Manual Training

Qazi Zain

What is Software testing?

It's a crucial part of the sdlc phase where software is evaluated to ensure that it meets specified requirements and bug free.

Why Software testing important?

To ensure that the software meets the user expectations.
To ensure the Quality of product.
To ensure that the software is bug free.

Functional Testing:

- **Black box testing:** it's a type of testing in which the user doesn't have to know the internal structure of the system.Also known as functional testing.

  Test behavior with respect to the user point of view.

  Real world Scenario: When registering for a driving license on an app, you should test the following user scenario: Enter an age that is less than 18 to verify whether the app correctly displays a message indicating that you are underage and thus ineligible for a license.

  Techniques:
  - Decision table
  - BVA (Boundary Value Analysis)

  Perform by (Tester/Developer/User)

- **White box testing:** it's a type of testing in which the user has to know the internal structure of the system. Also known as non functional testing.

  Test behavior with respect to the developers point of view.

  Real world Scenario: When we have the discount calculation function and we have to To test that the functionality is working properly or not so we test the control and data Flow in the correct structural manner.

  Techniques:
  - Control flow
  - Data flow
  - Branch flow

  Perform by (Tester/Developer)

- **Grey box testing:** it's a type of testing , we can say it's a combination of black box and white box testing, best suitable for web app testing because on the web we need to understand both how the software works & how different parts of the system interact with each other. (partial knowledge of the internal system).

Real world Scenario: Verify that users can successfully complete a transaction using the payment gateway.

Techniques:
- Regression testing
- Pattern testing

Perform by (Tester/Developer/User)

Non functional testing:

Performance testing:

- **Load testing:** put load in a range on a system of users to test the response time and stability.

  Tools: JMETER

- **Stress Testing:** put load out of the range on a system of users to test the response time and stability.

  Tools: JMETER

- **Volume Testing:** put the load on the database . Suppose we insert 10 gb of data in the database to check its capability.

  Tools: JMETER

Compatibility Testing: Also known as cross browser testing done to ensure compatibility of web across different browsers and web.

Tools:
- By Creating a selenium framework.
- Browsershots.
- comparium.

Security testing: in this we test the system to find the vulnerabilities
Present in a system and to fix it.

Tools:
- Burp Suite.

Manual Training

Qazi Zain

Unit testing: To test an individual unit of a system Suppose we have a calculator program and there is a sum module in it, if we test sum in a calculator program that will be known as unit test.

Tools: It may vary according to the language.

- Junit.
- Test Png
- Jest.

Smoke testing: To test the basic functionalities of a system when the build is unstable or in its initial stage, and before performing further testing. Like I am checking the web and its login functionality isnt working so I will reject the build. (can be done by dev can be done by tester.

Tools:

- Appium (for mobile)
- Selenium(for web)
- Postman(for Api)

Sanity testing: To test the particular component that was added to an existing system, that added component is working properly within the existing system. Like I have a calculator program, I added a new feature to find derivatives in it. I will check that this added feature is working properly or not.

Also we do it after bug fixing how the app is working.

Tools:

- Selenium
- Appium
- Postman

Integration testing: In this we test the two or more units that how they are working after integrating or combining together.

Like in shopping web login page, cart page , dashboard page all are individual components so in integration testing we will ensure that these components are interacting correctly.

Tools:

- Selenium
- Appium
- Cypress
- Postman

Manual Training

Qazi Zain

API Testing: In API Testing We usually test API , like HTTP errors.

In Todo list users have a rest api to manage his daily tasks , the api including creating task , retrieving task, update and deleting task so in api testing we have to ensure that these endpoints are working correctly.

Tools:

- RestAssured
- Soap ui
- postman

UI Testing: In UI testing we are not looking for the functionality of ui in this we usually looking for how ui is looking is it according to client requirement?.

Tools:

- Selenium
- Cypress
- Appium

System Testing: In this we evaluate the complete integrated system to ensure that it meets the specified requirements and also make sure the overall system Quality.

Like we have a banking application we have to ensure all the functionalities working together.

Regression Testing: in this we retest the already tested system to ensure that newly added functionality or a bug fix should not affect the existing system.

Real world example : Suppose we have an e commerce website and the team added new discount calculate feature and fix some existing bugs now team will test the whole website again that newly make changes aren't affect existing system.

Tools:

- **Selenium**
- **Appium**
- **postman**

Manual Training

Qazi Zain

Alpha testing: Done after system testing , it is usually done as black box as well as white box technique. Instantly fixes the bugs tested by testers and internal employees.

Beta testing: Done after alpha testing , it is usually done as black box technique, in which feedbacks are collected and improvisation in the new version is done by an external user , suppose we upload a beta version for some users to collect feedback.

how many levels of. testing are there and what are they?

There are 4 levels of testing:

1. Unit testing.
2. Integration testing.
3. System Testing.
4. Acceptance Testing (Alpha , Beta).

Bug Life Cycle: This life cycle helps in tracking and managing bugs systematically, ensuring that issues are resolved efficiently and effectively.

Defect : Deviation between actual and expected results.

Bug Life Cycle:

1. **New:** Tester finds a bug, makes it on jira and assigns it as (status : New).
2. **Assign:** Once the bug is posted , it will be approved by the testing lead that is assigned to the development team.
3. **Open:** The bug is analyzed by the development team and works on to fix it.
4. **Fixed:** After necessary changes made by developers the status assigns it fix.
5. **Pending retest:** Developers fix the bug and transfer the fix code to the tester team and it has to be retested that why status assigns pending retest.
6. **Retest:** Now testers start retesting the code to check whether the bug is fixed by the developer or not.
7. **Reopen:** If in the retesting phase the tester finds the bug after fixing he assigns the status as re open.
8. **Verified:** If after retesting the tester finds the bug is fixed he will assign status verified.

9. **Close:** If the developer fixes the bug and the testing team does not find the bug persist then the testing team assigns the status that bug is closed.

Difference between priority and severity ?

Priority: Urgency to fix a bug. (How Fast we have to work on bugs based on the impact).
Decide by Product manager.
Decided Status can be changed.
Example: (like we found a different status bug now we give priority that high severe bug should resolve first).

Severity: impact of a failure caused by a bug.
Example : (The Bug in phone pay is the send button isn't working which impacts major
Decide by Test Engineer.
Decided Status Doesn't change.
functionality so it means its most impactful and most severe.).

Difference between use case and test case:

USE Case	Test Case
Defies how actor interact with system	Define how to test the system to check it works correctly
Focus on overall functionality of system and user experience.	Focus on very specific functionality and how to fix it.
Develop during requirement gathering phase	Develop during the testing phase to ensure that user requirements meet their expectations.

Types of Test Cases:

8 Types of Test Cases in Software Testing

Given below 8 types of main test cases in software testing.

- **Functionality Test Case**
- **User Interface Test Case**
- **Performance Test Case**
- **Integration Test Case**
- **Usability Test Case**
- **Database Test Case**
- **Security Test Case**
- **User Acceptance Test Case**

Test Case Design Technique: Methods used to design test cases.

1. BVA: Test the value of design test cases for boundary values like input field accept the range of 1-10 so suppose 1 is A and 10 is B so design test cases for A-1,A,A+1 which are 0,1,2 and B-1,B,B+1 which are 9,10,11.

2. Equivalent class partitioning: let suppose we are checking the value i 1-500 so in equivalence class partitioning we will create different classes like 1-100, 101,200,201-300,301-400,401-500,501-600. From these classes we enter any single digit only and the whole class considers the same result in which class does exist.

3. Error Guessing: Based on user experience by guessing and testing the system.

Qazi Zain

What is the Test Scenario?

It is a high level description of what we need to test.

Example:

Test Scenario: Validate User can successfully Login .

Now it could vary. His scenario might include various test cases to cover different aspects like entering valid data, handling errors, and confirming registration.

Types :

Positive Scenario :   User Registration Success

Negative Scenario: User Registration Failure with Invalid Email

Qazi Zain

How to create a test case and for what purpose they are creating.

To detail a specific step to verify a particular aspect of a software application.

Writing pattern

Test Case Id	TC_LG_00
Test Case description	Example verify user login successfully
Pre Condition	User must present on login Screen
Steps	1. Enter Email format(abc@gmail.com) 2. Enter Password 3. Click Login Button
Expected Output	Login Successfully
Postcondition	Redirect to Dashboard

Manual Training

Qazi Zain

How to create Use Cases and what they Are Created for ?

Use Cases Are Created to map how the user is interacting with the system.(What the system should do from the user point of view.

Use case id	
Use Case Description	
Actors	
Pre condition	
Action	
Post condition	

What is jira? And it is used for bug tracking?

Jira is a tool which is used for project management and bug tracking.

How to create a bug on jira?

Click on create. Then select project , then select type (bug), fill summary what bug is , fill, describe detail with steps, actual output , expected output., decide the priority of bug., add image , assigne the member., Epic link , sprint also. Then click on create.

What are defects, bugs , errors, exceptions.

1. **Defect:** The imperfection in a software to perform un intended behavior.

   Example: A feature intended to sort user names alphabetically instead sorts them randomly.

2. **Bugs:** Specific type of Defect which means the error present in a code.

   Example: A typo in a function name that prevents the function from being called properly.

3. **Error:** Generally refers to mistakes made by humans.

   Example: A syntax error where a missing parenthesis causes the code to fail to compile.

4. **Exceptions:** Is an event which occurs at run time.

   Example: Trying to open a file that does not exist triggers a `FileNotFoundException`.

Difference between SDLC & STLC:

SDLC: Covers the entire software development process from requirements gathering to maintenance.

STLC: Focuses solely on the testing activities within the software development lifecycle.

Manual Training

Qazi Zain

SDLC Phases:

Requirements Gathering and Analysis: Collecting and analyzing the needs of stakeholders to define what the software should do.

Design: Creating architectural and detailed designs based on the requirements.

Implementation (Coding): Writing the actual code based on the design documents.

Testing: Verifying that the code meets the requirements and identifying defects. This phase is part of SDLC but is covered in more detail in STLC.

Deployment: Releasing the software to the users or market.

Maintenance: Providing ongoing support and updates to the software after deployment.

STLC Phases:

Requirement Analysis: Understanding the testing requirements based on the software requirements and specifications.

Test Planning: Developing a test plan that outlines the scope, resources, schedule, and strategy for testing.

Test Design: Creating detailed test cases and test scripts based on the requirements and design documents.

Test Execution: Running the test cases and scripts to identify defects or issues in the software.

Defect Reporting: Logging and reporting defects found during testing to the development team.

Test Closure: Finalizing the testing process, including preparing test summary reports, and evaluating the testing process.

Manual Training

Qazi Zain

Difference between Scrum & Agile:

Agile is a broader mindset in development to iterate the process and do collaboration in a general manner while scrum is the framework of agile in which the iteration is time-boxed known as sprint and roles are also defined.

What is the test strategy ?

The high level plan for accomplishing the testing objectives, what will be tested , how will be tested , who will perform the test, when will test.

A test strategy and a test plan are both essential components of a software testing process, but they serve different purposes and have different scopes. Here's a breakdown of each:

Test Strategy

Definition: A test strategy is a high-level document that outlines the overall approach and objectives for testing throughout the software development lifecycle. It is a broad and overarching plan that defines the testing goals, types of testing to be performed, resources required, and how testing will be integrated into the development process.

Components typically include:

- **Testing Objectives:** What the testing aims to achieve (e.g., ensuring software reliability, performance, security).
- **Testing Scope:** What will be tested (e.g., functionalities, performance, security).
- **Testing Levels:** Different levels of testing (e.g., unit testing, integration testing, system testing, acceptance testing).
- **Testing Types:** Various types of testing (e.g., manual, automated, exploratory, regression).
- **Tools and Resources:** Tools to be used for testing, and required resources (e.g., hardware, software, personnel).
- **Risk Management:** Identifying potential risks and mitigation strategies.
- **Test Environment:** The environment in which testing will be performed (e.g., staging, production).

Purpose: The test strategy provides a high-level vision and guidance for how testing will be approached across the entire project. It is a foundational document that helps align the testing process with the overall project objectives and ensures consistency.

Qazi Zain

Definition: A test plan is a detailed document that describes the specifics of how testing will be executed for a particular project or phase. It is more focused and granular compared to the test strategy.

Components typically include:

- **Test Objectives:** Specific goals for the particular testing phase or project.
- **Test Scope:** Detailed description of what is included and excluded from the testing effort.
- **Test Schedule:** Timelines for when testing will occur.
- **Test Resources:** Detailed list of resources required, including testers and tools.
- **Test Cases and Scenarios:** Specific test cases or scenarios to be executed, including input data and expected outcomes.
- **Roles and Responsibilities:** Who will perform each part of the testing.
- **Criteria for Pass/Fail:** How test results will be evaluated.
- **Risk and Contingency Plans:** Addressing specific risks related to the testing phase and how to handle them.
- **Deliverables:** What will be produced as a result of the testing (e.g., test reports, defect logs).

Purpose: The test plan translates the high-level test strategy into actionable steps for a specific project or testing cycle. It serves as a guide for executing testing activities, managing resources, and tracking progress.

In summary, the test strategy is an integral part of the test plan. It provides the foundational approach to testing, which the test plan then builds upon with detailed execution steps, schedules, and resource allocations.

What is test data?:

Test data is input used in software testing to ensure that a system functions correctly under various scenarios.

Test data can prepare :

- Manually.
- By Automation (Auto test data).

Defect Report

A well-documented bug report facilitates effective communication between testers and developers, ensuring that issues are addressed promptly and that the software improves in quality over time.

1. Defect ID:
DEF-2024-001

2. Title:

1. Invalid Email Accepted in Login
2. Incorrect Redirection After Login
3. Misuse of Red Button

3. Description:

1. **Invalid Email Accepted in Login:**
   The login system incorrectly accepts invalid email addresses without triggering an error, allowing users to log in with improperly formatted email addresses.
2. **Incorrect Redirection After Login:**
   After a successful login with valid credentials, users are redirected to a notification page rather than the intended dashboard.
3. **Misuse of Red Button:**
   The red button on the "User Settings" page does not perform its intended function and may be incorrectly labeled.

4. Severity:

1. Major

2. Critical
3. Minor

5. Priority:

1. High
2. High
3. Medium

6. Environment:
Production

7. Steps to Reproduce:

1. **Invalid Email Accepted in Login:**
    1. Navigate to the login page.
    2. Enter an invalid email address (e.g., "invalid-email").
    3. Enter any password.
    4. Click the "Login" button.
    5. Observe that the system allows login without error.
2. **Incorrect Redirection After Login:**
    1. Navigate to the login page.
    2. Enter valid credentials.
    3. Click the "Login" button.
    4. Observe that the user is redirected to the notification page instead of the dashboard.
3. **Misuse of Red Button:**
    1. Navigate to the "User Settings" page.
    2. Click the red button.
    3. Observe that the button does not perform the expected action or is incorrectly labeled.

8. Expected Result:

1. The login system should validate email formats and reject invalid email addresses.
2. After successful login, users should be redirected to the dashboard.
3. The red button should perform its intended function and be correctly labeled.

9. Actual Result:

1. The system accepts invalid email addresses without error.
2. Users are redirected to a notification page instead of the dashboard after logging in.
3. The red button does not perform its intended function or is mislabeled.

10. Attachments:

- [Screenshots of invalid email acceptance]
- [Screenshots of incorrect redirection]
- [Screenshots of the red button issue]
- [Logs, if available]

11. Reported By:
Qazi Zain, Arsam Ali, Athar

12. Date Reported:
September 16, 2024

13. Assigned To:
Development Team

14. Status:
New

15. Comments:
Please address the email validation issue urgently to prevent unauthorized access and fix the redirection to ensure a smooth user experience. The red button issue should be reviewed to confirm its intended functionality and correct any labeling issues.

Qazi Zain

Summary Report

Overview: We tested the new login feature of the XYZ app to ensure it works correctly before the upcoming release. Our testing aimed to check functionality, security, and usability.

Key Findings:

- **Total Test Cases:** 50
- **Passed:** 40
- **Failed:** 7
- **Blocked:** 3

Major Issues:

1. **Invalid Email Handling:** The system accepted improperly formatted email addresses during login.
2. **Redirection Error:** After successful login, users were incorrectly redirected to the notification page instead of the dashboard.

Performance Metrics:

- **Average Response Time:** 2 seconds
- **Load Handling:** Performed well under normal conditions.

Recommendations:

- Fix the email validation issue and correct the redirection to ensure users land on the dashboard.
- Continue testing with more edge cases and consider a final performance review.

Conclusion: Overall, the new login feature is mostly functional but requires fixes for a couple of critical issues before the release. The feature performs well under typical conditions, but attention to the identified bugs is needed to enhance user experience.

What is test plan and when it is creating:

Test plan is a detailed document to perform testing including (test data , test strategies, in scope, out scope).

A test plan is created during the early stages of the software development lifecycle, typically after the requirements phase and before the actual testing begins. It is usually developed.

What is An Api?

It's a kind of messenger which use to travel data from one place to another, it's a kind of middleware like you are using bykea app in which u can see the rider location because its showing from google map by sending some request through api and api proceed the request to google map and google ma returning the request after validating to api and api displaying the data in bykea app . It is safe because of the api key which is also used to monitor or secure the api.

Difference bw soap api and rest api?:

SOAP and REST are two ways to design web services. SOAP is a protocol that uses XML and has strict rules, which makes it complex but very secure and good for handling detailed transactions. REST, on the other hand, is an architectural style that uses standard HTTP methods and can work with different formats like JSON, making it simpler, faster, and more flexible. While SOAP is better for applications needing high security and complex operations, REST is generally preferred for its ease of use and efficiency, especially for web and mobile applications.

Qazi Zain

Status codes 100,200,300,400 and 500?

HTTP status codes are three-digit numbers returned by a server in response to a client's request. They indicate the result of the request and help the client understand whether it was successful, if there were errors, or if further action is needed. Here's a basic description of common status codes:

1. **100:** The server has received the initial part of the request and the client should continue sending the rest. (informational).

2. **200:** The request was successful, and the server has returned the requested data.(Successful).

3. **300:** Redirection.

4. **400:** The server cannot understand the request due to invalid syntax. The client should correct the request and try again. (client error).

5. **500:** The server encountered an unexpected condition that prevented it from fulfilling the request. It's a generic error message for server-side issues.

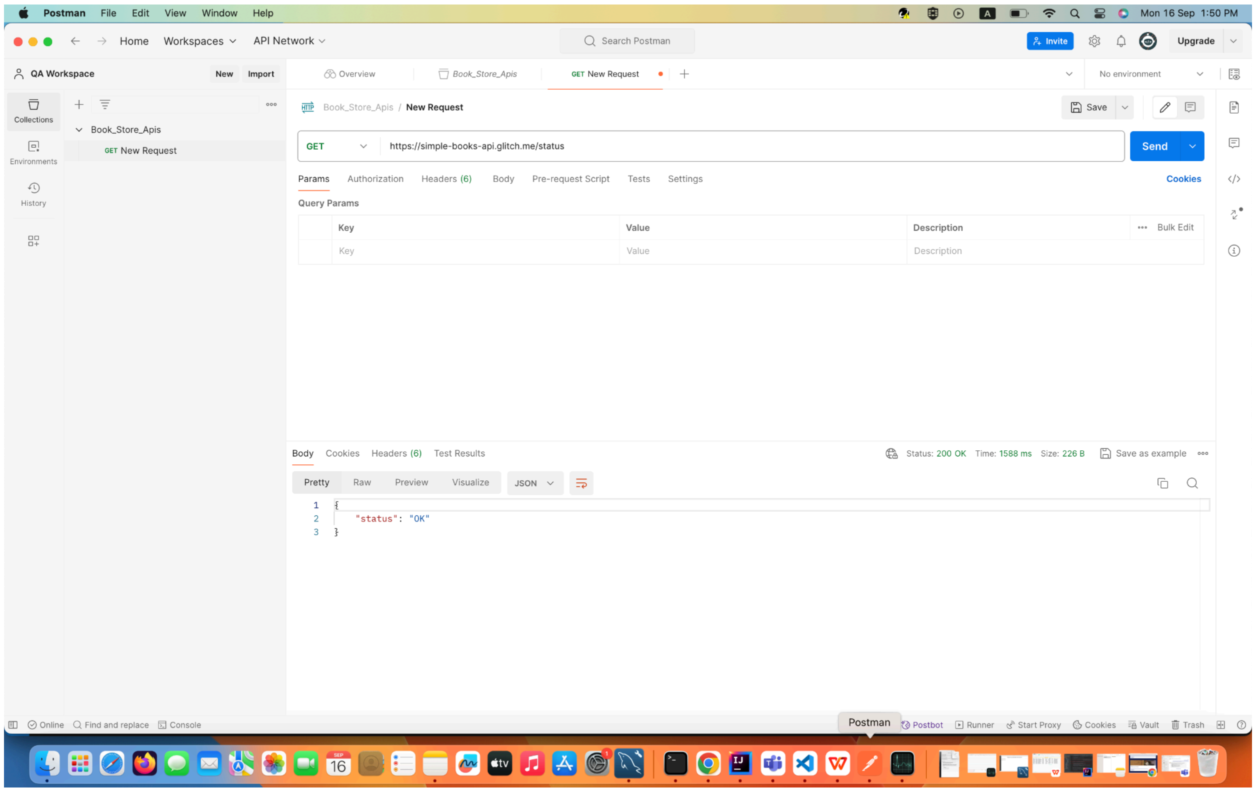
Postman:

Use for basic api testing.

Creating workspace inside this create collection inside this create request.

Get : use when we are reading data.(Apne request ki aapko data mil gaya).

Post: use when we are sending data.


Postman File Edit View Window Help
Mon 16 Sep 1:50 PM
Home Workspaces API Network
Search Postman
Invite Upgrade
QA Workspace New Import
Collections
Environments
History
Book_Store_Apis
GET New Request
Overview
No environment
Book_Store_Apis / New Request
Save
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
Query Params
Key Value Description Bulk Edit
Body Cookies Headers (6) Test Results
Save as example
Pretty Raw Preview Visualize JSON
Online Find and replace Console
Postbot Runner Start Proxy Cookies Vault Trash

GET https://simple-books-api.glitch.me/status

Status: 200 OK Time: 1588 ms Size: 226 B

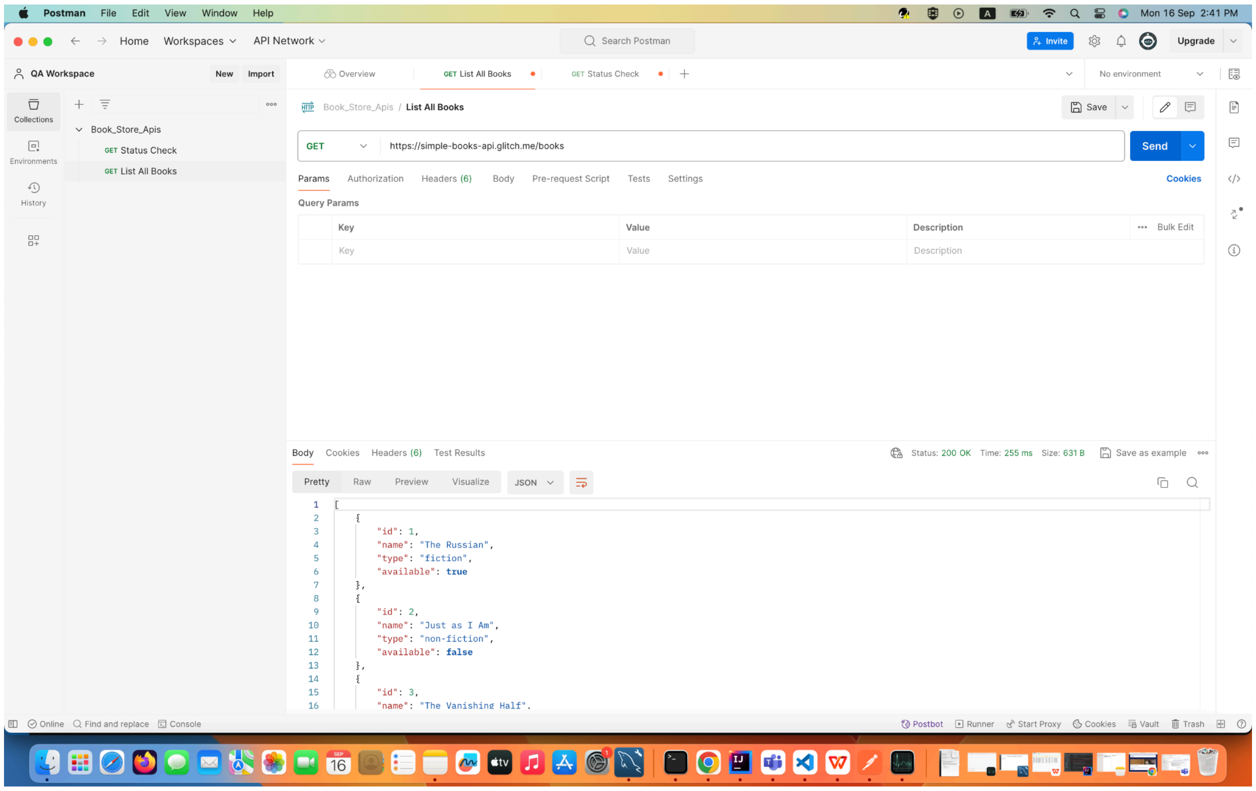
```
{
    "status": "OK"
}
```

In this you can see the status code coming 200 which means the request was successful.

Inside the header we can pass key and value by manually according to what api asks.

Manual Training

Qazi Zain

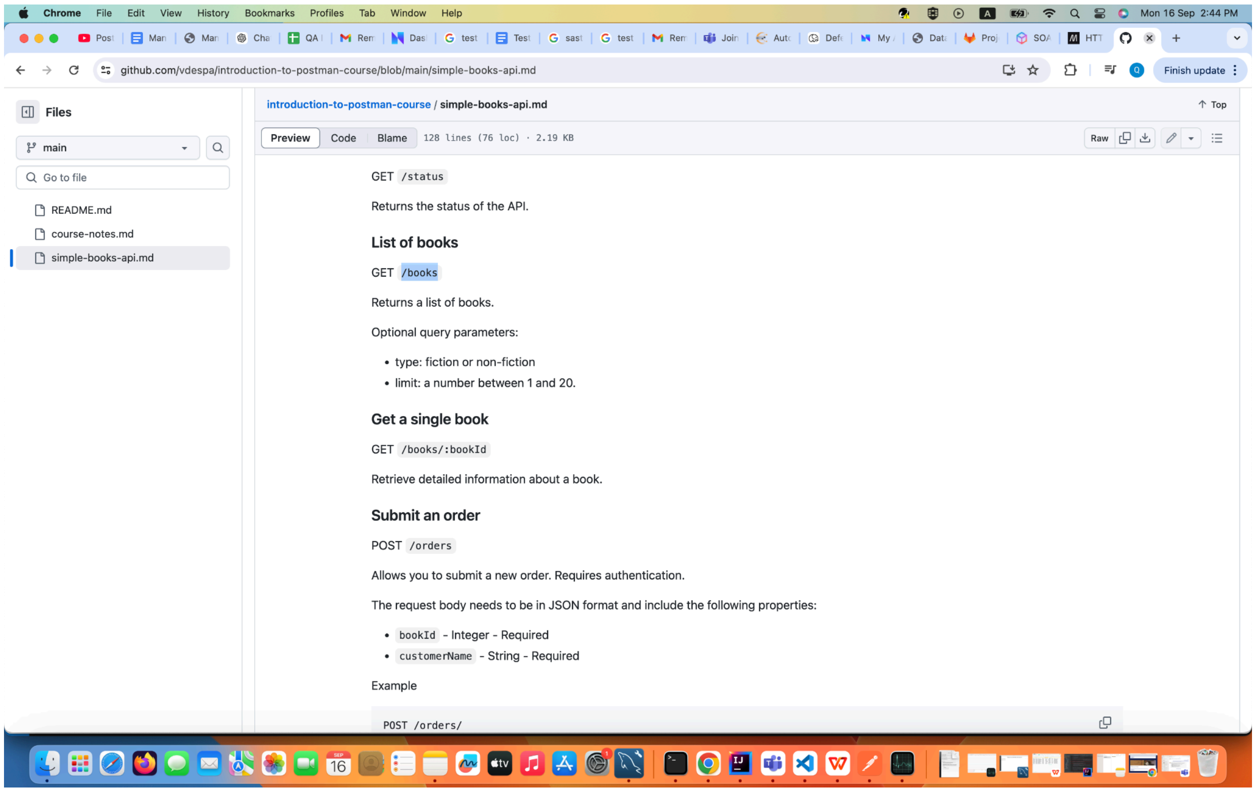

Postman File Edit View Window Help
Mon 16 Sep 2:41 PM
Home Workspaces API Network
Search Postman
Invite Upgrade
QA Workspace New Import
Overview
GET List All Books
GET Status Check
No environment
Collections
Environments
History
Book_Store_Apis
GET Status Check
GET List All Books
Book_Store_Apis / List All Books
Save
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
Query Params
Key Value Description Bulk Edit
Body Cookies Headers (6) Test Results
Save as example
Pretty Raw Preview Visualize JSON
Online Find and replace Console
Postbot Runner Start Proxy Cookies Vault Trash

GET https://simple-books-api.glitch.me/books

Status: 200 OK Time: 255 ms Size: 631 B

```
[
    {
        "id": 1,
        "name": "The Russian",
        "type": "fiction",
        "available": true
    },
    {
        "id": 2,
        "name": "Just as I Am",
        "type": "non-fiction",
        "available": false
    },
    {
        "id": 3,
        "name": "The Vanishing Half",
```

If we make a mistake in passing the url we get 400 which is a client error. Below is an example:
Here see the path is : /books/:bookId and where i pass /status/:bookId.


github.com/vdespa/introduction-to-postman-course/blob/main/simple-books-api.md
Files
main
Go to file
README.md
course-notes.md
simple-books-api.md
introduction-to-postman-course / simple-books-api.md
Preview
Code
Blame
128 lines (76 loc) · 2.19 KB
Raw

GET `/status`

Returns the status of the API.

List of books

GET `/books`

Returns a list of books.

Optional query parameters:

- type: fiction or non-fiction
- limit: a number between 1 and 20.

Get a single book

GET `/books/:bookId`

Retrieve detailed information about a book.

Submit an order

POST `/orders`

Allows you to submit a new order. Requires authentication.

The request body needs to be in JSON format and include the following properties:

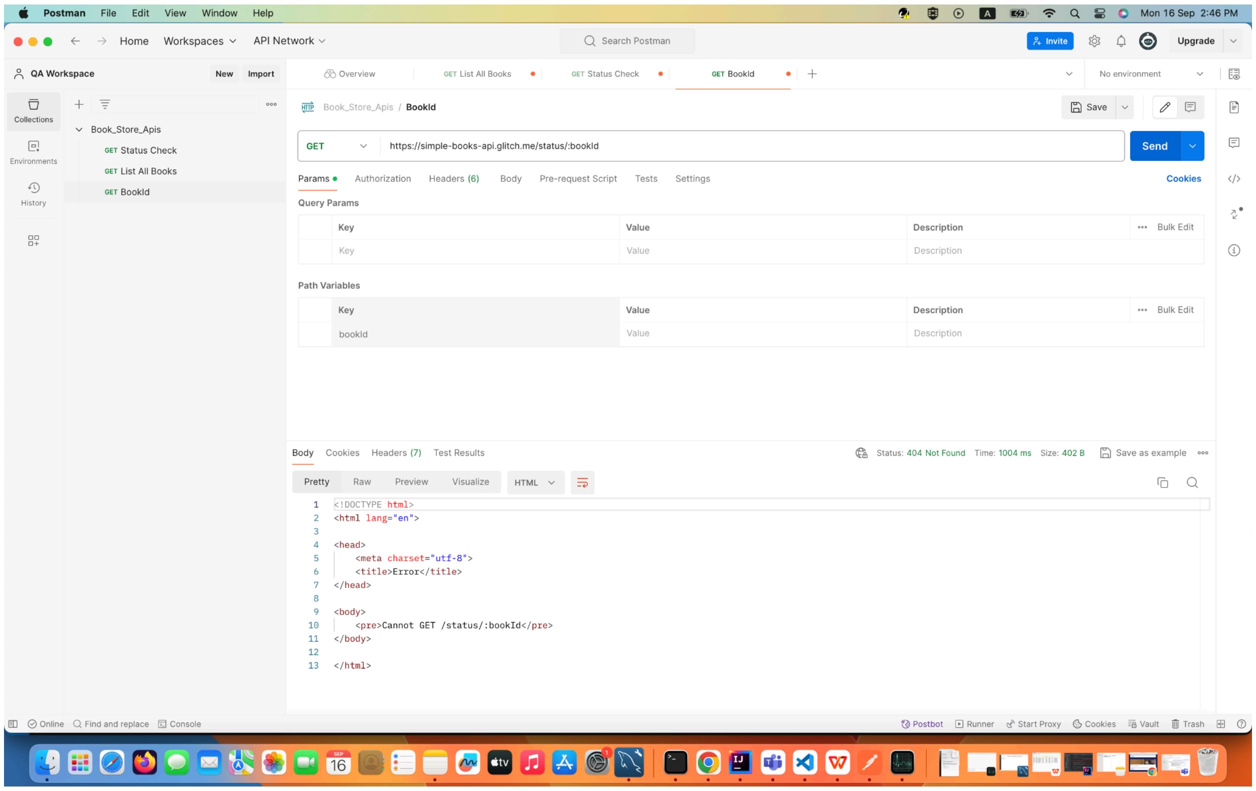
- `bookId` - Integer - Required
- `customerName` - String - Required

Example

```
POST /orders/
```

Manual Training

Qazi Zain



GET https://simple-books-api.glitch.me/status/:bookId

Path Variables

Key	Value	Description
bookId		

Status: 404 Not Found Time: 1004 ms Size: 402 B

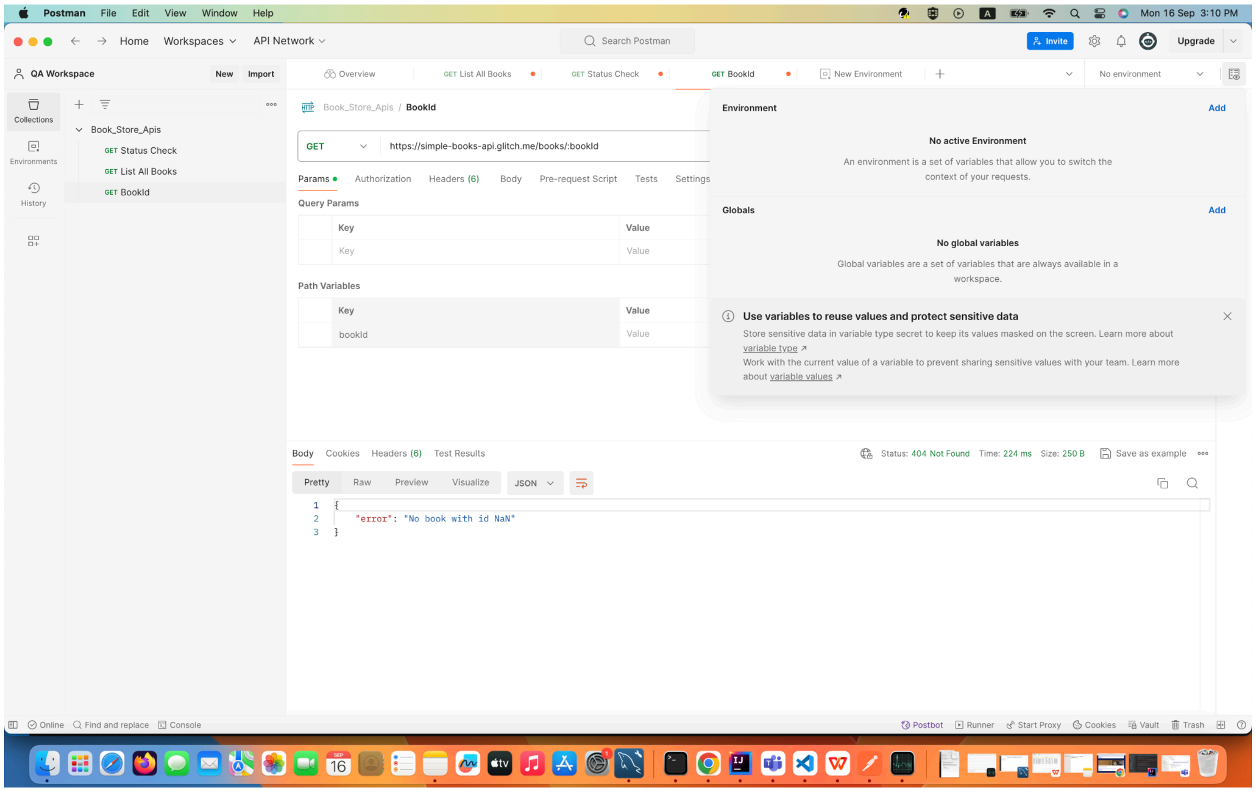
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Error</title>
</head>

<body>
    <pre>Cannot GET /status/:bookId</pre>
</body>

</html>
```

Here you can see the status code coming 400 means client error.

Manual Training

Qazi Zain

Now i am creating environment Variable:



GET https://simple-books-api.glitch.me/books/:bookId

Query Params

Key	Value
Key	Value

Path Variables

Key	Value
bookId	Value

Environment Add

No active Environment

An environment is a set of variables that allow you to switch the context of your requests.

Globals Add

No global variables

Global variables are a set of variables that are always available in a workspace.

Use variables to reuse values and protect sensitive data

Store sensitive data in variable type secret to keep its values masked on the screen. Learn more about variable type ↗

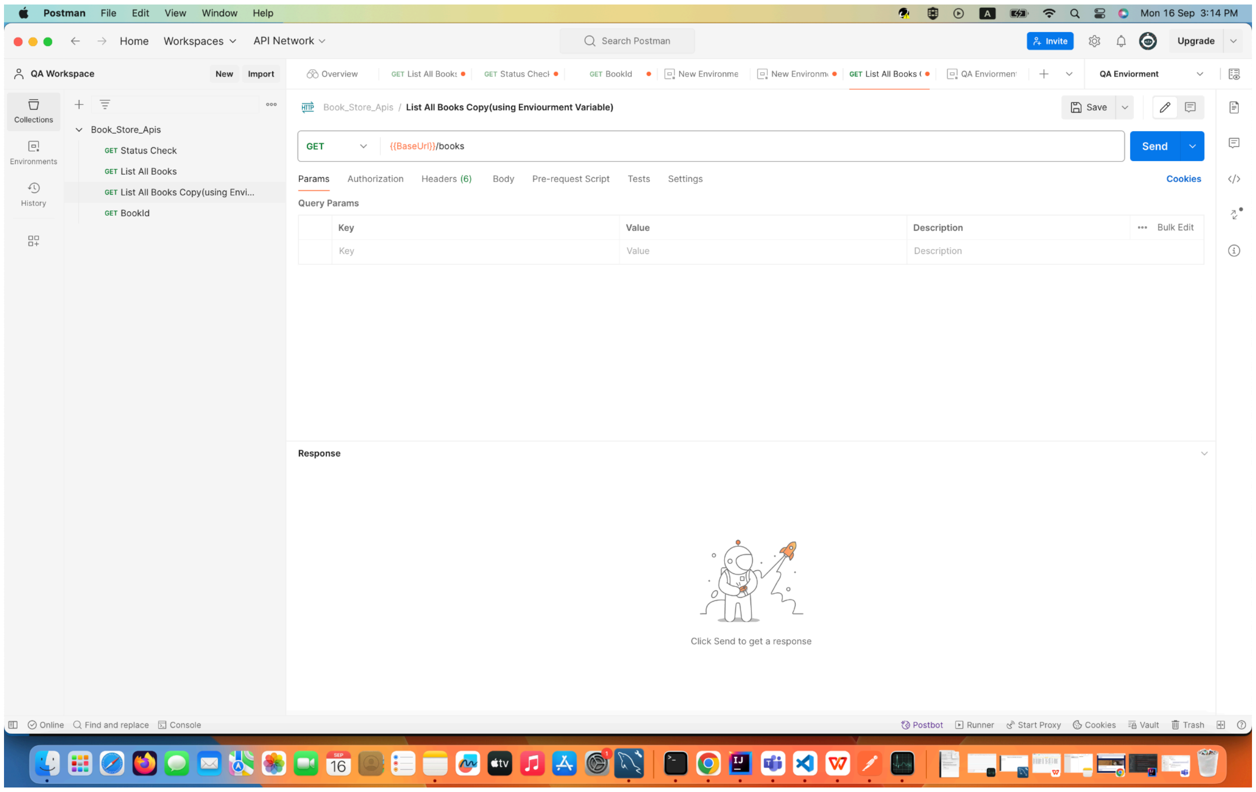
Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about variable values ↗

Body Cookies Headers (6) Test Results

Status: 404 Not Found Time: 224 ms Size: 250 B Save as example

Pretty Raw Preview Visualize JSON

```
{
    "error": "No book with id NaN"
}
```



Book_Store_Apis / List All Books Copy(using Enviourment Variable)

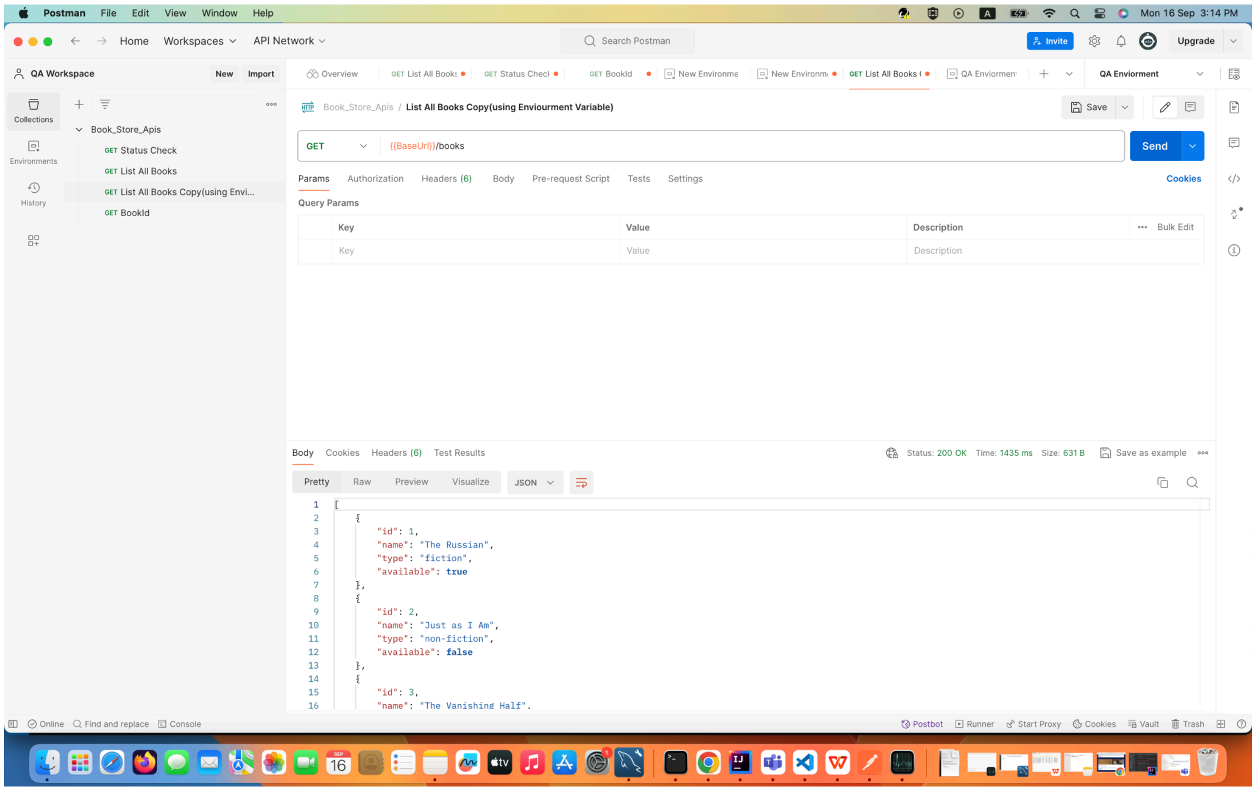
GET {{BaseUrl}}/books

Query Params

Key	Value	Description
Key	Value	Description

Manual Training

Qazi Zain



Book_Store_Apis / **List All Books Copy(using Enviourment Variable)**

GET {{BaseUrl}}/books

Query Params

Key	Value	Description
Key	Value	Description

Status: 200 OK Time: 1435 ms Size: 631 B

```
[
    {
        "id": 1,
        "name": "The Russian",
        "type": "fiction",
        "available": true
    },
    {
        "id": 2,
        "name": "Just as I Am",
        "type": "non-fiction",
        "available": false
    },
    {
        "id": 3,
        "name": "The Vanishing Half",
```