COM661 Full-Stack Strategies and Development

Practical B6: API Testing and Documentation

Aims

- To appreciate the importance of through API testing
- To develop a coherent test plan for the ***Biz*** API
- To introduce the Postman testing facility
- To develop an automated test script for the ***Biz*** API
- To introduce the Postman topol for automated API documentation
- To document the requests and parameters of the ***Biz*** API
- To illustrate the addition of examples to the test documentation
- To publish the API documentation on the Web using Postman's automated toolkit

Contents

B6.1 Automated API Testing

API testing involves comparing the result of requests to API endpoints with the intended behavior. For example, in the API developed so far, a GET request to the endpoint http://localhost:5000/api/v1.0/businesses should result in the retrieval of a JSON formatted collection of business information, with the HTTP status code 200. If a code other than 200 is returned, then the request has failed due to incorrect input, program code error or non-availability of the database.

Comprehensive testing of an API can be a very laborious process as each request to each endpoint should be tested each time the code is modified or deployed to a new environment. This situation is greatly eased, however, by Postman, which provides a useful tool that allows us to specify a collection of tests that can be run with a single button click.

B6.1.1 Creating a test plan

Our API supports the following endpoints and requests.

Endpoint	Method	Meaning	User type
/api/v1.0/businesses	GET	Fetch list of businesses	All
	POST	Add new business	Registered
/api/v1.0/businesses/<id>	GET	Fetch a single business	All
	PUT	Modify a business	Registered
	DELETE	Delete a business	Admin
/api/v1.0/businesses/<id>/reviews	GET	Fetch list of reviews	All
	POST	Add new review	Registered
/api/v1.0/businesses/<id>/reviews/<id>	GET	Fetch single review	All
	PUT	Edit single review	Registered
	DELETE	Delete a review	Admin

Our test plan should provide for all of these combinations, making sure that all provide the desired response for any combination of input. As our database is a "living" one, with businesses and reviews being added, modified and deleted as time progresses, the

appropriate order is to begin with actions available to *Registered* users so that we can test the addition and modification of businesses and reviews, before proceeding to test the general retrieval operations open to all – in case the retrieval operations are adversely affected by addition and modification of the data. Finally, we will test the endpoints available to *Admin* users to delete the previously added reviews and businesses, so leaving the database in the state it was in before testing.

B6.1.2 Registered user testing

Postman allows us to arrange our tests in collections so that we can keep the tests for an API together and run them as a batch on request. Launch Postman and sign in by either creating a new account or using your Google account. Make sure that the sidebar is visible and click on "My Workspace" to reveal a screen as shown in Figure B6.1, below.

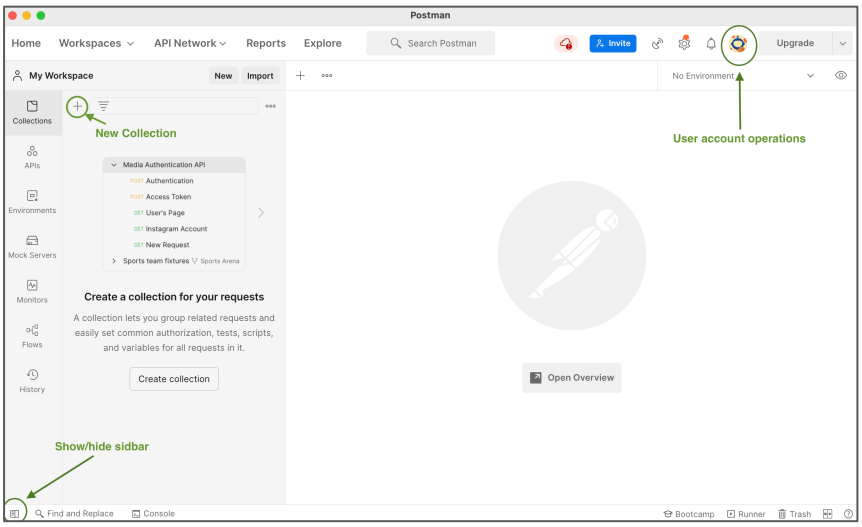

Figure B6.1 Postman Account Operations and Sidebar

Now use the ***New Collection*** button to create a new collection called ***Biz*** to hold your tests. Before we can add a new business to the database, we need to log in to the application and generate a JWT token, so create a new request in the Biz collection and specify a GET request to the endpoint http://localhost:5000/api/v1.0/login. In the ***Authorization*** tab, select ***Basic Auth*** and provide the username and password for one of your registered (but not admin) users. If you click ***Send*** and issue the request, you will see that the request is processed and the JWT token is presented in the Postman output pane. Previously we manually copied this token value and pasted it as a **x-access-token** header, but we want the test process to run without user intervention, so we need to have the token value saved in a variable that we can then use in subsequent requests.

Postman provides a global variable structure that allows us to save values returned from a request so that they can be used later. We do this by selecting the ***Tests*** tab as shown in Figure B6.2, below and specifying the JavaScript code that retrieves the value from the result and saves it.

First, click on the ***Set a global variable*** option from the ***SNIPPETS*** menu in the right-hand pane. This causes the line of code

```
pm.globals.set("variable_key", "variable_value");
```

to be added to the tests pane. Change the "**`variable_key`**" entry to read **`"jwt_token"`** (the name of our global variable). Now, we will retrieve the token value from the **`pm.response`** object by adding the line of code

```
response = pm.response.json();
```

above the previously inserted line. Finally, we replace the "**`variable_value`**" entry with **`response.token`** to extract the value of the token object and set it as the value of the global variable.

Note:	Any valid JavaScript code can be used to specify the test, so we could check for the presence of a particular key field, that an array returned is of a certain length (or within particular boundaries) or any other format or value check.

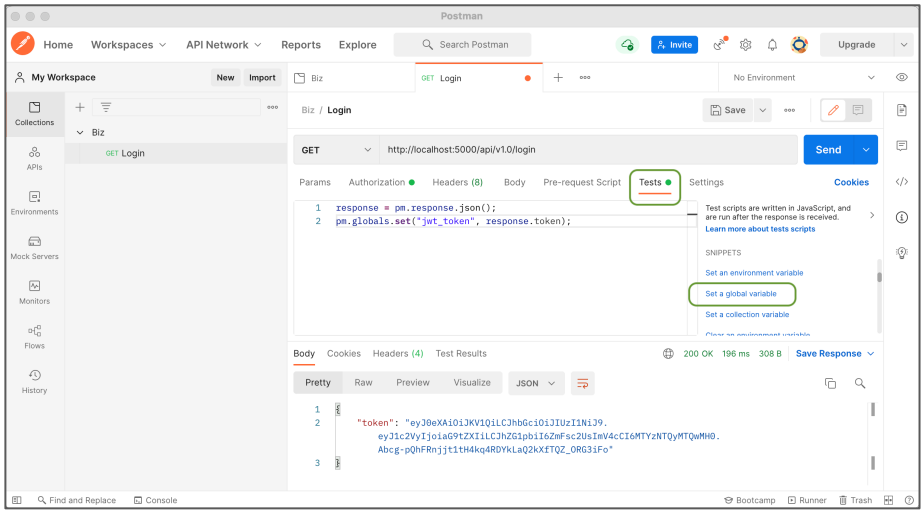

Figure B6.2 Fetch token from response object

Now, issue the request by clicking ***Send*** and view the value of the newly created global variable by clicking the ***Show Global Variables*** icon as highlighted in Figure B6.3. The new token value should now be seen against the **`jwt_token`** entry in the variable list.

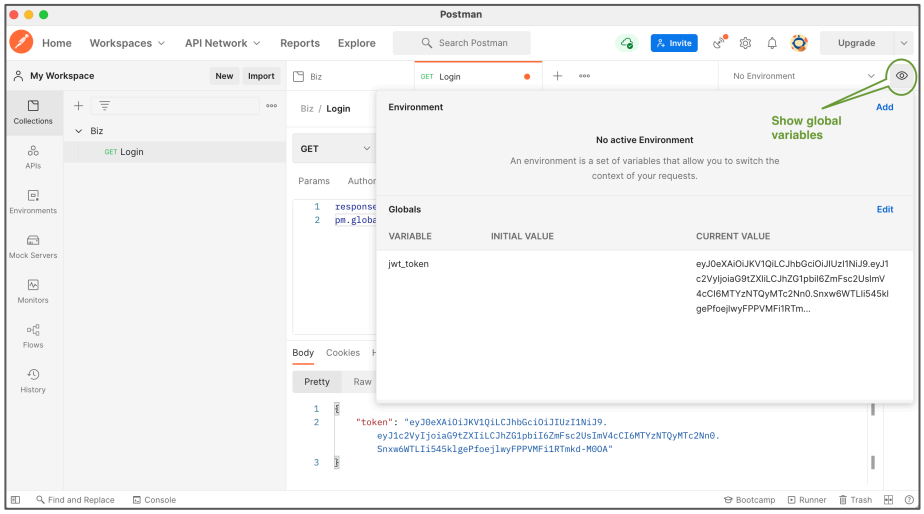

Figure B6.3 Inspect global variables

Now that the login operation is complete and the new JWT token has been saved, we can use the token in a request to add a new business. Save the current request and add a new request called "*Add new business*" to the collection as a POST request to http://localhost:5000/api/v1.0/businesses. Specify values for the parameters **`name`**, **`city`** and **`stars`** in the ***Body*** tab as shown in Figure B6.4, below.

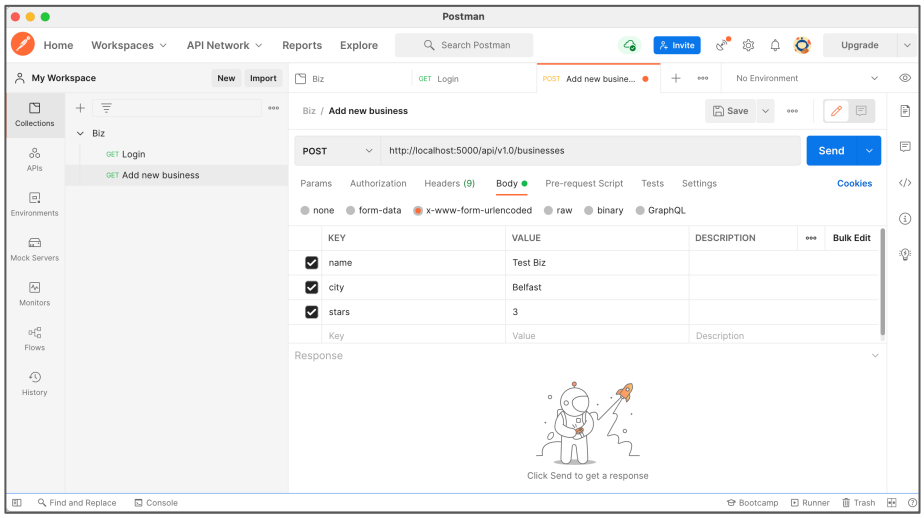

Figure B6.4 Parameters for new business

Now add the JWT token to the request by visiting the ***Headers*** tab and creating a new header called ***x-access-token***, as previously. For the value of the token, we can retrieve the

global variable value by specifying the name of the variable in double curly brackets, such as **`{{jwt_token}}`**, as shown in Figure B6.5

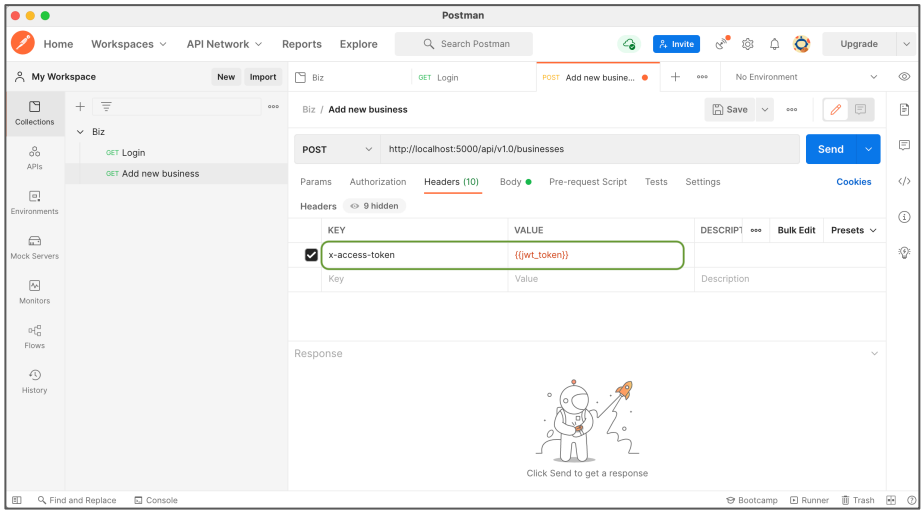

Figure B6.5 Add JWT token to header

Now, we can specify the tests for the ***Add New Business*** request. The operation has been successful when a HTTP Status code of 201 is returned, so we begin by selecting the ***Tests*** tab, clicking on the ***Status code: Code is 200*** option from the ***SNIPPETS*** list and editing the code so that it checks for return code **201** instead of **200**. Next, we repeat the previous operation to set a global variable by retrieving the URL returned from the response and setting it to a new global variable called **`new_url`** as shown in Figure B6.6, below.

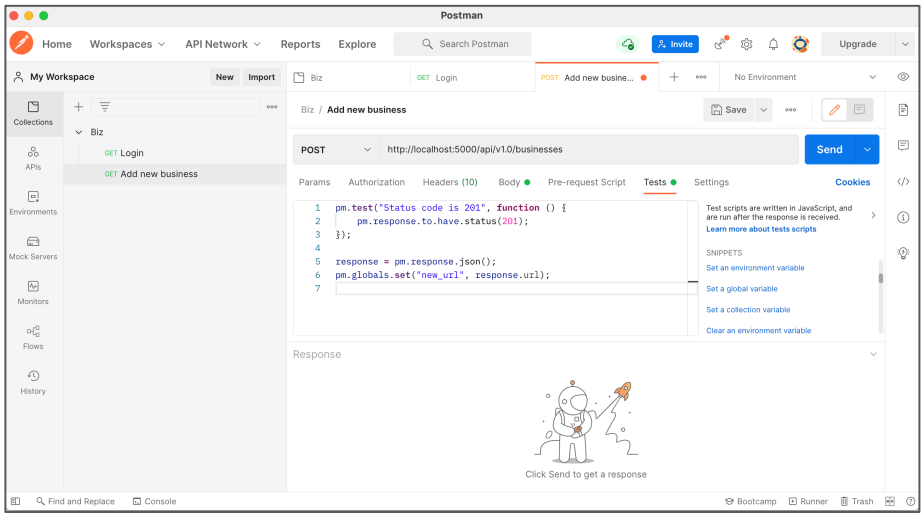

Figure B6.6 Specify tests for add business

Now, when we issue the request and check the global variables, we see that the new global variable has been created with a value of the URL of the new resource, shown in Figure B6.7, below.

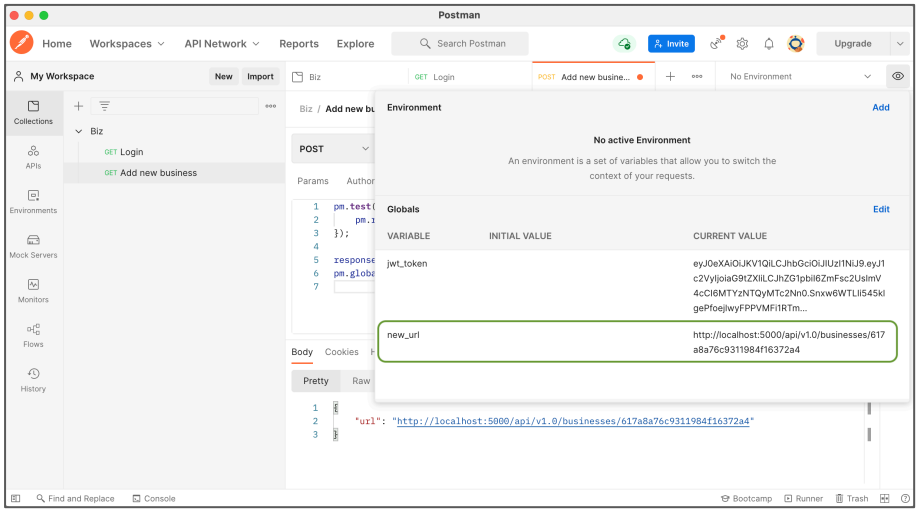

Figure B6.7 New URL generated and saved

Finally, we can run the two tests specified so far as a single operation (make sure that both tests have been saved). Open the Collection pane by clicking on the arrow icon next to the collection name and open the ***Collection Runner*** by the ***Run*** button, highlighted in Figure B6.8, below.

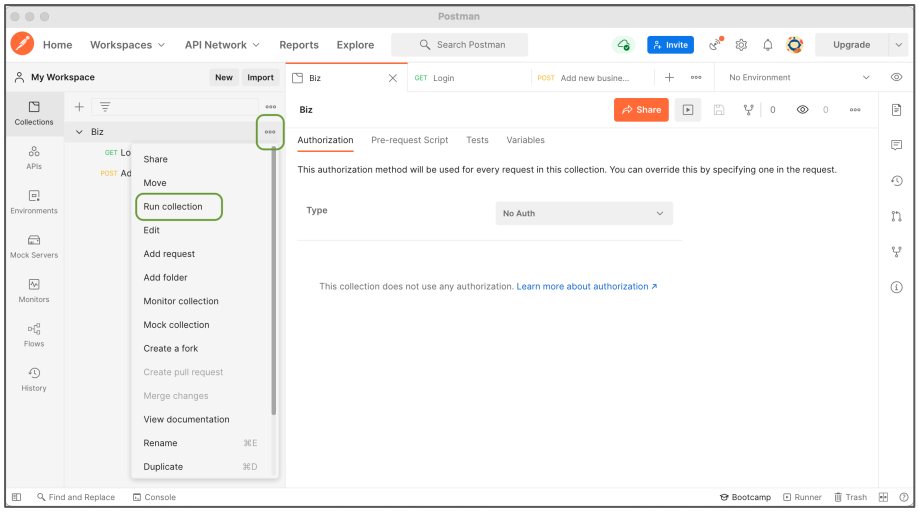

Figure B6.8 Open the Collection Runner

From the ***Collection Runner*** window shown in Figure B6.9 below, click the ***Run Biz*** button to run the tests.

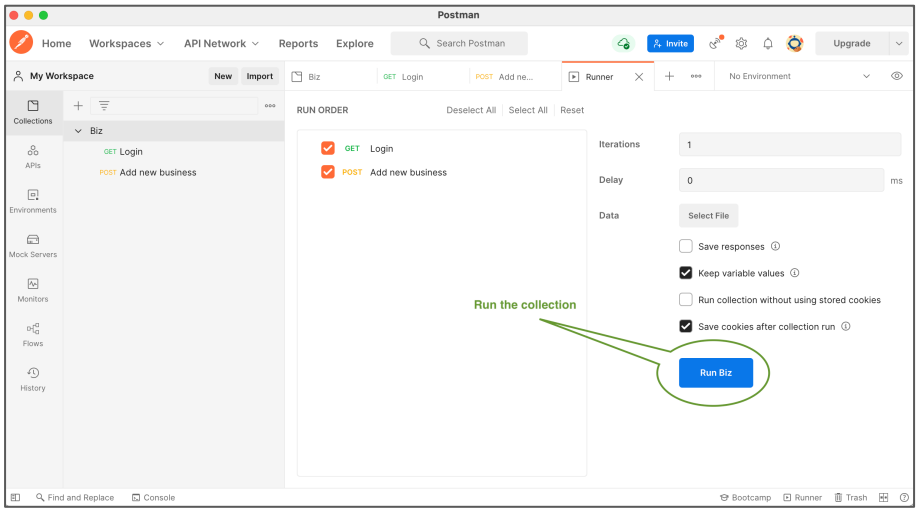

Figure B6.9 Collection Runner

Finally, you should see the result of the tests as described in Figure B6.10, below.

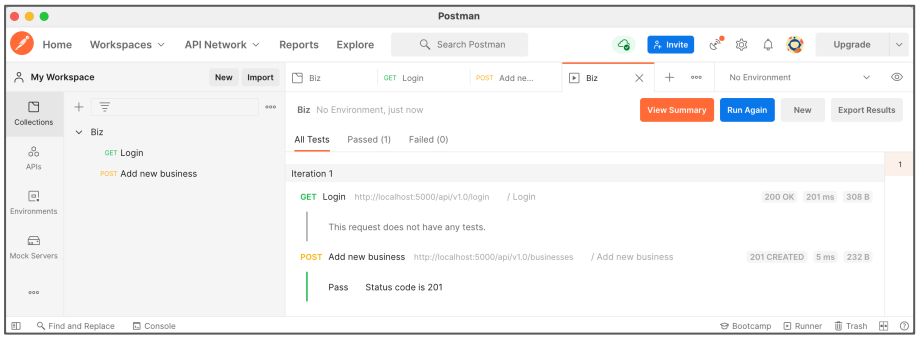

Figure B6.10 Test results

Do it now!	Follow the steps outlined above to create the ***login*** and ***add new business*** tests and verify their operation. Make sure that you have first run the MongoDB database server and the ***app.py*** application file.

Do it now!	Add a test to the login entry so that it checks for status code 200 being returned. Run the collection of tests and verify that they work as expected.

Try it now!	It is important to check that tests fail when invalid input is provided. Change the ***add business*** test so that it checks for status code 200 instead of 201 and re-run the collection. Verify that you receive output such as that shown in Figure B6.11, below.

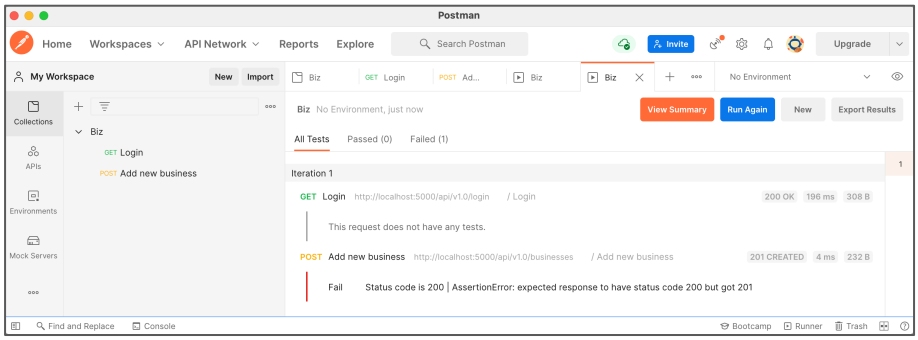

Figure B6.11 Tests fail

Try it now!	Restore the status code check to 201 and remove one of the input parameters name, city or stars. Make sure that the test fails with a "Missing form data" message.

We can now repeat the previous process to test the endpoint that adds a new review to a business. Remember that we created the **`new_url`** global variable to hold the url of a newly added business, so we can make use of this to build the endpoint that allows us to add a review to that business. The **`new_url`** value will be in the form http://localhost:5000/api/v1.0/<id>, so the URL for the request to add a review can be expressed as **`{{new_url}}/reviews`**.

Create a new test in the ***Biz*** collection to generate a POST request to **`{{new_url}}/reviews`** and provide values for the input fields ***username***, ***comment*** and ***stars*** as shown in Figure B6.12 below. Remember to also add the **`x-access-token`** header as **`{{jwt_token}}`** as for adding a business earlier.

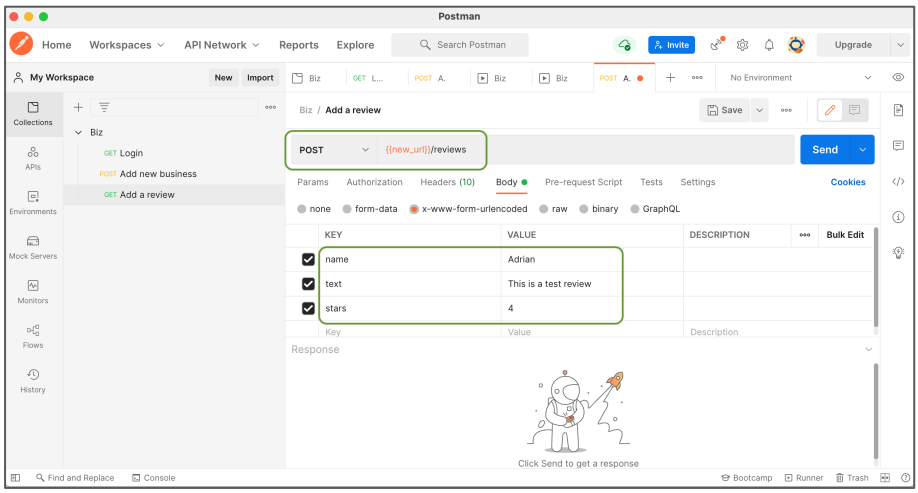

Figure B6.12 Add a new review

Now we can add a test for the request as for adding a new business, by checking that the HTTP Status code returned is **201** and setting a new global variable called ***new_review_url*** to the value returned in the response body. The code is illustrated in Figure B6.13, below.

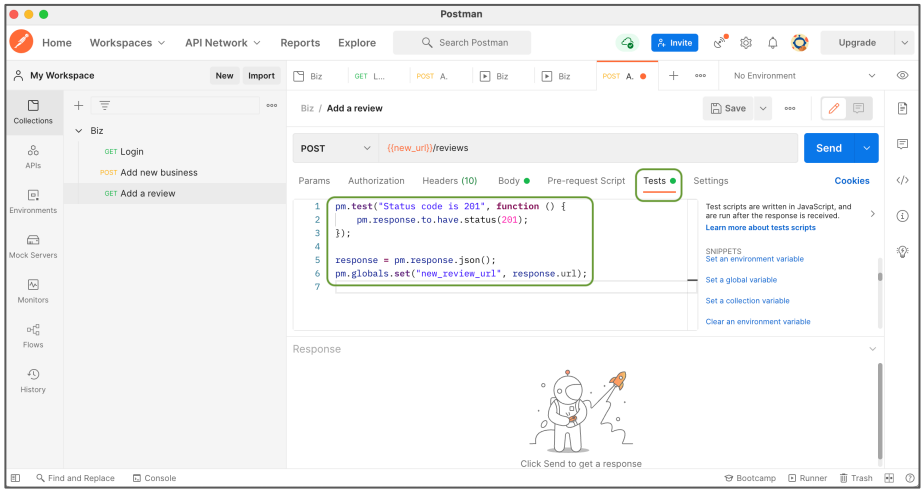

Figure B6.13 Test for new review

Finally, we run the collection of tests and verify that they all succeed as shown in Figure B6.13, below.

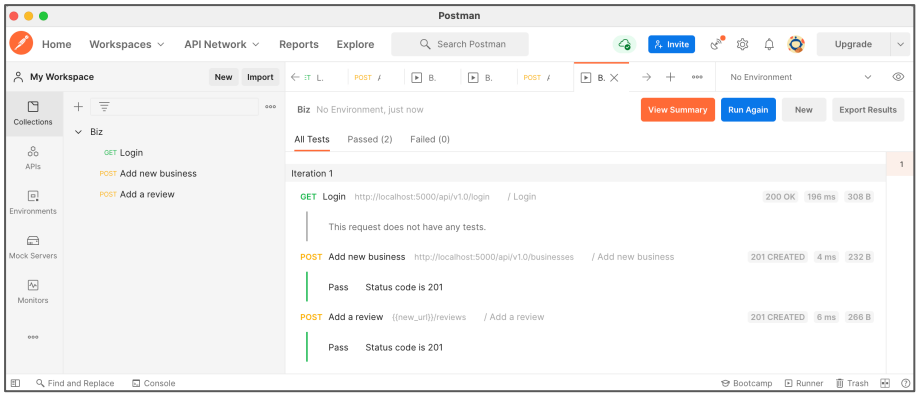

Figure B6.14 Check tests

Do it now!	Follow the steps illustrated above to add a test for the ***add review*** endpoint and verify that they are successful. Check the global variables to make sure that the new **`new_review_url`** variable has been created.

Try it now!	Add additional tests to validate the **PUT** requests that allow a registered user to edit the details of a specified business and specified review by using the global variables **`new_url`** and **`new_review_url`** to construct the endpoints. In each case, the HTTP Status code returned should be 200.

Now that all of the registered user functionality has been added to the test collection, we can add an additional step to the collection to log the user out of the application. This is achieved by simply adding a new entry for a GET request to the URL http://localhost:5000/api/v1.0/logout as illustrated in Figure B6.14, below.

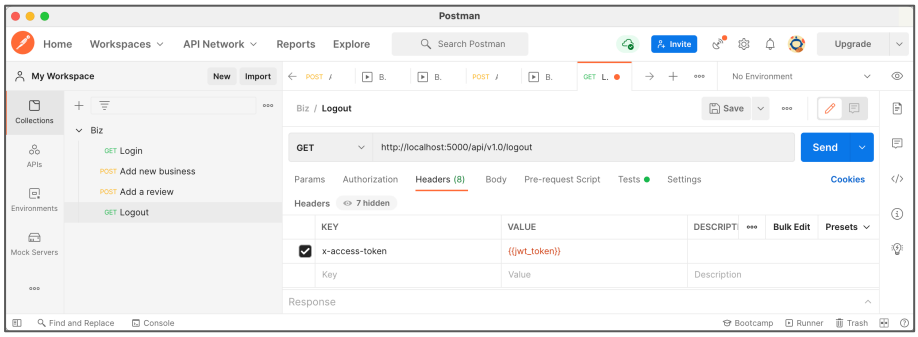

Figure B6.15 Logout user

Do it now!	Add the new entry to the collection to logout the current user. Add a test to the entry so that the HTTP response code is verified to be 200. Now, run the collection of tests and check that they still run successfully.

B6.1.3 General user testing

Now that the endpoints to add and edit businesses and reviews have been tested, we can implement tests for the routes available to general (i.e. non-logged in) users as described below.

Endpoint	Action	Test
GET /api/v1.0/businesses	Get all businesses	Status code is 200
GET /api/v1.0/businesses/<id>	Get a specific business	Status code is 200
GET /api/v1.0/businesses/<id>/reviews	Get all reviews for a business	Status code is 200
GET /api/v1.0/businesses/<id>/reviews/<id>	Get a specific review for a business	Status code is 200

Try it now!	Add the tests above to the ***Biz*** collection and verify that all operate as expected, generating the specified HTTP status code. Use the value of the global variables **`new_url`** and **`new_review_url`** in the tests to retrieve a specific business and review, respectively.

B6.1.4 Admin user testing

The final endpoints to be added to the test collection are those which are reserved for admin users to delete a specific business or review. These are described in the table below.

Endpoint	Action	Test
DELETE /api/v1.0/businesses/<id>	Delete a specific business	Status code is 204
DELETE /api/v1.0/businesses/<id>/reviews/<id>	Delete a specific review for a business	Status code is 204

Try it now!	Add the tests above to the Biz collection and verify that all operate as expected. Use the value of the global variables **`new_url`** and **`new_review_url`** in the tests to retrieve a specific business and review, respectively. Remember to test the delete review route first, before testing the route to delete a business. **Hint:** Before the DELETE requests are sent, you should add an action to log in a user with "Admin" privileges. Add an additional action to log out the admin user after the tests are complete.

B6.2 Automated API Documentation

Effective documentation is a very important aspect of API development. When an API to a data collection is released for use by the wider community, its potential for uptake is greatly enhanced by clear documentation that describes each endpoint in terms of its effect, the parameters required for it to be invoked and the format and organisation of the data that will be returned. Postman provides a very useful tool for the construction, ongoing management and online publication of API documentation in three areas – requests, parameters and collections. We will examine each of these in the sections that follow.

B6.2.1 Documenting Requests

Each request can be documented by providing it with a meaningful name and a short piece of text that describes the purpose of the request. Figure B6.16, below, shows a description applied to the user login request.

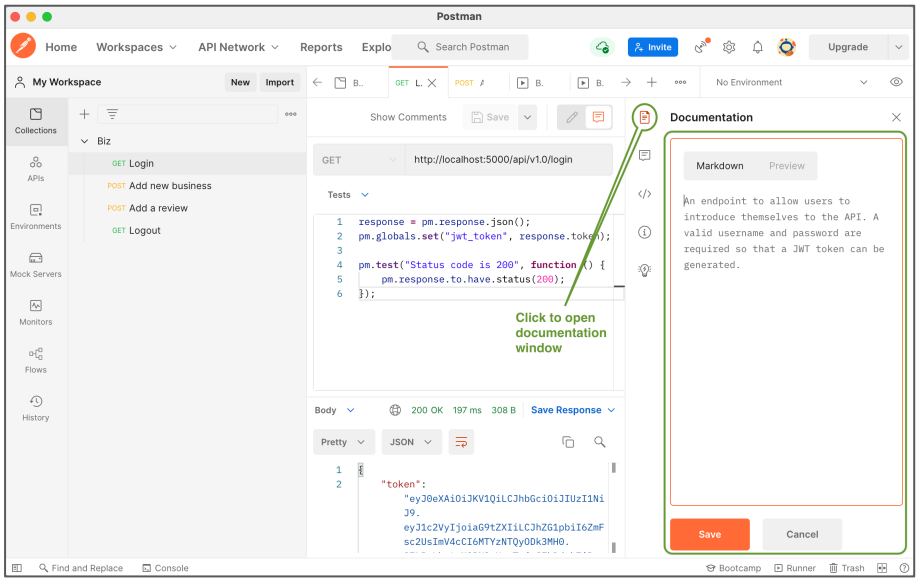

Figure B6.16 Documenting a request

The ***Documentation*** panel is accessed by clicking on the "document" icon in the menu bar on the right hand side (highlighted in Figure B6.16 above). name of the request (the text "User Login" in the highlighted area of the example shown). Hovering over the text area reveals a pencil icon that, when clicked, allows the user to enter or modify the text description of the request. This field can be completed in plain text or in ***Markdown*** – a text formatting notation that provides HTML-type notation for specification of lists, tables, images and other formatting options.

Note:	Markdown is beyond the scope of this module but links to information on it can be found in Section B6.3. You are encouraged to explore this and to use it to provide more professional documentation for your API.

Do it now!	Follow the steps above to create a documentation entry for the "User Login" request.

Try it now!	Create documentation entries for all of the requests in the Biz API

B6.2.2 Documenting Parameters

Our API frequently uses parameters specified as key/value pairs to provide information to be used in requests. These might be form fields (entered in the ***Body/x-www-form-urlencoded*** tab), headers (such as ***x-access-token*** values) or other data.

Anywhere that Postman provides a key/value data input facility also includes provision for a ***Description*** field to provide illustrative information for the value. This can be information regarding the allowable values (text, numeric, allowed range, etc.), the purpose of the parameter, whether it is compulsory or optional, or any other information that helps the user of the API provide the correct request format.

Figure B6.17, below, demonstrates the provision of description information for the parameters for the ***Add New Business*** request. Here, we specify that the business name and location are text strings, while the rating should be an integer value in the range 0-5.

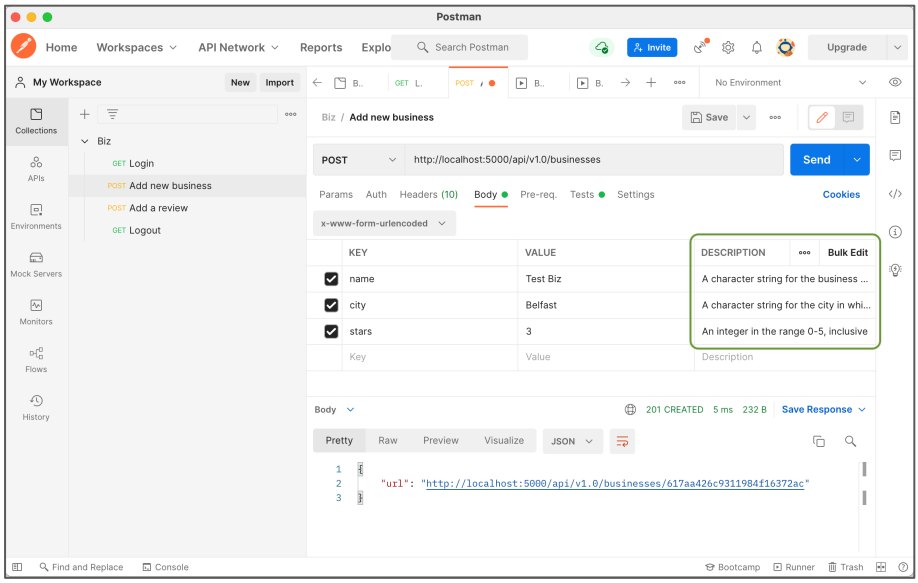

Figure B6.17 Documenting parameters

Do it now!	Provide description values for the parameters to the ***Add New Business*** request as shown in Figure B6.17.

Try it now!	Add documentation for all parameters in your collection – including those in the Headers tab.

B6.2.3 Documenting Collections

The final option for providing textual help for the API consumer is summary documentation that describes the collection as a whole. This is useful in producing a general description of your API, indicating its purpose, intended use and limitations for certain types of users.

The collection documentation is accessed by clicking on the ellipsis (…) to the right of the collection name and selecting ***View Documentation*** from the drop-down menu that appears (illustrated by Figure B6.18, below).

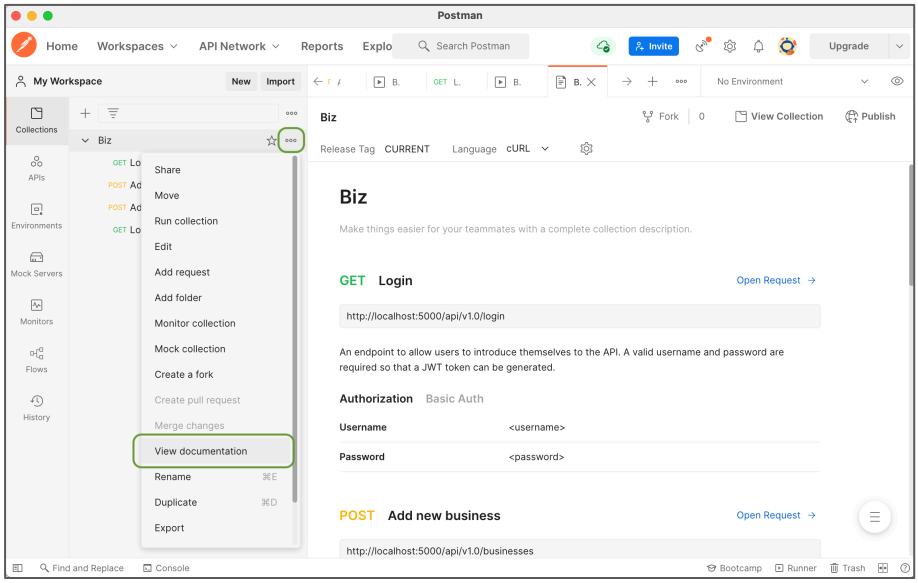

Figure B6.18 Edit a collection description

This reveals the ***Describe Collection*** window as shown in Figure B6.19. below. Here, we hover over on the prompt or previously provided text and provide some summary text to give an overview of the API, detailing its purpose and usage restrictions.

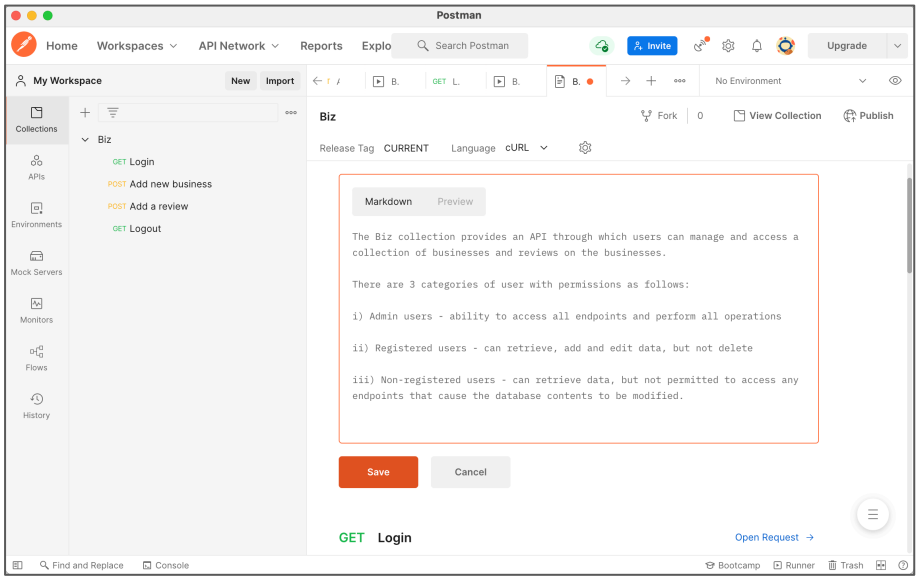

Figure B6.19 Provide a collection description

Do it now!	Add documentation to your collection as shown in Figure B6.19

B6.2.5 Adding Examples

Another powerful feature of Postman's documentation tool is the ability to include examples of API requests to show the format of the request and sample data returned. This can be very useful for users in clearly illustrating how the API can be integrated with their application and how the data returned can be parsed and processed.

To add an example to the documentation, first select the request that you want to demonstrate from the list shown in the collection. Then, ensure that all of the request parameters are set up and click ***Send***. If you are happy with the response generated, click on the ***Save Response*** link on the top-right of the response panel and click on ***Save as example*** as shown in Figure B6.20, below. The example will then be added as a sub-item below the request in the Collections list in the sidebar from where its properties can be manipulated through the menu accessible from the ellipsis to the right of the example name. This is illustrated in Figure B6.21, where the name of the saved example has been changed to "Test Biz added".

Note that you can provide as many examples as you want for each request, if there are multiple elements of functionality that you want to demonstrate.

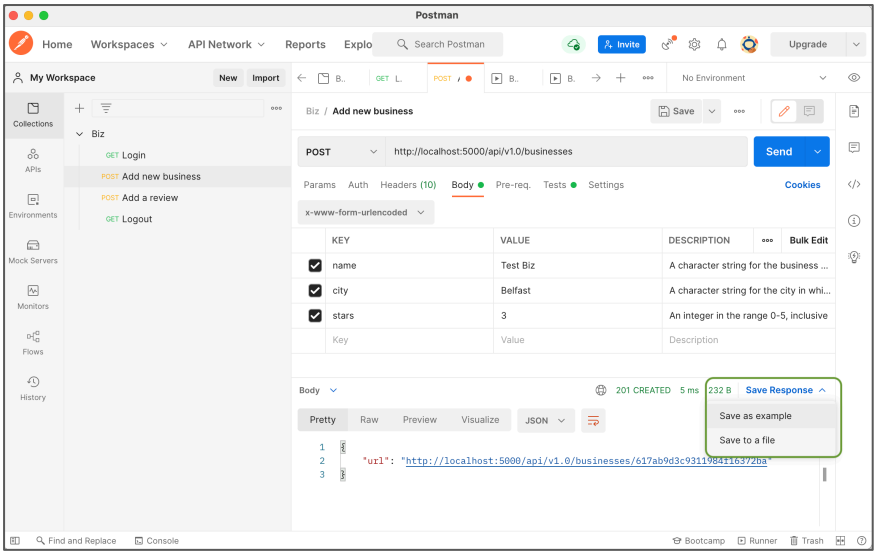

Figure B6.20 Adding an example

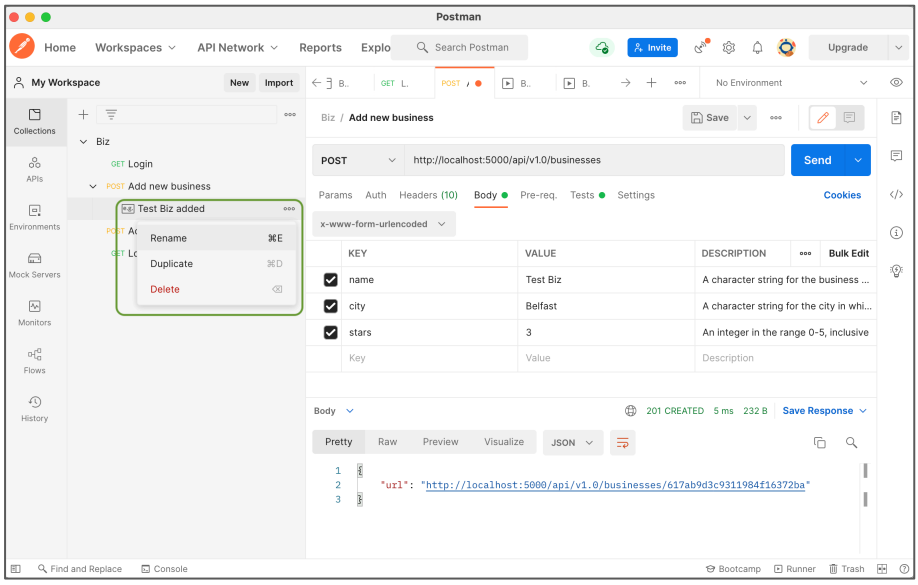

Figure B6.21 Example saved

Do it now!	Add an example to your ***Add New Business*** request as illustrated in Figure B6.20. Remember that you will need to have previously logged in to the application and provided a valid JWT token.

Try it now!	Add additional examples for the other requests in your collection.

B6.2.6 View documentation

Once all of the documentation and examples have been provided, we can publish the documentation to the Web. Click on the ellipsis to the right of the collection name (in the sidebar) and then on the ***View documentation*** option on the menu that is revealed to see a preview of the documentation you have provided. Then, in the documentation panel, click the ***Publish*** icon on the top-right, as illustrated in Figure B6.22, below.

This opens a web browser on the ***Publish Collection*** page, containing a form that enables you to customize your online documentation. Leave the various options at their default values and click on the ***Preview documentation*** link on the left-hand side. This opens another web browser with a live preview of how your online documentation will be presented.

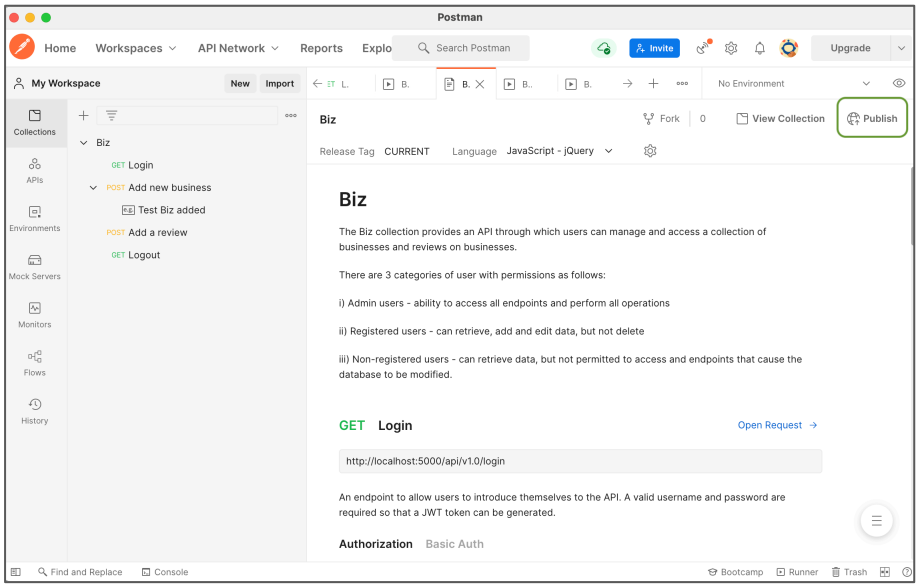

Figure B6.22 View documentation

Note how the list of requests is presented on the left-hand side as a clickable menu giving direct access to each of your requests. The information about every request is presented in the centre pane with all of the descriptive information you have provided; while the right-hand pane shows code demonstrating how to call the endpoint and your example responses. The language used to demonstrate the calls to the API can be changed by the drop-down menu at the top of the pane. Figure B6.23, below, illustrates the web view of the documentation generated so far.

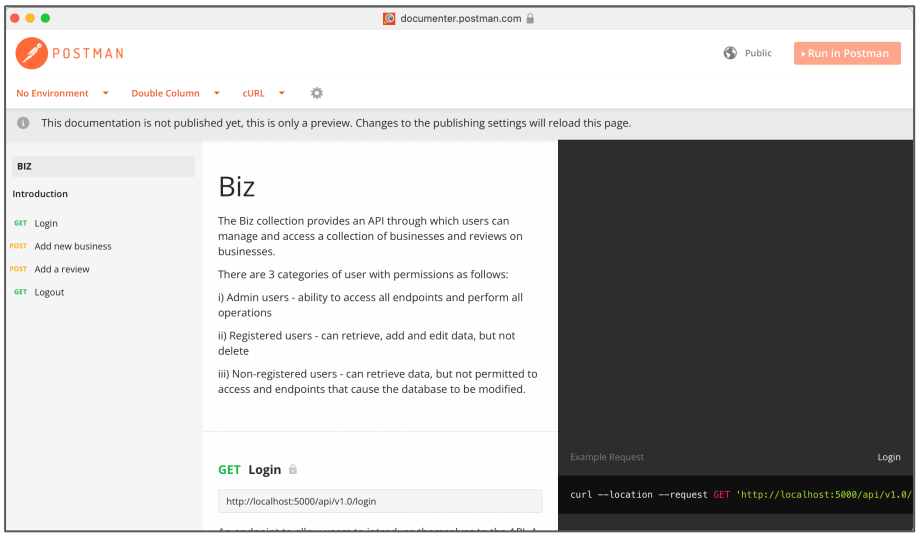

Figure B6.23 Preview documentation

Finally, when you are satisfied with the content and appearance of the documentation, you can publish it publicly by clicking on the ***Publish Collection*** button at the bottom of the ***Publish Collection*** page, shown in Figure B6.24, below.

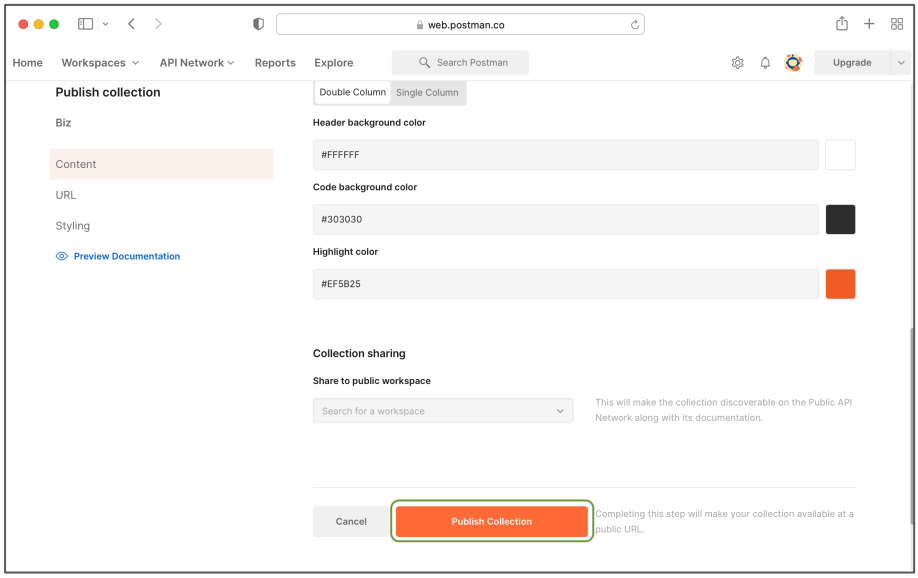

Figure B6.24 Publish documentation on the Web

Once published, clicking on the Publish link in the Documentation panel within Postman will open the ***Publication settings*** page, shown in Figure B6.25 below, which contains the URL that you can distribute for others to access your documentation and use your API.

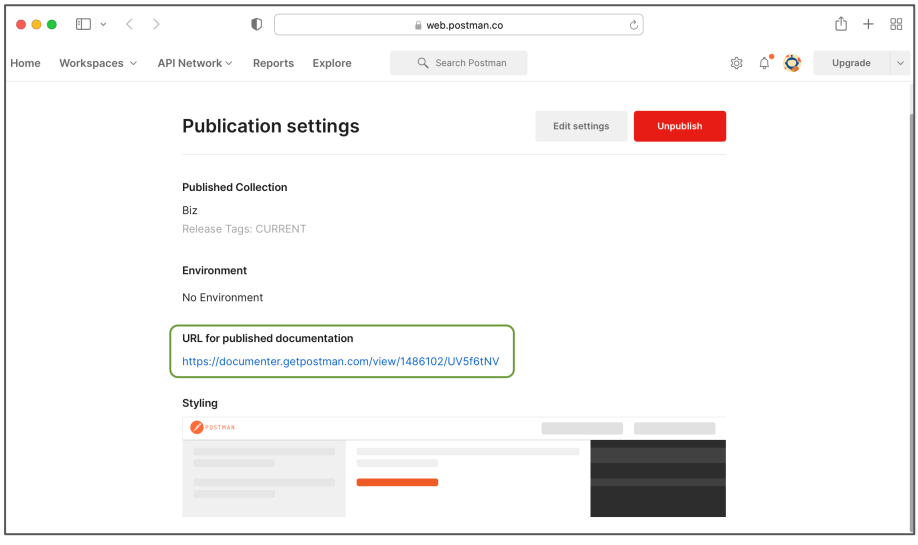

Figure B6.25 Documentation published

One of the best features of the Postman documentation tool is that any updates you make to the documentation are reflected in real-time in the public Web version – your documentation will always be up to date!

Do it now!	Follow the steps above to publish your completed API documentation on the Web. Open a new Web browser and verify that the documentation package is available at the URL provided to you at the final stage (highlighted in Figure B6.25).

B6.3 Further Information

- https://en.wikipedia.org/wiki/API_testing
  API testing overview

- https://www.katalon.com/resources-center/blog/api-testing-tips/
  10 API Testing Tips for Beginners

- https://www.softwaretestingmaterial.com/api-testing/
  API Testing Tutorial

- https://www.youtube.com/watch?v=t5n07Ybz7yI
  The Basics of Using Postman for API Testing (YouTube)

- https://www.guru99.com/postman-tutorial.html
  Postman Tutorial for Beginners with API Testing Example

- https://learning.getpostman.com/docs/postman/scripts/test_scripts
  Postman Learning Centre – Test Scripts

- https://medium.com/aubergine-solutions/api-testing-using-postman-323670c89f6d
  API Testing Using Postman

- https://www.markdownguide.org/
  Markdown Guide

- https://www.markdowntutorial.com/
  Markdown Tutorial

- https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
  Markdown Cheatsheet