Testing Restful APIs with Postman

Arka Jain University

Course:- Master of Computer Application

Information Technology

PREFACE

In this study I learned a lot about testing and Postman tool that I use in my daily work. With this new information about Postman, I do see new ways to use the tool and I most likely will in the future. Surprising to me was that, what at first seemed like a straight forward thing to do, turned out to be more complex what I imagined.

Big thanks to allow to choose the topic by own and my colleagues for support they provided.

- Akhilesh Kushwaha

Contents

List of Abbreviations

API Application programming interface JWT Json Web Token

Bcrypt Data Hashing

MFA Multi-factor authentication QA Quality assurance

SC System Configurator

Abstract

Author: Akhilesh Kushwaha

Title: Postman (Rest API Testing for MERN Project)

Number of Pages: 34 pages

Date: 4th of March 2024

Degree: MCA

Degree Programs: Information Technology Professional Major: Backend (API Testing)

As software systems become increasingly complex, ensuring their quality and reliability through rigorous testing has become paramount. Application Programming Interfaces (APIs) serve as the backbone of modern software architectures, facilitating communication and data exchange between various components. Effective testing of APIs is therefore crucial to guaranteeing the functionality, performance, and security of software applications. In recent years, Postman has emerged as a leading tool for API testing, offering a plethora of features to streamline the testing process. This review paper explores the capabilities of Postman in API testing, highlighting its benefits, best practices, and potential challenges. Through a comprehensive analysis of Postman's key features, integration capabilities, automation tools, and collaboration functionalities, this paper aims to provide insights into how **Postman** can be leveraged to enhance software quality and accelerate the development lifecycle.

Keywords: **POSTMAN, Jwt, Bcrypt, MongoDB, Restful API, HTTPS, Testing**

Introduction

API (Application Programming Interface) testing is a critical aspect of software development, ensuring the reliability, security, and performance of APIs that facilitate communication between different software components.

In recent years, the importance of API testing has grown significantly, driven by the increasing adoption of web services, microservices architecture, and cloud computing. This introduction provides an overview of the significance of API testing, highlighting its role in ensuring software quality, accelerating development cycles, and enhancing user experience. Additionally, it introduces Postman as a prominent tool for API testing and sets the context for the subsequent sections of this review paper.

To reduce overlapping tools and methods, same tools were used as for functionality testing in development phase. Quality Assurance (QA) team was interviewed for tools that they use and also for any possible existing scripts that could be used for Operations team testing.

Goal is also to automate, as much as possible, testing for production environment customer base after product update. Depending on the automation outcome, testing could also be run on the back ground and notify for any anomalies found.

What is Postman?

Postman is a popular API testing tool that offers a wide range of features for designing, testing, and debugging APIs.

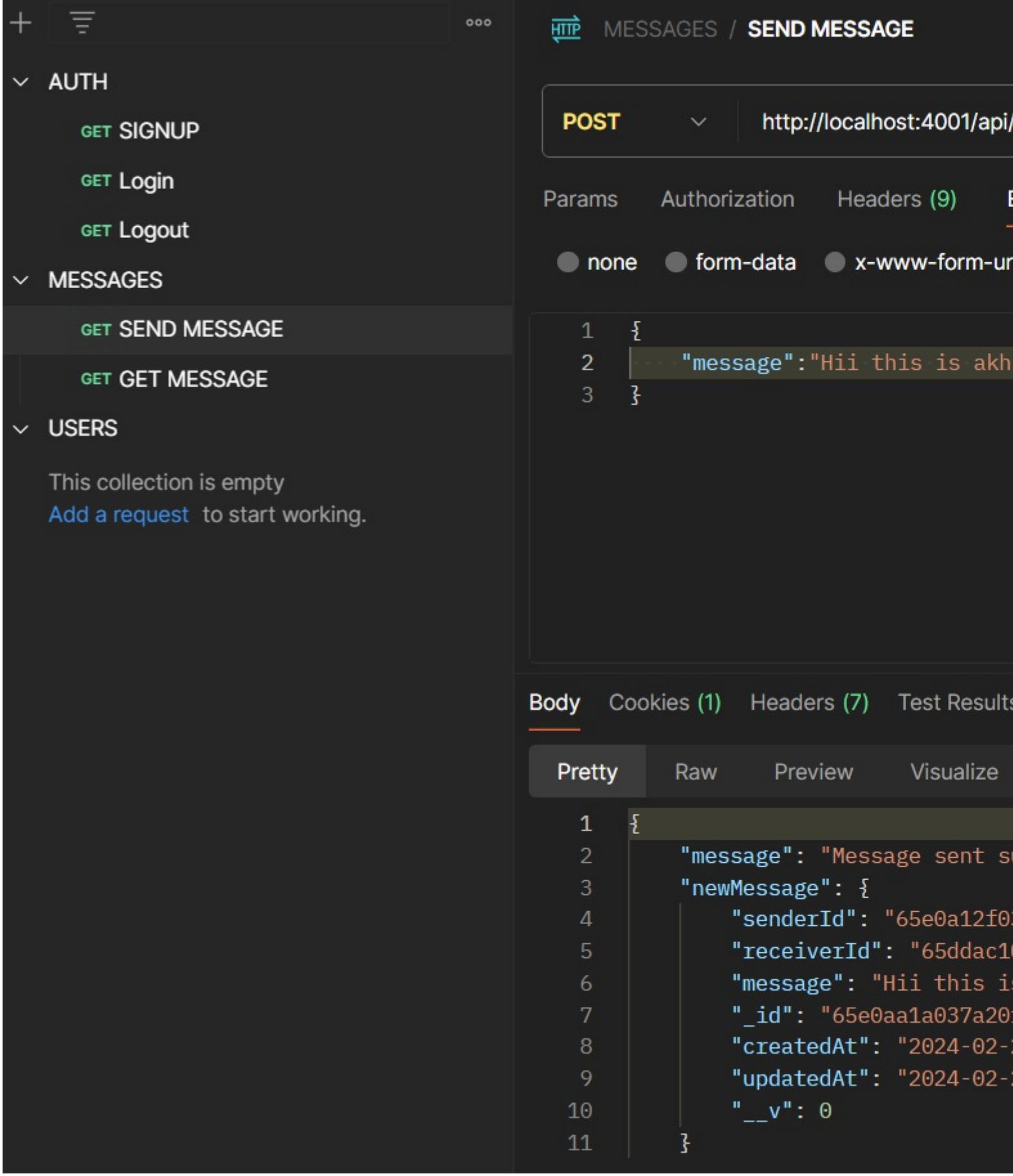

Figure 1. Postman user interface from Postman.

Postman software is either downloaded to the users´ computer (Windows, Mac and Linux operation systems are supported) or used from the web browser with Postman Web version.

Postman itself is then used to send requests to APIs. “Requests can retrieve, add, delete or update data” [9]. Requests can be used to send parameters, login details, authorization information or any other body data that is needed [9].

These requests are then added to collections and collection can host multiple requests. Data that request receives is called response.

2. Literature Review

(API Testing and Postman)

Introduction:

API testing plays a fundamental role in modern software development, ensuring that APIs function as intended and meet the expectations of stakeholders. This section explores the significance of API testing, the emergence of Postman as a dominant tool in this domain, and the key features that make it indispensable for developers and testers.

Importance of API Testing:

APIs serve as the primary means of communication between different software systems, making effective API testing essential to verify functionality, adherence to specifications, and security. API testing helps uncover defects, vulnerabilities, and performance issues early in the development lifecycle, leading to improved software quality and faster time-to-market.

Evolution of Postman:

Postman has evolved from a simple Chrome extension for API development and testing into a comprehensive platform with a wide range of features. Founded in 2014 by Abhinav Asthana, Postman gained popularity due to its intuitive interface, automation capabilities, and collaboration features. Over the years, Postman has expanded its offerings to include features such as monitoring, documentation, and integration with other development tools.

Key Features of Postman:

Postman offers a range of features that make it an indispensable tool for API testing and development. These include a user-friendly interface, automation capabilities, collaboration features, environment management, assertions, testing, and monitoring.

Testing APIs

Test is a functionality in Postman that can be “added to individual responses, collections and folders in collection” [7]. Tests tab of individual request can be seen in user interface picture of Postman. Test scripts need to be written in programming language Javascript [7]. Javascript for testing scripts can be written manually or by using Snippets in the code editor. Snippets are API requests in code format [10].

Test scrips that Postman use “can use dynamic variables, carry out test assertions on response data, and pass data between requests.” [7] In practice this means that one test can lead to another depending on the received response and requests can use data received in previous response.

HTTPS page request

HTTP is technology commonly used in internet for browsing webpages. HTTP supports with the HTTP/1.1 specification GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS and TRACE request methods. The combination of allowed methods can be configured per HTTP server. Server can support all of the possible request methods or just some of them. Methods are case sensitive.

Used in customer link page is HTTPS version of HTTP technology. HTTPS is secured version of HTTP protocol as it encrypts the data on Transport layer. This is important because it protects sent data against different capture methods and false data provider identity type of attacks. Using HTTP on a website can be seen as deprecated technology and ”HTTPS is now used more often by web users than the original non-secure HTTP.”

HTTPS requests can be made with Postman to fetch data available on a website. Received information can be further processed into new requests. [9]

API: Restful interface

API can mean different things in terms of how each software communicates with each other. REST (Representational State Transfer) APIs however try to unison this by using a definitive architecture on how these REST APIs should work. This helps when creating a module to communicate with this interface since it is not completely black box on how it operates.

Terms Restful API and REST API can be a bit confusing, Restful meaning the Restful web APIs but these terms Restful and REST can be used interchangeably. In common lingo, Restful API or REST API is usually meaning the same thing.

When Restful client requires a resource, it connects to the server by using the Restful API. Restful API request and response will follow the basic flow below

1. Client sends a request to the server.
2. Server authenticates the sender client and confirms that client has the rights for required access.
3. Server then starts processing the request if rights check is cleared.
4. Response is returned to the client from server. Response will tell if the request was successful or if any errors occurred. If the request was successful, the response will contain the information requested.

Testing Methods:-

1. **GET**
2. **POST**
3. **PATCH**
4. **DELETE**
5. **PUT**
6. **HEAD**
7. **OPTIONS**

Out of all testing tools used, Postman was selected during the meeting to be used for this new testing setup. Postman provides tools that can be used for testing API's [2] and is used for both QA and Operations team. Application programming interface (API) is used to fetch or modify existing data in other application. API´s can be seen as a way for different applications to communicate with each other.

Testing the REST APIs

This chapter will go through the current state of Contact Pro usage for both customers and internal Sinch usage, Operations teams testing for its usability and where current setup could be improved.

Signup API Testing

Testing signup APIs involves ensuring that the API endpoints responsible for user registration are functioning correctly and securely. Here's a structured approach to testing signup APIs:

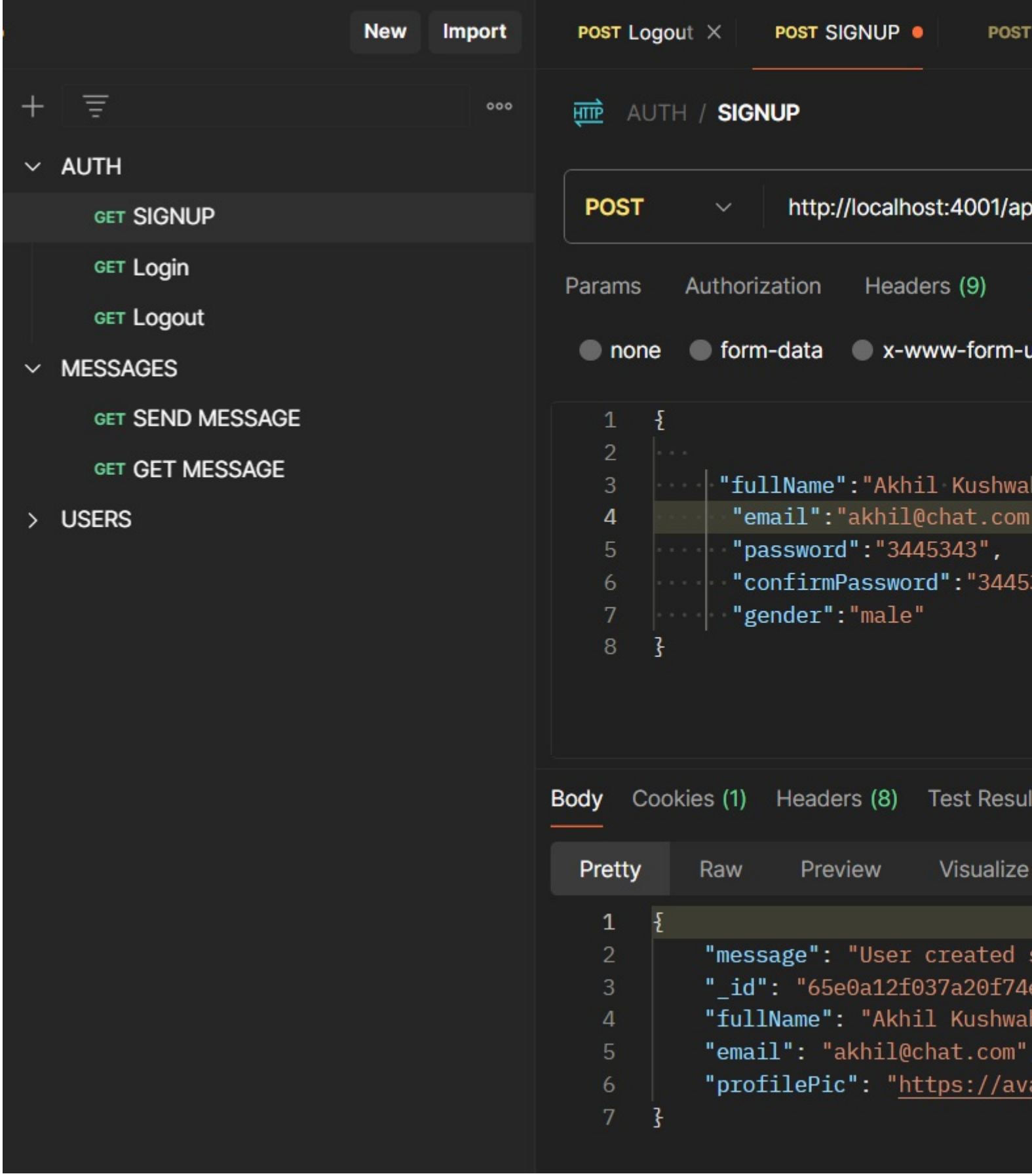

Figure 1. Testing Signup API Using POST Request

Login API Testing

Testing login APIs is crucial to ensure that user authentication and authorization mechanisms work as intended. Here's a structured approach to testing login APIs:

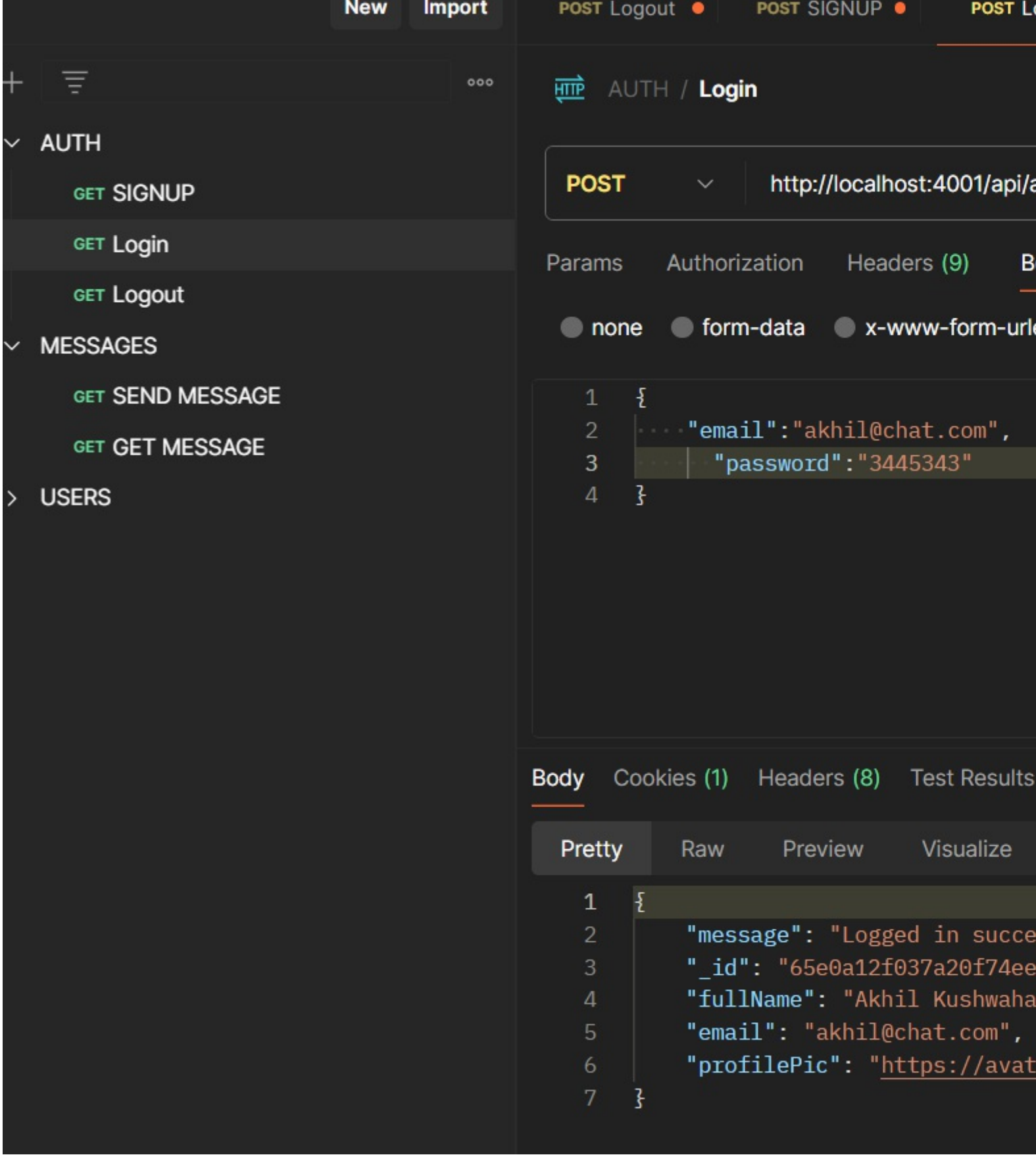

Figure 2. Testing Login API Using POST Request

API´s in this test scenario are tested only two respond. To fully test an API, usually API secret and user id with password, are required to receive the requested information from API.

Functionality testing is also done if customer reports an incident. Functionality is tested by either Operations team or Service Desk team before investigating the issue further from the Contact Pro logs. No active and alerting log reading tool is in use and logs are read only for incident solving and product development.

Jwt

Testing JWT (JSON Web Token) APIs for login and signup involves verifying that the authentication and token generation processes work correctly and securely. Here's how you can approach testing JWT APIs for both login and signup functionalities:

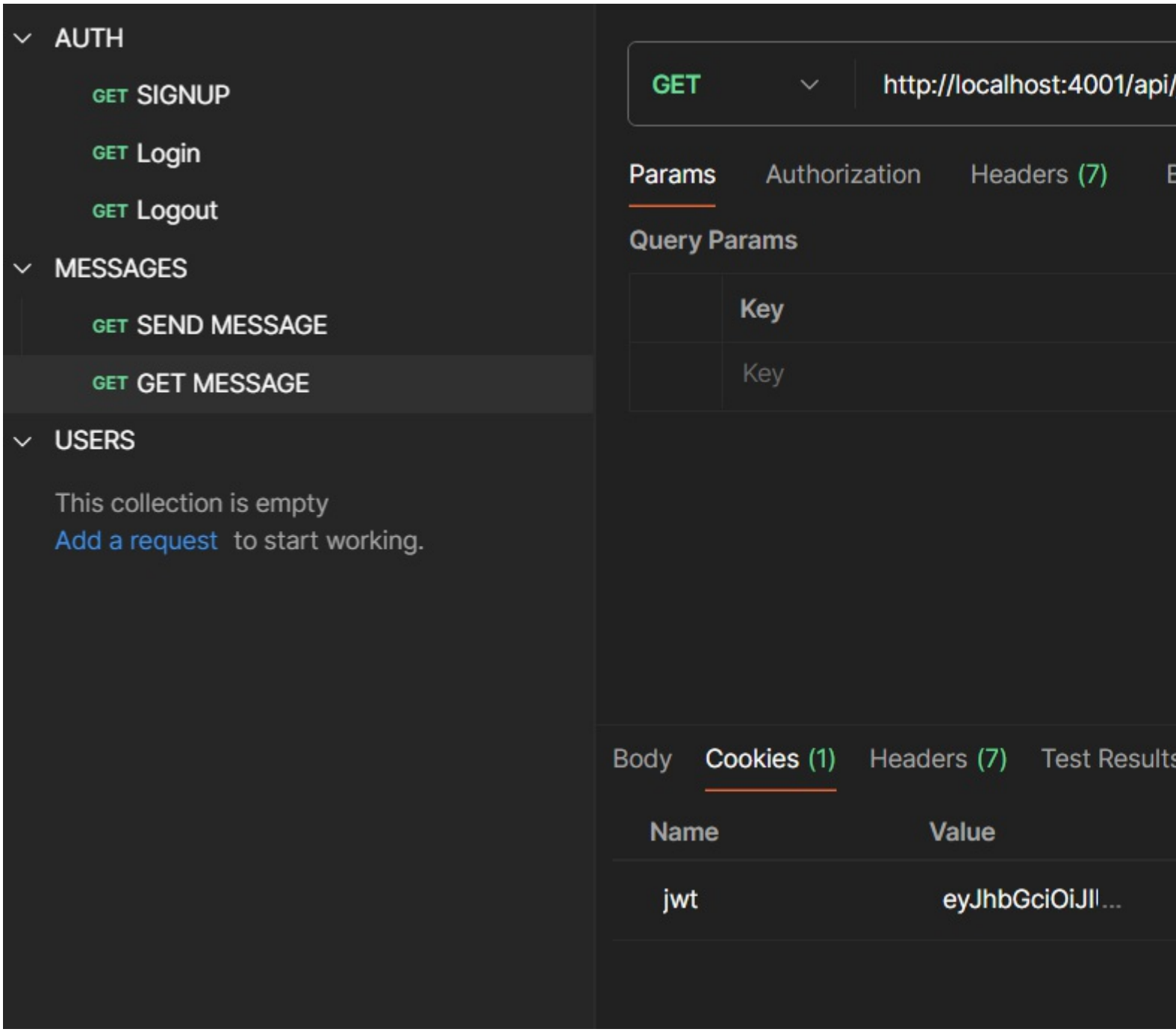

Figure 3. Authenticating Logged in user with Jwt Using GET Request

Bcrypt

bcrypt is a cryptographic hash function designed to securely hash passwords. It is commonly used for password hashing in software applications to protect user passwords from being compromised in the event of a data breach. Here's how bcrypt is typically used:

1. Hashing Passwords: When a user creates an account or updates their password, bcrypt is used to hash the password before storing it in the database. The bcrypt hash function takes the password as input and generates a secure hash value that is then stored in the database.
2. Salting: bcrypt automatically handles salting, which involves adding random data (salt) to the password before hashing it. Salting adds an extra layer of security by ensuring that even if two users have the same password, their hashed passwords will be different due to the unique salts.
3. Verification: When a user attempts to log in, the password they provide is hashed using bcrypt and compared to the stored hash in the database. If the hashes match, the password is considered valid, and the user is granted access.

```
const bcrypt = require('bcryptjs');

// Hashing a password
const plaintextPassword = 'userPassword123';
bcrypt.genSalt(10, function(err, salt) {
    bcrypt.hash(plaintextPassword, salt, function(err, hash) {
        // Store the hash in the database
        console.log('Hashed Password:', hash);
    });
});
```

Database (MongoDB)

MongoDB is a popular, open-source, NoSQL database management system (DBMS) designed for high-performance, high availability, and scalability. It is part of the document-oriented database category, which means it stores data in a flexible, JSON-like format called BSON (Binary JSON).

Here are some key characteristics and features of MongoDB:

Document-Oriented: MongoDB stores data in collections, which are containers for documents. Each document is a JSON-like data structure composed of field-value pairs, allowing for flexible schema design.

Scalability: MongoDB is designed to scale out horizontally across multiple servers, enabling it to handle large volumes of data and high throughput. It supports sharding, which distributes data across multiple machines, and replication, which provides data redundancy and fault tolerance.

High Performance: MongoDB is optimized for high-performance read and write operations. It uses various optimizations such as indexing, query optimization, and in-memory storage engine to ensure fast data access.

Flexible Schema: Unlike traditional relational databases, MongoDB does not require a predefined schema. Fields within documents can vary from document to document, providing flexibility in data modeling.

Rich Query Language: MongoDB supports a rich query language that includes features such as filtering, sorting, aggregation, and geospatial queries. It also provides full-text search capabilities.

Automatic Failover: MongoDB provides automatic failover and data redundancy through replica sets. In the event of a primary node failure, a secondary node automatically takes over to ensure continuous availability.

Community and Enterprise Editions: MongoDB is available in both community and enterprise editions. The community edition is free to use and open-source, while the enterprise edition offers additional features such as advanced security, monitoring, and support services.

Overall, MongoDB is widely used in modern web applications, big data solutions, and real-time analytics platforms due to its flexibility, scalability, and performance characteristics.

Background information

In this section background information is provided to give some understanding for the methods and software involved.

Testing practices and needs

Here are some reasons why testing APIs in Postman is beneficial:

1. Ease of Use: Postman provides a user-friendly interface that allows testers to easily create and execute API requests. Its intuitive design and features such as pre-request scripts and variables make it easy to set up and automate API tests.
2. Collaboration: Postman offers collaboration features that enable team members to share collections, test suites, and environments. This fosters collaboration among developers, testers, and other stakeholders involved in API development and testing.
3. Automation: Postman allows testers to automate API tests by creating collections and test suites that can be run repeatedly. This is useful for regression testing, where tests need to be executed frequently to ensure that new changes do not introduce regressions or break existing functionality.
4. Environment Management: Postman provides support for managing different environments such as development, testing, and production. This allows testers to easily switch between environments and configure variables such as URLs, authentication tokens, and headers.
5. Assertions: Postman allows testers to define assertions to validate the responses returned by API endpoints. Assertions can be used to verify status codes, response bodies, headers, and other aspects of API responses, ensuring that the API behaves as expected.
6. Debugging: Postman offers debugging tools such as console logs, response visualizers, and request/response history, making it easier to debug issues and troubleshoot API problems.
7. Monitoring: Postman provides monitoring capabilities that allow testers to monitor API performance, uptime, and response times. This helps identify performance bottlenecks and ensure that APIs meet service-level agreements (SLAs).
8. Integration: Postman integrates with other tools and services such as version control systems, continuous integration (CI) pipelines, and API documentation platforms, enabling seamless integration into the development workflow.

Overall, testing APIs in Postman offers several benefits including ease of use, collaboration, automation, environment management, assertions, debugging, monitoring, and integration, making it an essential tool for API development and testing.

Conclusions and further improvement points

The created test collection can be considered to be success. Test collection can be run by anyone on Operations team so running the tests is not limited to just few persons. Only limitation, outside of Operations teams know-how, comes from the fact that in Chapter 4, CognitoAsfData access token was created using *amazon-cognito-identity-js* JavaScript package. This token was created with QA teams help and running such packages is not something Operations team does. Token is long lived, active for one year once created, but replacement is required in time.

Full test run taking less than 4 minutes, when 504 individual tests are executed on 168 tenants. Active tenants are always fetched from customer link page when collection is run so up-to-date information is used in execution. Fast execution time also means that at this point all tenants can be tested at once. If tenant amount increases by a lot, regional runs may need to be created in order to test only the region that has been upgraded.

Improvement points for the future use would be:

1. Adding Communications panel login test to this Postman collection would be beneficial.
2. Generate new CognitoAsfData access token automatically, or atleast add notification when this is about to expire.
3. Automate this Postman collection run to run in background, either using Postman's own tools or with other means. Notifications about possible failures would be needed to fully utilize this automation.
4. Regional runs, instead of testing all tenants, if full test run time increases to tens of minutes.

Results

The results section presents the findings of the review, summarizing the key insights, trends, and patterns identified in the literature related to Postman API testing. It discusses the advantages, limitations, and potential challenges associated with using Postman for API testing.

Discussion

The discussion section interprets the results of the review in the context of existing knowledge and explores implications for practice, research, and future developments in API testing. It examines the strengths and weaknesses of Postman as a tool for API testing and considers its broader impact on software development processes.

Conclusion

The conclusion section summarizes the main findings of the review and reiterates the significance of Postman API testing in contemporary software development. It offers insights into the implications of the review findings and suggests areas for further research and exploration in the field of API testing.

References

1. Sinch. Contact Pro | Sinch's Omnichannel Cloud Contact Center [Internet]. Stockholm: Sinch; 2022 [Cited 2022 Oct 25]. Available from: https://www.sinch.com/products/customer-engagement/contact-pro/
2. Postman. Postman API Platform [Internet]. San Francisco: Postman; 2022 [Cited 2022 Oct 10]. Available from: https://www.postman.com/product/what-is-postman/
3. Lee Copeland. A Practitioner's Guide to Software Test design [Internet]. London: Artech House; 2003 [cited 2023 Nov 10]. Chapter 12. Available from: https://ebookcentral.proquest.com/lib/metropolia- ebooks/detail.action?docID=227688
4. Sinch Contact Pro Service Description [Internet]. Sinch; 2022. [Cited 2023 Jan 10]. [40 p]. Available from: https://docs.cc.sinch.com/cloud/service- description/en/Service_Description.pdf
5. Sinch. Service Configuration [Internet]. Stockholm: Sinch; 2022. [Cited 2022 Oct 25]. Available from: https://docs.cc.sinch.com/cloud/service- configuration/en/index.html visited
6. Amazon Web Services. Configuring MFA-protected API access – AWS Identity and Access Management [Internet]. Seattle: Amazon Web Services; 2022. [Cited 2022 Nov 26]. Available from: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_c onfigure-api-require.html
7. Postman. Writing tests | Postman Learning Center [Internet]. San Francisco: Postman; 2022. [Cited 2022 Dec 05]. Available from: https://learning.postman.com/docs/writing-scripts/test-scripts/
8. Postman. Download Postman | Get Started for Free [Internet]. San Francisco: Postman; 2022. [Cited 2022 Dec 05]. https://www.postman.com/downloads/
9. Postman. Building requests | Postman Learning Center [Internet]. San Francisco: Postman; 2022. [Cited 2022 Dec 05]. Available from: https://learning.postman.com/docs/sending-requests/requests/

10. Postman. Generating client code | Postman Learning Center [Internet]. San Francisco: Postman; 2022. [Cited 2022 Dec 05]. Available from: https://learning.postman.com/docs/sending-requests/generate-code- snippets/
11. Wikipedia. Hypertext Transfer Protocol – Wikipedia [Internet]. San Francisco: Wikipedia; 2022. [Cited 2022 Dec 29]. Available from: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
12. Wikipedia. HTTPS [Internet]. San Francisco: Wikipedia; 2022. [Cited 2022 Dec 05]. Available from: https://en.wikipedia.org/wiki/HTTPS
13. Amazon Web Services. What is Amazon Cognito? – Amazon Cognito [Internet]. Seattle: Amazon Web Services; 2022. [Cited 2022 Dec 06]. Available from: https://docs.aws.amazon.com/cognito/latest/developerguide/what-is- amazon-cognito.html
14. Amazon Web Services. What is RESTful API? – RESTful API Beginner's guide [Internet]. Seattle: Amazon Web Services; 2022. [Cited 2022 Dec 10]. Available from: https://aws.amazon.com/what-is/restful-api/
15. Sinch. Contact Pro Documentation [Internet]. Stockholm: Sinch; 2022. [Cited 2022 Dec 25]. Available from: https://docs.cc.sinch.com/cloud/api.html
16. Open source. Cheerio [Internet]. Global: Open source; 2022. [Cited 2022 Dec 28]. Available from: https://cheerio.js.org/
17. Amazon Web Services. Integrating Amazon Cognito with web and mobile apps – Amazon Cognito [Internet]. Seattle: Amazon Web Services; 2022. [Cited 2022 Dec 28]. Available from: https://docs.aws.amazon.com/cognito/latest/developerguide/cognito- integrate-apps.html
18. Amazon Web Services. AuthenticateCognitoActionConfig – Elastic Load Balancer [Internet]. Seattle: Amazon Web Services; 2022. [Cited 2022 Dec 29]. Available from: https://docs.aws.amazon.com/elasticloadbalancing/latest/APIReference/A PI_AuthenticateCognitoActionConfig.html