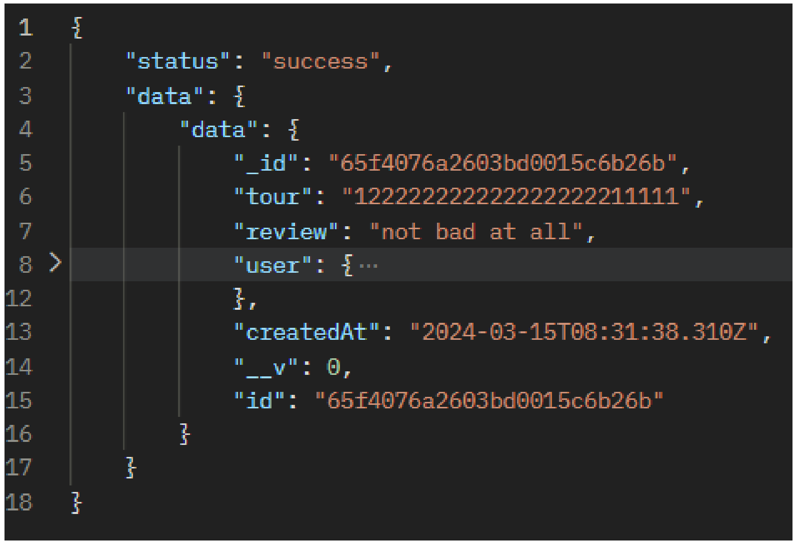
Examples of APIs response:

- Login



```
{
    "status": "success",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVjOGEyM2M4MmY4ZmI4MTRiNTZmYTE4ZCIsImlhdCI6MTcyOTA2ODgyNSwiZXhwIjoxNzM2ODQ0ODI1fQ.
        i05cCtig0rI344Co4DGUd_z-q7T9mFGNLDQTJsC4U8k",
    "data": {
        "user": {
            "photo": "user-14.jpg",
            "role": "user",
            "_id": "5c8a23c82f8fb814b56fa18d",
            "name": "Laura Sarah Wilson",
            "email": "laura@example.com",
            "__v": 0,
            "passwordChangedAt": "2019-05-14T12:55:15.782Z"
        }
    }
}
```

- Get Review By ID



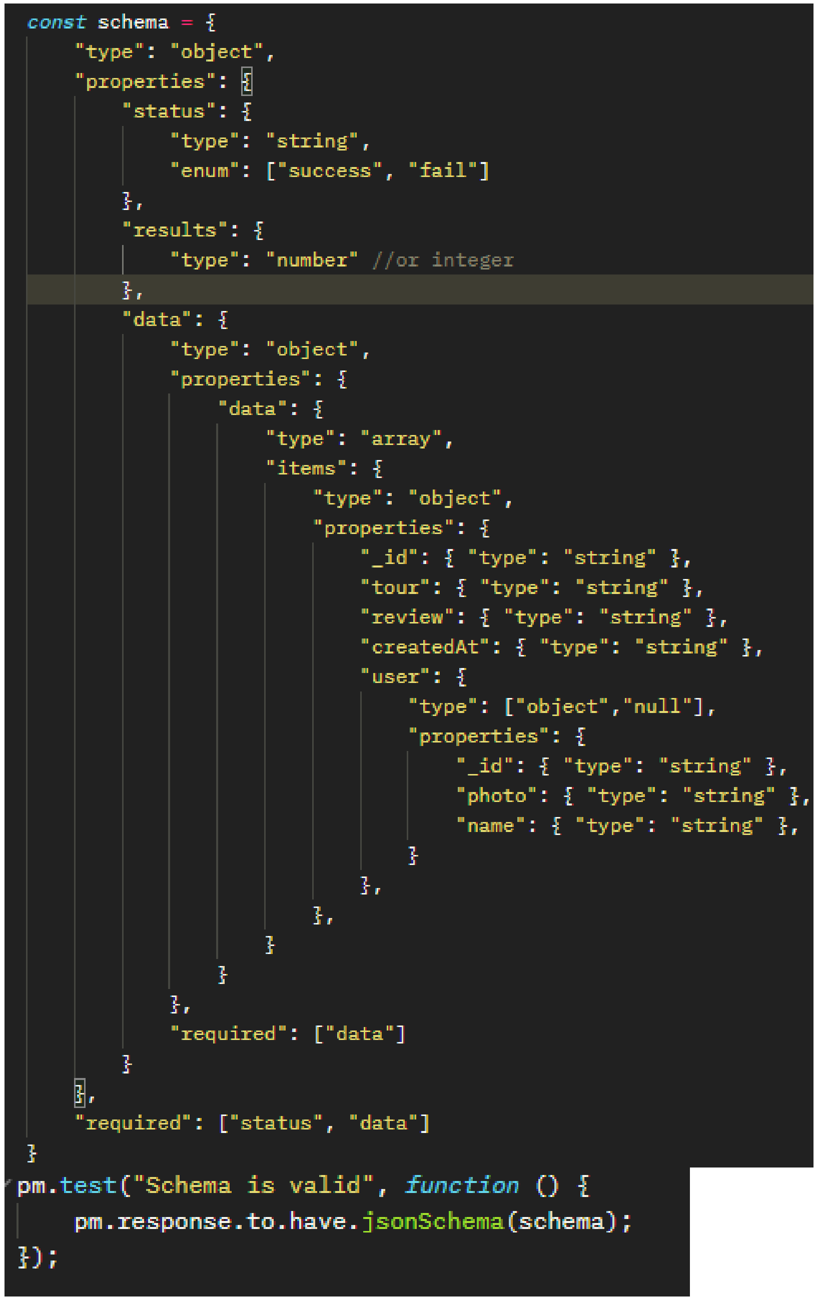
```
{
    "status": "success",
    "data": {
        "data": {
            "_id": "65f4076a2603bd0015c6b26b",
            "tour": "122222222222222222211111",
            "review": "not bad at all",
            "user": {…
            },
            "createdAt": "2024-03-15T08:31:38.310Z",
            "__v": 0,
            "id": "65f4076a2603bd0015c6b26b"
        }
    }
}
```

- Code snippets (with output):

Code for API testing is repetitive, so the same code snippet that tests the Json response schema will be the same for all APIs except for the schema itself, also testing the request method, Authorization header, URL structure.

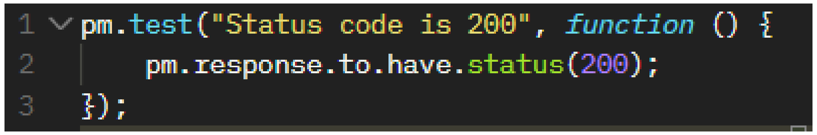
They will only differ in slight change depending on the need of each request.

Testing schema



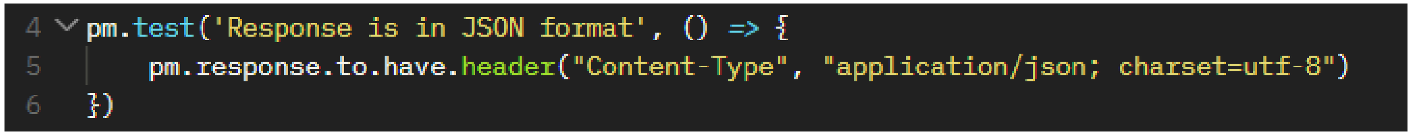
```
const schema = {
    "type": "object",
    "properties": {
        "status": {
            "type": "string",
            "enum": ["success", "fail"]
        },
        "results": {
            "type": "number" //or integer
        },
        "data": {
            "type": "object",
            "properties": {
                "data": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "_id": { "type": "string" },
                            "tour": { "type": "string" },
                            "review": { "type": "string" },
                            "createdAt": { "type": "string" },
                            "user": {
                                "type": ["object","null"],
                                "properties": {
                                    "_id": { "type": "string" },
                                    "photo": { "type": "string" },
                                    "name": { "type": "string" },
                                }
                            },
                        },
                    }
                }
            },
            "required": ["data"]
        }
    },
    "required": ["status", "data"]
}
pm.test("Schema is valid", function () {
    pm.response.to.have.jsonSchema(schema);
});
```

- Testing Response Status



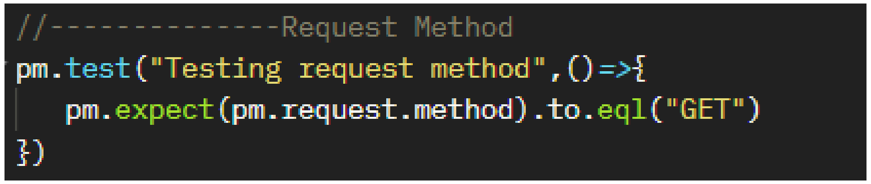
```javascript
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

- Testing Response Format Is Json



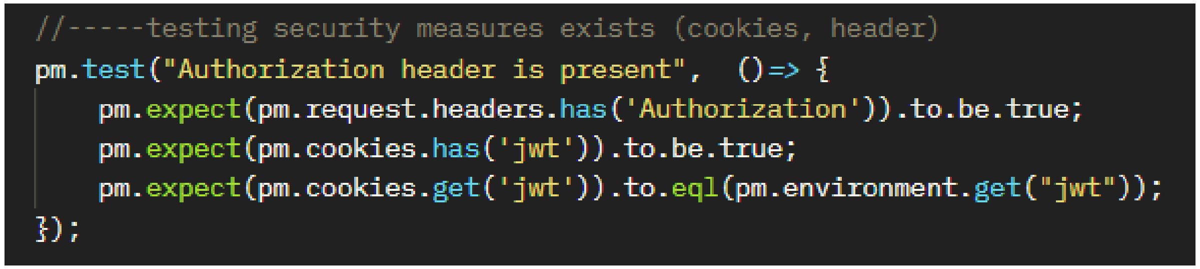
```javascript
pm.test('Response is in JSON format', () => {
    pm.response.to.have.header("Content-Type", "application/json; charset=utf-8")
})
```

- Testing Request Method



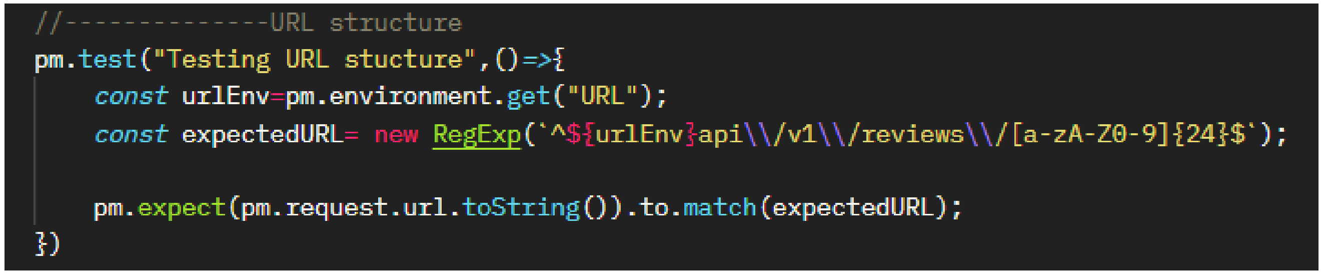
```javascript
//-------------Request Method
pm.test("Testing request method",()=>{
   pm.expect(pm.request.method).to.eql("GET")
})
```

- Testing Authorization header and JWT token



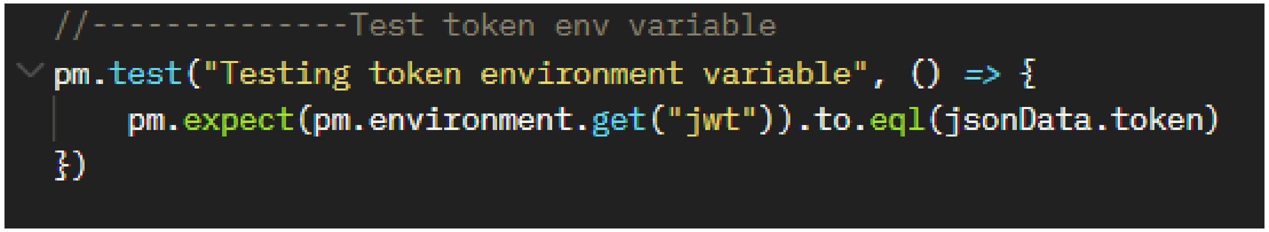
```javascript
//-----testing security measures exists (cookies, header)
pm.test("Authorization header is present",  ()=> {
    pm.expect(pm.request.headers.has('Authorization')).to.be.true;
    pm.expect(pm.cookies.has('jwt')).to.be.true;
    pm.expect(pm.cookies.get('jwt')).to.eql(pm.environment.get("jwt"));
});
```

- Testing URL structure



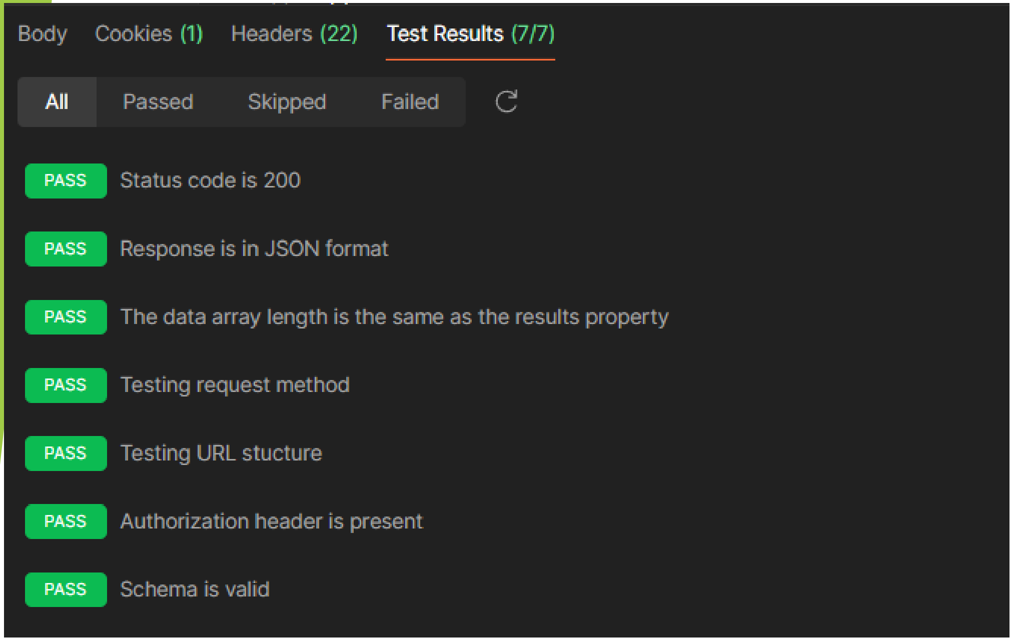
```
//--------------URL structure
pm.test("Testing URL stucture",()=>{
    const urlEnv=pm.environment.get("URL");
    const expectedURL= new RegExp(`^${urlEnv}api\\/v1\\/reviews\\/[a-zA-Z0-9]{24}$`);

    pm.expect(pm.request.url.toString()).to.match(expectedURL);
})
```

- Testing Environment variable exists



```
//--------------Test token env variable
pm.test("Testing token environment variable", () => {
    pm.expect(pm.environment.get("jwt")).to.eql(jsonData.token)
})
```

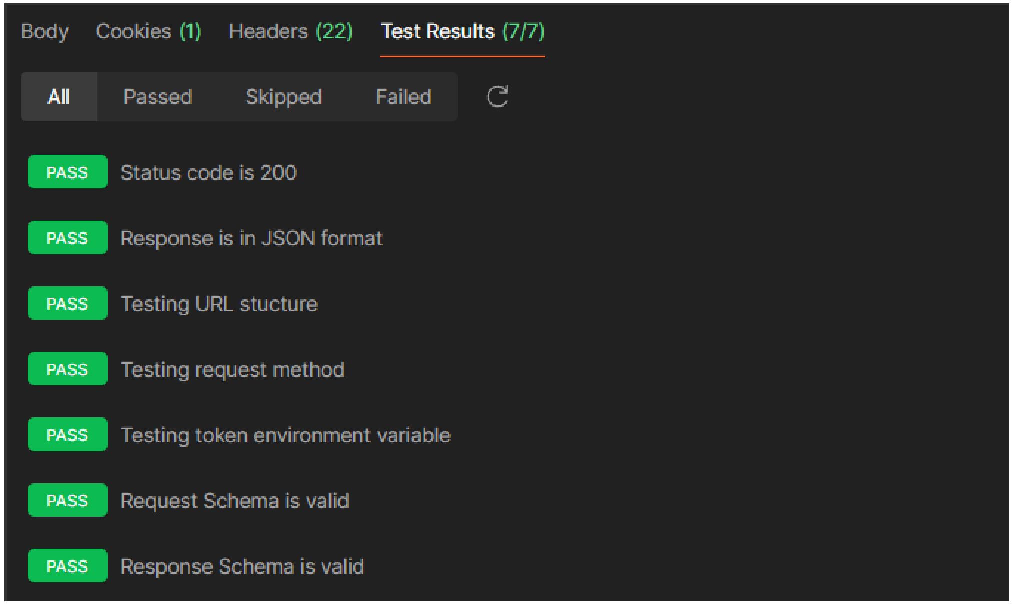
- Output example for 2 endpoints test


Body Cookies (1) Headers (22) Test Results (7/7)

All Passed Skipped Failed

```
PASS Status code is 200
PASS Response is in JSON format
PASS The data array length is the same as the results property
PASS Testing request method
PASS Testing URL stucture
PASS Authorization header is present
PASS Schema is valid
```

Get All Reviews

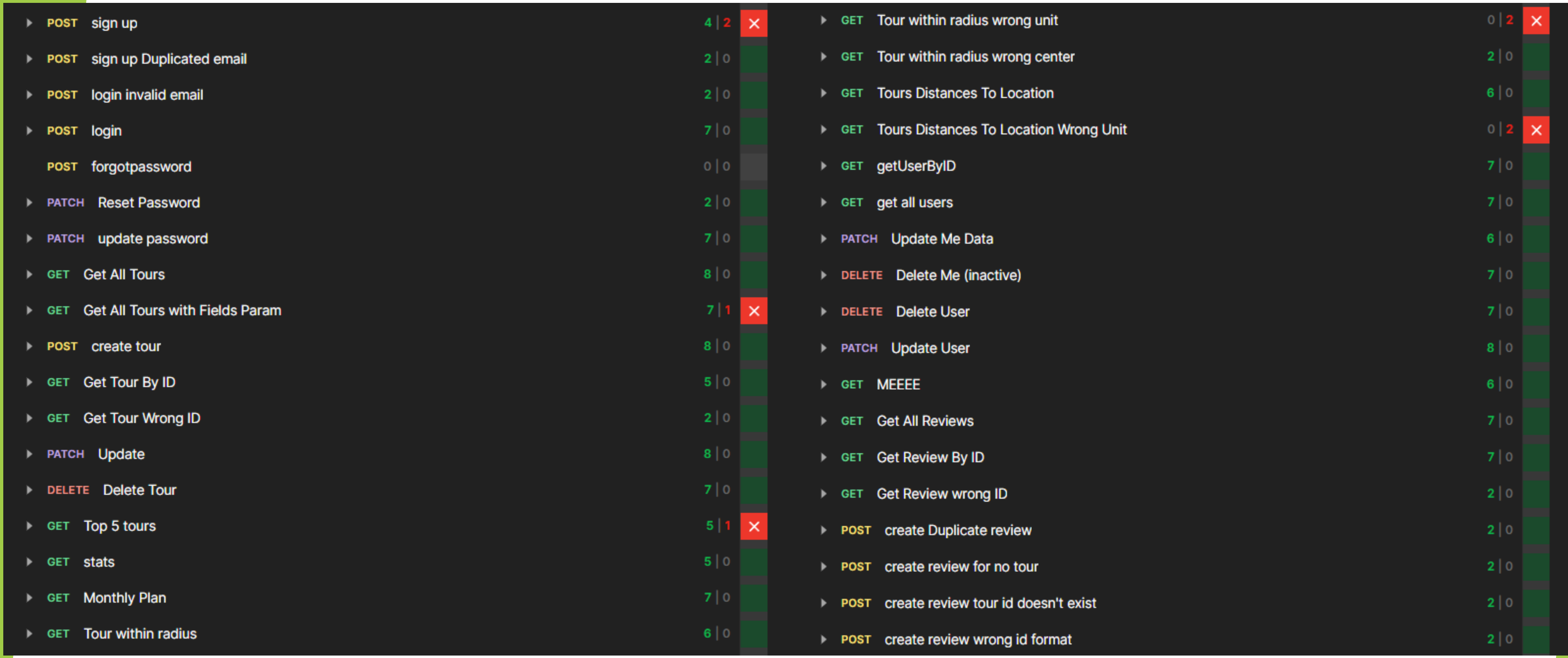

Body Cookies (1) Headers (22) Test Results (7/7)

All Passed Skipped Failed

```
PASS Status code is 200
PASS Response is in JSON format
PASS Testing URL stucture
PASS Testing request method
PASS Testing token environment variable
PASS Request Schema is valid
PASS Response Schema is valid
```

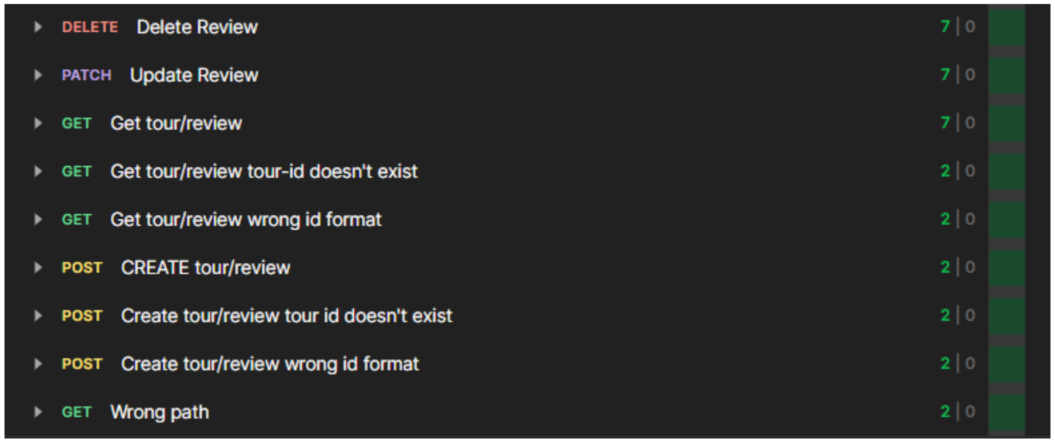
Login

- Test run for all endpoints (bugs will be identified in next section)



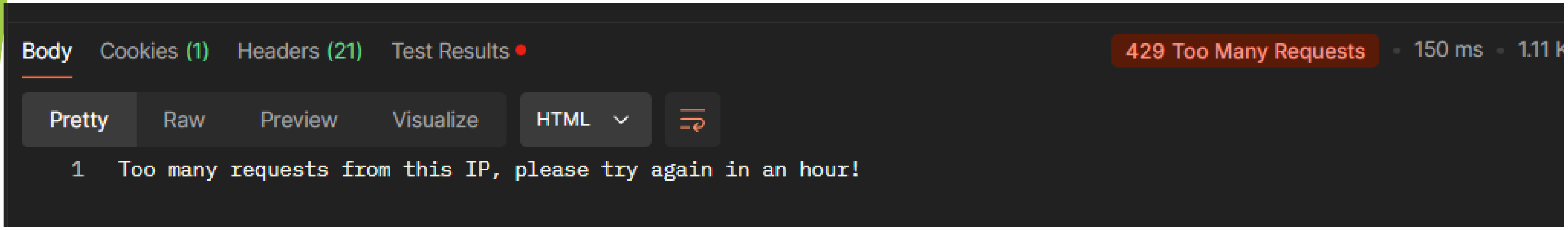
```
POST   sign up                                   4 | 2  ✕
POST   sign up Duplicated email                  2 | 0
POST   login invalid email                       2 | 0
POST   login                                     7 | 0
POST   forgotpassword                            0 | 0
PATCH  Reset Password                            2 | 0
PATCH  update password                           7 | 0
GET    Get All Tours                             8 | 0
GET    Get All Tours with Fields Param           7 | 1  ✕
POST   create tour                               8 | 0
GET    Get Tour By ID                            5 | 0
GET    Get Tour Wrong ID                         2 | 0
PATCH  Update                                    8 | 0
DELETE Delete Tour                               7 | 0
GET    Top 5 tours                               5 | 1  ✕
GET    stats                                     5 | 0
GET    Monthly Plan                              7 | 0
GET    Tour within radius                        6 | 0
```

```
GET    Tour within radius wrong unit             0 | 2  ✕
GET    Tour within radius wrong center           2 | 0
GET    Tours Distances To Location               6 | 0
GET    Tours Distances To Location Wrong Unit    0 | 2  ✕
GET    getUserByID                               7 | 0
GET    get all users                             7 | 0
PATCH  Update Me Data                            6 | 0
DELETE Delete Me (inactive)                      7 | 0
DELETE Delete User                               7 | 0
PATCH  Update User                               8 | 0
GET    MEEEE                                     6 | 0
GET    Get All Reviews                           7 | 0
GET    Get Review By ID                          7 | 0
GET    Get Review wrong ID                       2 | 0
POST   create Duplicate review                   2 | 0
POST   create review for no tour                 2 | 0
POST   create review tour id doesn't exist       2 | 0
POST   create review wrong id format             2 | 0
```



```
DELETE Delete Review                             7 | 0
PATCH  Update Review                             7 | 0
GET    Get tour/review                           7 | 0
GET    Get tour/review tour-id doesn't exist     2 | 0
GET    Get tour/review wrong id format           2 | 0
POST   CREATE tour/review                        2 | 0
POST   Create tour/review tour id doesn't exist  2 | 0
POST   Create tour/review wrong id format        2 | 0
GET    Wrong path                                2 | 0
```

- A stress test was done, by sending too many requests, and the test, returning status (429 Too Many Requests), passed by blocking the IP for an hour.



```
Too many requests from this IP, please try again in an hour!
```

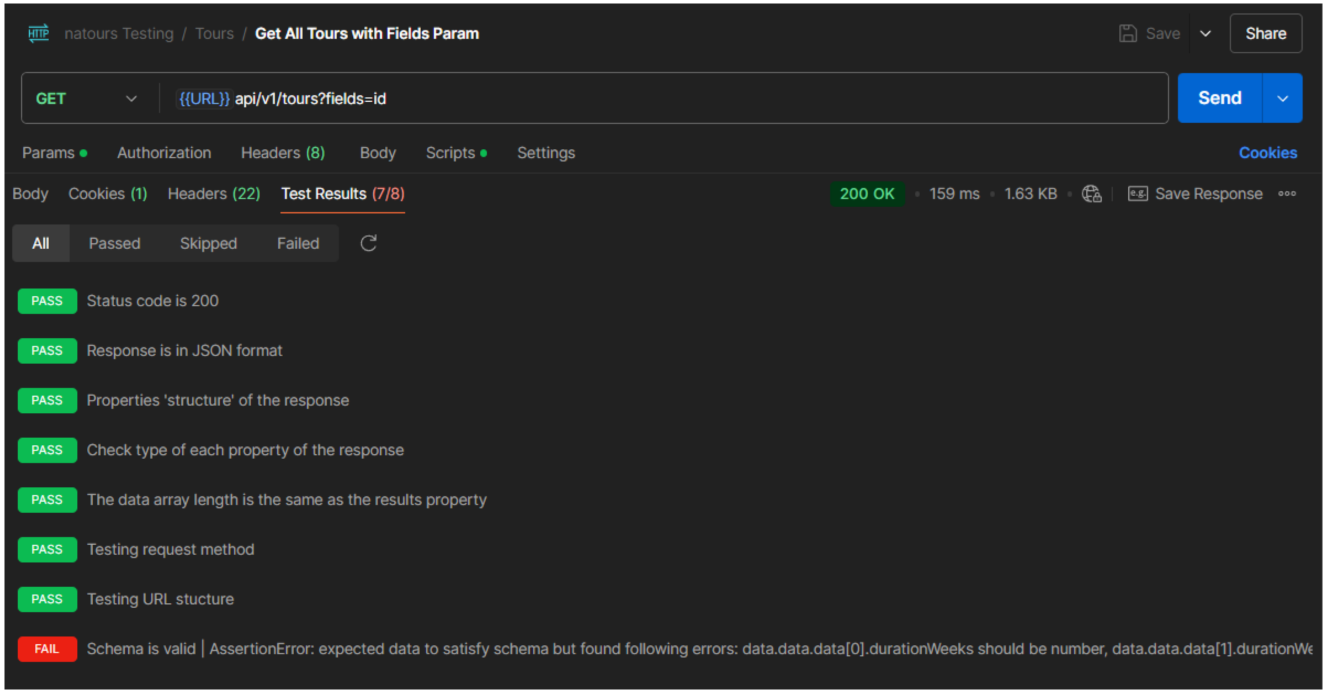
- Bugs identified:

The bugs identified are mostly due to expectations on the schema response, like not filling a virtual variable so they are minor bugs.

Some bugs were due to wrong response to using the URL in wrong format (expected “bad request” but got “ok”).

- Examples of bugs:
  1. Get all tours with fields param:

  This API request should result in response body with some fields, one of them is “durationWeeks” and it should be a number, but it returned as null.

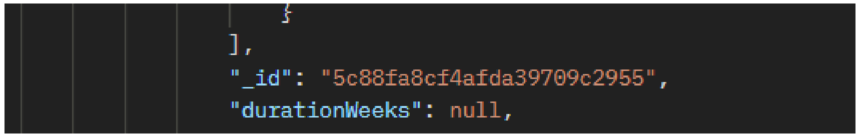


natours Testing / Tours / **Get All Tours with Fields Param**

GET {{URL}} api/v1/tours?fields=id

Body Cookies (1) Headers (22) Test Results (7/8) — 200 OK · 159 ms · 1.63 KB

```
PASS Status code is 200
PASS Response is in JSON format
PASS Properties 'structure' of the response
PASS Check type of each property of the response
PASS The data array length is the same as the results property
PASS Testing request method
PASS Testing URL stucture
FAIL Schema is valid | AssertionError: expected data to satisfy schema but found following errors: data.data.data[0].durationWeeks should be number, data.data.data[1].durationWe
```

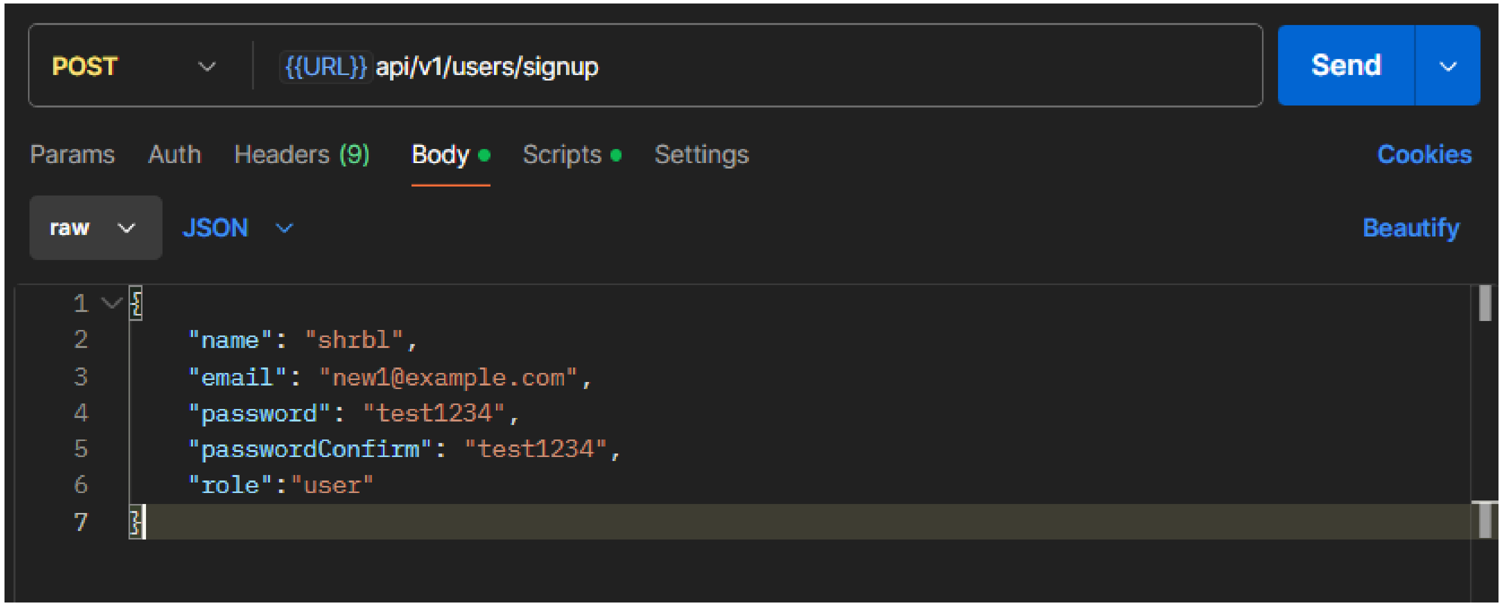


```
        }
    ],
    "_id": "5c88fa8cf4afda39709c2955",
    "durationWeeks": null,
```

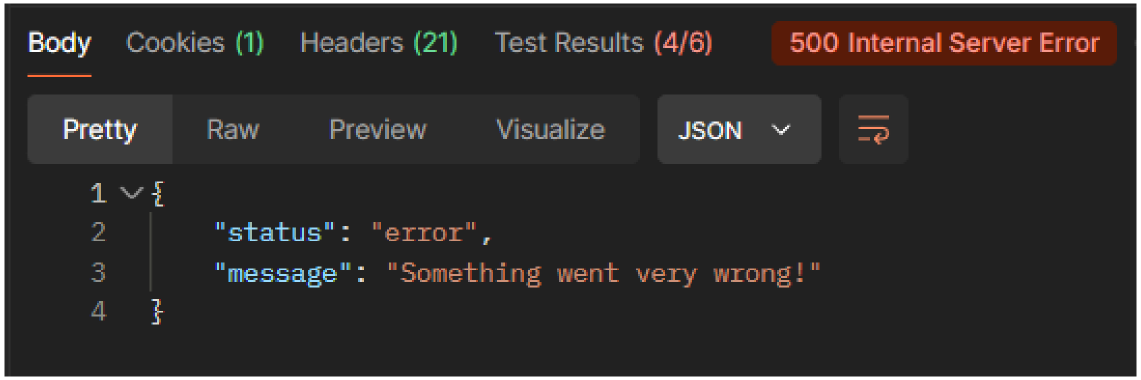
2. Sign up:

This API request should result in user logged in with a token for the session, and the response be user details and the token, instead it returned 500 error code.

Yet user can log in later with these credentials.



```
{
    "name": "shrbl",
    "email": "new1@example.com",
    "password": "test1234",
    "passwordConfirm": "test1234",
    "role":"user"
}
```

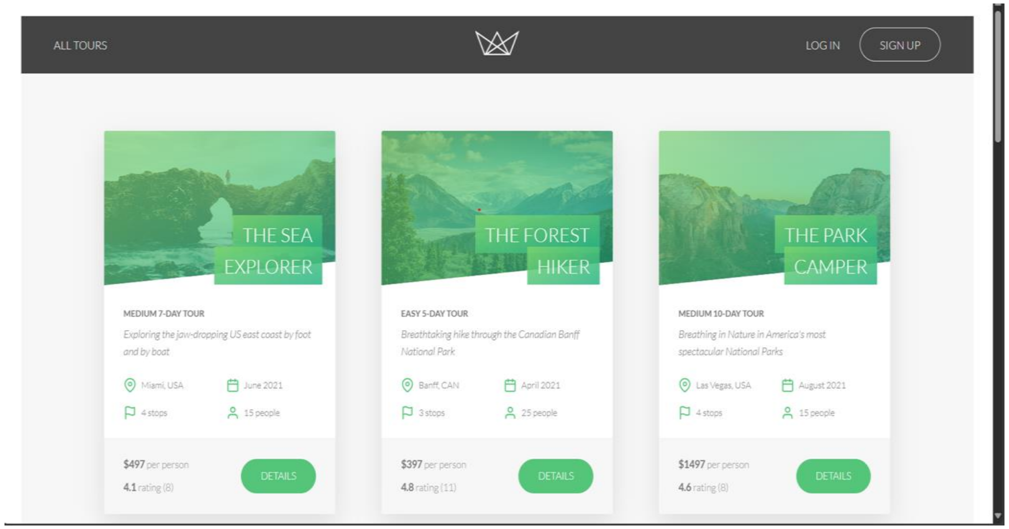


```
{
    "status": "error",
    "message": "Something went very wrong!"
}
```

Manual Testing

- Demonstration of the system:

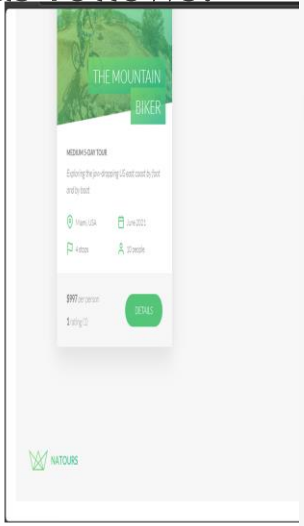
The system consists of , and they are listed as follows:

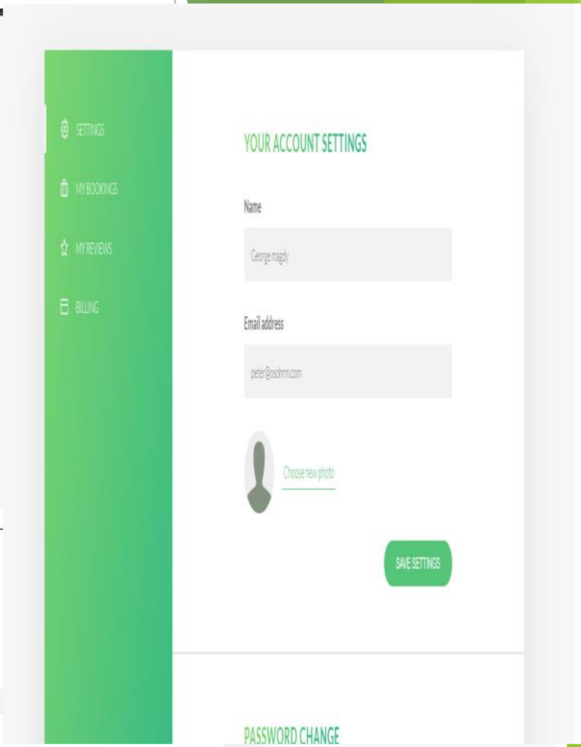

ALL TOURS LOG IN SIGN UP

THE SEA EXPLORER
MEDIUM 7-DAY TOUR
Exploring the jaw-dropping US east coast by foot and by boat
Miami, USA · June 2021 · 4 stops · 15 people
$497 per person · 4.1 rating (8) · DETAILS

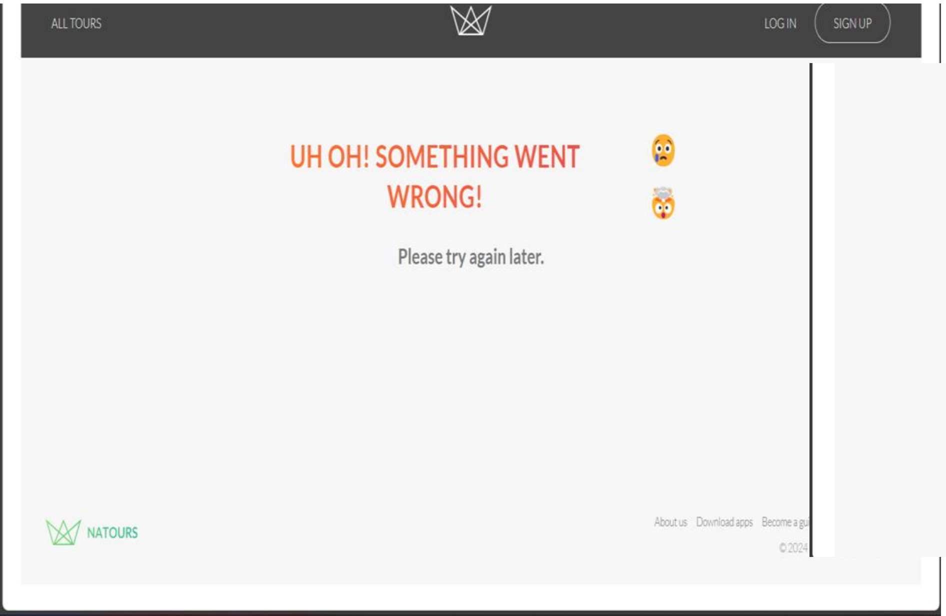
THE FOREST HIKER
EASY 5-DAY TOUR
Breathtaking hike through the Canadian Banff National Park
Banff, CAN · April 2021 · 3 stops · 25 people
$397 per person · 4.8 rating (11) · DETAILS

THE PARK CAMPER
MEDIUM 10-DAY TOUR
Breathing in Nature in America's most spectacular National Parks
Las Vegas, USA · August 2021 · 4 stops · 15 people
$1497 per person · 4.6 rating (8) · DETAILS


THE MOUNTAIN BIKER
MEDIUM 5-DAY TOUR
Exploring the jaw-dropping US east coast by foot and by boat
Miami, USA · June 2021 · 4 stops · 10 people
$997 per person · 1 rating (1) · DETAILS
NATOURS


SETTINGS
MY BOOKINGS
MY REVIEWS
BILLING

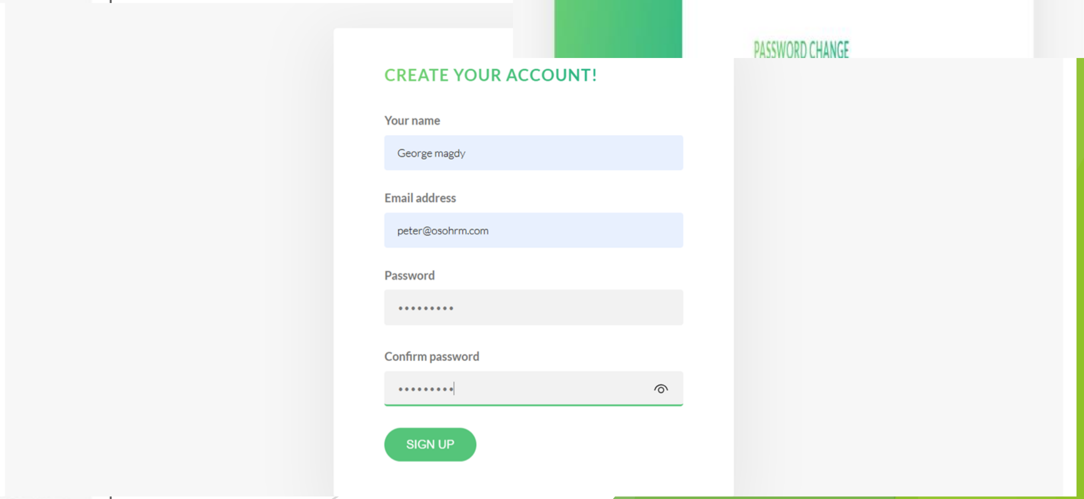
YOUR ACCOUNT SETTINGS
Name: George magdy
Email address: peter@osohrm.com
Choose new photo
SAVE SETTINGS
PASSWORD CHANGE


ALL TOURS LOG IN SIGN UP

UH OH! SOMETHING WENT WRONG! 😨 🤯

Please try again later.

NATOURS
About us Download apps Become a gu
© 2024

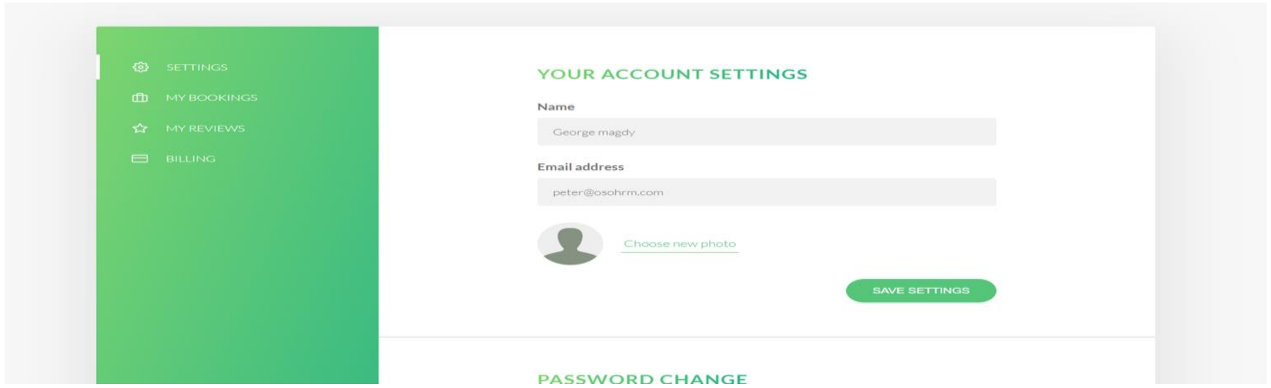

CREATE YOUR ACCOUNT!

Your name: George magdy
Email address: peter@osohrm.com
Password: •••••••••
Confirm password: •••••••••

SIGN UP

Examples of APIs response:

- Login



YOUR ACCOUNT SETTINGS

Name

George magdy

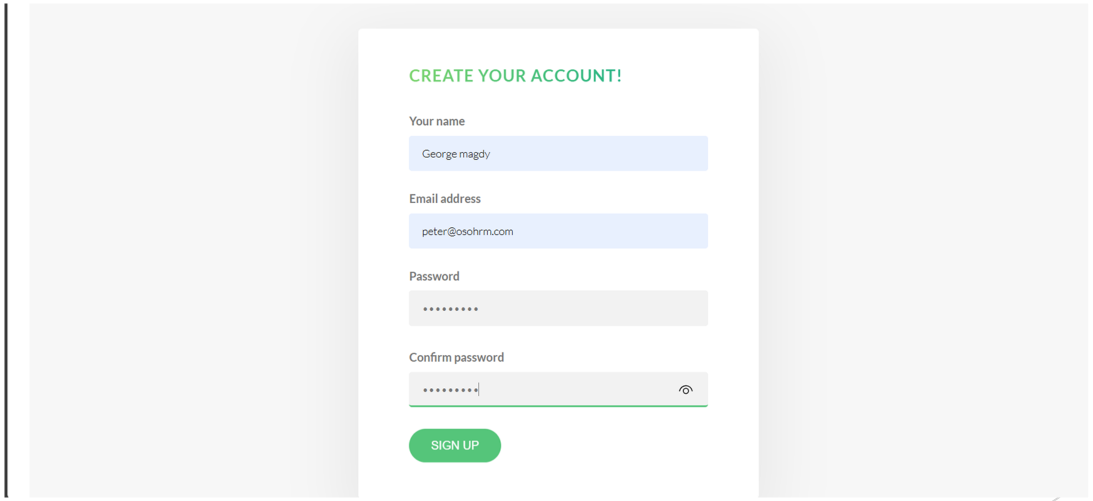
Email address

peter@osohrm.com

Choose new photo

PASSWORD CHANGE

- Singup



CREATE YOUR ACCOUNT!

Your name

George magdy

Email address

peter@osohrm.com

Password

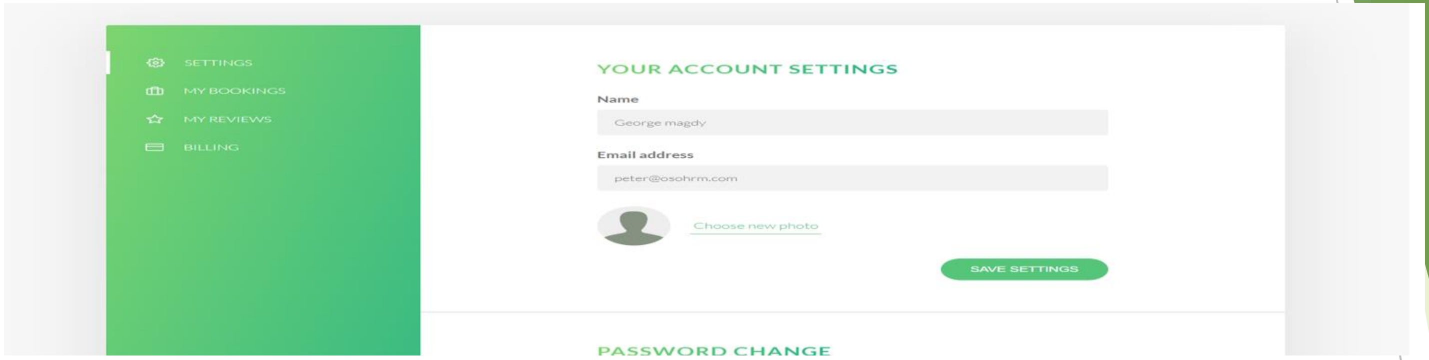
•••••••••

Confirm password

•••••••••

Examples of APIs response:

- Profile



SETTINGS
MY BOOKINGS
MY REVIEWS
BILLING

YOUR ACCOUNT SETTINGS

Name
George magdy

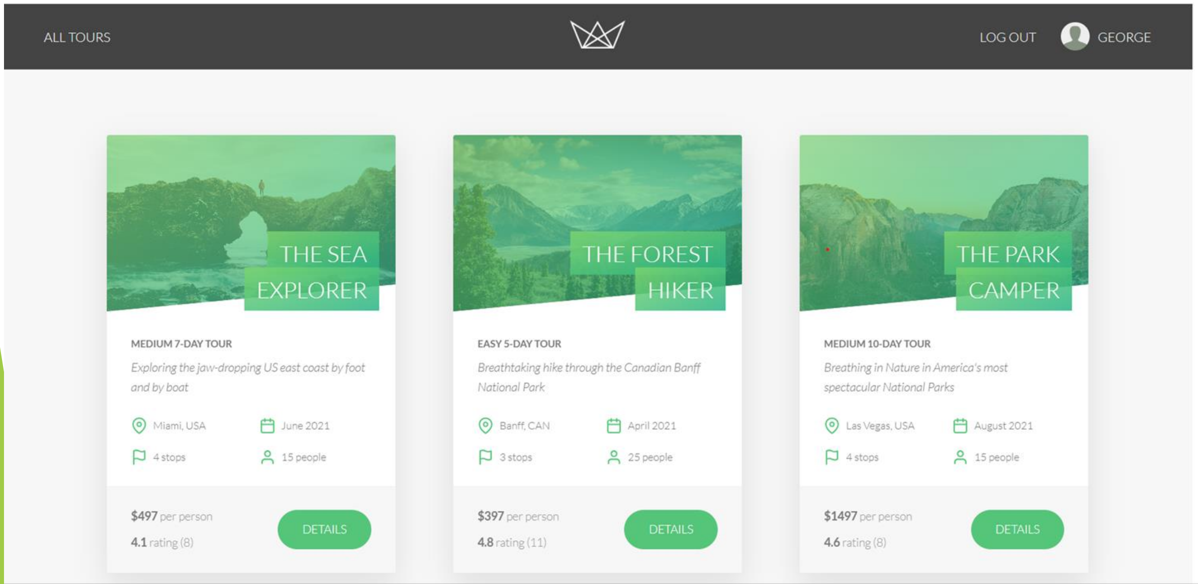
Email address
peter@osohrm.com

Choose new photo

SAVE SETTINGS

PASSWORD CHANGE

- page after login



ALL TOURS   LOG OUT   GEORGE

THE SEA EXPLORER
MEDIUM 7-DAY TOUR
Exploring the jaw-dropping US east coast by foot and by boat
Miami, USA | June 2021
4 stops | 15 people
$497 per person
4.1 rating (8)
DETAILS

THE FOREST HIKER
EASY 5-DAY TOUR
Breathtaking hike through the Canadian Banff National Park
Banff, CAN | April 2021
3 stops | 25 people
$397 per person
4.8 rating (11)
DETAILS

THE PARK CAMPER
MEDIUM 10-DAY TOUR
Breathing in Nature in America's most spectacular National Parks
Las Vegas, USA | August 2021
4 stops | 15 people
$1497 per person
4.6 rating (8)
DETAILS

- Bugs identified:

The bugs identified are mostly due to expectations on the schema response, like not filling a virtual variable so they are minor bugs.

Some bugs were due to wrong response to using the URL in wrong format (expected “bad request” but got “ok”).

Challenges and Solutions

- Challenges encountered
- Solutions implemented

API Testing

- Challenges encountered:

Most of the challenges faced were easy and solvable like:

1. understanding the APIs responses and setting up the collection variables.
2. Also determining the order of running APIs so all the requests would rum smoothly with no unexpected results.
3. The most difficult one was some APIs required an admin email to log in which wasn’t provided, also there was provided only a demo email for testing which some APIs were blocked for.

- Solutions implemented:

The first 2 challenges were solvable:

1. Looking for API documentation and comparing the output to his course content and also to convention. (like when to use 400 or 200 as the response status).
2. Understanding the order was easy to solve.

The third challenge remains unsolved as it needs to contact the owner of the website to allow full access to the APIs.

Manual Testing

Future Work/Improvements

- Future enhancements
- Areas for expansion

API Testing

- Future enhancement:
  1. Adding a more comprehensive test suite testing real depth in the API response body.
  2. The Json response body form a very deep tree containing image extension and real data existing in the system, such as reviews, users, tour and more.
  3. Collecting repeating test cases as checking the request method, auth header and URL structure to the pre-collection script.

- Areas for expansion:
  1. Contacting the website owner to give access to run a more comprehensive test suite that covers 100% of the endpoints.

Manual Testing

Conclusion

- Summary of achievements
- Reflection on impact
- Final remarks

- Summary of achievements:

The project involved developing a simple website dedicated to showcasing various products. Throughout the process, we encountered several challenges that highlighted the importance of addressing certain issues that ultimately affected the website's overall functionality and user experience.

Despite its straightforward nature, we realized that several crucial features that should have been implemented either were not functioning as intended or were overlooked entirely. This oversight hindered the potential effectiveness of the website, limiting user engagement and satisfaction

- Reflection on impact:

  The project involved creating a simple website dedicated to showcasing various products. While the initial goal was to provide users with an accessible and engaging platform, we encountered several challenges that significantly impacted the website's effectiveness.

- Final remarks:

In conclusion, the development of this simple product showcase website has been a valuable journey filled with learning experiences and insights. While our initial goal was to create a straightforward platform for presenting products, we encountered several challenges that ultimately impacted the website's functionality and overall user experience.

Acknowledgments

We would like to thank our **course instructor Eng.george** for their guidance and support throughout this project.

A big thanks to all our **team members** for their collaboration and hard work, as well as to our **peers** for their valuable feedback.

This project was a great learning experience, and we appreciate everyone who contributed to its success.

Questions

Invite questions from the audience