**Prerequisites**

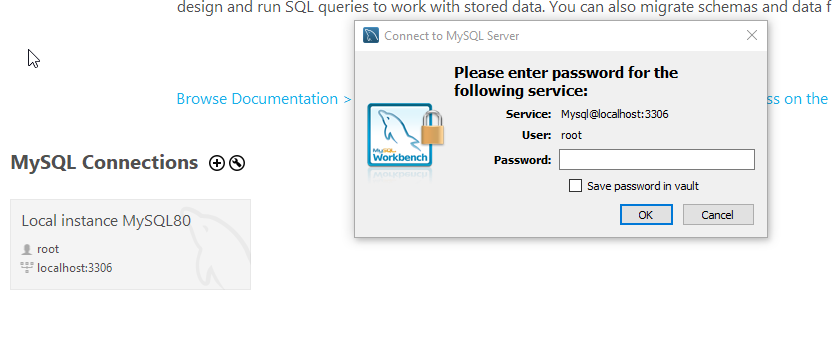
* Source Files from GitHub
* Python 3
* Postman
* MySQL Workbench

**Introduction** The program is a REST API application that can perform HTTP Requests such as *GET, POST, PUT, and DELETE* to send and receive information in JSON*.* It is built using thePython language and the Flask framework. The data being accessed is stored in a database called *api* and a table named *books* which contains information about books with the attributes id, title, author, and description. The user can perform GET requests to access book information which are translated in JSON format and filter results using desired parameters. The user can also perform POST requests to send data by providing a request body in JSON format which will be stored in the database. The PUT request allows the user to make changes from existing data by providing a request body in JSON containing the information of changes to be made with the id attribute used to distinguish the book to be updated. To delete an existing book, the DELETE request will perform such task by providing parameters to filter and delete the desired data. An application called Postman is used to easily perform HTTP requests and test the REST API application while MySQL Workbench is used to develop and visualize the database.

**Database Setup**

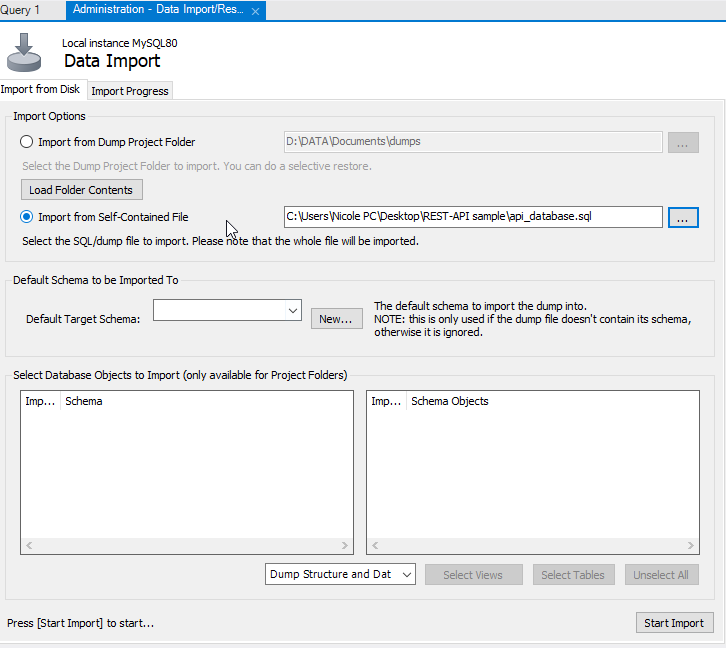
The database can be configured by following the steps:

1. Open the MySQL Workbench and connect to local database.



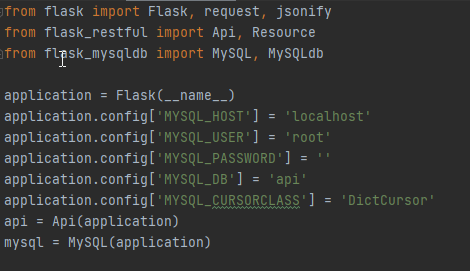
*Figure 1: MySQL Workbench*

1. Once connected, import the *api\_database.sql* file as shown.



*Figure 2: MySQL Workbench import database*

1. On the program, edit and provide the MYSQL\_PASSWORD to connect to the database.

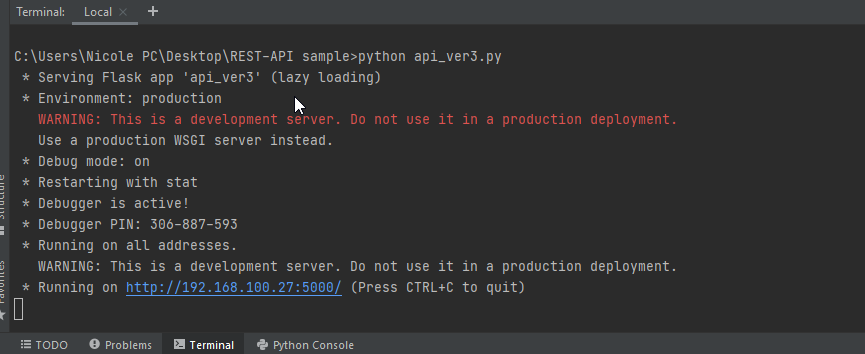


*Figure 3: MySQL Workbench password*

**Execution of the REST API Application**

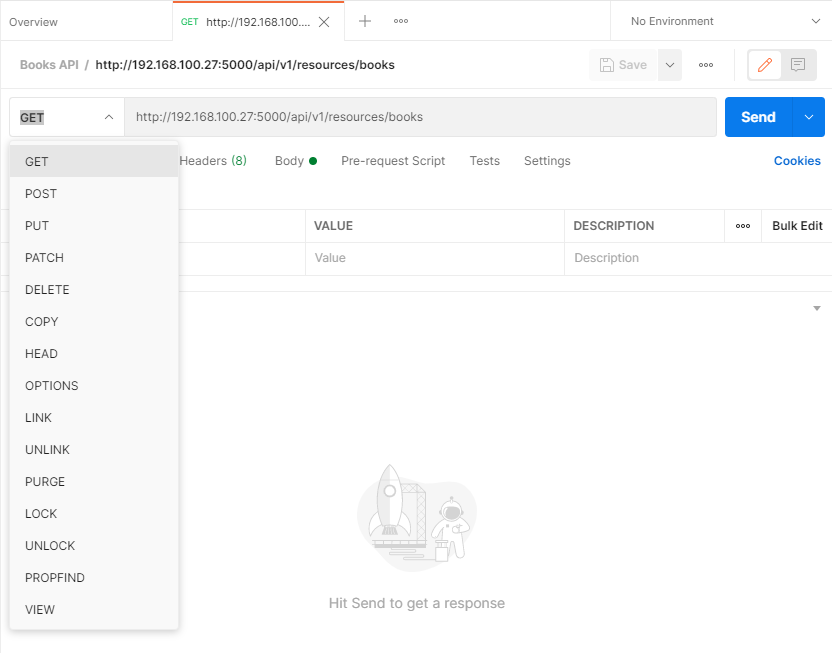
The program can be executed by following the steps:

1. Open the terminal and run the REST API application on a local server.
2. Locate the directory of the application and execute the command *python api\_ver3.py* on the terminal.
3. Once the application is running, open the Postman app and copy the URL to test the application.



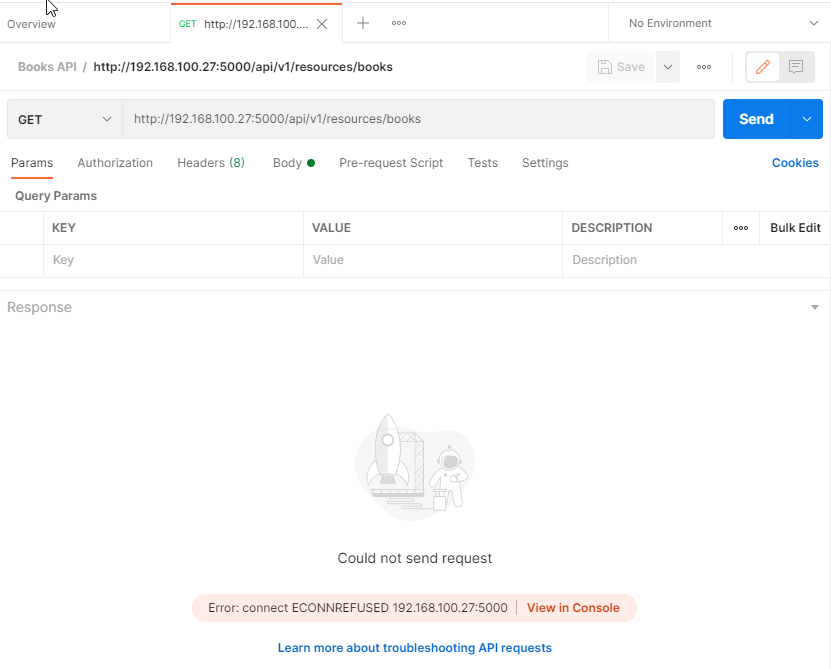
*Figure 2: REST API application execution*

1. On the Postman app, the user can perform various HTTP requests to access data such as *GET, POST, PUT and DELETE.* The data that the user can access is in JSON and is integrated and stored in a database.



*Figure 3: Postman application execution*

1. If the application is not running locally when performing a request, the Postman app will generate an error prompt.

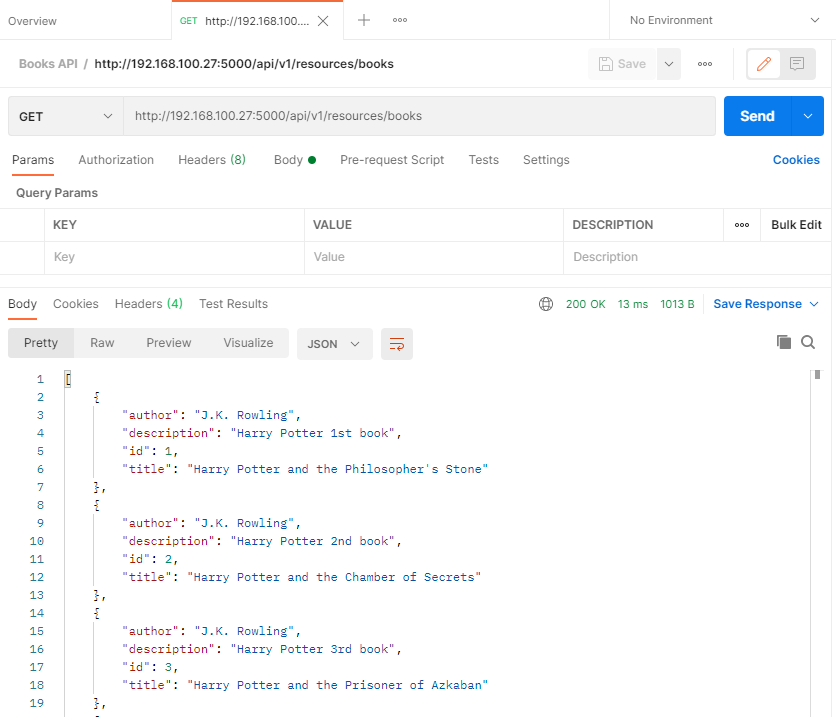


*Figure 4: Postman error connection prompt*

**GET Request**

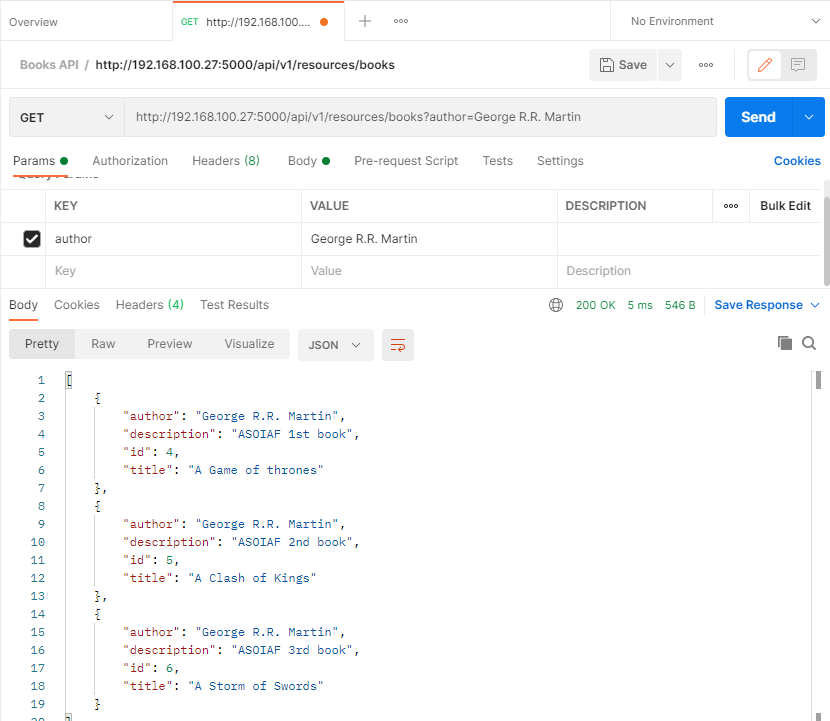
Steps in performing a GET request:

1. On the Postman app, choose the *GET* request and click the *Send* buttonto access and retrieve data. This will retrieve all books stored in the database and the response is translated in JSON format.



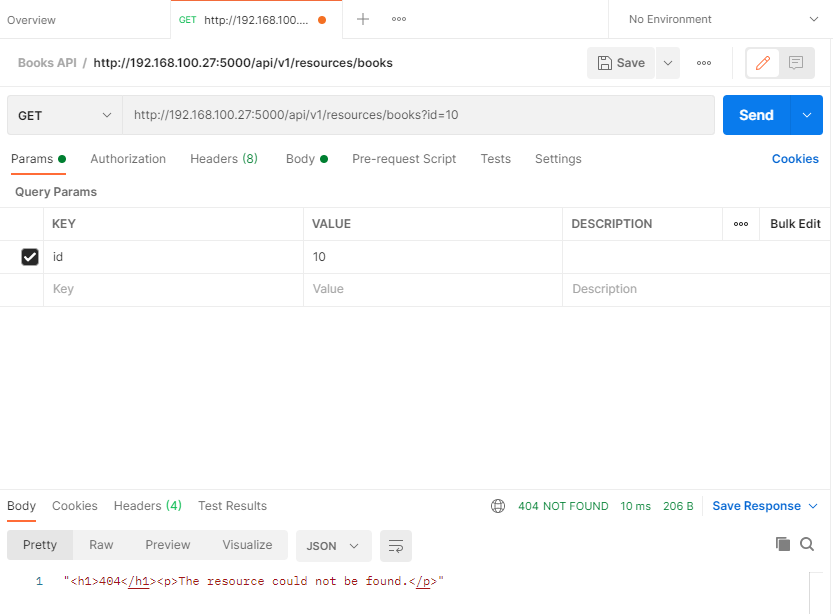
*Figure 5: GET Request*

1. If the user wants to retrieve books that are filtered according by their attributes, parameters should be included. In this case, the user should define the desired key and value to filter the data to be retrieved.



*Figure 6: GET Request with parameters*

1. Should the user provide an invalid or non-existing parameter, an error prompt is displayed.

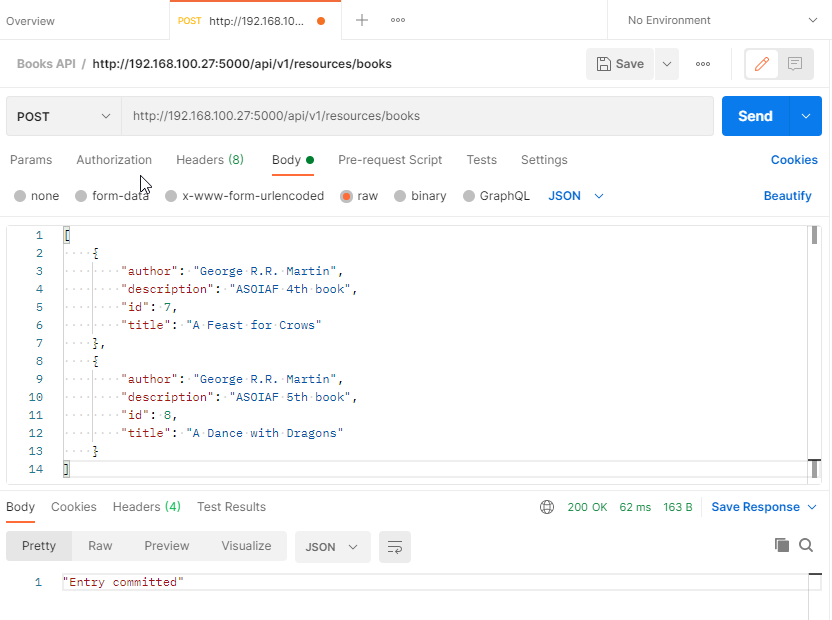


*Figure 7: GET Request invalid parameter*

**POST Request**

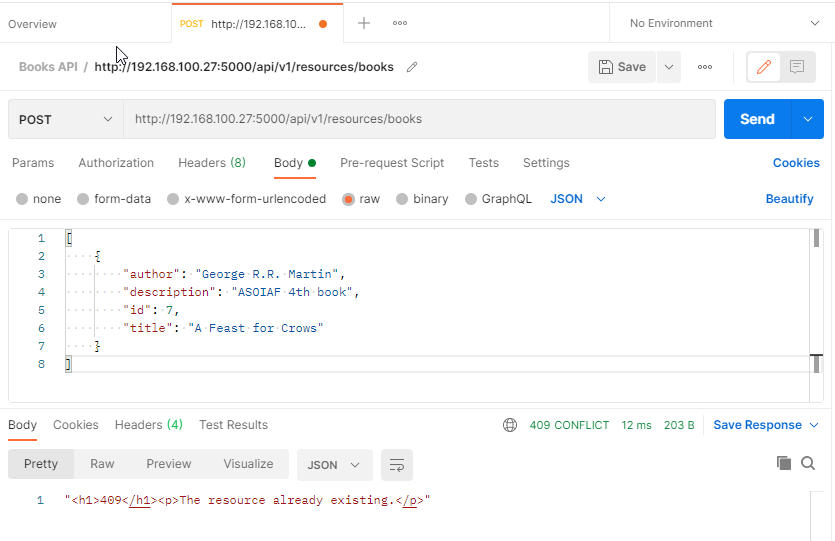
Steps in performing a POST request:

1. On the Postman app, choose the *POST* request and provide the data in the body section to be stored in the database. The data in the request message can include more than one book that the user wants to store which is in JSON format.
2. Once the request is successful, a prompt shall be displayed.



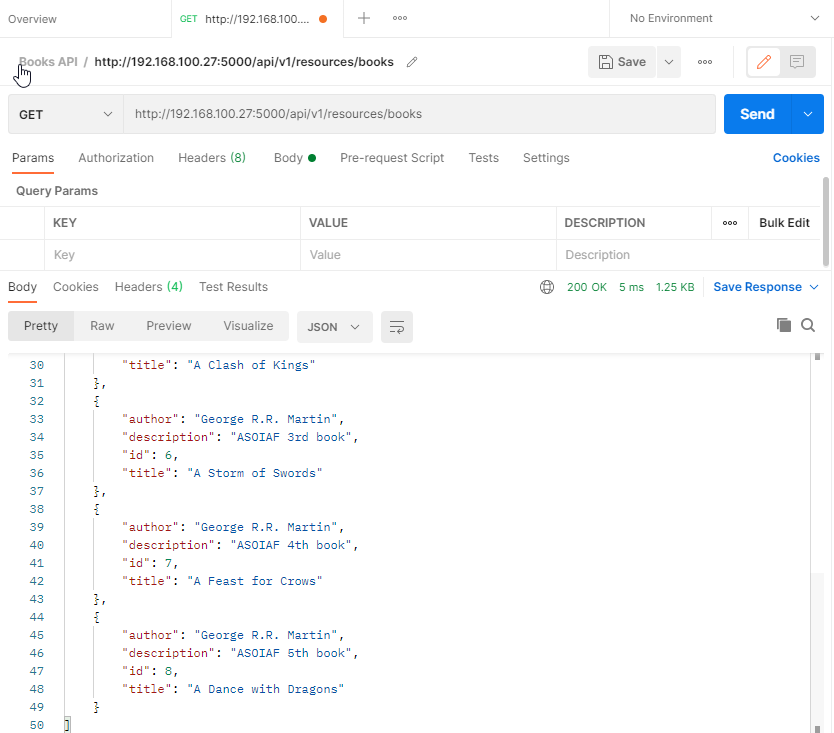
*Figure 8: POST Request*

1. Should the user provide an existing data, an error prompt is displayed.



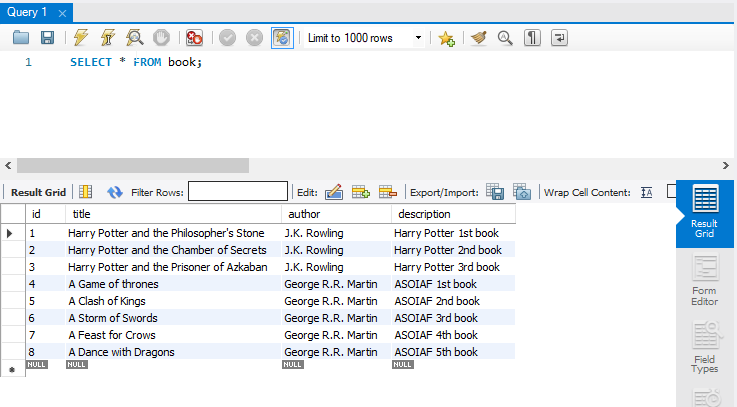
*Figure 9: POST Request existing data*

1. To ensure the request has been made, perform a GET request to verify that the books were accepted and stored in the database. As shown, the books with id 7 and 8 were included from the POST request made earlier.



*Figure 10: Verify POST Request with GET Request*

1. If we check the database in the MySQL Workbench, it is verified that the requested data is stored in the database.

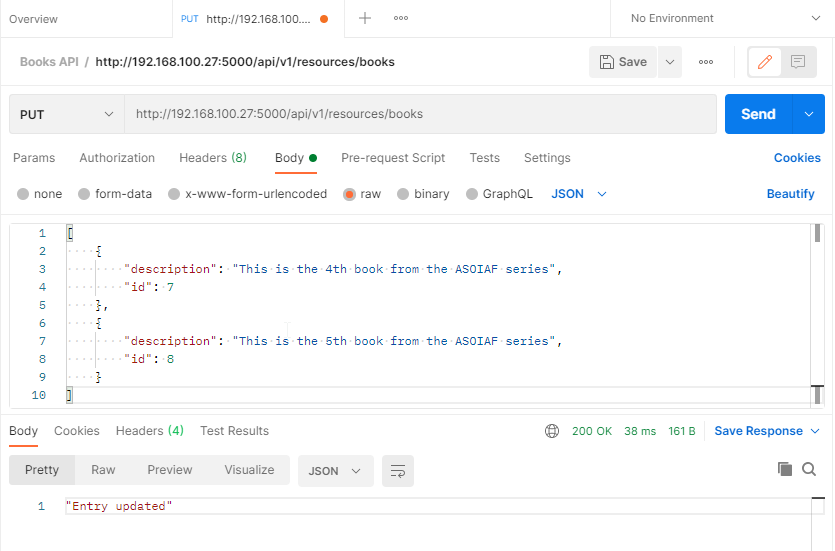


*Figure 11: Verify POST Request with MySQL Workbench*

**PUT Request**

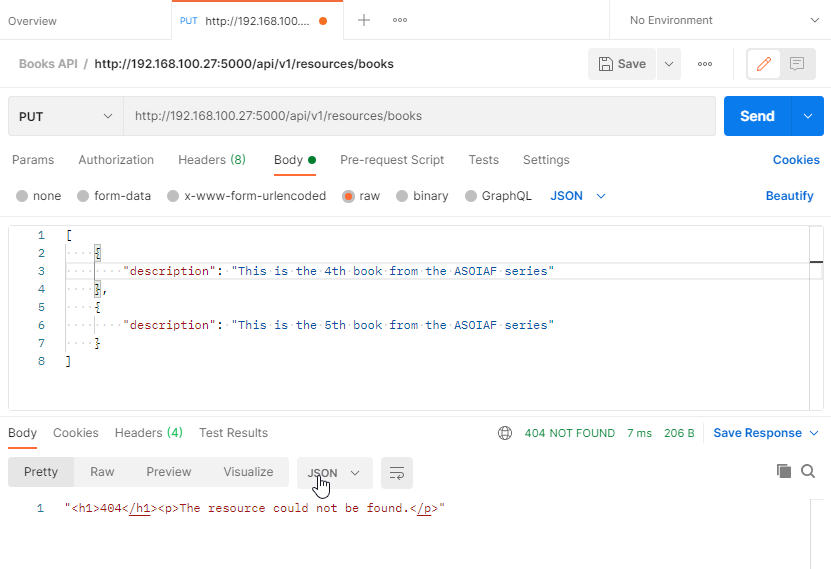
Steps in performing a PUT request:

1. On the Postman app, choose the *PUT* request and provide the data in the body section to update an existing data from the database. The data in the request message can include more than one book that the user wants to update which is in JSON format.
2. Once the request is successful, a prompt shall be displayed.



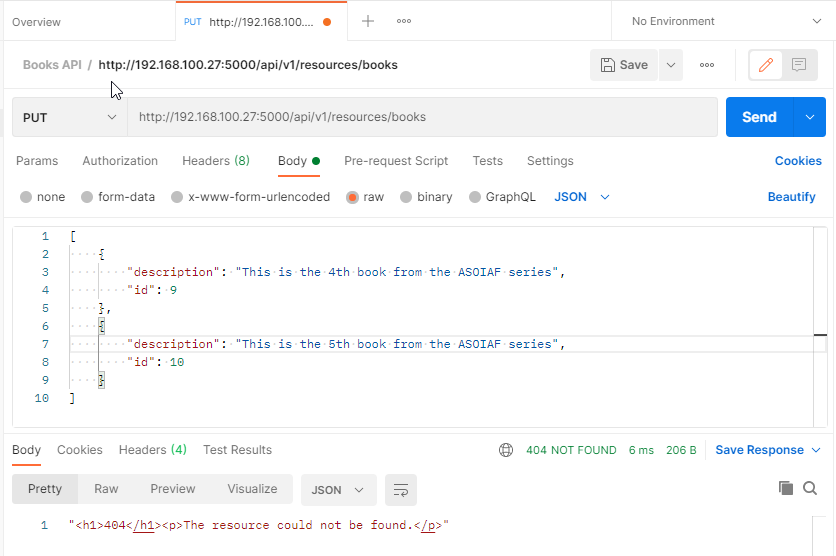
*Figure 12: PUT Request*

1. It should be noted that the id of the book to be updated must be included since the id is the parameter used for identifying which book shall be updated.
2. If the id is not included, an error prompt shall be displayed.



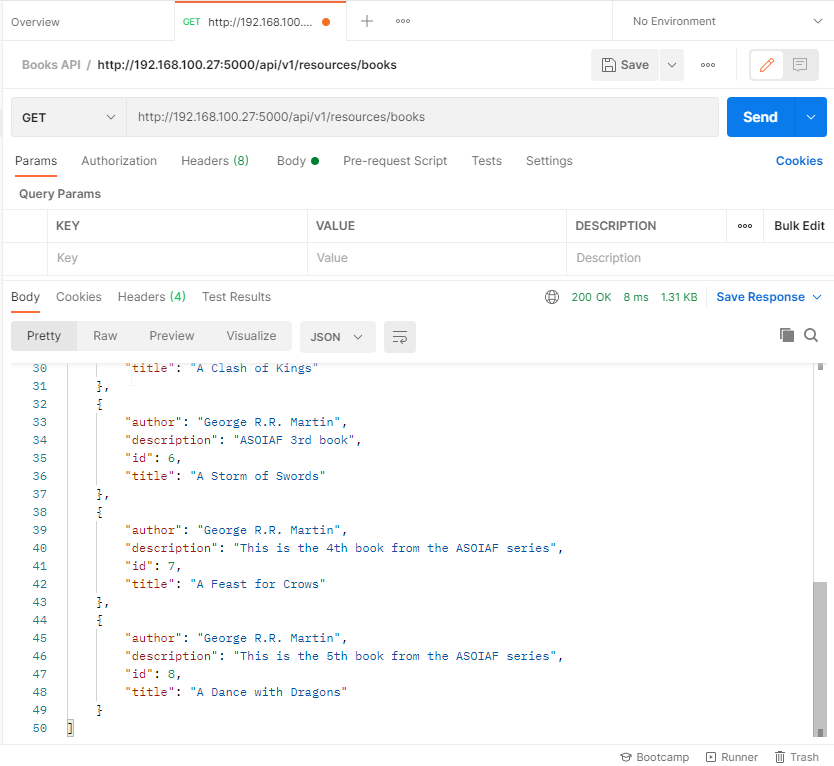
*Figure 13: PUT Request invalid id*

1. Furthermore, should the user include an invalid or non-existing id, an error prompt shall also be displayed.



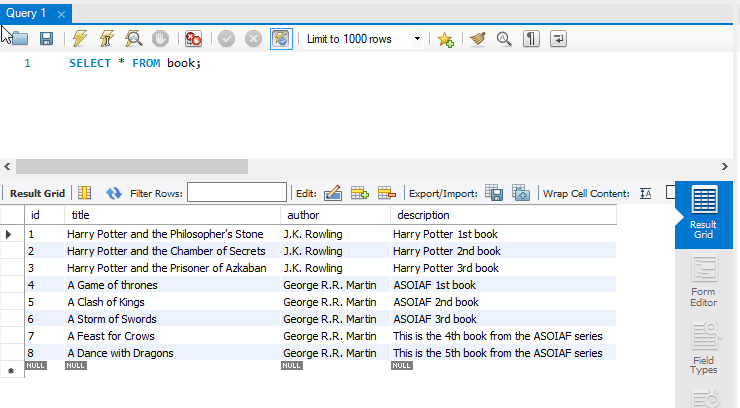
*Figure 14: PUT Request non-existing id*

1. To ensure the changes made on the data, perform a GET request to verify that the books were updated. As shown, the description for the books with id 7 and 8 were changed from the updates made earlier.



*Figure 15: Verify PUT Request with GET Request*

1. If we check the database in the MySQL Workbench, it is verified that the changes made in the data are reflected in the database.

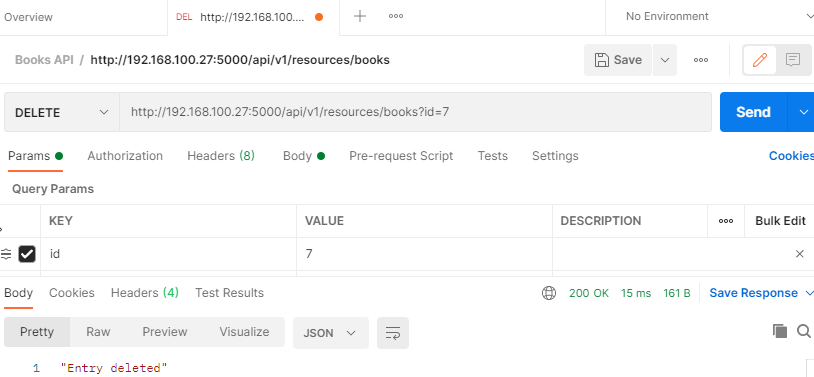


*Figure 16: Verify PUT Request with MySQL Workbench*

**DELETE Request**

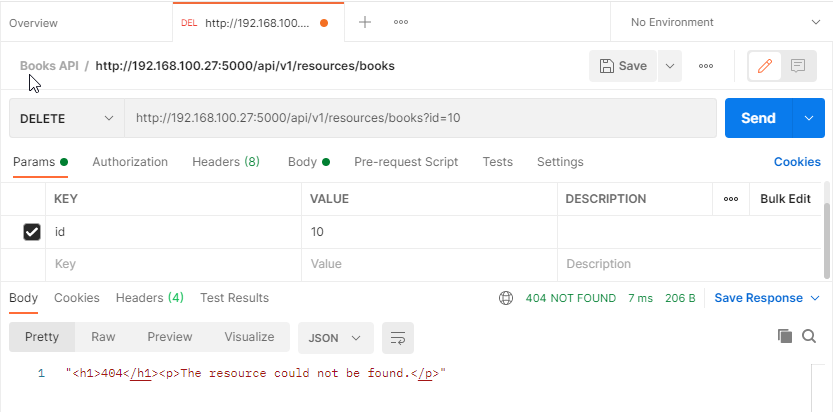
Steps in performing a DELETE request:

1. On the Postman app, choose the *DELETE* request and provide the parameters to filter the desired books to be deleted from the database.
2. Once the request is successful, a prompt shall be displayed.



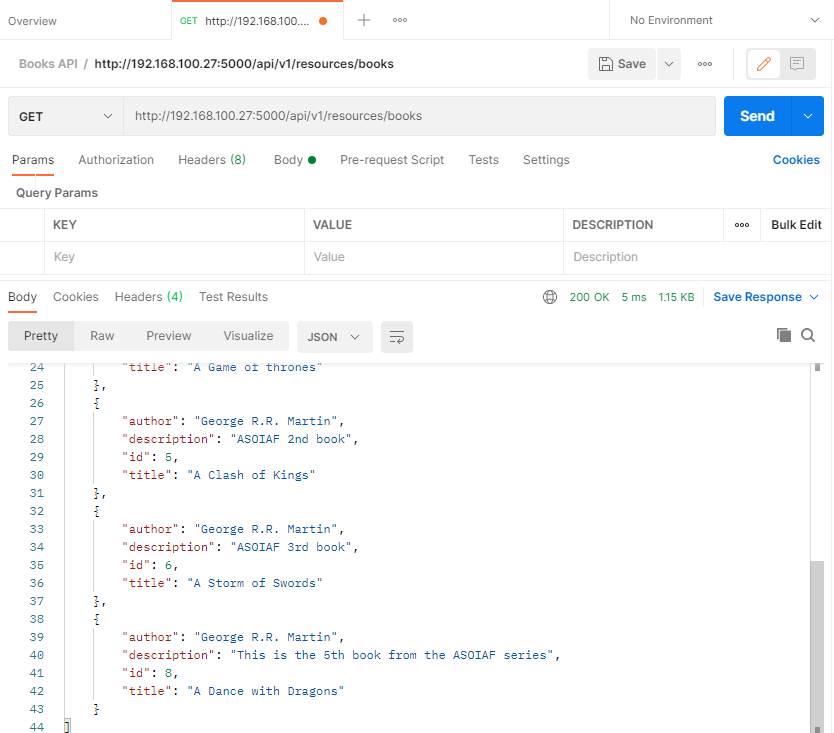
*Figure 17: DELETE Request*

1. Should the user provide an invalid or non-existing parameter, an error prompt is displayed.



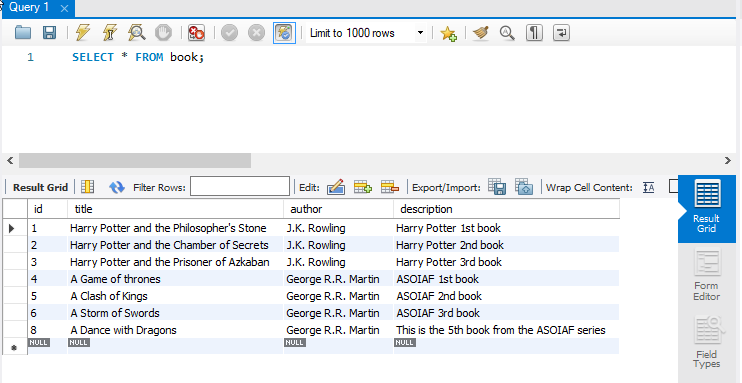
*Figure 18: DELETE Request invalid parameter*

1. To ensure the changes made on the data, perform a GET request to verify that the books were deleted. As shown, the book with id 7 was deleted from the request made earlier.



*Figure 19: Verify DELETE Request with POST Request*

1. If we check the database in the MySQL Workbench, it is verified that the data is deleted in the database.



*Figure 20: Verify DELETE Request with MySQL Workbench*