

## Environment:

### OS Support:

- Windows 10

### IDE:

- PyCharm 64-Bit
  - Can be found at:  
<https://www.jetbrains.com/pycharm/download/#section=windows>

### Python Version:

- Python 3.8.4
  - Can be found at: <https://www.python.org/downloads/>

### Testing Apparatus:

- Postman

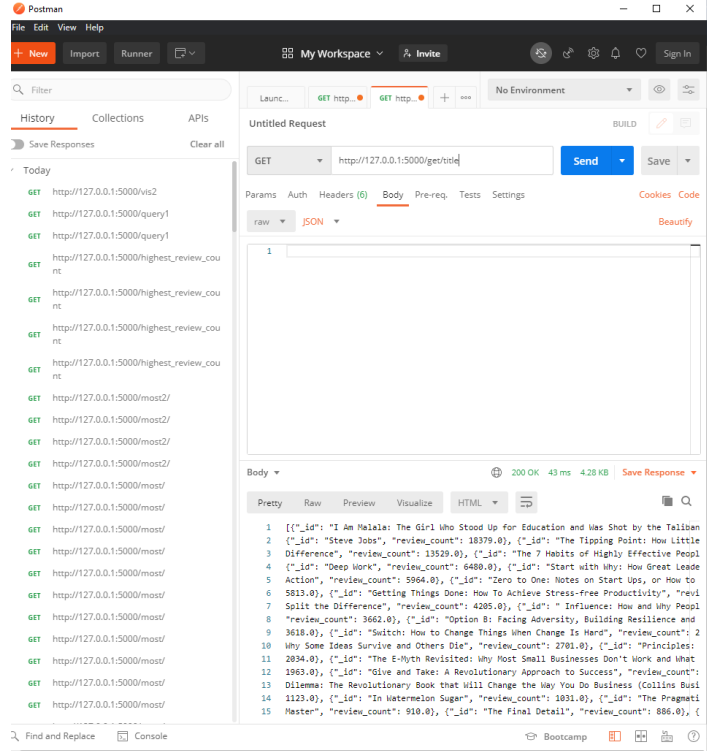
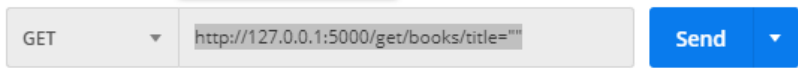
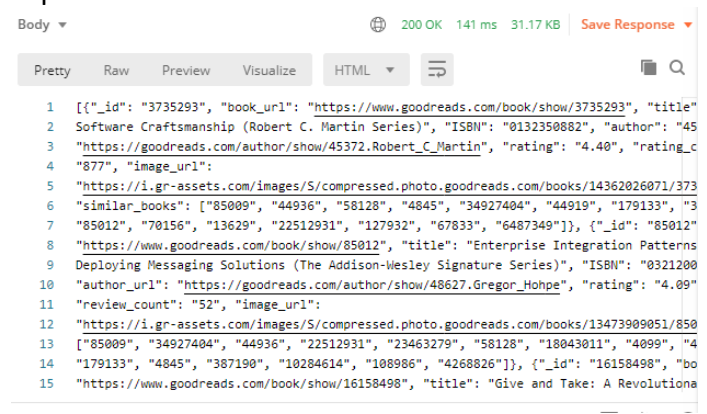
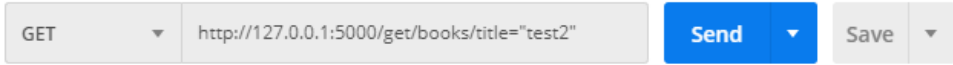
Newer versions of Python may work but downgrade if any errors are encountered.

### Modules Needed:

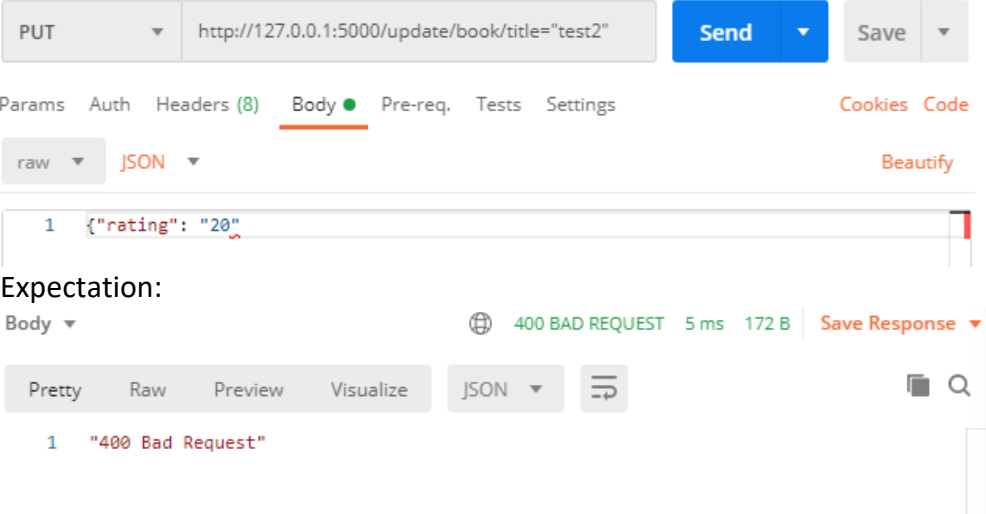
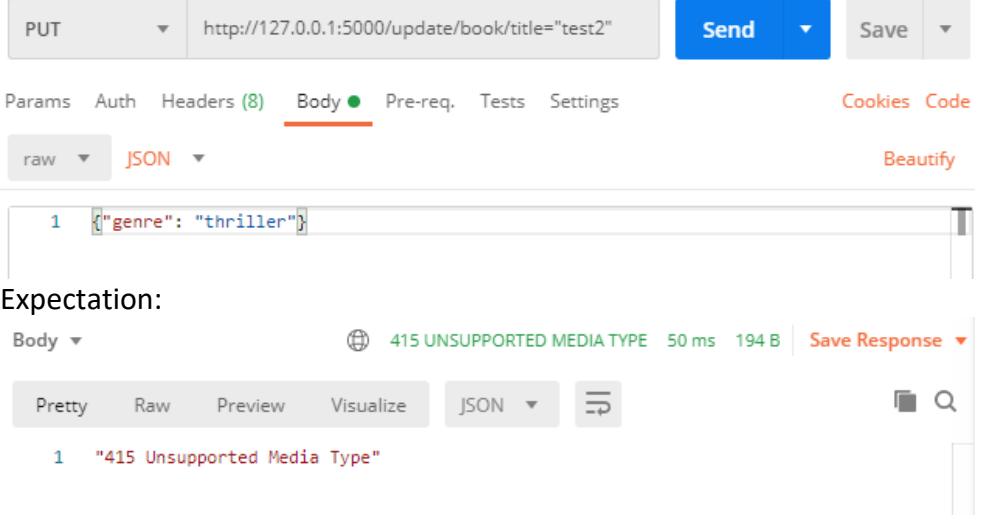
- Django
- Requests
- Flask






Simply use pip-install if you do not have these modules

# Manual Tests:

Image	Test
 <p>The screenshot shows the Postman application interface. On the left, there's a sidebar with 'History' and 'Collections' tabs. The 'History' tab is active, showing a list of recent requests. The main area displays an 'Untitled Request' with a GET method and the URL 'http://127.0.0.1:5000/get/title'. The 'Body' tab is selected, showing a JSON response. The response is a list of 15 book objects, each containing fields like '_id', 'title', 'author', 'rating', and 'image_url'. The status bar at the bottom indicates a '200 OK' response with a response time of 43 ms and a size of 4.28 KB.</p>	<p>To use postman select the functionality (GET, POST, PUT,DELETE), input the url, and if needed input the json string.</p>
<p><b>Input:</b></p>  <p><b>Expectation:</b></p>  <p>The screenshot shows the Postman interface for a GET request. The method is 'GET' and the URL is 'http://127.0.0.1:5000/get/books/title=""'. The 'Send' button is visible. Below the input field, the 'Body' tab is selected, showing a JSON response. The response is a list of 15 book objects, each containing fields like '_id', 'title', 'author', 'rating', and 'image_url'. The status bar at the bottom indicates a '200 OK' response with a response time of 141 ms and a size of 31.17 KB.</p>	<p>First, we will test a simple get request. I left the search parameter as a 0 length string since I am trying to retrieve all the books in the database. The program correctly returns the books and gives a “200 OK” response.</p>
<p><b>Input:</b></p> 	<p>Now we can try to get a specific book. With the</p>

<p>Expectation</p> <p>Body ▾</p> <p>200 OK 50 ms 719 B Save Response ▾</p> <p>Pretty Raw Preview Visualize HTML ▾</p> <pre> 1 [{"_id": "5678", "book_url": "https://www.goodreads.com/book/show/85012", "title": "Te 2 "author": "48627", "author_url": "https://goodreads.com/author/show/48627.Gregor_Hohpe 3 "rating_count": "5", "review_count": "52", "image_url": 4 "https://i.gr-assets.com/images/S/compressed.photo.goodreads.com/books/13473909051/850 5 ["85009", "34927404", "44936", "22512931", "23463279", "58128", "18043011", "4099", "4 6 "179133", "4845", "387190", "10284614", "108986", "4268826"]}]] </pre>	<p>search parameter only one book is found. The "200 OK" response is once again received.</p>
<p>Input:</p> <p>PUT http://127.0.0.1:5000/update/book/title="test2" Send Save ▾</p> <p>Params Auth Headers (8) Body ● Pre-req. Tests Settings Cookies Code</p> <p>raw ▾ JSON ▾ Beautify</p> <pre> 1 {"rating": "20"} </pre> <p>Expectation:</p> <p>Body ▾</p> <p>200 OK 156 ms 153 B Save Response ▾</p> <p>Pretty Raw Preview Visualize JSON ▾</p> <pre> 1 "200 OK" </pre>	<p>Now let's update the rating of the book with a put request. The json object is passed to the api and the book is updated. The response shows everything went correctly.</p>
<p>Input:</p> <p>GET http://127.0.0.1:5000/get/books/title="test2" Send Save ▾</p> <p>Expectation:</p> <p>Body ▾</p> <p>200 OK 55 ms 717 B Save Response ▾</p> <p>Pretty Raw Preview Visualize HTML ▾</p> <pre> 1 ww.goodreads.com/book/show/85012", "title": "Test2", "ISBN": "0321200683", 2 ://goodreads.com/author/show/48627.Gregor_Hohpe", "rating": "20", "rating_count": 3 4 essed.photo.goodreads.com/books/13473909051/85012.jpg", "similar_books": 5 1", "23463279", "58128", "18043011", "4099", "44919", "1069827", "127932", 6 "108986", "4268826"]}]] </pre>	<p>If we get the same book again, we can see that the rating is changed.</p>
<p>Input:</p>	<p>If we try and change the rating using wrong json syntax, the api</p>

 <p>PUT <span>▼</span> http://127.0.0.1:5000/update/book/title="test2" <span>Send</span> <span>Save</span> <span>▼</span></p> <p>Params Auth Headers (8) <u>Body</u> ● Pre-req. Tests Settings Cookies Code</p> <p>raw <span>▼</span> JSON <span>▼</span> Beautify</p> <pre>1 {"rating": "20"}</pre> <p>Expectation:</p> <p>Body <span>▼</span> 400 BAD REQUEST 5 ms 172 B <span>Save Response</span> <span>▼</span></p> <p>Pretty Raw Preview Visualize JSON <span>▼</span> <span>↺</span></p> <pre>1 "400 Bad Request"</pre>	<p>responds with a “400 Bad Request” error.</p>
<p>Input:</p>  <p>PUT <span>▼</span> http://127.0.0.1:5000/update/book/title="test2" <span>Send</span> <span>Save</span> <span>▼</span></p> <p>Params Auth Headers (8) <u>Body</u> ● Pre-req. Tests Settings Cookies Code</p> <p>raw <span>▼</span> JSON <span>▼</span> Beautify</p> <pre>1 {"genre": "thriller"}</pre> <p>Expectation:</p> <p>Body <span>▼</span> 415 UNSUPPORTED MEDIA TYPE 50 ms 194 B <span>Save Response</span> <span>▼</span></p> <p>Pretty Raw Preview Visualize JSON <span>▼</span> <span>↺</span></p> <pre>1 "415 Unsupported Media Type"</pre>	<p>If we try and change an attribute that is not present, the api responds with a “415 Unsupported Media Type Error” error.</p>
<p>Input:</p>	<p>To add a book, use a post request and input the attributes of the book as a json object. The api responds with a “201 Created” response indicating that the adding of</p>

<p>Untitled Request <span>BUILD</span>  </p> <p>POST <span>▼</span> http://127.0.0.1:5000/add/books <span>Send</span> <span>Save</span> <span>▼</span></p> <p>Params Auth Headers (8) <u>Body</u> ● Pre-req. Tests Settings <span>Cookies</span> <span>Code</span></p> <p>raw <span>▼</span> JSON <span>▼</span> <span>Beautify</span></p> <pre> 1 [{"_id": "7889", "book_url": "https://www.goodreads.com/book/show/85012", "title": 2   "man_test", "ISBN": "0321200683", 3   "author": "48627", "author_url": "https://goodreads.com/author/show/48627. 4     Gregor_Hohpe", "rating": "20", "rating_count": 5     "5", "review_count": "52", "image_url": 6     "https://i.gr-assets.com/images/S/compressed.photo.goodreads.com/books/13473909051/       85012.jpg", "similar_books": 7     ["85009", "34927404", "44936", "22512931", "23463279", "58128", "18043011", 8       "4099", "44919", "1069827", "127932", 9       "179133", "4845", "387190", "10284614", "108986", "4268826"]} </pre> <p>Expectation:</p> <p>Body <span>▼</span> <span>201 CREATED</span> <span>445 ms</span> <span>164 B</span> <span>Save Response</span> <span>▼</span></p> <p>Pretty Raw Preview Visualize JSON <span>▼</span> </p> <pre> 1 "201 CREATED" </pre>	<p>the book was successful.</p>
<p>Input:</p> <p>POST <span>▼</span> http://127.0.0.1:5000/delete/book/man_test <span>Send</span> <span>Save</span> <span>▼</span></p> <p>Expectation:</p> <p>Body <span>▼</span> <span>200 OK</span> <span>56 ms</span> <span>153 B</span> <span>Save Response</span> <span>▼</span></p> <p>Pretty Raw Preview Visualize JSON <span>▼</span> </p> <pre> 1 "200 OK" </pre>	<p>We can also delete a book. A “200 OK” response indicates the book has been deleted successfully.</p>
<p>Input:</p> <p>GET <span>▼</span> http://127.0.0.1:5000/query1 <span>Send</span> <span>Save</span> <span>▼</span></p> <p>Expectation:</p> <p>Body <span>▼</span> <span>200 OK</span> <span>80 ms</span> <span>644 B</span> <span>Save Response</span> <span>▼</span></p> <p>Pretty Raw Preview Visualize HTML <span>▼</span> </p> <pre> 1 [[{"_id": "45372", "name": "Robert C. Martin", "author_url": "https://www.goodreads.co 2   "4.34", "rating_count": "27019", "review_count": "1766", "image_url": 3   "https://images.gr-assets.com/authors/1490470967p5/45372.jpg", "related_authors": ["28 4     25215", "32731", "48622", "48660", "104368", "7127583"], "author_books": ["3735293", 5     "84983", "45280021", "13136186", "873375", "79774", "25936819"]}]] </pre>	<p>The first query should return only one author, who has the greatest number of books in the book collection.</p>

Input:

GET	http://127.0.0.1:5000/vis1	Send	Save
-----	----------------------------	------	------

Expectation:

Body	200 OK 43 ms 1.36 KB	Save Response
Pretty Raw Preview Visualize		
<pre>[{"_id": "Chris Voss", "rating": 4.4}, {"_id": "Robert C. Martin", "rating": 4.34}, {"_id": "Damien Lewis", "rating": 4.25}, {"_id": "Ben Horowitz", "rating": 4.23}, {"_id": "Andy Hunt", "rating": 4.22}, {"_id": "Erich Gamma", "rating": 4.2}, {"_id": "Peter Thiel", "rating": 4.17}, {"_id": "Robert B. Cialdini", "rating": 4.16}, {"_id": "Cal Newport", "rating": 4.15}, {"_id": "Ray Dalio", "rating": 4.15}, {"_id": "Malala Yousafzai", "rating": 4.14}, {"_id": "Walter Isaacson", "rating": 4.12}, {"_id": "Stephen R. Covey", "rating": 4.11}, {"_id": "Gregor Hohpe", "rating": 4.11}, {"_id": "Kent Beck", "rating": 4.1}, {"_id": "Simon Sinek", "rating": 4.09}, {"_id": "James C. Collins", "rating": 4.08}, {"_id": "Clayton M. Christensen", "rating": 4.05}, {"_id": "Michael C. Feathers", "rating": 4.03}, {"_id": "Michael E. Gerber", "rating": 4.02}, {"_id": "Malcolm Gladwell", "rating": 4.01}, {"_id": "Chip Heath", "rating": 4.0}, {"_id": "Josh Kaufman", "rating": 3.99}, {"_id": "Richard Brautigan", "rating": 3.98}, {"_id": "David Allen", "rating": 3.98}, {"_id": "Adam M. Grant", "rating": 3.95}, {"_id": "Marcus Buckingham", "rating": 3.95}, {"_id": "Sheryl Sandberg", "rating": 3.94}, {"_id": "Jack Welch", "rating": 3.83}]</pre>		

The first visualization should return the authors in order of their ratings. Make sure that the authors are in the correct order.

Input:

GET	http://127.0.0.1:5000/vis2	Send	Save
-----	----------------------------	------	------

Expectation:

<pre>[{"_id": "I Am Malala: The Girl Who Stood Up for Education and Was Shot by the Taliban", "review_count": 20651.0}, {"_id": "Steve Jobs", "review_count": 18379.0}, {"_id": "The Tipping Point: How Little Things Can Make a Big Difference", "review_count": 13529.0}, {"_id": "The 7 Habits of Highly Effective People", "review_count": 10341.0}, {"_id": "Deep Work", "review_count": 6480.0}, {"_id": "Start with Why: How Great Leaders Inspire Everyone to Take Action", "review_count": 5964.0}, {"_id": "Zero to One: Notes on Start Ups, or How to Build the Future", "review_count": 5813.0}, {"_id": "Getting Things Done: How To Achieve Stress-free Productivity", "review_count": 5055.0}, {"_id": "Never Split the Difference", "review_count": 4205.0}, {"_id": "Influence: How and Why People Agree to Things", "review_count": 3662.0}, {"_id": "Option B: Facing Adversity, Building Resilience and Finding Joy", "review_count": 3618.0}, {"_id": "Switch: How to Change Things When Change Is Hard", "review_count": 2834.0}, {"_id": "Made to Stick: Why Some Ideas Survive and Others Die", "review_count": 2701.0}, {"_id": "Principles: Life and Work", "review_count": 2034.0}, {"_id": "The E-Myth Revisited: Why Most Small Businesses Don't Work and What"</pre>		
--	--	--

The second visualization should return the books in order of their review\_counts . Make sure that the books are in the correct order.