

(IT 314) Software Engineering

Group 21

Renting System

Testing Documentation

Group Members

Name	ID
Aviraj Karangiya	202001185
Vikas Patni	202001192
Kadam Darji	202001198
Ayush Gandhi	202001200
Khush Shah	202001203
Hemang Joshi	202001212
Harsh Shah	202001221
Shiv Baria	202001222
Kunj Parekh	202001223
Naman Patel	202001236

1. Unit Testing

We have used Postman to perform Unit Testing in our project, Renting System. We made test cases for different scenarios and tested them by passing different kinds of inputs and noting the response received from the backend server.

Scenario 1: Accessing the backend Server

Code:

```
pm.test("accessing server", function() {
  pm.response.to.have.status(200);
  pm.response.to.be.ok;
});
pm.test("greeting message", function() {
  pm.expect(pm.response.text()).to.include("Welcome to Renting System!");
});
```

Output:

The screenshot shows the Postman interface for a GET request to `https://rentbuddy.onrender.com/`. The 'Tests' tab is active, displaying the following code:

```
1 pm.test("accessing server", function(){
2   pm.response.to.have.status(200);
3   pm.response.to.be.ok;
4 });
5 pm.test("greeting message", function(){
6   pm.expect(pm.response.text()).to.include("Welcome to Renting System!");
7 });
```

Below the code editor, the 'Test Results (2/2)' section shows two tests, both of which passed:

- PASS** accessing server
- PASS** greeting message

The overall status is **Status: 200 OK** and the execution time is **Time: 907 ms**.

Scenario 2: Password length less than 8**Code:**

```
pm.test("Password less than the given size", function(){
    pm.expect(pm.response.text()).to.include("Please enter a valid password")
})
```

Output:

The screenshot displays the Postman interface. The top bar includes tabs for Params, Authorization, Headers (8), Body, Pre-request Script, Tests, and Settings. The 'Tests' tab is active, showing a script with three lines of code: `pm.test("Password less than the give size", function(){`, `pm.expect(pm.response.text()).to.include("Please enter a valid password")`, and `})`. The bottom section shows the 'Test Results (1/1)' tab, which includes filters for All, Passed, Skipped, and Failed. A single result is shown: 'PASS Password less than the give size'. The status bar at the bottom right indicates a 200 OK response with a 2.79 s execution time.

Scenario 3: User is not registered**Code:**

```
pm.test("User is not registered", function(){
  pm.expect(pm.response.text()).to.include("User is not registered")
})
```

Output:

The screenshot displays the Postman interface. The top bar includes tabs for Params, Authorization, Headers (8), Body, Pre-request Script, Tests, and Settings. The 'Tests' tab is active, showing a test script with the following code:

```
1 pm.test("User is not registered", function(){
2   pm.expect(pm.response.text()).to.include("User is not registered")
3 }
4
```

Below the script editor, the 'Test Results (1/1)' tab is selected, showing a single test result:

Test Results (1/1)
PASS User is not registered

On the right side of the interface, there is a sidebar with sections for 'Test scripts' and 'Snippets'. The 'Test scripts' section contains links for 'Get an env', 'Get a glob', 'Get a varia', 'Get a colle', and 'Set an env'. The 'Snippets' section contains links for 'Get an env', 'Get a glob', 'Get a varia', 'Get a colle', and 'Set an env'.

Scenario 4: Password incorrect**Code:**

```
pm.test("Password incorrect", function(){  
  pm.expect(pm.response.text()).to.include("Password is not valid")  
})
```

Output:

The screenshot displays the Postman application interface. At the top, there are tabs for 'Params', 'Authorization', 'Headers (9)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Tests' tab is selected and highlighted with an orange underline. Below the tabs, a code editor shows the following JavaScript code:

```
1 pm.test("Password incorrect", function(){  
2   pm.expect(pm.response.text()).to.include("Password is not valid")  
3 })  
4 |
```

Below the code editor, there is another set of tabs: 'Body', 'Cookies', 'Headers (15)', and 'Test Results (1/1)'. The 'Test Results (1/1)' tab is selected and highlighted with an orange underline. To the right of these tabs, the status '200 OK' and a version number '2.7' are visible. Below the tabs, there are four filter buttons: 'All', 'Passed', 'Skipped', and 'Failed'. The 'All' button is selected. Below the filters, a single test result is shown: a green button labeled 'PASS' followed by the text 'Password incorrect'.

Scenario 5: Validation of email**Code:**

```
pm.test("Vaildation of email", function(){  
  pm.expect(pm.response.text()).to.include("Enter a valid email")  
})
```

Output:

The screenshot displays the Postman interface with the 'Tests' tab selected. The test code is visible in the editor, and the 'Test Results' tab at the bottom shows a single test named 'Vaildation of email' with a 'PASS' status.

Params Authorization Headers (9) Body ● Pre-request Script Tests ● Settings

```
1 pm.test("Vaildation of email", function(){  
2   pm.expect(pm.response.text()).to.include("Enter a valid email")  
3   })  
4
```

Body Cookies Headers (15) Test Results (1/1) 200 OK

All Passed Skipped Failed

PASS Vaildation of email

Scenario 6: Signup missing firstname**Code:**

```
pm.test("Signup testing for missing firstname", ()=>{  
  pm.expect(pm.response.text()).to.include('Enter a valid Firstname');  
})
```

Output:

The screenshot displays the Postman interface for a POST request to `https://rentbuddy.onrender.com/lender/signup?firstname&lastname=joshi&email=hsj@gmail.com&idproof=hs12345`. The 'Tests' tab is active, showing the test code: `pm.test("Signup testing for missing firstname", ()=>{ pm.expect(pm.response.text()).to.include('Enter a valid Firstname'); })`. Below the code, the 'Test Results (1/1)' section shows a single result: **PASS** Signup testing for missing firstname. The overall status is **200 OK** with a time of **2.09 s**.

Scenario 7: Successful Signup

Code:

```
pm.test("Signup testing for missing firstname", ()=>{
  pm.expect(pm.response.text()).to.include('Enter a valid Firstname');
})

pm.test("Signup successful", ()=>{
  const resJson = pm.response.json();
  pm.expect(pm.response.text()).to.include('userid');
})
```

Output:

The screenshot displays the Postman interface for a POST request to `https://rentbuddy.onrender.com/lender/signup`. The 'Tests' tab is active, showing two test cases. The first test, 'Signup testing for missing firstname', has failed with an assertion error. The second test, 'Signup successful', has passed.

POST `https://rentbuddy.onrender.com/lender/signup`

Params • Authorization Headers (9) Body • Pre-request Script **Tests •** Settings

```
1 pm.test("Signup testing for missing firstname", ()=>{
2   pm.expect(pm.response.text()).to.include('Enter a valid Firstname');
3 })
4
5 pm.test(["Signup successful", ()=>{
6   const resJson = pm.response.json();
7   pm.expect(pm.response.text()).to.include('userid');
8 }])
```

Body Cookies Headers (15) **Test Results (1/2)** Status: 200 OK Time: 3.73 s

All Passed Skipped Failed

FAIL Signup testing for missing firstname | AssertionError: expected '{"error":false,"userid":"644bc5073efd..."' to include 'Enter a valid

PASS Signup successful

Scenario 8: Product Present/Missing**Code:**

```
pm.test("accessing server", function() {
  pm.response.to.have.status(200);
  pm.response.to.be.ok;
});

pm.test("Response JSON check", () => {
  const resJson = pm.response.json();
  pm.expect(resJson.title).to.eql("House 1");
})

pm.test("Product not present", ()=>{
  pm.expect(pm.response.text()).to.be.empty;
})
```

Output:

Product not present

The screenshot shows the Postman interface for a GET request to `https://rentbuddy.onrender.com/products/642e85973ecfa602c993d775`. The 'Tests' tab is active, displaying the following test code:

```
1 pm.test("accessing server", function() {
2   pm.response.to.have.status(200);
3   pm.response.to.be.ok;
4 });
5 pm.test("Response JSON check", () => {
6   const resJson = pm.response.json();
7   pm.expect(resJson.title).to.eql("House 1");
8 })
9
10 pm.test("Product not present", ()=>{
11   pm.expect(pm.response.text()).to.be.empty;
12 })
```

The 'Test Results' tab is also active, showing the following results:

- PASS** accessing server
- PASS** Response JSON check
- FAIL** Product not present | AssertionError: expected '{"_id":"642e85973ecfa602c993d775","ti..."' to be empty

The overall status is **200 OK** and the time taken is **1929 ms**.

Product present

The screenshot displays a REST client interface for a GET request to the URL `https://rentbuddy.onrender.com/products/642e85973ecfa602c993d774`. The interface includes tabs for Params, Authorization, Headers (7), Body, Pre-request Script, Tests, and Settings. The Tests tab is active, showing a list of three tests:

```
1 pm.test("accessing server", function() {
2   pm.response.to.have.status(200);
3   pm.response.to.be.ok;
4 });
5 pm.test("Response JSON check", () => {
6   const resJson = pm.response.json();
7   pm.expect(resJson.title).toEqual("House 1");
8 });
9
10 pm.test(["Product not present", ()=>{
11   pm.expect(pm.response.text()).to.be.empty;
12 }]);
```

Below the code editor, the Test Results section shows the following outcomes:

- PASS** accessing server
- FAIL** Response JSON check | `JSONError: No data, empty input at 1:1 ^`
- PASS** Product not present

The overall status is **200 OK** and the execution time is **2.32 s**.

Scenario 9: Product Add**Code:**

```
pm.test("Product Add testing missing price", ()=>{
  pm.expect(pm.response.text()).to.include('Please enter the price');
})

pm.test("Product Added Succsessfully", ()=>{
  pm.expect(pm.response.text()).to.include('Succsessfully Saved Product');
})
```

Output:

Price field is missing

POST <https://rentbuddy.onrender.com/products/lender/>

Params Authorization Headers (9) Body Pre-request Script Tests Settings

```
1 pm.test("Product Add testing missing price", ()=>{
2   pm.expect(pm.response.text()).to.include('Please enter the price');
3 })
4
5 pm.test("", ()=>{
6   pm.expect(pm.response.text()).to.include('Succsessfully Saved Product');
7 })
```

Body Cookies Headers (15) Test Results (1/2) Status: 200 OK Time: 2.08 s

All Passed Skipped Failed

PASS Product Add testing missing price

FAIL | AssertionError: expected '{"error":true,"msg":"Please enter the..."' to include 'Succsessfully Saved Product'

Product added successfully

The screenshot shows a REST client interface with a POST request to `https://rentbuddy.onrender.com/products/lender/`. The **Tests** tab is active, displaying two tests:

```
1 pm.test("Product Add testing missing price", ()=>{
2   pm.expect(pm.response.text()).to.include('Please enter the price');
3 })
4
5 pm.test("Product Added Successfully", ()=>{
6   pm.expect(pm.response.text()).to.include('Successfully Saved Product');
7 })
```

Below the tests, the **Test Results (1/2)** section shows the following results:

- FAIL** Product Add testing missing price | AssertionError: expected '{"error":false,"msg":"Successfully S...}' to include 'Please enter the price'
- PASS** Product Added Successfully

The overall status is **200 OK** and the time taken is **2.79 s**.

Scenario 10: Fetching details of already existing lender**Code:**

```
pm.test("Fetching details of lender", function() {  
  pm.response.to.be.json;  
})
```

Output:

The screenshot displays the Postman interface for a GET request to `https://rentbuddy.onrender.com/lender/detail`. The 'Tests' tab is active, showing a JavaScript test script:

```
1 pm.test("Fetching details of lender", function(){  
2   pm.response.to.be.json;  
3 }  
4
```

Below the script, the 'Test Results (1/1)' section shows a single test that passed:

Test Results (1/1)
PASS Fetching details of lender

The status bar at the bottom indicates a 200 OK response, a duration of 2.23 s, and a body size of 1.02 KB. A 'Save as Example' button is also visible.

Scenario 11: Order placing:**Code:**

```
pm.test("return date missing", function(){
  pm.expect(pm.response.text()).to.include('Please enter a valid return date');
})

pm.test("", ()=>{
  pm.expect(pm.response.text()).to.include('Successsfully Order Placed');
})
```

Output:

When a field is empty:

The screenshot shows a Postman interface for a POST request to `https://rentbuddy.onrender.com/order/`. The 'Tests' tab is active, displaying two test scripts. The first test, 'return date missing', passes. The second test, which checks for 'Successsfully Order Placed' in the response text when the return date is empty, fails. The 'Test Results' tab at the bottom shows the failure details: 'AssertionError: expected '{"error":true,"msg":"Please enter a v...'} to include 'Successsfully Order Placed''.

```
1 pm.test("return date missing", function(){
2   pm.expect(pm.response.text()).to.include('Please enter a valid return
   date');
3 })
4
5 pm.test("", ()=>{
6   pm.expect(pm.response.text()).to.include('Successsfully Order Placed')
7   ;
8 })
```

Test Results (1/2)

PASS return date missing

FAIL | AssertionError: expected '{"error":true,"msg":"Please enter a v...' to include 'Successsfully Order Placed'

When all things are added valid:

The screenshot shows a REST client interface with a POST request to `https://rentbuddy.onrender.com/order/`. The **Tests** tab is active, displaying two test scripts in a code editor:

```
1 pm.test("return date missing", function(){
2   pm.expect(pm.response.text()).to.include('Please enter a valid return
   date');
3 })
4
5 pm.test("", ()=>{
6   pm.expect(pm.response.text()).to.include('Successfully Order Placed')
7   ;
8 })
```

On the right, a sidebar provides information about test scripts and snippets. Below the code editor, the **Test Results (1/2)** tab is active, showing a summary of the test outcomes:

- FAIL** return date missing | AssertionError: expected '{"error":false,"msg":"Successfully O...}' to include 'Please enter a valid return
- PASS**

The status bar at the bottom indicates a **200 OK** response with a time of **1978 ms** and a size of **813 B**. A **Save** button is also visible.

Scenario 12: Product Delete**Code:**

```
pm.test("Product to be deleted Not Found", ()=>{
  pm.expect(pm.response.text()).to.include('Product Not Found');
})

pm.test("Product Deleted Successfully", ()=>{
  pm.expect(pm.response.text()).to.include('Successfully Deleted');
})
```

Output:

Product key wrong:

The screenshot shows a REST client interface with a DELETE request to `https://rentbuddy.onrender.com/products/lender/644bd52a3efdf091c3298c07`. The 'Tests' tab is active, displaying two test cases:

```
1 pm.test("Product to be deleted Not Found", ()=>{
2   pm.expect(pm.response.text()).to.include('Product Not Found');
3 })
4
5 pm.test("Product Deleted Successfully", ()=>{
6   pm.expect(pm.response.text()).to.include('Successfully Deleted');
7 })
```

The 'Test Results' tab shows the following results:

Test Case	Result
Product to be deleted Not Found	PASS
Product Deleted Successfully	FAIL

The failure message for the second test case is: `AssertionError: expected '{"error":true,"msg":"Product Not Foun...}' to include 'Successfully Deleted'`

Correct Product Key

The screenshot displays a REST client interface for a DELETE request. The URL is `https://rentbuddy.onrender.com/products/lender/644bd52a3efdf091c3298c08`. The interface includes tabs for Params, Authorization, Headers (7), Body, Pre-request Script, Tests, and Settings. The Tests tab is active, showing two test cases:

```
1 pm.test("Product to be deleted Not Found", ()=>{
2   pm.expect(pm.response.text()).to.include('Product Not Found');
3 })
4
5 pm.test(["Product Deleted Successfully", ()=>{
6   pm.expect(pm.response.text()).to.include('Successfully Deleted');
7 })]
```

Below the tests, the Test Results section shows the status of the tests. The first test, "Product to be deleted Not Found", failed with the message: "AssertionError: expected '{\"error\":false,\"msg\":\"Successfully D...\"}' to include 'Product Not Found'". The second test, "Product Deleted Successfully", passed.

Test Results (1/2)

Test Name	Status	Message
Product to be deleted Not Found	FAIL	AssertionError: expected '{\"error\":false,\"msg\":\"Successfully D...\"}' to include 'Product Not Found'
Product Deleted Successfully	PASS	

Scenario 13: Order found or not using order_id:

Code:

```
pm.test("Order found", function() {  
    pm.response.to.be.json;  
})
```

Output:

The screenshot displays the Postman interface for a GET request to the URL `https://rentbuddy.onrender.com/order/lender/644796ea6fa97d608e5a19f4`. The 'Tests' tab is selected, showing the following test script:

```
1 pm.test("Order found", function() {  
2     pm.response.to.be.json;  
3 })  
4
```

On the right side, there is a sidebar with 'Test scripts' and 'Snippets' sections. The 'Test Results' tab at the bottom shows a single result: **PASS** Order found.

2. GUI Testing:

Manual testing is a common approach used to perform GUI Testing, which involves a human tester executing specific actions on the software application and observing the outcomes. During GUI Testing, the tester may also examine the various GUI components' behavior, including their responsiveness, ease of use, and consistency in appearance and functionality across different platforms and devices.

GUI Testing Manually

- **All the web pages in the RentBuddy.**
 1. Home (All Products)
 2. Login
 3. SignUp
 4. Profile
 5. Category

Home

Functionality	Workability
Text - field	Not present
Text	UI-done , responsive
Dropdown	Not present
Buttons	Buttons working properly
Check-box	Not present
radio-button	Not present
Hyperlink	Link is working fine.
Image	Present and taking to view product page
Num-input fields	Not present
scroll-bar	Present and can see more products
Message	Not present
Logo	No need to direct to the home page
Menus its operation	Not present
Maps	Not present

General	Working all right
---------	-------------------

Login

Functionality	Workability
Text - field	Password-no special character is needed
Text	Clear and Proper alignment
Dropdown	Not present
Buttons	Good UI and working and responsive when hover
Check-box	Present and working great
radio-button	Not present
Hyperlink	Link is fine
Image	Set fine
Num-input fields	Not present
scroll-bar	Not present

Message	All messages related to the text field present
Logo	redirect to the home page
Menus its operation	Not present
Maps	Working
General	All well , responsive

Signup

Functionality	Workability
Text - field	Email and password are working fine, and the name should not have spaces.
Text	UI-done
Dropdown	Not present
Buttons	Responsive when hover
Check-box	Not present
radio-button	Not present

Hyperlink	The hyperlink is working fine.
Image	All right
Num-input fields	Not present
scroll-bar	present
Message	All messages related to the text field present
Logo	redirect to the home page
Menus its operation	Not present
Maps	Not present
General	Quiet good, responsive

Profile

Functionality	Workability
Text	Design is ok
Text - field	Present
Dropdown	Not present
Buttons	Good UI amd hover effect ok
Check-box	Not present
radio-button	Not present
Hyperlink	Present and working fine
Image	Not present
Num-input fields	Not present
scroll-bar	Not present
Message	All message-related present
Logo	Redirect to all products
Menus its operation	Not present
Maps	zoom in + out fine

General	All good
---------	----------

Category

Functionality	Workability
Text - field	Not present
Text	UI-done , responsive
Dropdown	Not present
Buttons	Buttons working properly
Check-box	Not present
radio-button	Not present
Hyperlink	Link is working fine.
Image	Present and taking to category selected
Num-input fields	Not present
scroll-bar	Not needed
Message	Not present

Logo	No need to direct to the home page
Menus its operation	Not present
Maps	Not present
General	Working all right

3. System Testing

1. Testing of sign up page:

- Invalid Input Field and their error:

Invalid Input Field	Showing Error
FirstName	Please write firstname in the field
LastName	Please write lastname in the field
Address	Please write address in the field
Email	It should have '@' and after it should have '.'
Password	It should have atleast 8 characters


Test Cases:

Test case	FirstName	LastName	Address	Email	Password	Expected outcome
-----------	-----------	----------	---------	-------	----------	------------------

1	Harsh	Shah	DAIICT	h@ac.i n	12216678892	Successful signup
2	1233	1223	1223	h@ac.i n	12216678892	Successful signup
3	Harsh	Shah	DAIICT	y@ac	123456789	Please enter valid email
4	Neha	patel	Nirma	y@ac.i n	12345	Please enter valid password
5	Saunak	Giri	LD	y.in@a c	123456789	Please enter valid email

ScreenShots of some invalid inputs:

Buyer Sign up



Harsh

Shah

DAIICT

h@ac.in

....


Sign up


[Existing user? Log in](#)


[Sign up as a Seller](#)


PLEASE ENTER A VALID PASSWORD


Buyer Sign up




 Harsh

 Shah



 DAIICT

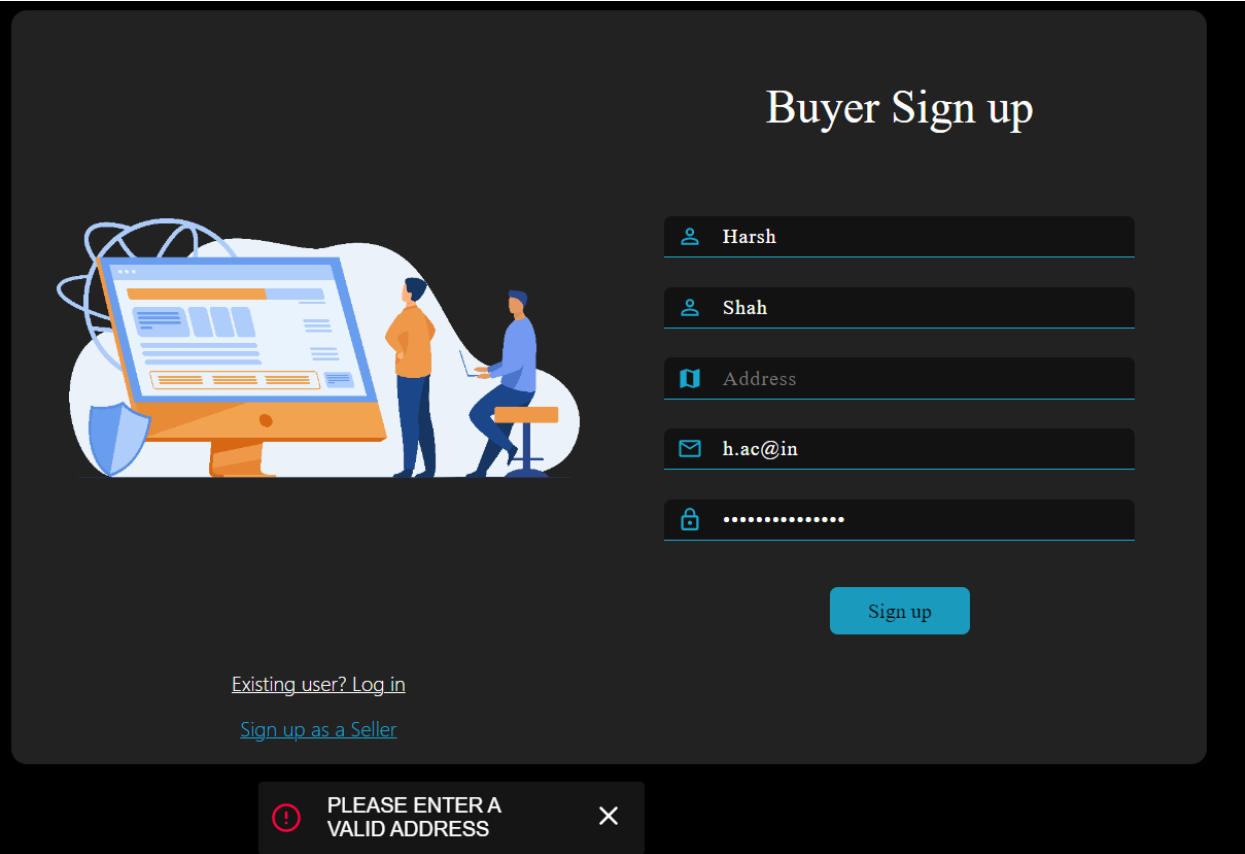
 h.ac@in



Sign up

[Existing user? Log in](#)
[Sign up as a Seller](#)

 PLEASE ENTER A
VALID E-MAIL 



The image shows a 'Buyer Sign up' form on a dark background. On the left, there is an illustration of two people, one standing and one sitting at a desk with a large monitor displaying a website. The form on the right has five input fields: two for names ('Harsh' and 'Shah'), one for 'Address', one for email ('h.ac@in'), and one for password (masked with dots). A blue 'Sign up' button is below the fields. At the bottom left of the form area, there are links: 'Existing user? Log in' and 'Sign up as a Seller'. At the bottom center, there is a red error message box that says 'PLEASE ENTER A VALID ADDRESS' with a close button (X).

Buyer Sign up

Harsh

Shah

Address

h.ac@in

.....

Sign up

Existing user? [Log in](#)

[Sign up as a Seller](#)

PLEASE ENTER A VALID ADDRESS

2. Testing of the error messages that are getting displayed perfectly in the case of any error i.e. When we enter the wrong password, an error message is displayed.

Invalid Input Field and their error:






Invalid Input Field	Showing Error
Email	It should have '@' and after it should have '.'
Password	It should have atleast 8 characters

Testcase:

Test case	Email	Password	Expected outcome
1	y@ac	123456789	Please enter valid email
2	y@ac.in	12345	Please enter valid password
3	y.in@ac	123456789	Please enter valid email


RentBuddy


Rent More, Worry Less



Renting System © 2023. All rights reserved.

Buyer Sign in

 nm@gamil.com





☐ Remember me

Log in

Create an account

Sign in as a Seller

 PLEASE ENTER A VALID PASSWORD



Seller sign in

☐ Remember me


Log in

[Create an account](#) [Sign in as a Buyer](#)

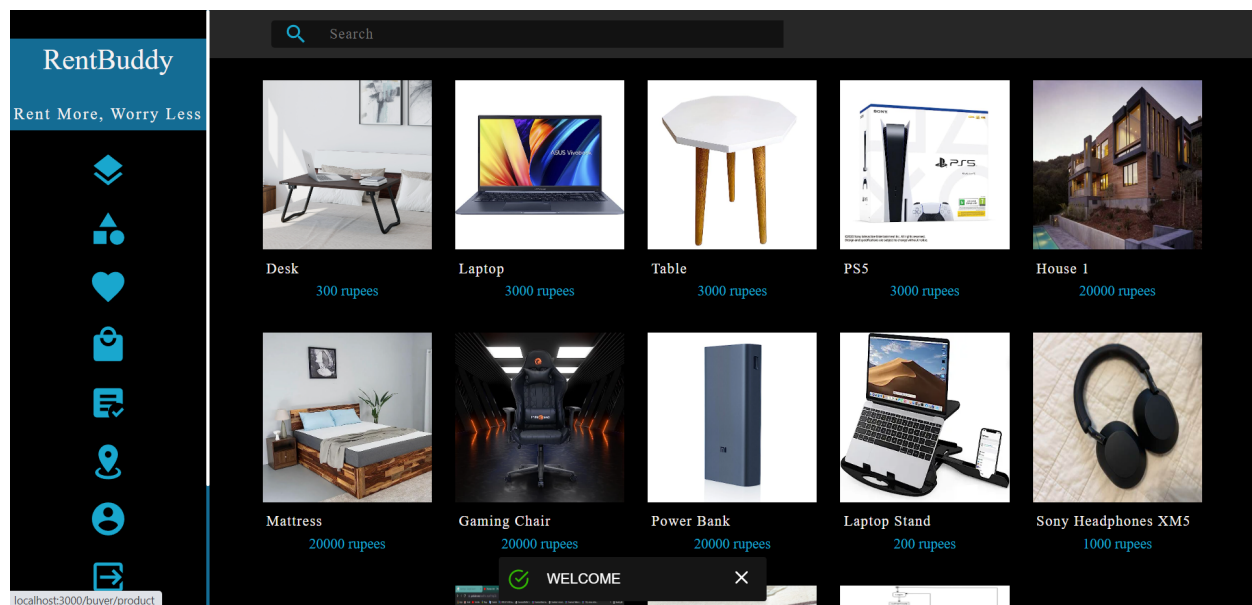
!

ENTER A VALID EMAIL ADDRESS

×

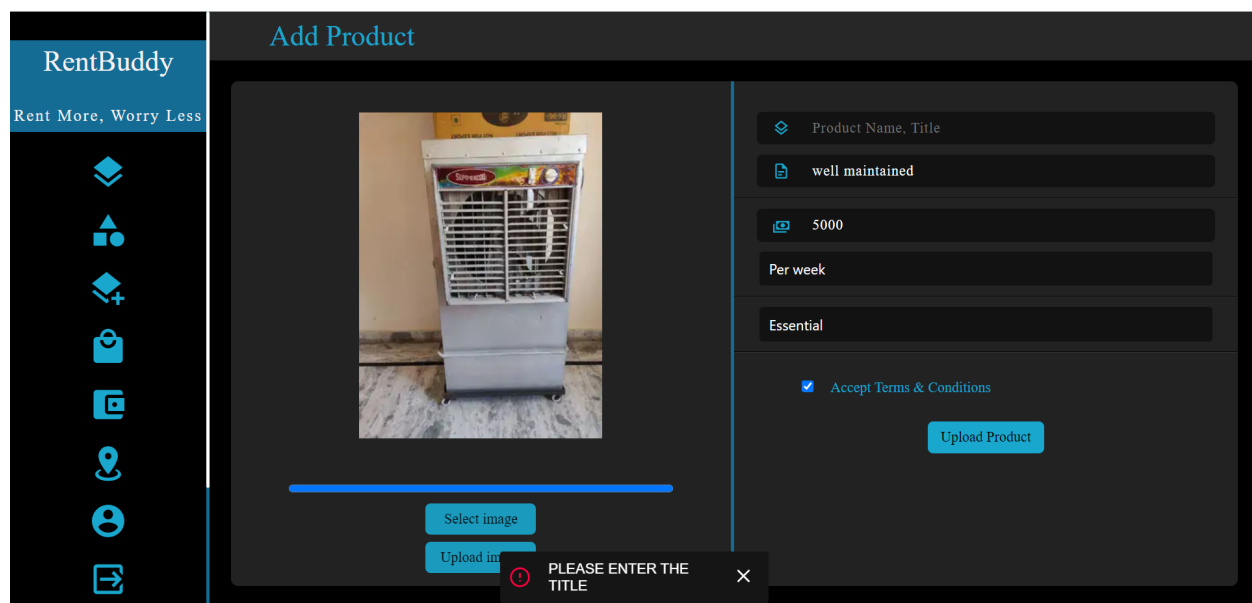


3. When you log in to your account, a welcome message is displayed, and all product pages are opened.



→ All error messages are displayed in red color, and messages without errors are displayed in green color.

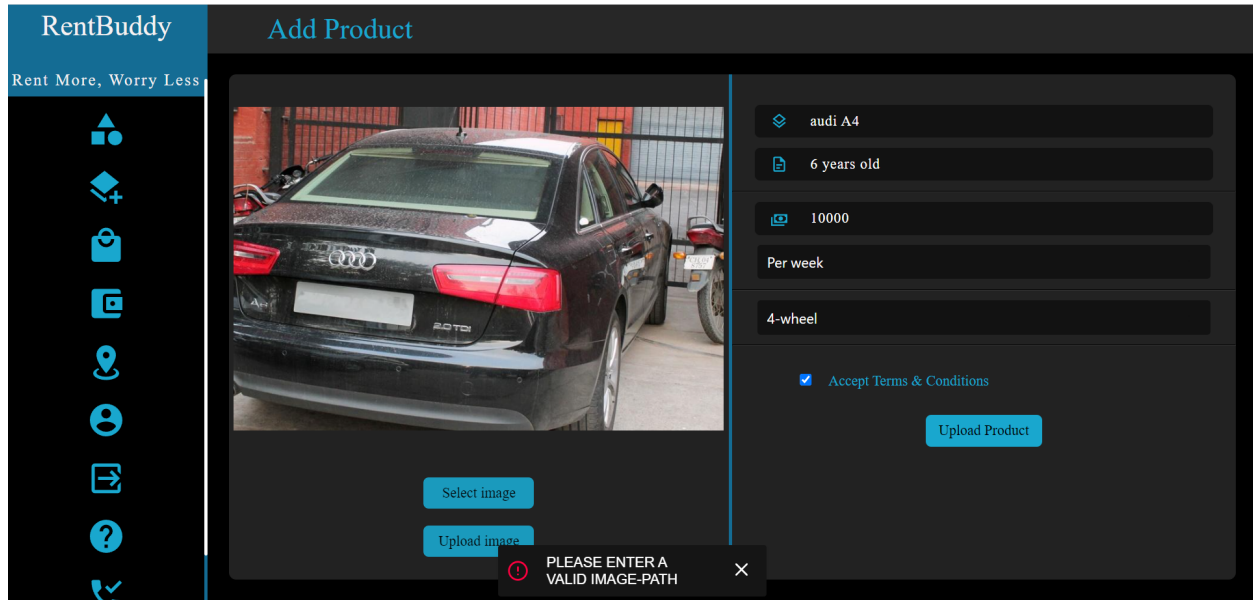
4. As you can see, if you forget to enter any information while uploading a product, an error message will be displayed.



→ This will enable you to understand where you have made a mistake and what needs to be corrected.

→ As you can see, our website has a simple and user-friendly interface. We have used simple metaphors and a straightforward logo that is easy for users to understand.

5. Here, as you can see, if you try to upload a product without adding an image, you will receive an error message that says, "PLEASE ENTER A VALID IMAGE-PATH."



→ As a result of this, users are made aware that they need to upload the image first.

4. Non-Functional Testing

All Products

HTTP Request

Name: All Products

Comments:

Basic

Advanced

Web Server

Protocol [http]: Server Name or IP: rentbuddy-ylsr.onrender.com Port Number:

HTTP Request

GET Path: / Content encoding:

☐ Redirect Automatically

☒ Follow Redirects

☒ Use KeepAlive

☐ Use multipart/form-data

☐ Browser-compatible headers

Parameters

Body Data

Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail

Add

Add from Clipboard

Delete

Up

Down

Category

HTTP Request

Name: Category

Comments:

▲ ▼

...

Basic

Advanced

Web Server

Protocol [http]: Server Name or IP: rentbuddy-ylsr.onrender.com Port Number:

HTTP Request

GET Path: /category Content encoding:

☐ Redirect Automatically

☒ Follow Redirects

☒ Use KeepAlive

☐ Use multipart/form-data

☐ Browser-compatible headers

Parameters

Body Data

Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail

Add

Add from Clipboard

Delete

Up

Down

FAQ

HTTP Request

Name:

FAQ

Comments:

▲ ▼

...

Basic

Advanced

Web Server

Protocol [http]:

Server Name or IP: rentbuddy-ylsr.onrender.com

Port Number:

HTTP Request

GET

▼

Path: /help

Content encoding:

☐ Redirect Automatically

☒ Follow Redirects

☒ Use KeepAlive

☐ Use multipart/form-data

☐ Browser-compatible headers

Parameters

Body Data

Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail

Add

Add from Clipboard

Delete

Up

Down

Seller Login

HTTP Request

Name: Seller Login

Comments:

Basic

Advanced

Web Server

Protocol [http]: Server Name or IP: rentbuddy-ylsr.onrender.com Port Number:

HTTP Request

POST Path: /seller/login Content encoding:

☐ Redirect Automatically

☒ Follow Redirects

☒ Use KeepAlive

☐ Use multipart/form-data

☐ Browser-compatible headers

Parameters

Body Data

Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
email	202001223@daiict.ac.in	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>
password	k202001223	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>

Detail

Add

Add from Clipboard

Delete

Up

Down

Contact Us

HTTP Request

Name:

Contact Us

Comments:

▲ ▼

...

Basic

Advanced

Web Server

Protocol [http]:

Server Name or IP:

rentbuddy-ylsr.onrender.com

Port Number:

HTTP Request

GET

▼

Path:

/contactus

Content encoding:

☐ Redirect Automatically

☒ Follow Redirects

☒ Use KeepAlive

☐ Use multipart/form-data

☐ Browser-compatible headers

Parameters

Body Data

Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail

Add

Add from Clipboard

Delete

Up

Down

Buyer Order

HTTP Request

Name: Buyer Order

Comments:

Basic

Advanced

Web Server

Protocol [http]: Server Name or IP: rentbuddy-ylsr.onrender.com Port Number:

HTTP Request

GET Path: /buyer/order Content encoding:

☐ Redirect Automatically

☒ Follow Redirects

☒ Use KeepAlive

☐ Use multipart/form-data

☐ Browser-compatible headers

Parameters

Body Data

Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail

Add

Add from Clipboard

Delete

Up

Down

Buyer Profile

HTTP Request

Name: Buyer Profile

Comments:

Basic

Advanced

Web Server

Protocol [http]: Server Name or IP: rentbuddy-ylsr.onrender.com Port Number:

HTTP Request

GET Path: /buyer/profile Content encoding:

☐ Redirect Automatically

☒ Follow Redirects

☒ Use KeepAlive

☐ Use multipart/form-data

☐ Browser-compatible headers

Parameters

Body Data

Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail

Add

Add from Clipboard

Delete

Up

Down

Seller Requests

HTTP Request

Name: Seller Requests

Comments:

Basic

Advanced

Web Server

Protocol [http]: Server Name or IP: rentbuddy-ylsr.onrender.com Port Number:

HTTP Request

GET Path: /seller/request Content encoding:

☐ Redirect Automatically

☒ Follow Redirects

☒ Use KeepAlive

☐ Use multipart/form-data

☐ Browser-compatible headers

Parameters

Body Data

Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail

Add

Add from Clipboard

Delete

Up

Down

Seller register

HTTP Request

Name: Seller register

Comments:

▲ ▼

...

Basic

Advanced

Web Server

Protocol [http]: Server Name or IP: rentbuddy-yls.onrender.com Port Number:

HTTP Request

GET Path: /seller/register Content encoding:

☐ Redirect Automatically

☒ Follow Redirects

☒ Use KeepAlive

☐ Use multipart/form-data

☐ Browser-compatible headers

Parameters

Body Data

Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail

Add

Add from Clipboard

Delete

Up

Down

Seller Add Product

HTTP Request

Name: Seller AddProduct

Comments:

▲ ▼

...

Basic

Advanced

Web Server

Protocol [http]: Server Name or IP: rentbuddy-ylsr.onrender.com/ Port Number:

HTTP Request

GET Path: seller/addproduct Content encoding:

☐ Redirect Automatically

☒ Follow Redirects

☒ Use KeepAlive

☐ Use multipart/form-data

☐ Browser-compatible headers

Parameters

Body Data

Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail

Add

Add from Clipboard

Delete

Up

Down

Sold Products

HTTP Request

Name:

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: Port Number:

HTTP Request

Path: Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail Add Add from Clipboard Delete Up Down

Seller Profile

HTTP Request

Name:

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: Port Number:

HTTP Request

Path: Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail Add Add from Clipboard Delete Up Down

Seller My Products

HTTP Request

Name:

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: Port Number:

HTTP Request

GET Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail Add Add from Clipboard Delete Up Down

b. Results obtained

Table form

View Results in Table

Name:

Comments:

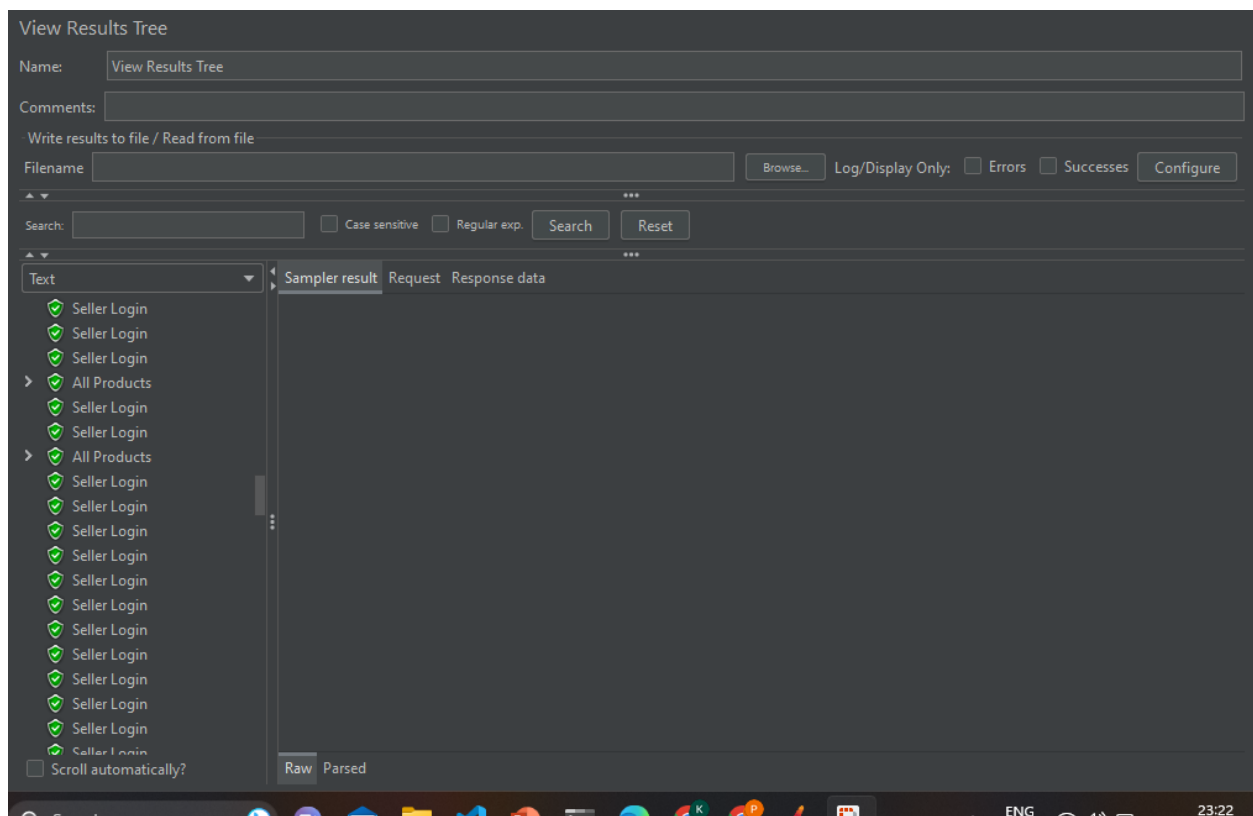
Write results to file / Read from file

Filename Log/Display Only: ☐ Errors ☐ Successes

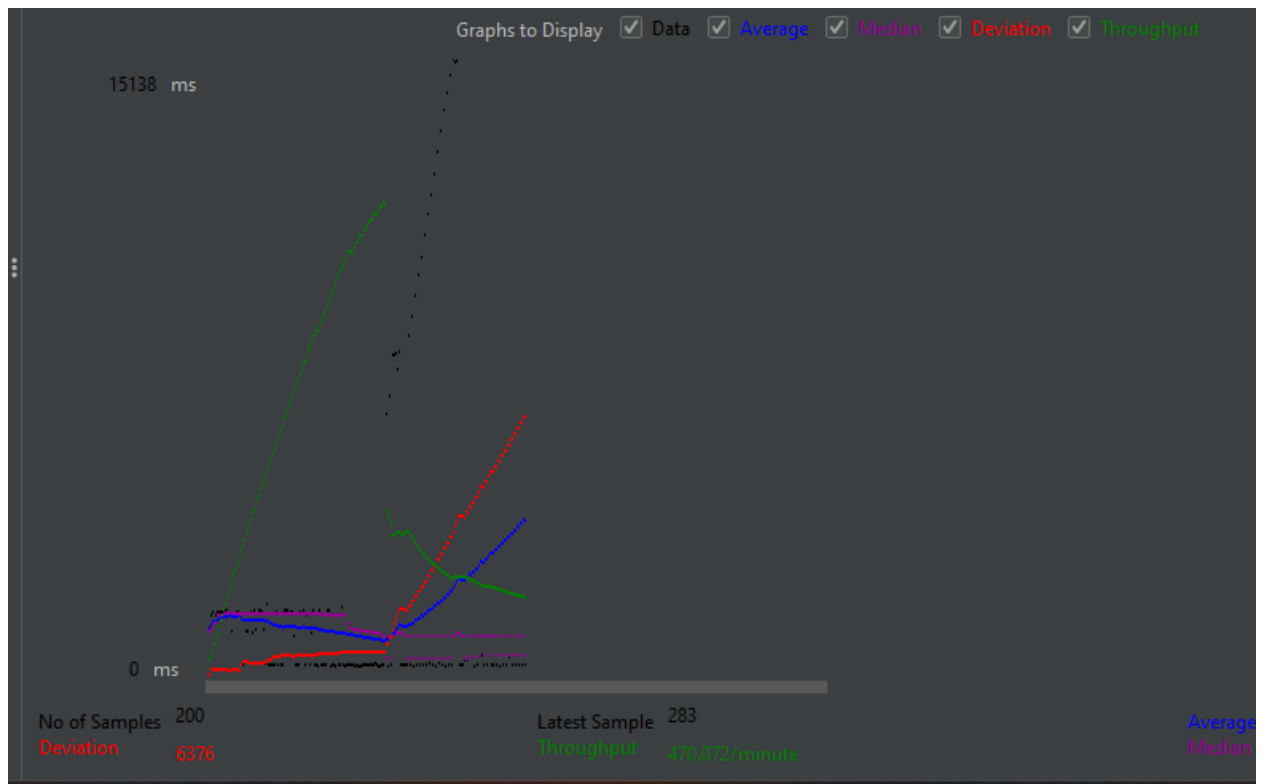
Sample #	Start Time	Thread Name	Label	Sample Time(...	Status	Bytes	Sent Bytes	Latency	Connect Time(...
1	23:20:23.398	User 1-36	All Products	1126		4607	264	356	55
2	23:20:23.322	User 1-28	All Products	1213		4607	264	422	124
3	23:20:23.088	User 1-5	All Products	1520		4608	264	784	14
4	23:20:23.048	User 1-1	All Products	1571		4607	264	793	10
5	23:20:23.319	User 1-25	All Products	1308		4608	264	562	126
6	23:20:23.137	User 1-10	All Products	1499		4607	264	788	12
7	23:20:23.057	User 1-2	All Products	1597		4608	264	822	20
8	23:20:23.067	User 1-3	All Products	1587		4608	264	834	13
9	23:20:23.079	User 1-4	All Products	1578		4607	264	785	12
10	23:20:23.118	User 1-8	All Products	1571		4608	264	853	14
11	23:20:23.149	User 1-11	All Products	1540		4608	264	824	10
12	23:20:23.097	User 1-6	All Products	1627		4607	264	828	11
13	23:20:23.110	User 1-7	All Products	1614		4608	264	825	11
14	23:20:23.211	User 1-17	All Products	1526		4608	264	815	38
15	23:20:23.157	User 1-12	All Products	1581		4608	264	871	15
16	23:20:23.639	User 1-60	All Products	1120		4607	264	360	16
17	23:20:23.212	User 1-15	All Products	1547		4608	264	818	92
18	23:20:23.211	User 1-14	All Products	1567		4608	264	837	62
19	23:20:23.211	User 1-13	All Products	1587		4607	264	882	108
20	23:20:23.211	User 1-16	All Products	1603		4608	264	882	73

☐ Scroll automatically? ☐ Child samples? No of Samples 200 Latest Sample 283 Average 3840 Deviation 6376

- Result in tree form



Result in Graph form



Test 1: a.

Test Parameters

- 10 users
- 1 sec time interval between two consecutive requests
- 10 requests per users

Table form

View Results in Table

Name:

Comments:

Write results to file / Read from file

Filename Log/Display Only: ☐ Errors ☐ Successes

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
1	23:33:46.063	User 1-1	All Products	571	✓	4607	264	340	12
2	23:33:46.188	User 1-2	All Products	611	✓	4607	264	367	25
3	23:33:46.279	User 1-3	All Products	611	✓	4607	264	379	26
4	23:33:46.634	User 1-1	Seller Login	283	✓	603	279	283	0
5	23:33:46.364	User 1-4	All Products	564	✓	4607	264	334	30
6	23:33:46.465	User 1-5	All Products	530	✓	4607	264	311	23
7	23:33:46.798	User 1-2	Seller Login	285	✓	603	279	285	0
8	23:33:46.565	User 1-6	All Products	554	✓	4607	264	330	20
9	23:33:46.928	User 1-4	Seller Login	279	✓	603	279	279	0
10	23:33:46.665	User 1-7	All Products	560	✓	4607	264	334	29
11	23:33:46.763	User 1-8	All Products	612	✓	4607	264	356	34
12	23:33:47.120	User 1-6	Seller Login	288	✓	603	279	288	0
13	23:33:46.890	User 1-3	Seller Login	518	✓	603	279	518	0
14	23:33:46.864	User 1-9	All Products	578	✓	4607	264	341	23
15	23:33:46.996	User 1-5	Seller Login	482	✓	603	279	482	0
16	23:33:47.225	User 1-7	Seller Login	278	✓	603	279	278	0
17	23:33:46.965	User 1-10	All Products	542	✓	4607	264	332	24
18	23:33:47.375	User 1-8	Seller Login	289	✓	603	279	289	0
19	23:33:46.916	User 1-1	Category	748	✗	1368	280	295	0
20	23:33:47.443	User 1-9	Seller Login	280	✓	603	279	280	0

- Result in Graph form



Result in tree form

