BSc. (Hons) in Software Engineering
Department of Information Technology
Faculty of Computing

Module: Software Testing
Submission Date: 19th Sept 2025

Table of Contents

Test Plan and Final Report – Marimar Hotel Reservation System

Test Plan

Introduction

The 'Marimar' Hotel Reservation System (HRS) is a web-based application developed in PHP to simplify the process of hotel booking for customers and management for admins. The system allows customers to search for available rooms by entering check-in and check-out dates, number of persons and accommodation type. Customers can add rooms to the cart, register or log in to confirm bookings, view booking details, edit user account and print invoices after confirmation.

Admins are provided with features to view, confirm, edit, cancel and delete reservations, with customers notified of any updates via system messages. Additionally the system includes reporting functionality to help admin to generate booking and revenue reports.

This Test Plan outlines the objectives, scope and schedule for validating the functional and non-functional requirements of the Hotel Reservation System.

Objectives

- Ensure that customers can successfully search, add rooms to the cart, register/login and confirm bookings.
- Validate that core customer features such as view booking list, invoice printing and notification system work as expected.
- Verify that the admin features including reservation management (view, edit, confirm, cancel, delete) and report generation function correctly.
- Identify and document functional bugs and performance issues.
- Evaluate the system under concurrent user loads using JMeter.

Scope

I. **Customer Side:**

- Room search (with/without login)
- Cart operations (add/clear cart)
- User registration and login for booking confirmation
- Viewing booking details, list and invoice printing
- Receiving notifications when admin updates reservation

II. **Admin Side:**

- Manage reservations (view, edit, confirm, cancel, delete)
- Generate system reports

Team Roles

- Biyanga Kalupahana(QA Lead): Prepare test plan, assign roles, final reporting.
- Prabash Lakshitha(Functional Tester): Write Selenium test cases, execute functional tests.
- Kaveesha Muthugala(Performance Tester): Configure and execute JMeter load tests, analyze graphs.
- Biyanga Kalupahana(Documentation): Maintain bug report, compile screenshots, prepare final PDF.

Test schedule

Task	Tool	Duration	Responsible member
Test plan and setup	-	1 day	Biyanga Kalupahana (IT_UGC_001/B004/0031)
Functional testing	Selenium	2 days	Prabash Lakshitha (IT_UGC_001/B004/0011)
Performance testing	JMeter	1 day	Kaveesha Muthugala (IT_UGC_001/B004/0032)
Bug reporting and final report	-	2 day	Biyanga Kalupahana (IT_UGC_001/B004/0031)

Test cases

Test case ID:	TC 01 – TC 04					
Unit:	Search rooms					
Assumptions (If any):	1. Customer uses this system after hosting 2. The computer must connected to a network connection					
Test case No	**Description**	**Input data**	**Expected result**	**Actual result**	**Pass/ Fail**	**Comments**
TC 01	Searching for details for upcoming date	Check in: 2025 Sep 18 Check out: 2025 Sep 19 Person: 1 Accommodation: standard room	Display the details of "Wing A standard room" and "Wing B and ground floor standard room" details	Displayed the details of "Wing A standard room" and "Wing B and ground floor standard room" details	Pass	
TC 02	Searching for details for previous date	Check in: 2025 Sep 09 Check out: 2025 Sep 12 Person: 2 Accommodation: standard room	Customer could not have access to reserve a room that have already gone. Must have display an error message "Please select upcoming date"	Displayed the details of "Wing A standard room" details	Fail	
TC 03	Enter selections without No. of persons and then click Book Now button	Check in: 2025 Sep 19 Check out: 2025 Sep 22 Person: Do not select	Should not display suitable rooms for the reservation	Display a message "Available room From: 09/19/2020	Pass	

		Accommodation: standard room		To: 09/22/2025 | Standard Room" without displaying room details to make a booking		
TC 04	Enter selections without accommodation type and then click Book Now button	Check in: 2025 Sep 18 Check out: 2025 Sep 19 Person: 1 Accommodation: Do not select	Should not display suitable rooms for the reservation	Display a message "Available room From: 09/18/2020 To: 09/19/2025 |0" without displaying room details to make a booking	Pass	

Test case ID:	TC 01 – TC 02					
Unit:	Making bookings					
Assumptions (If any):	1. Customer uses this system after hosting 2. The computer must connected to a network connection 3. Customer has already logged in to the system					
Test case No	**Description**	**Input data**	**Expected result**	**Actual result**	**Pass/ Fail**	**Comments**
TC 01	Cancel reservation	Add multiple rooms to the cart. Navigate to the cart page. Click on the "Remove" to cancel the booking.	The cart should be updated after removing	The cart updated as expected	Pass	
TC 02	Complete the booking process	Successfully login to the system. Click on the "Submit booking" button.	System should display a success message	After successful login, the reservation details displayed. After click on the "Submit booking" button, system display success message as "Booking is successfully submitted!"	Pass	

Test case ID:	TC 01 – TC 04

Unit:	Registration					
Assumptions (If any):	1. Customer uses this system after hosting 2. The computer must connected to a network connection 3. Customer still don't have an account to login					
Test case No	**Description**	**Input data**	**Expected result**	**Actual result**	**Pass/ Fail**	**Comments**
TC 01	Fill all the details in the form correctly.	Input data to all the fields (first name: test1, last name: system, username: test1, password:test1, city: kaduwela, address: 20/A, DOB: 09/12/2002, nationality: Sinhala, zip code:12, phone: 0123456789, company: XYZ, Email: test1@gmail.com) Tick the checkbox of the terms and conditions. Click "Confirm" button.	Customer should successfully redirect to the payment page	Successfully redirected to the payment page	Pass	
TC 02	Fill the data and leave space to one/ or multiple data	Input data to the fields and leave space for "Nationality" (first name: test1, last name: system, username: test1, password:test1, city: kaduwela, address: 20/A, DOB: 09/12/2002, zip code:12, phone: 0123456789, company: XYZ, Email: test1@gmail.com)	Display an error message	Display an error message "All fields are required!"	Pass	

		Tick the checkbox of the terms and conditions. Click “Confirm” button.				
TC 03	Provide email address with incorrect format	Input data to all the fields (first name: test3, last name: system, username: test3, password:test3, city: kaduwela, address: 20/A, DOB: 09/12/2002, nationality: Sinhala, zip code:12, phone: 0123456789, company: XYZ, Email: test3com) Tick the checkbox of the terms and conditions. Click “Confirm” button.	System should display an error message	Display an error message “Please include an ‘@’ in the email address ‘test3’ is missing an ‘@’.”	Pass	
TC 04	Do not tick to the terms and conditions checkbox	Input data to all the fields (first name: test1, last name: system, username: test1, password:test1, city: kaduwela, address: 20/A, DOB: 09/12/2002, nationality: Sinhala, zip code:12, phone: 0123456789, company: XYZ, Email: test1@gmail.com) Click “Confirm” button.	System should display an error message	Display an error message “pls. agree the term and condition of this hotel”	Pass	

Test case ID:	TC 01 – TC 04					
Unit:	Login					
Assumptions (If any):	1. Customer uses this system after hosting 2. The computer must connected to a network connection					
Test case No	**Description**	**Input data**	**Expected result**	**Actual result**	**Pass/ Fail**	**Comments**
TC 01	Successfully enter the username and password	Enter username: test1 Password: test1 Then click "Sign In" button	Successfully redirect to the payment page	Successfully redirected to the payment page with booking details	Pass	
TC 02	Enter incorrect password	Enter username: test1 Password: test12 Then click "Sign In" button	Display an error message	Display an error message "Invalid Username and Password! Please contact administrator".	Pass	
TC 03	Enter incorrect username	Enter username: anne Password: test1 Then click "Sign In" button	Display an error message	Display an error message "Invalid Username and Password! Please contact administrator".	Pass	
TC 04	Try to login without signup to the system	Enter username: test12 Password: test12 Then click "Sign In" button	Display an error message	Display an error message "Invalid Username and Password! Please contact administrator".	Pass	

Test case ID:	TC 01 – TC 03
Unit:	User icon and Booking details

Assumptions (If any):	1. Customer uses this system after hosting 2. The computer must connected to a network connection 3. Customer have logged in to the system successfully					
Test case No	**Description**	**Input data**	**Expected result**	**Actual result**	**Pass/ Fail**	**Comments**
TC 01	View reservation details	Successfully login to the system. Navigate to the Booking in the user icon.	Display the list of booked rooms	Display all reservations under that user account.	Pass	
TC 02	Edit account details	Successfully login to the system. Navigate to the Account section in the user icon. Edit the city as "Kottawa" and save.	Display the new account details after update	City changed as "Kottawa"	Pass	
TC 03	Add image for the account	Successfully login to the system. Navigate to the "Use Avatar" in the user icon. Choose a file and click on the "Upload photo" button.	The photo should be display in the user icon.	The photo displays above to the first name and last name of the user.	Pass	

Test case ID:	TC 01 – TC 04
Unit:	Contact form

Assumptions (If any):	1. Customer uses this system after hosting 2. The computer must connected to a network connection					
Test case No	**Description**	**Input data**	**Expected result**	**Actual result**	**Pass/ Fail**	**Comments**
TC 01	Fill all the fields of contact form	Type Name: Biyanga Email: biya@gmail.com Subject: new Message: Hello guys! Click on the "Send message" button	Display a successful message	Display a successful message "Message sent successfully!"	Pass	
TC 02	Leave the fields as empty	Click on "Send message" button	Display an error message	Display an error message "Please fill out this field!"	Pass	
TC 03	Leave one field of the form empty (Email/ Message)	Type Name: Biyanga Subject: new Message: Hello guys! Click on the "Send message" button	Display an error message	Display an error message "Please fill out this field!"	Pass	
TC 04	Leave one field of the form empty (Subject)	Type Name: Biyanga Email: biya1@gmail.com Message: Hello guys! Click on the "Send message" button	System should display a successful message	Display a successful message "Message sent successfully!"	Pass	

Functional testing

Room search and booking

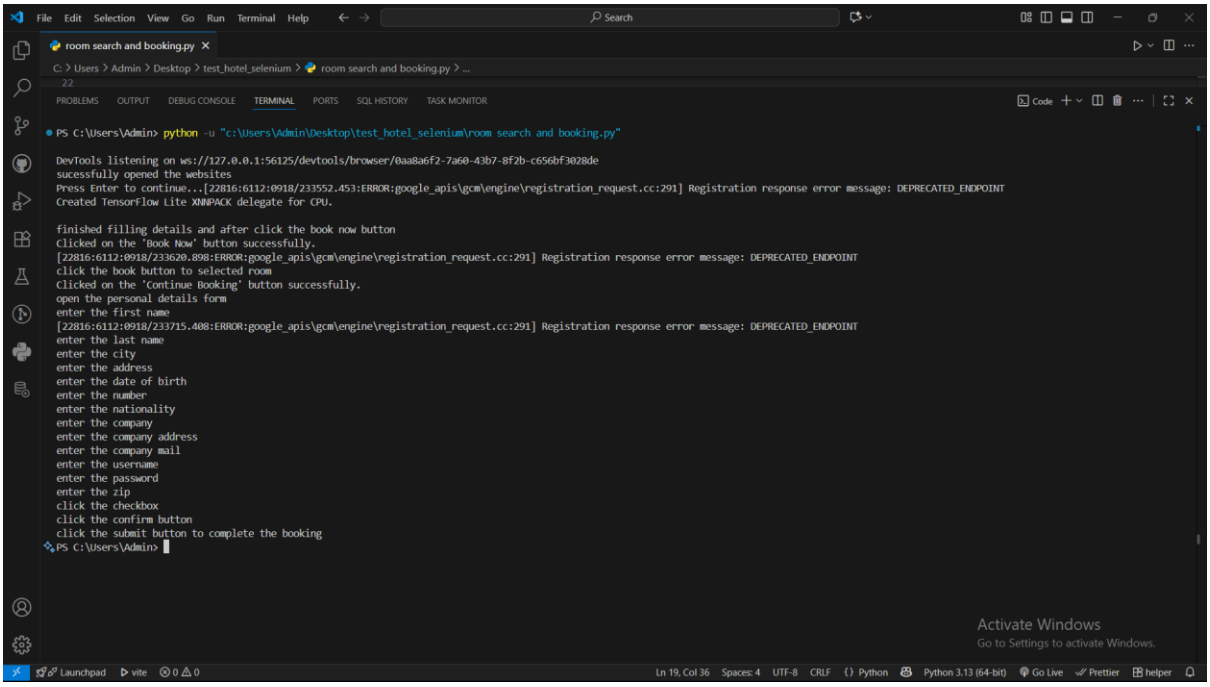
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.service import Service
import time
#create service with path to chromedriver
service=Service(ChromeDriverManager().install())

driver = webdriver.Chrome(service=service)


driver.get("http://localhost/marimar/")
print("sucessfully opened the websites")
input("Press Enter to continue...")




print("finished filling details and after click the book now button")
time.sleep(5)


try:
    book_button = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.CLASS_NAME, "booking_form_button"))
    )
    book_button.click()
    print("Clicked on the 'Book Now' button successfully.")

except Exception as e:
    print(f"Error: The button could not be clicked. The element might not be
available. Details: {e}")


#keep open browser seconds 15
time.sleep(15)
```

```
book1_buttom= WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.ID,"booknow"))
)

book1_buttom.click()
print("click the book button to selected room")

time.sleep(15)

try:

    continue_button = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.XPATH, "//a[contains(text(), 'Continue
Booking')]"))
    )
    continue_button.click()
    print("Clicked on the 'Continue Booking' button successfully.")




except Exception as e:
    print(f"Error: The 'Continue Booking' link could not be clicked. Details:
{e}")

time.sleep(15)

#user register and fill the personal details form

register_link = driver.find_element(By.XPATH, "//a[contains(text(),
'Register')]")
register_link.click()

time.sleep(5)
form=driver.find_element(By.NAME,"personal")
print("open the personal details form")
time.sleep(2)

first_name = form.find_element(By.ID, "name")
first_name.clear()
first_name.send_keys("prabash")
print("enter the first name")
time.sleep(2)

last_name = form.find_element(By.ID, "last")
```

```
last_name.clear()
last_name.send_keys("lakshitha")
print("enter the last name")
time.sleep(2)

city = form.find_element(By.ID, "city")
city.clear()
city.send_keys("colombo")
print("enter the city")
time.sleep(2)

address = form.find_element(By.ID, "address")
address.clear()
address.send_keys("155/a,kolonnawa,colombo 15")
print("enter the address")
time.sleep(2)

driver.execute_script("document.getElementById('dbirth').value = '2000-01-01'")
print("enter the date of birth")
time.sleep(2)

num = form.find_element(By.ID, "phone")
num.clear()
num.send_keys("0772087845")
print("enter the number")
time.sleep(2)

nationality = form.find_element(By.ID, "nationality")
nationality.clear()
nationality.send_keys("sri lankan")
print("enter the nationality")
time.sleep(2)

company = form.find_element(By.ID, "company")
company.clear()
company.send_keys("GMC PVT LTD")
print("enter the company")
time.sleep(2)

caddress = form.find_element(By.ID, "caddress")
caddress.clear()
caddress.send_keys("12/a,kolonnawa,colombo 15")
print("enter the company address")
time.sleep(2)
```

```
cemail = form.find_element(By.ID, "cemail")
cemail.clear()
cemail.send_keys("GMC35@gmail.com")
print("enter the company mail")
time.sleep(2)

element = form.find_element(By.ID, "username")
driver.execute_script("arguments[0].scrollIntoView(true);", element)
element.send_keys("prabash123")
print("enter the username")
time.sleep(2)

password = form.find_element(By.ID, "password")
password.clear()
password.send_keys("Prabash@123")
print("enter the password")
time.sleep(2)

zip = form.find_element(By.ID, "zip")
zip.clear()
zip.send_keys("110856")
print("enter the zip")
time.sleep(2)

checkbox = form.find_element(By.NAME, "condition")
checkbox.click()
print("click the checkbox")
time.sleep(2)

confirm_button = form.find_element(By.NAME, "submit")
confirm_button.click()
print("click the confirm button")

time.sleep(15)

submit_button = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.NAME, "btnsubmitbooking"))
)
submit_button.click()
print("click the submit button to complete the booking")
time.sleep(10)
```

Output screenshots:



(Figure 1: All details are correctly executes)

Registration

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.service import Service
import time

#Room search and booking automation script

#create service with path to chromedriver
service=Service(ChromeDriverManager().install())

driver = webdriver.Chrome(service=service)


driver.get("http://localhost/marimar/")
print("sucessfully opened the websites")

input("After filling details and press enter to continue...")
```

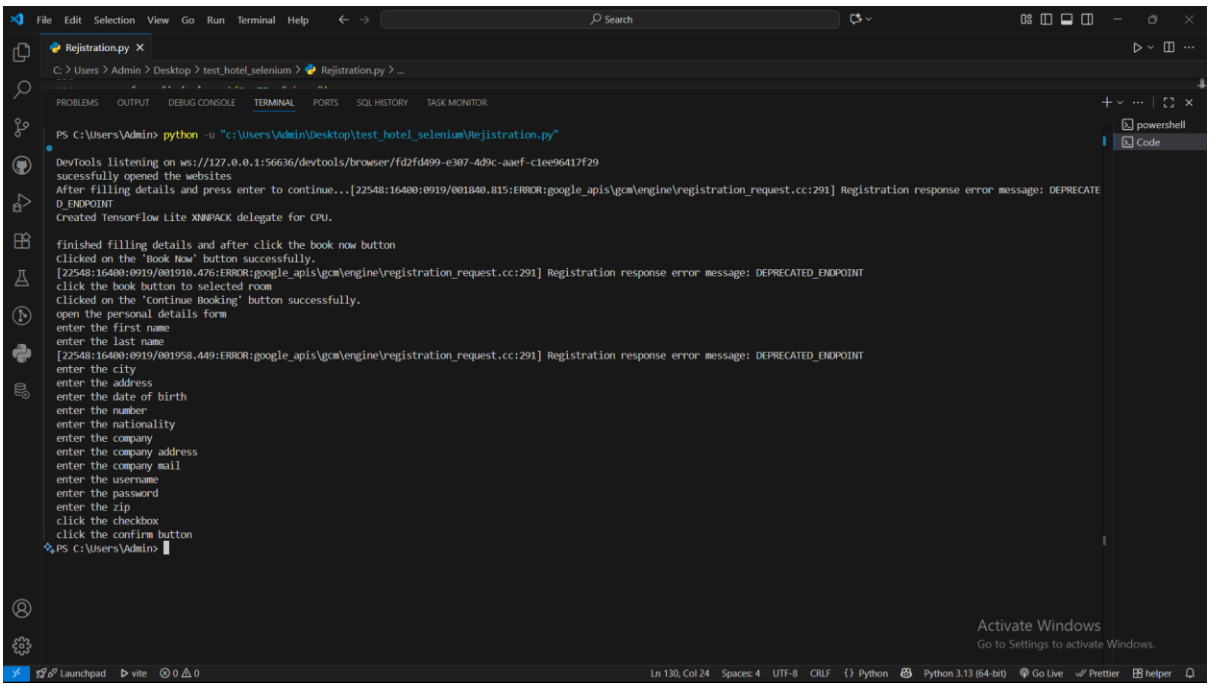
```
print("finished filling details and after click the book now button")
time.sleep(5)


try:
    book_button = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.CLASS_NAME, "booking_form_button"))
    )
    book_button.click()
    print("Clicked on the 'Book Now' button successfully.")

except Exception as e:
    print(f"Error: The button could not be clicked. The element might not be
available. Details: {e}")


#keep open browser seconds 15
time.sleep(15)


book1_buttom= WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.ID,"booknow"))
)

book1_buttom.click()
print("click the book button to selected room")

time.sleep(15)

try:

    continue_button = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.XPATH, "//a[contains(text(), 'Continue
Booking')]"))
    )
    continue_button.click()
    print("Clicked on the 'Continue Booking' button successfully.")




except Exception as e:
    print(f"Error: The 'Continue Booking' link could not be clicked. Details:
{e}")
```

```
time.sleep(15)


#user register and fill the personal details form

register_link = driver.find_element(By.XPATH, "//a[contains(text(),
'Register')]")
register_link.click()

time.sleep(5)
form=driver.find_element(By.NAME,"personal")
print("open the personal details form")
time.sleep(2)

first_name = form.find_element(By.ID, "name")
first_name.clear()
first_name.send_keys("prabash")
print("enter the first name")
time.sleep(2)

last_name = form.find_element(By.ID, "last")
last_name.clear()
last_name.send_keys("lakshitha")
print("enter the last name")
time.sleep(2)

city = form.find_element(By.ID, "city")
city.clear()
city.send_keys("colombo")
print("enter the city")
time.sleep(2)

address = form.find_element(By.ID, "address")
address.clear()
address.send_keys("212/a,kolonnawa,colombo 15")
print("enter the address")
time.sleep(2)

driver.execute_script("document.getElementById('dbirth').value = '2000-01-01'")
print("enter the date of birth")
time.sleep(2)

num = form.find_element(By.ID, "phone")
num.clear()
num.send_keys("0772087845")
```

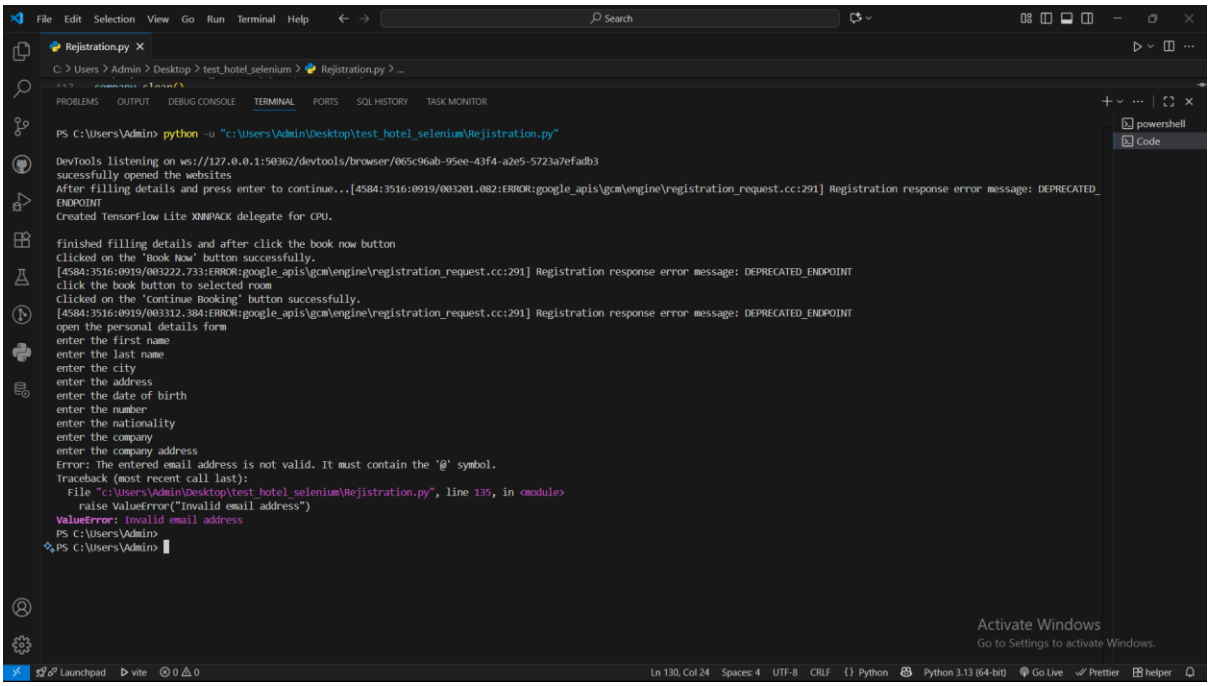
```
print("enter the number")
time.sleep(2)

nationality = form.find_element(By.ID, "nationality")
nationality.clear()
nationality.send_keys("sri lankan")
print("enter the nationality")
time.sleep(2)

company = form.find_element(By.ID, "company")
company.clear()
company.send_keys("GMC PVT LTD")
print("enter the company")
time.sleep(2)

caddress = form.find_element(By.ID, "caddress")
caddress.clear()
caddress.send_keys("12/a,kolonnawa,colombo 15")
print("enter the company address")
time.sleep(2)

cemail = form.find_element(By.ID, "cemail")
cemail.clear()
cemail.send_keys("GMC35@gmail.com")

if '@' not in cemail.get_attribute('value'):
    print("Error: The entered email address is not valid. It must contain the '@'
symbol.")
    driver.close()
    raise ValueError("Invalid email address")
print("enter the company mail")
time.sleep(2)

element = form.find_element(By.ID, "username")
driver.execute_script("arguments[0].scrollIntoView(true);", element)
element.send_keys("prabash123")
print("enter the username")
time.sleep(2)

password = form.find_element(By.ID, "password")
password.clear()
password.send_keys("Prabash@123")
print("enter the password")
time.sleep(2)
```

```
zip = form.find_element(By.ID, "zip")
zip.clear()
zip.send_keys("110856")
print("enter the zip")
time.sleep(2)

checkbox = form.find_element(By.NAME, "condition")
checkbox.click()



if not checkbox.is_selected():
    print("want agree the term and condition")
    driver.close()
    raise Exception("The terms and conditions checkbox was not selected.")
print("click the checkbox")
time.sleep(2)

confirm_button = form.find_element(By.NAME, "submit")
confirm_button.click()
print("click the confirm button")
```

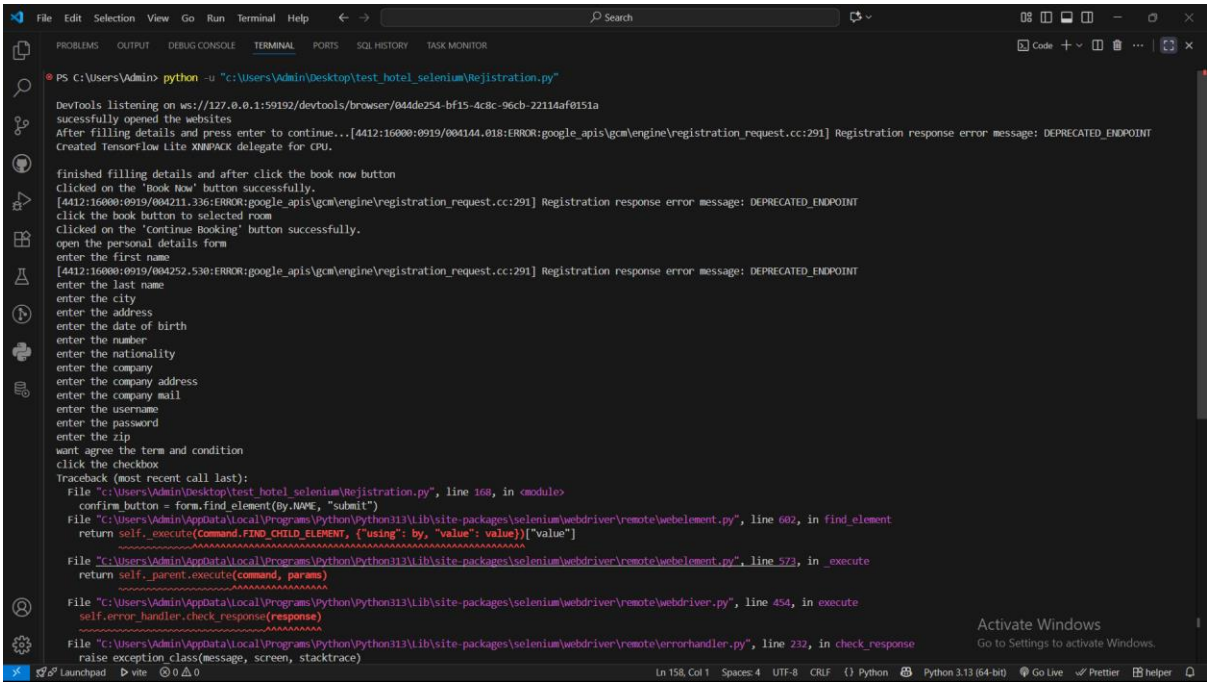
Output screenshots:



(Figure 2: Successful registration)



(Figure 3: Email field missing an '@' symbol)



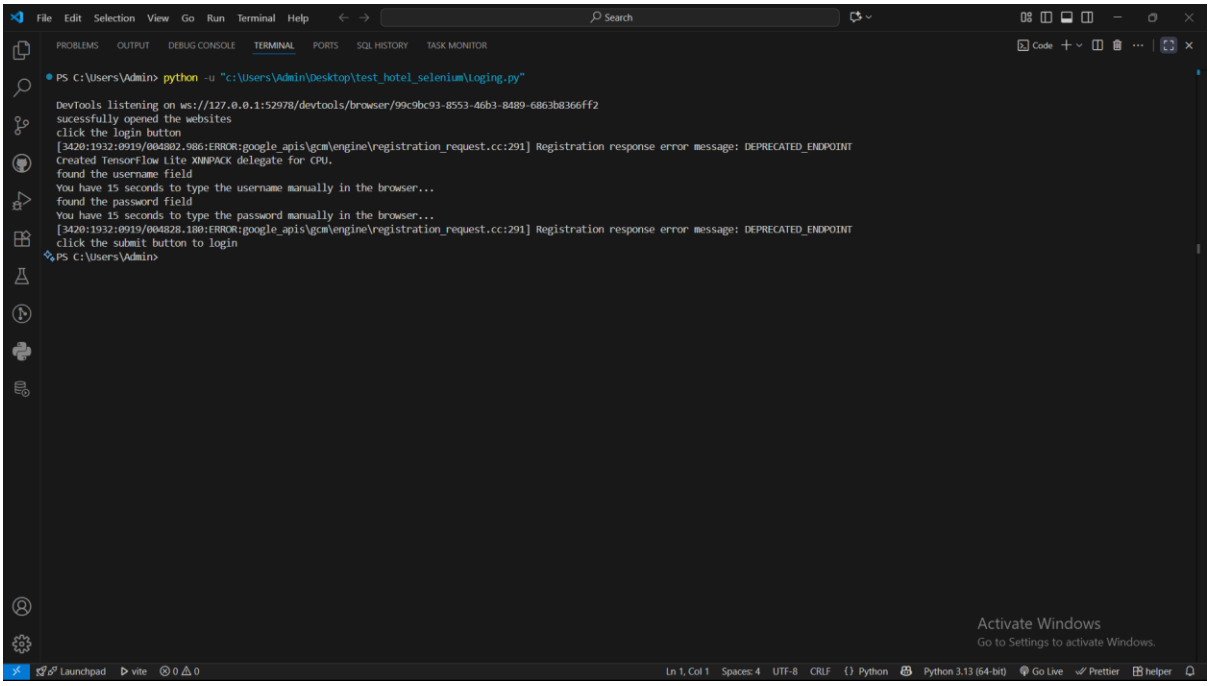
(Figure 4: Do not tik the checkbox in front of terms and conditions)

Login

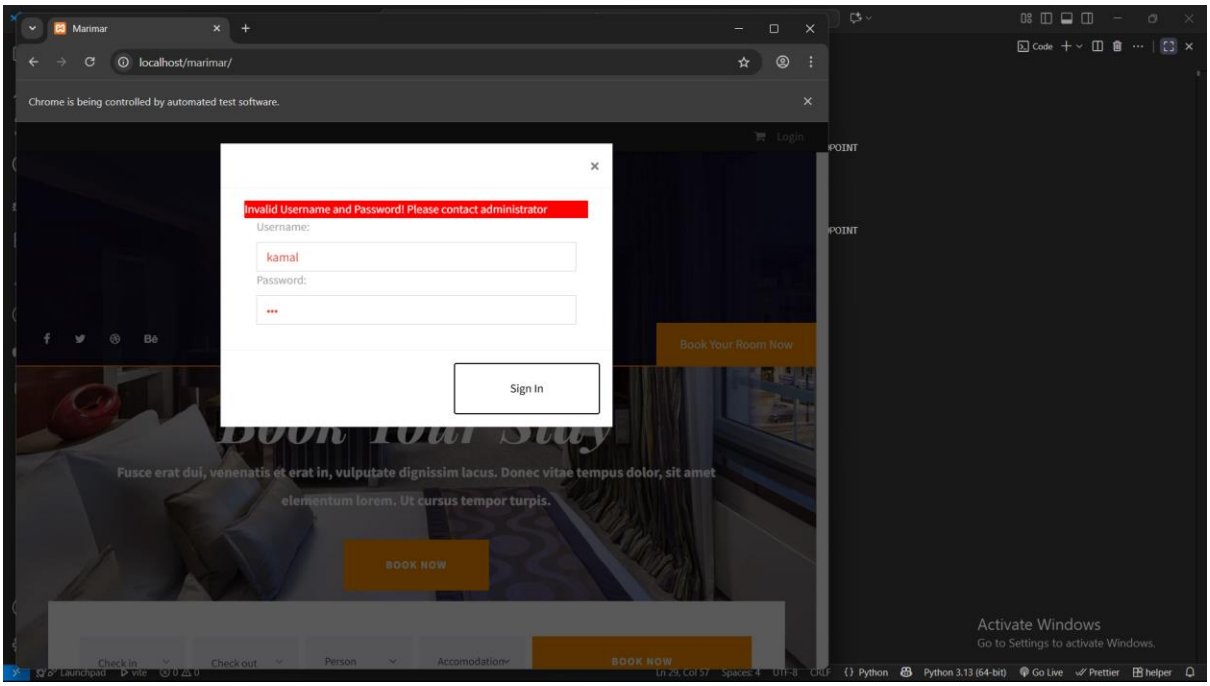
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.service import Service
import time

#Room search and booking automation script

#create service with path to chromedriver
service=Service(ChromeDriverManager().install())

driver = webdriver.Chrome(service=service)


driver.get("http://localhost/marimar/")
print("sucessfully opened the websites")


login_button = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.XPATH, "//a[@title='Login Guest']"))
)
login_button.click()
print("click the login button")
time.sleep(10)

username_field=WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.ID,"U_USERNAME"))
)
print("found the username field")
print("You have 15 seconds to type the username manually in the browser...")
time.sleep(15)

passwor_field=WebDriverWait(driver,10).until(
    EC.presence_of_element_located((By.ID,"U_PASS"))
)
print("found the password field")
print("You have 15 seconds to type the password manually in the browser...")
time.sleep(15)

time.sleep(7)
submit_button=WebDriverWait(driver,10).until(
```

```
    EC.element_to_be_clickable((By.NAME,"btnLogin"))
)
submit_button.click()
print("click the submit button to login")
time.sleep(10)
```

Output screenshots:



(Figure 5: Successful login)



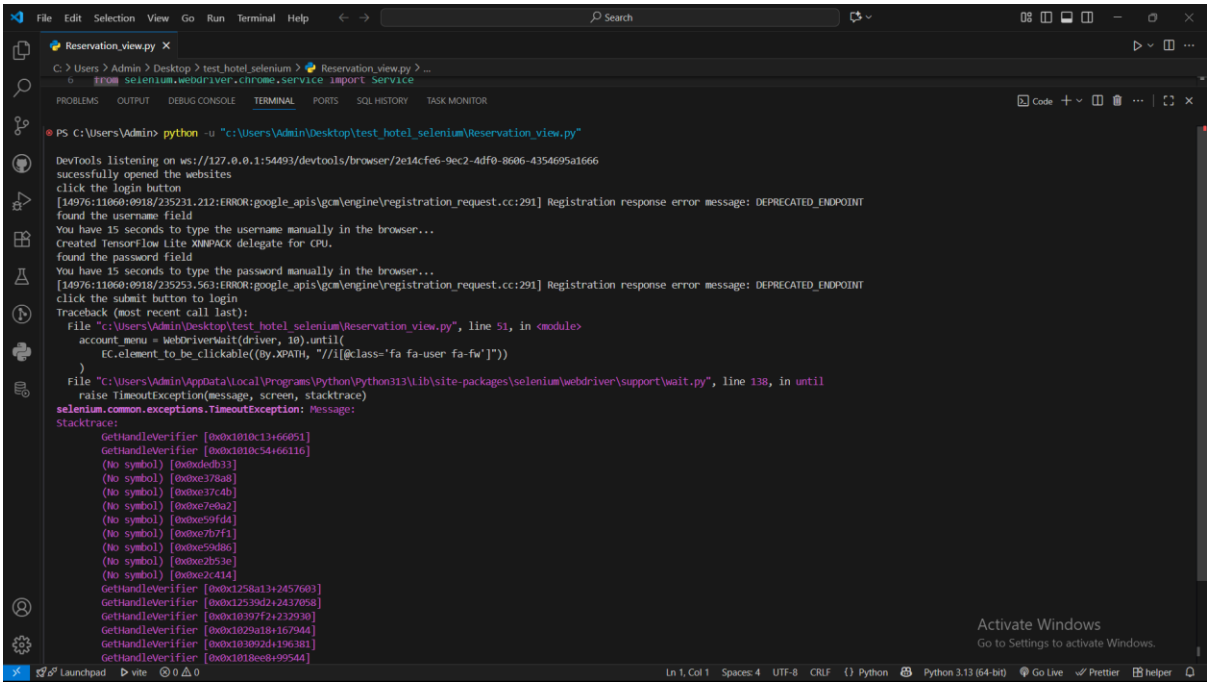
(Figure 6: Enter incorrect username or password)

View booking history

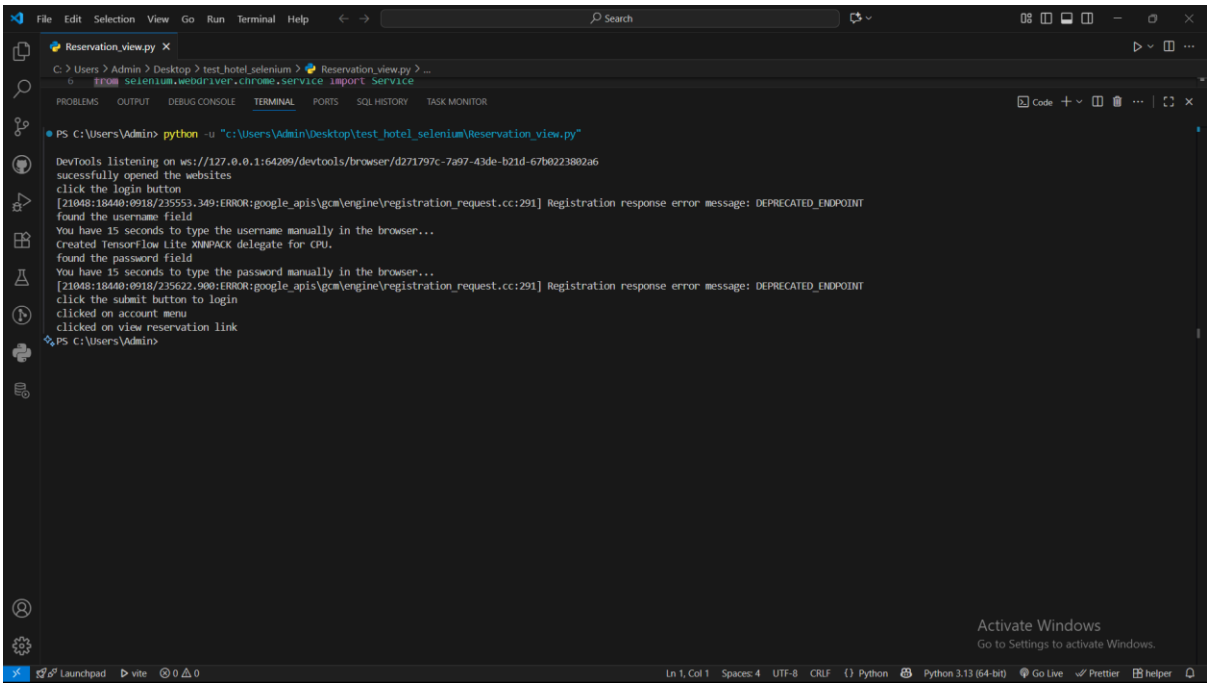
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.service import Service
import time

#Room search and booking automation script

#create service with path to chromedriver
service=Service(ChromeDriverManager().install())

driver = webdriver.Chrome(service=service)


driver.get("http://localhost/marimar/")
print("sucessfully opened the websites")


login_button = WebDriverWait(driver, 10).until(
```

```
    EC.element_to_be_clickable((By.XPATH, "//a[@title='Login Guest']"))
)
login_button.click()
print("click the login button")
time.sleep(10)

username_field=WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.ID,"U_USERNAME"))
)
print("found the username field")
print("You have 15 seconds to type the username manually in the browser...")
time.sleep(15)

passwor_field=WebDriverWait(driver,10).until(
    EC.presence_of_element_located((By.ID,"U_PASS"))
)
print("found the password field")
print("You have 15 seconds to type the password manually in the browser...")
time.sleep(15)

time.sleep(7)
submit_button=WebDriverWait(driver,10).until(
    EC.element_to_be_clickable((By.NAME,"btnLogin"))
)
submit_button.click()
print("click the submit button to login")
time.sleep(10)


account_menu = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.XPATH, "//i[@class='fa fa-user fa-fw']"))
)
account_menu.click()
print("clicked on account menu")
time.sleep(3)

view_resevation= booking_link = driver.find_element(By.XPATH,
"//a[@href='/marimar/guest/bookinglist.php']")
booking_link.click()
print("clicked on view reservation link")
time.sleep(10)
```

Output screenshots:



(Figure 7: Enter incorrect password or username to logged in to view bookings. Customer must have successfully logged in to the system to view booking history)



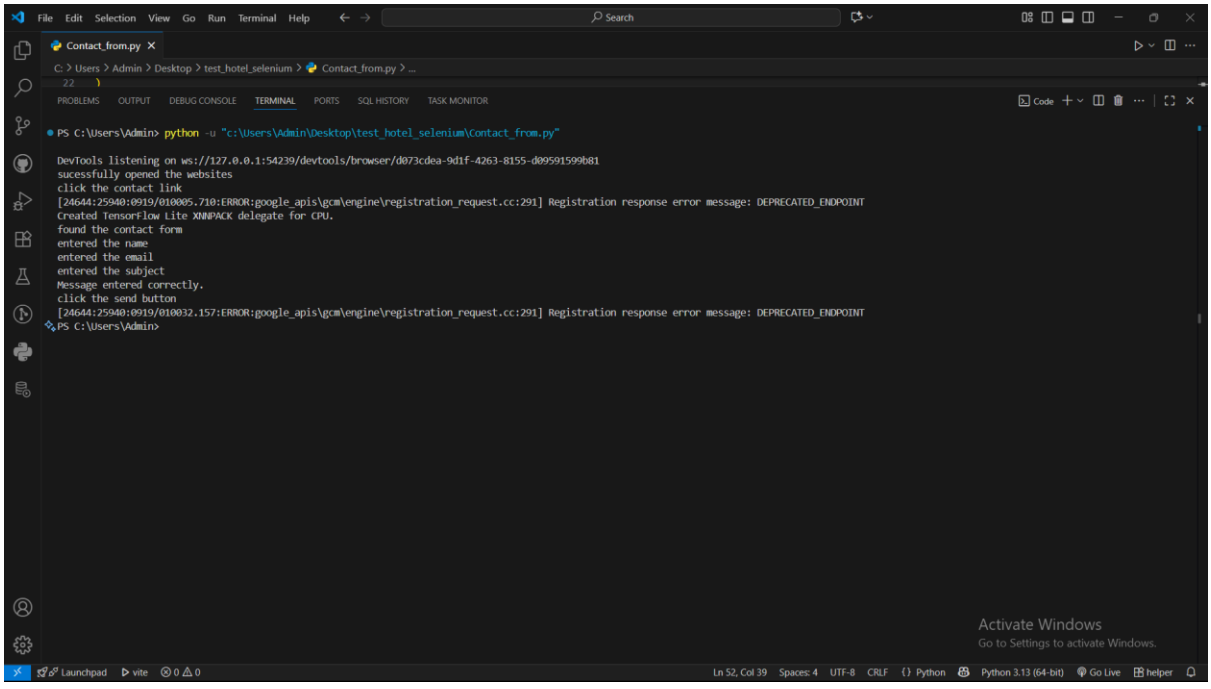
(Figure 8: Successful view of reservations)

Contact form

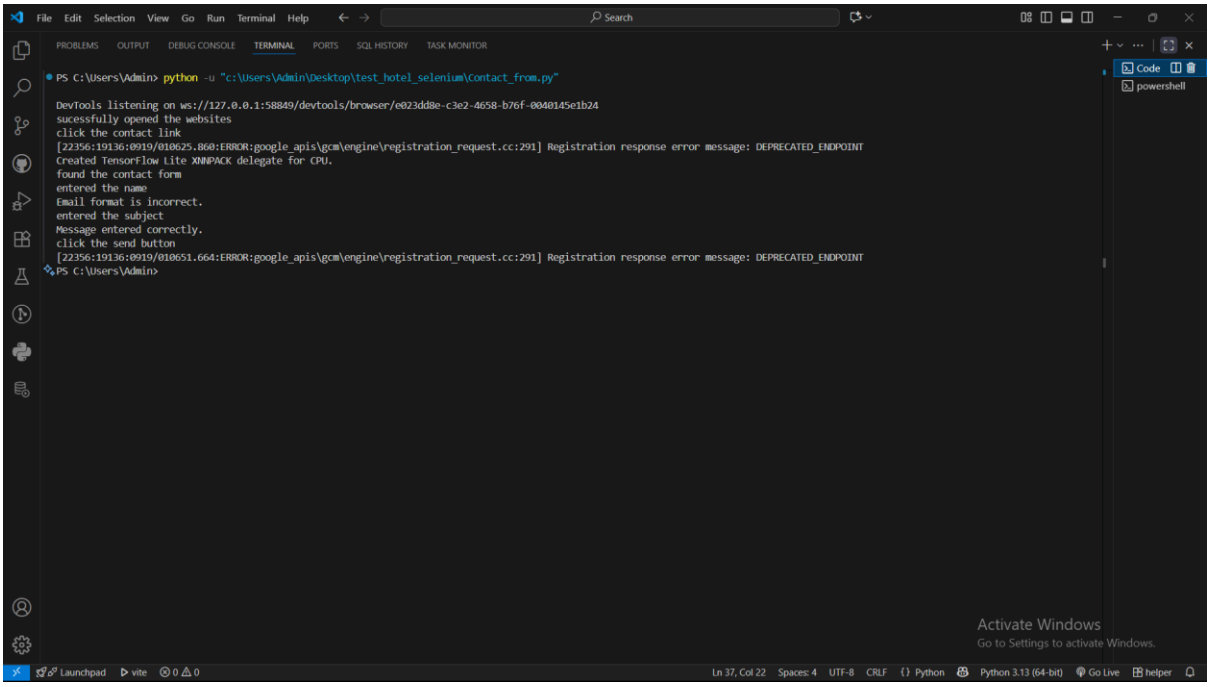
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.service import Service
import time

#Room search and booking automation script

#create service with path to chromedriver
service=Service(ChromeDriverManager().install())

driver = webdriver.Chrome(service=service)


driver.get("http://localhost/marimar/")
print("sucessfully opened the websites")

contact_link = WebDriverWait(driver,10).until(
    EC.element_to_be_clickable((By.XPATH,
"//a[@href='/marimar/index.php?p=contact']"))
)
contact_link.click()
print("click the contact link")
time.sleep(15)

form=driver.find_element(By.XPATH, "//form[@action='contact.php']")
print("found the contact form")
time.sleep(3)

name=form.find_element(By.NAME,"name")
name.send_keys("Test User")
print("entered the name")
time.sleep(2)

email=form.find_element(By.NAME,"email")
email.send_keys("tet1@gmail.com")
if "@"in email.get_attribute("value"):
        print("Email format is correct.")
else:
        print("Email format is incorrect.")
```

```
time.sleep(2)

sublect=form.find_element(By.NAME,"subject")
sublect.send_keys("Test Subject")
print("entered the subject")
time.sleep(2)

message=form.find_element(By.NAME,"message")
message.send_keys()

if message.get_attribute('value') == "This is a test message for contact form.":
        print("Message entered correctly.")
else:
        print("Message entry failed.")

time.sleep(2)

button=driver.find_element(By.XPATH, "//button[@class='contact_button']")
button.click()
print("click the send button")
time.sleep(10)
```

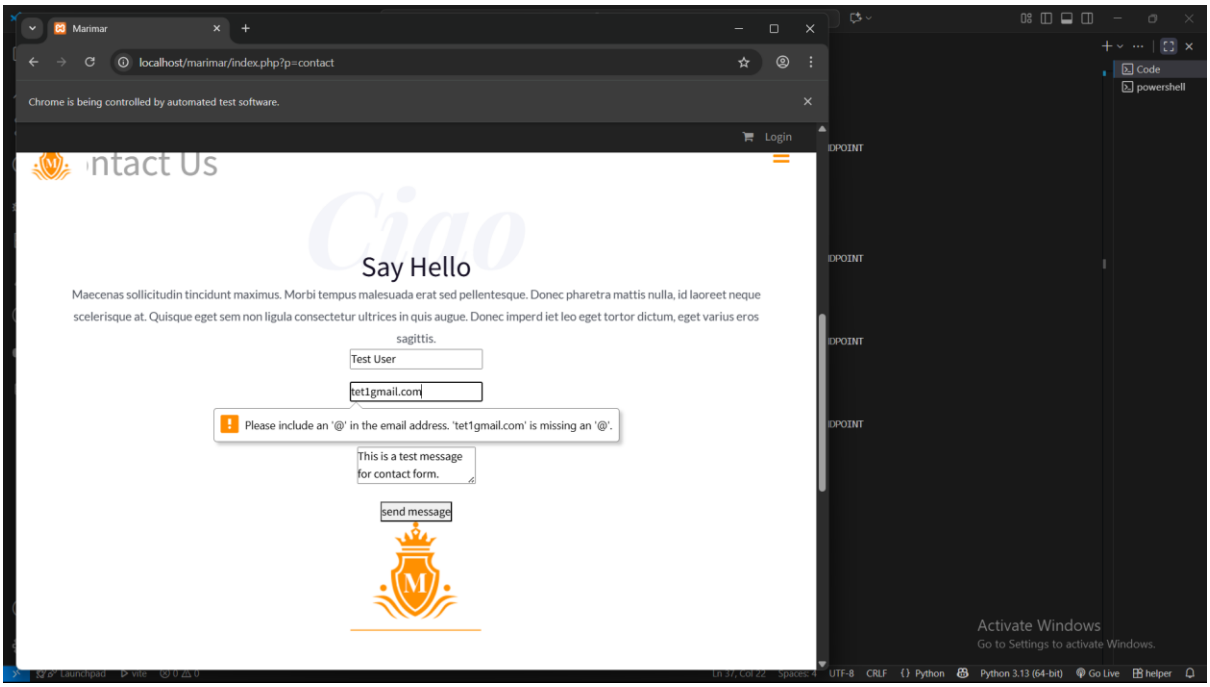
Output screenshots:



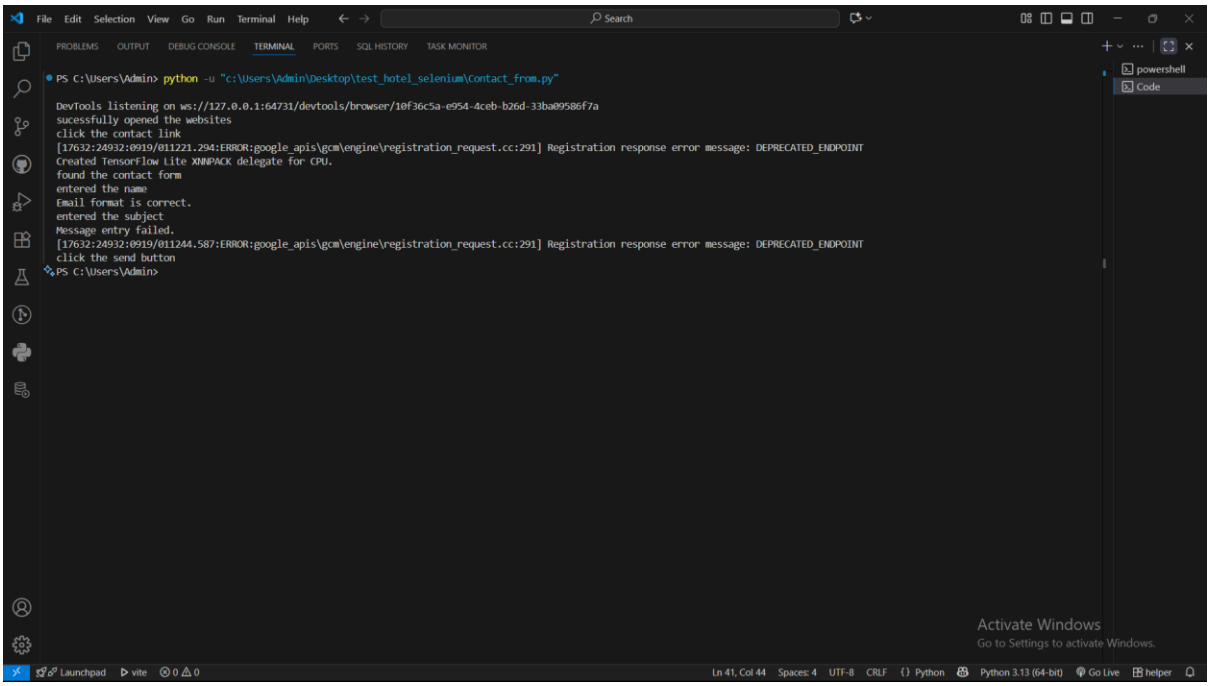
(Figure 9: Fill all fields of the contact form)



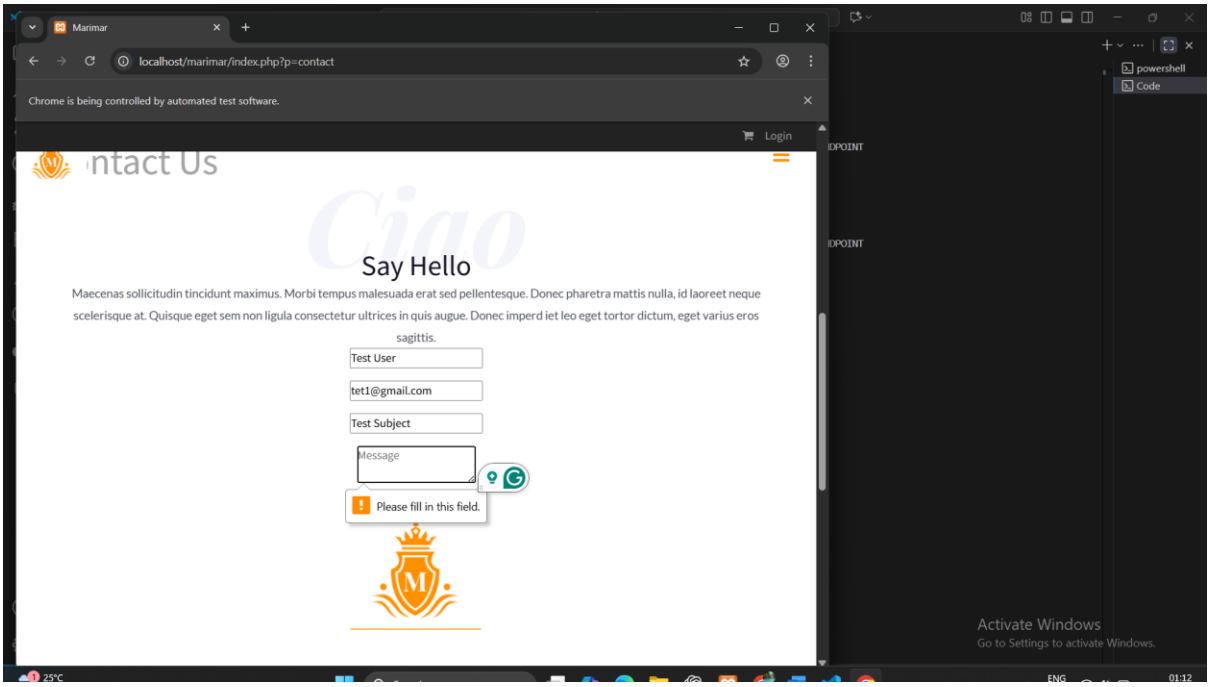
(Figure 10)



(Figure 11: Do not use the correct format(missing '@') of email)



(Figure 12)



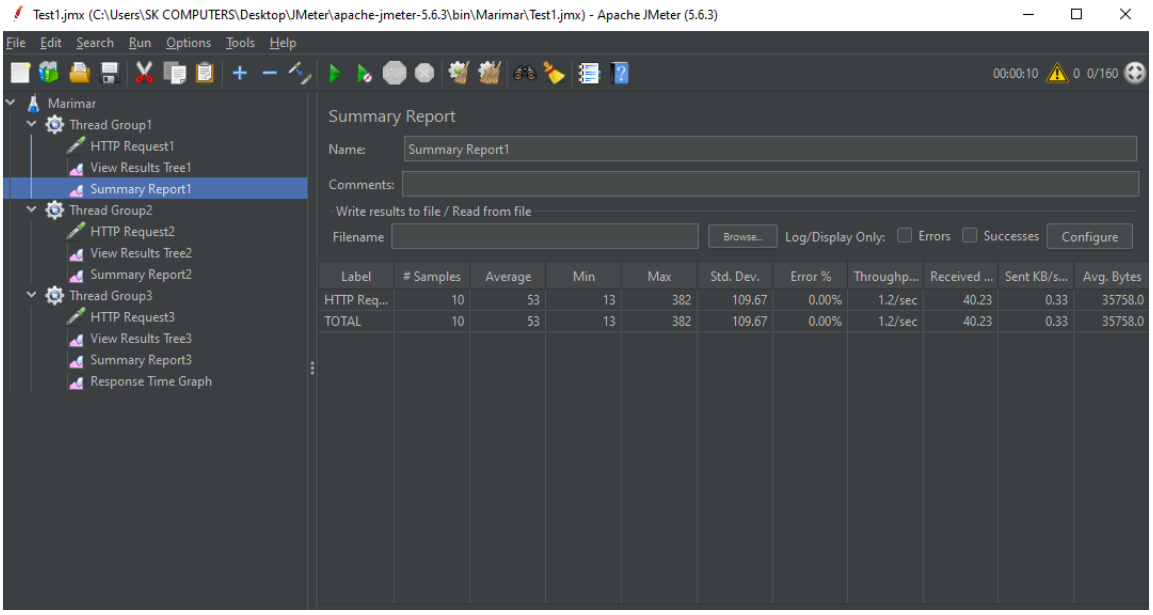
(Figure 13: Form fill and submit without filling message field)

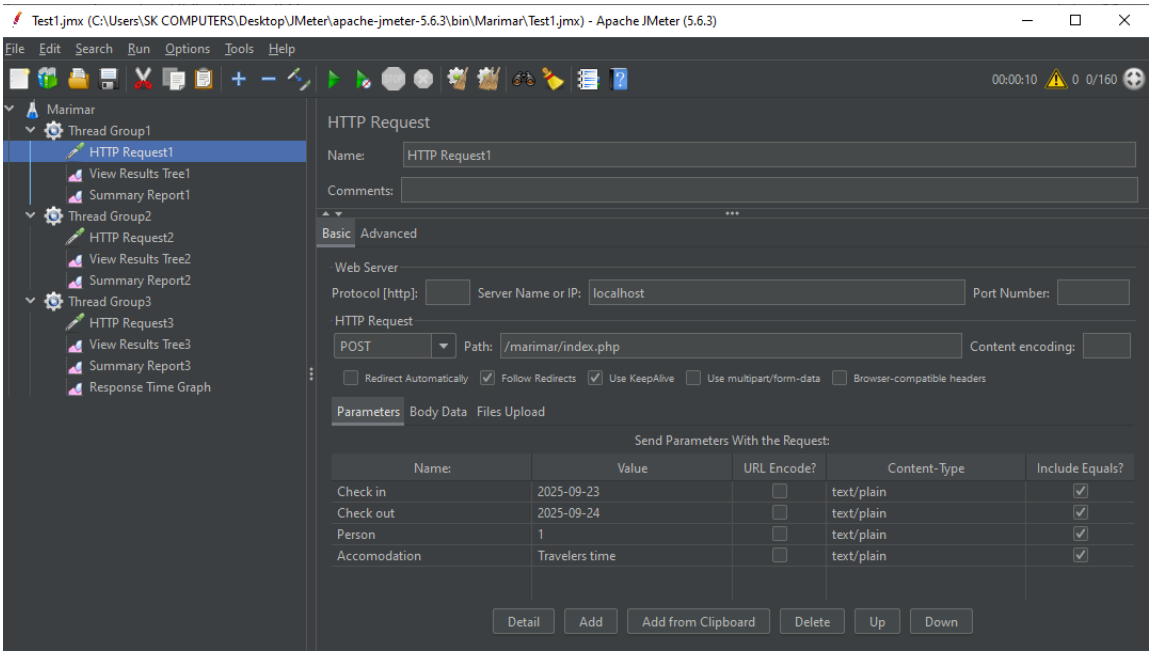
Performance testing

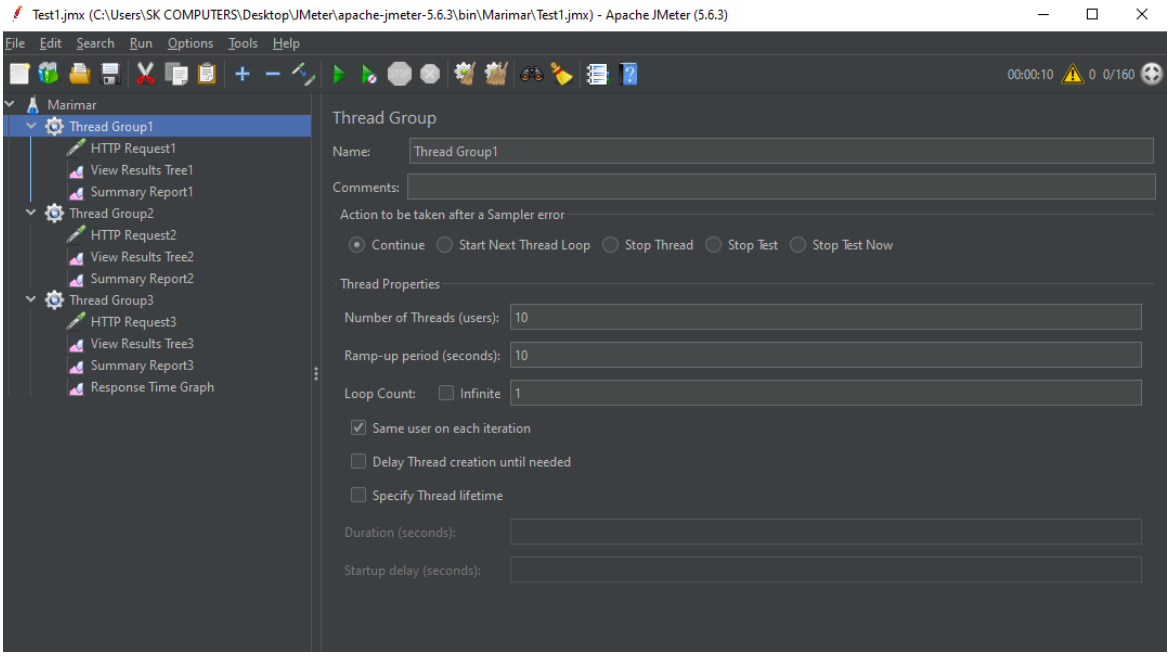
Search room function

- All the graphs for each thread group in the zip file.

10 concurrent users:

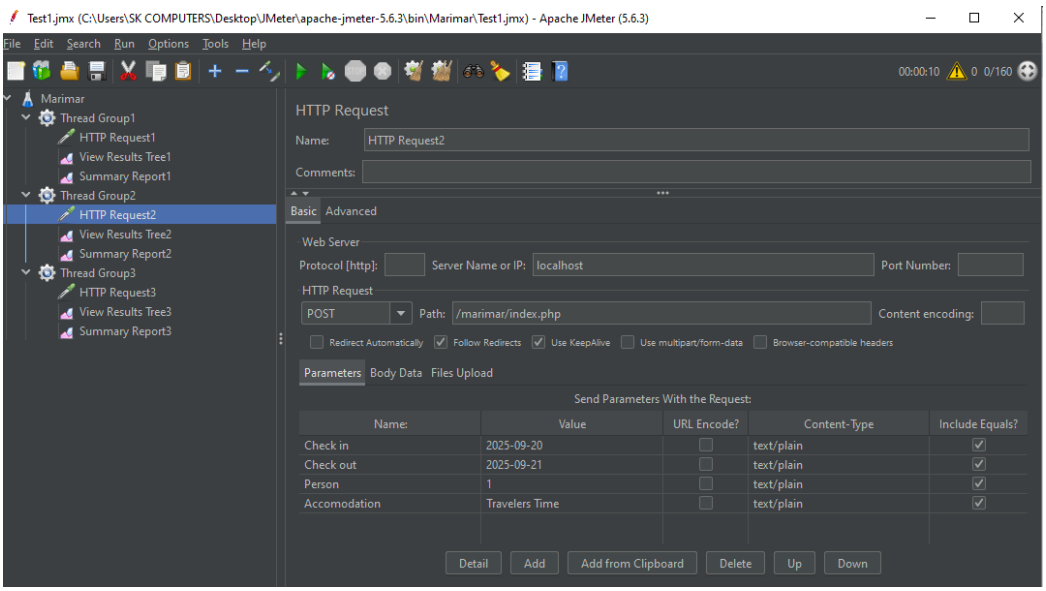


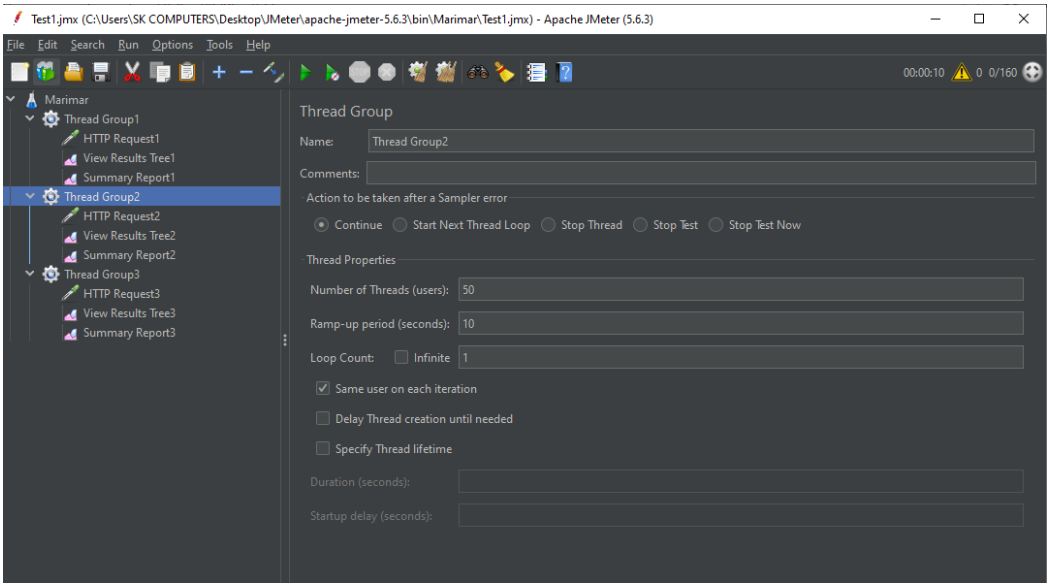


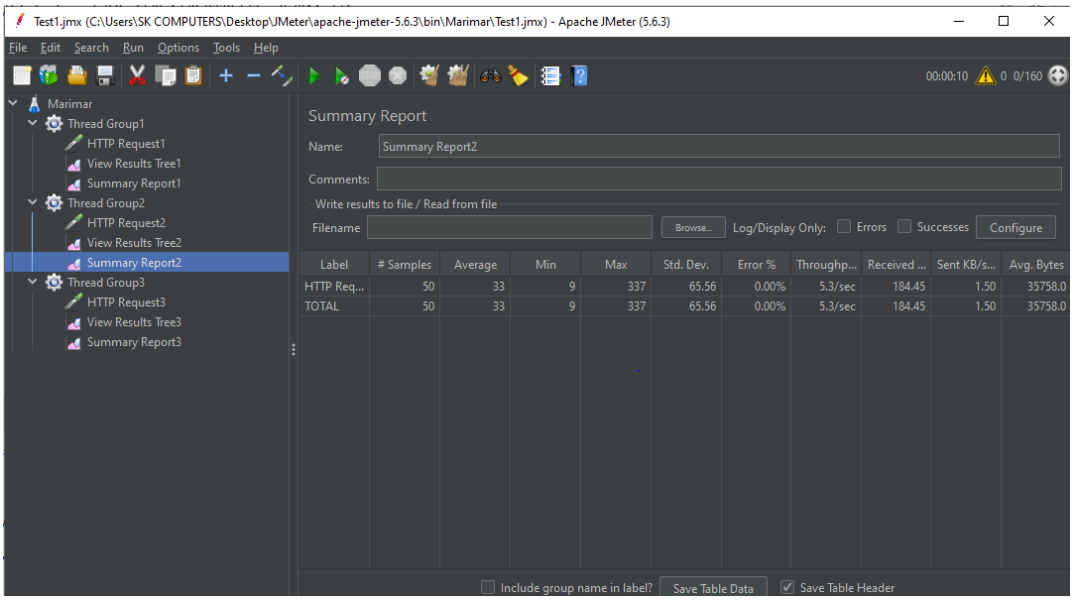


Metrics	**Value**
Users	10 requests were sent
Average	The average response time is a very good 53 milliseconds.
Min/ Max	The minimum response time was 13 ms, while the maximum was 382 ms. This maximum time is still well within an acceptable range.
Std.dev	109.67 ms. The standard deviation is high compared to the average, indicating that there was a large variance in response times among the requests.
Error %	All requests succeeded.
Throughput	The server processed 1.2 requests per second.
Received KB/sec	40.23 Data received from server per second.
Sent KB/sec	0.33 Data sent to server per second.
Avg.bytes	35758.0 Average size of each response.

50 concurrent users:

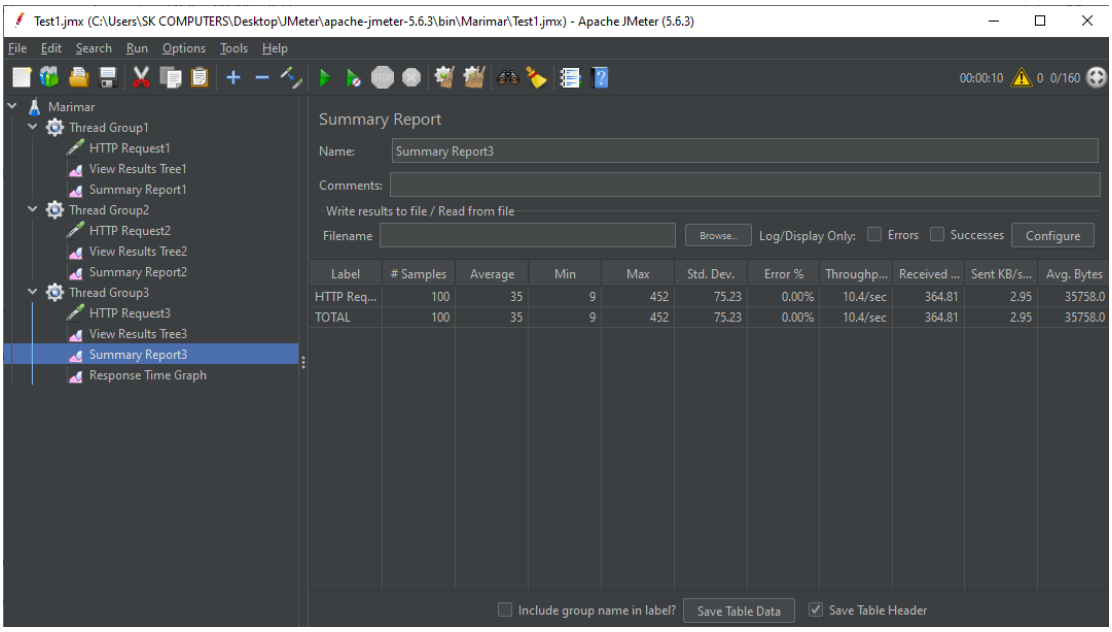


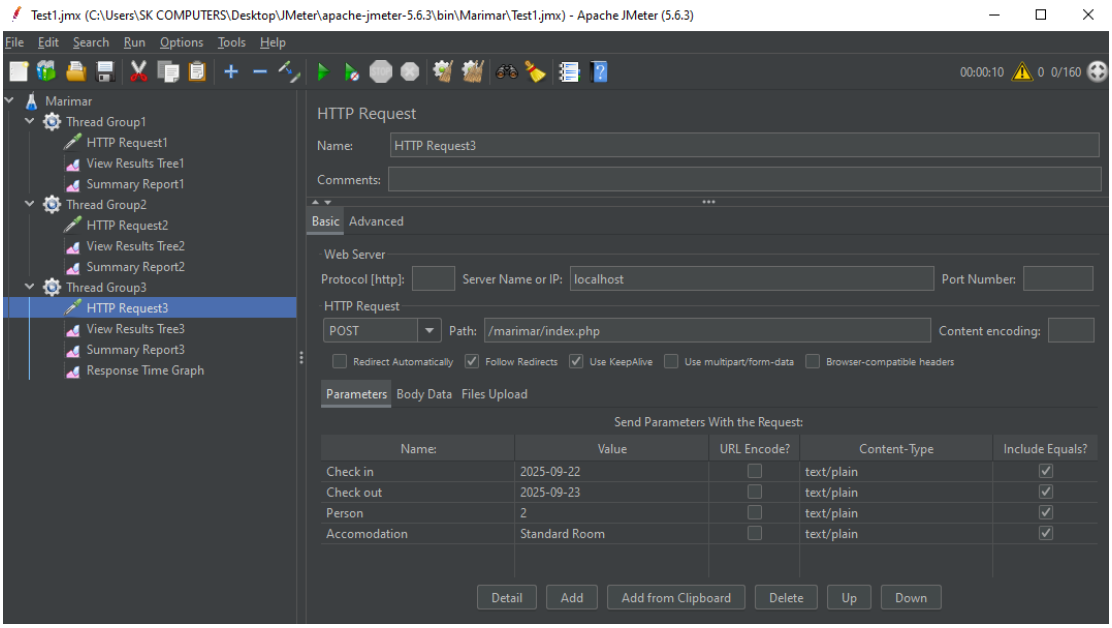


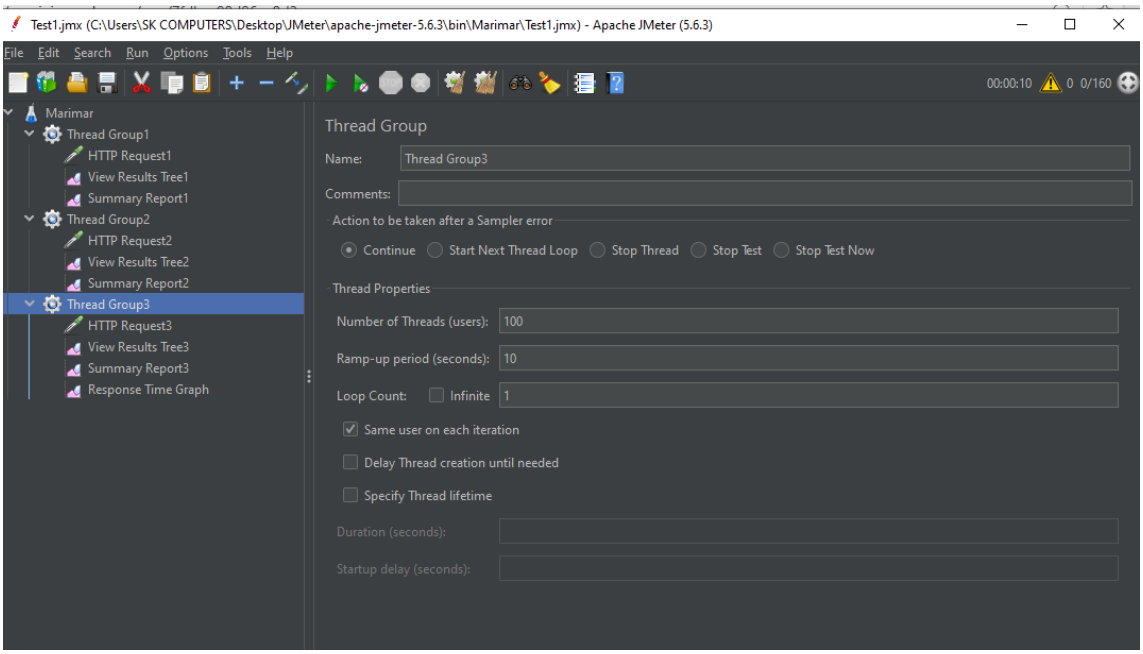


Metrics	**Value**
Users	50 requests were sent
Average	This is the average response time for all 50 requests. A very low average response time of 33 milliseconds indicates that the server is responding very quickly to search requests.
Min/ Max	The minimum response time was 9 ms, while the maximum was 337 ms. The maximum time is also very low, which is excellent. It shows that even under load, there were no extremely slow requests.
Std.dev	value of 65.56 ms is higher than the average, which means there was a noticeable variance in response times among the 50 requests.
Error %	All requests succeeded.
Throughput	The server processed 5.3 requests per second. This is a good metric for understanding the server's capacity under this specific load.
Received KB/sec	184.45 Data received from server per second.
Sent KB/sec	1.5 Data sent to server per second.
Avg.bytes	35758.0 Average size of each response.

100 concurrent users:







Metrics	Value
Users	100 requests were sent
Average	the server is still responding quickly even with double the load from previous 50 user test.
Min/ Max	The minimum response time remained 9 ms, but the maximum response time increased to 452 ms. Some requests took longer as the server handled more concurrent connections.
Std.dev	The standard deviation increased to 75.23 ms. This indicates a wider spread of response times compared to the 50 user test
Error %	All requests succeeded.
Throughput	Nearly doubled from the 50 user test (5.3/sec), processing 10.4 requests per second. Application scales well with an increased number of users.
Received KB/sec	364.81 Data received from server per second.
Sent KB/sec	2.95 Data sent to server per second.
Avg.bytes	35758.0 Average size of each response.

Interpretation of Search function:

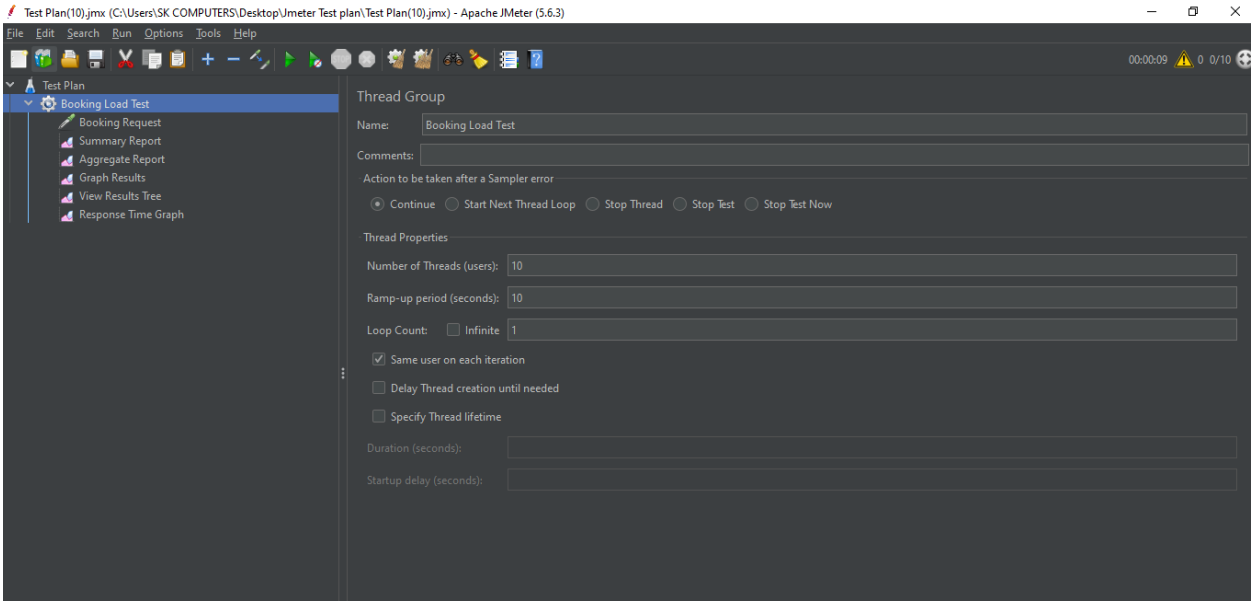
- Average Response Time**:** The average response time remains consistently low across all test scenarios, which is excellent.
- Error Rate: The most impressive finding is the 0% error rate across all three tests. This demonstrates that the application is stable and can handle increasing user loads without failing.

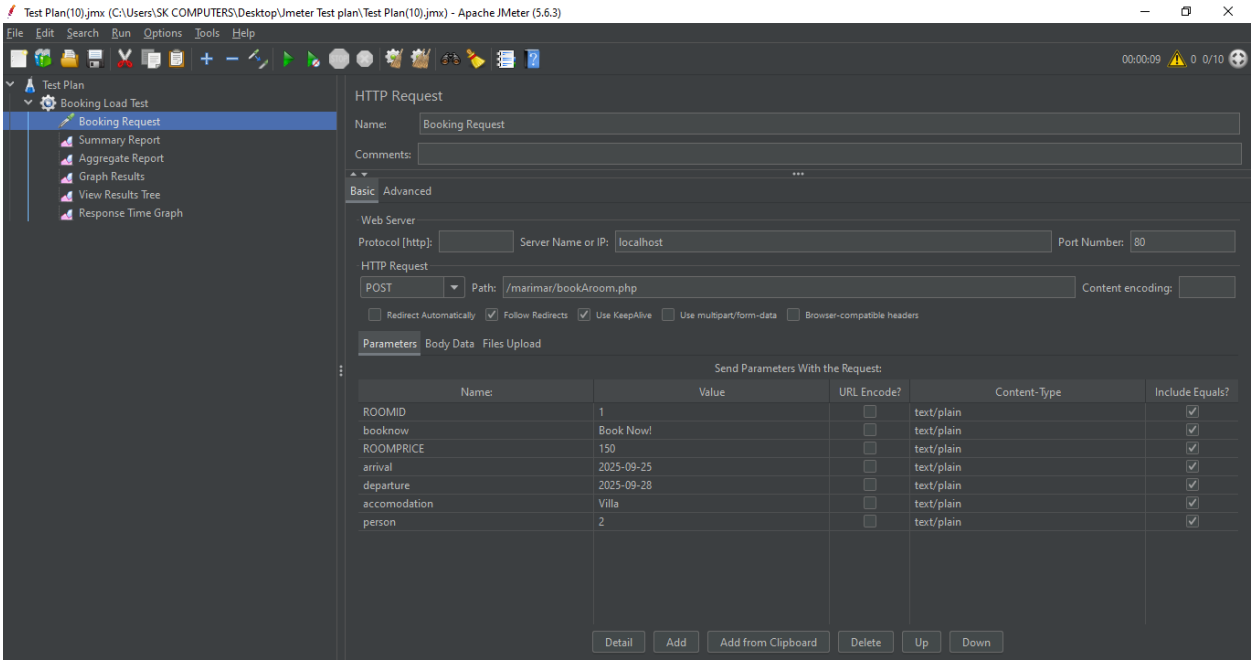
- Scalability: The throughput increases proportionally with the number of users (from 1.2/sec at 10 users to 10.4/sec at 100 users). This is a strong indicator of good scalability.
- Response Time Variance: As the number of users increases, the standard deviation and max response time also increase. This suggests some requests take longer than others, likely due to resource contention, but it's not a major issue since the overall performance remains high and there are no errors.

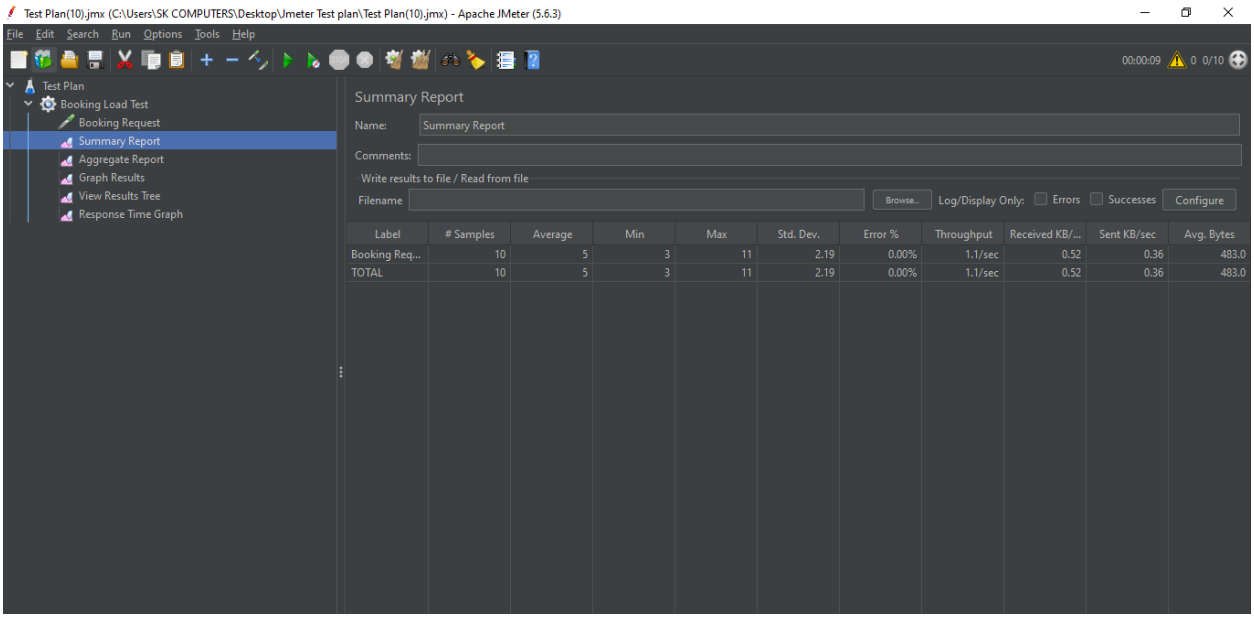
Booking function

- All the graphs for each thread group in the zip file.

10 concurrent users:



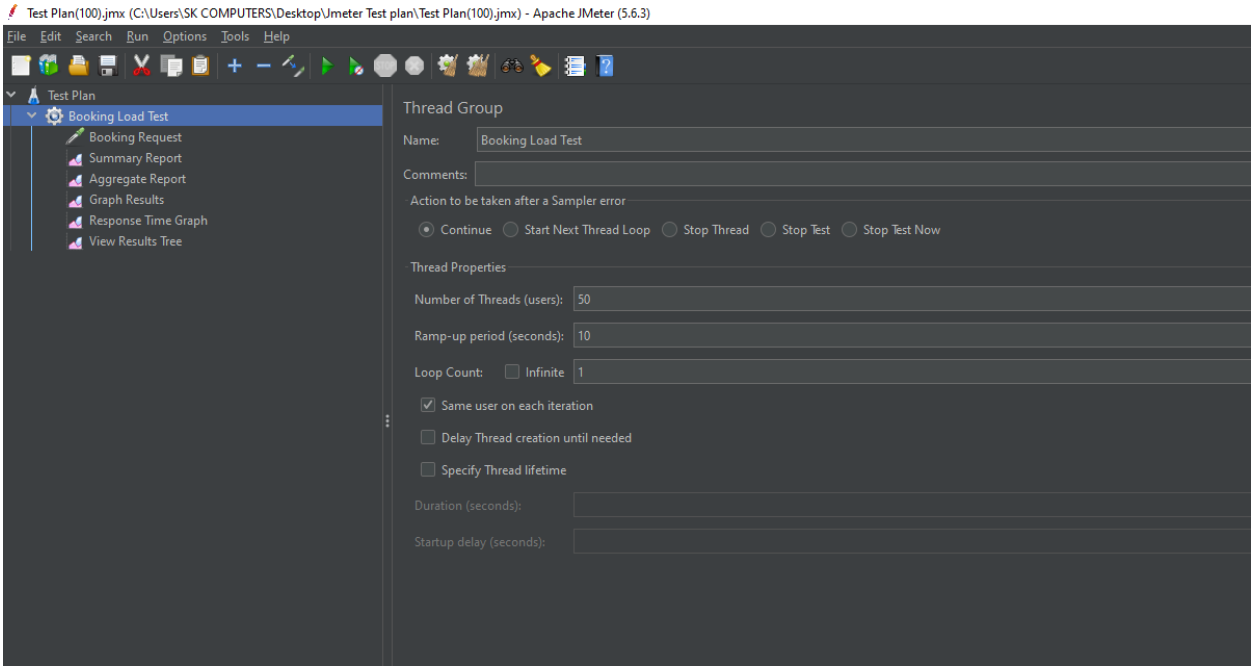


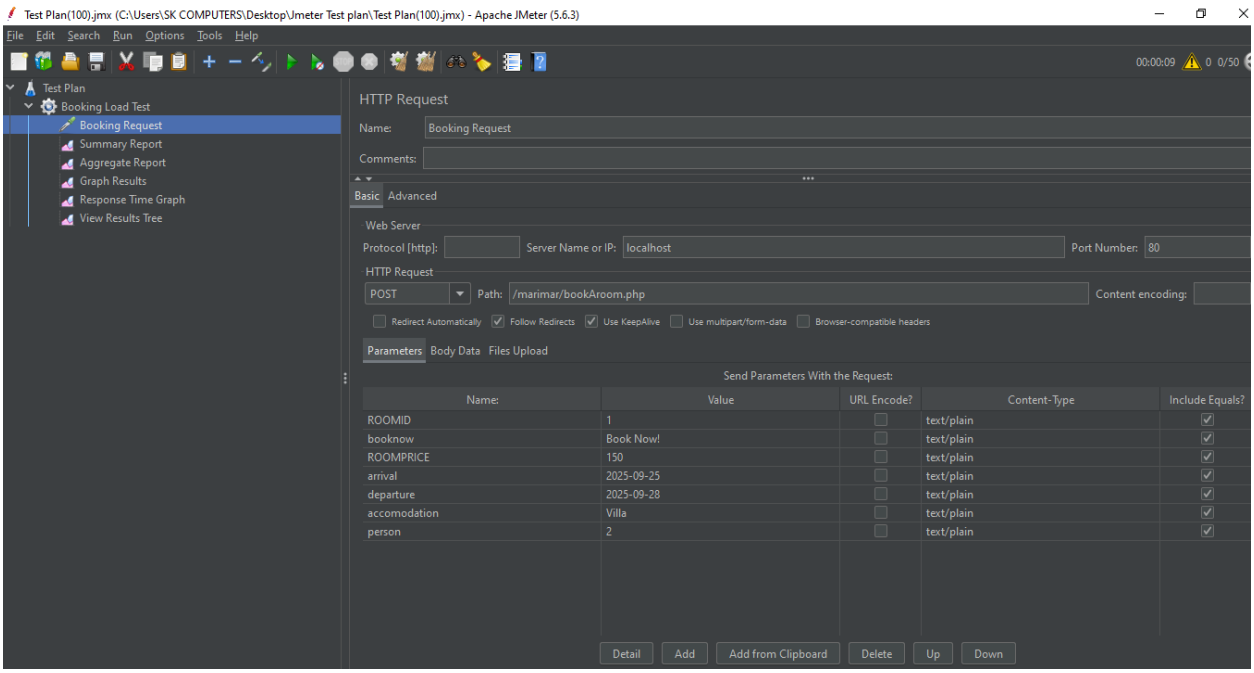


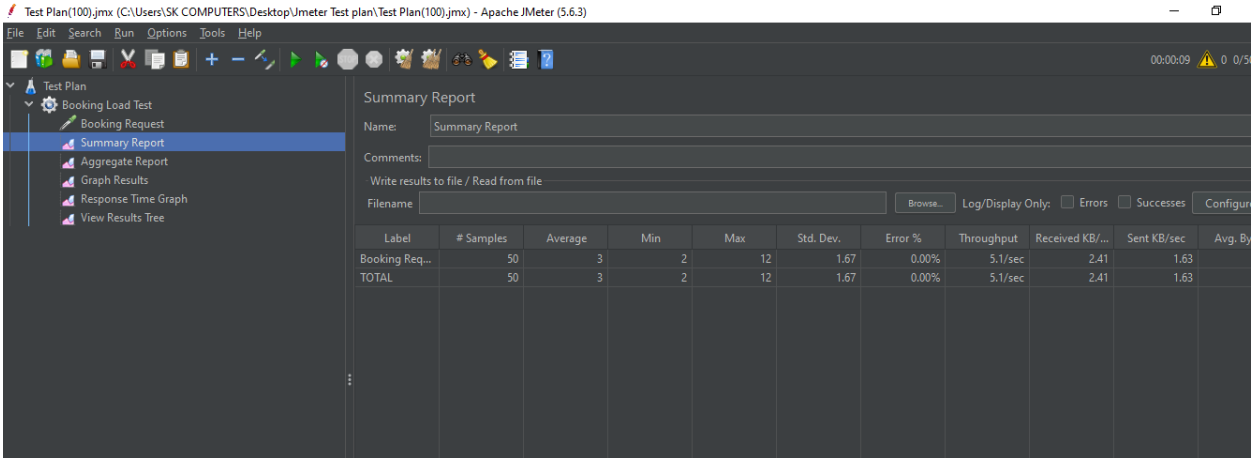
Metrics	**Value**
Users	10 requests were sent
Average	A 5-millisecond average means the server is processing booking requests almost instantly
Min/ Max	The response times are very consistent, with a narrow range between the fastest (3 ms) and slowest (11 ms) requests.
Std.dev	The 2.19 ms standard deviation is very low, confirming the high consistency of the response

	times. This shows there was very little variance in how long each booking request took.
Error %	All requests succeeded.
Throughput	Server processed 1.1 booking requests per second, which is a good baseline for a booking transaction
Received KB/sec	0.52 Data received from server per second.
Sent KB/sec	0.36 Data sent to server per second.
Avg.bytes	483 Average size of each response.

50 concurrent users:



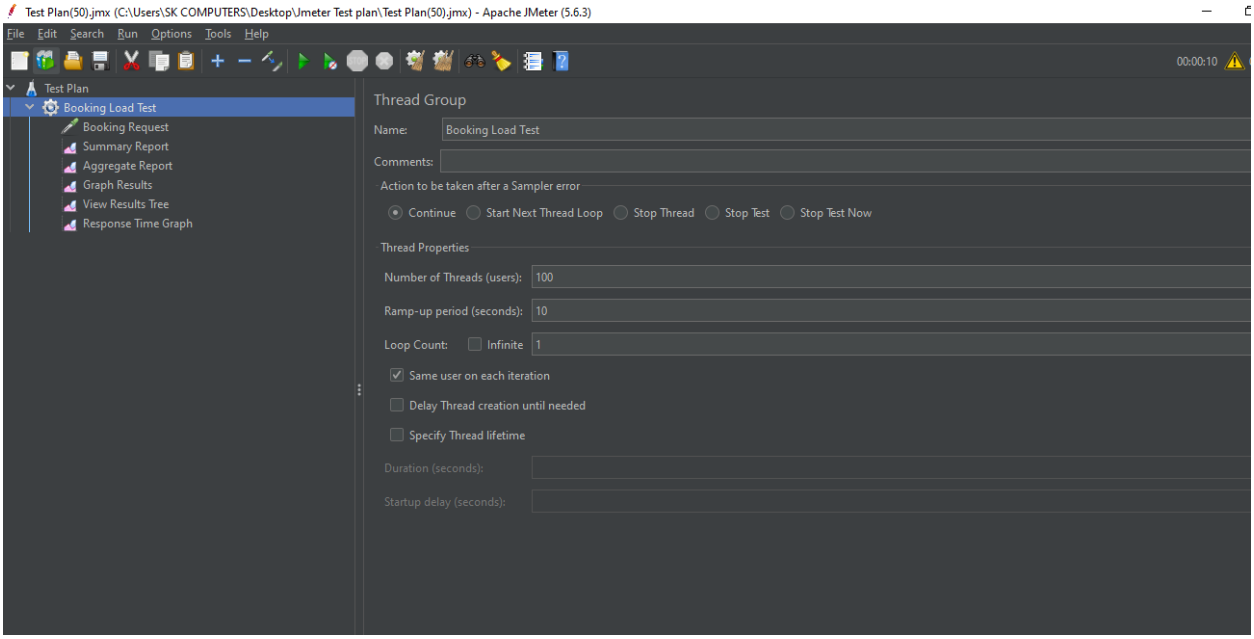


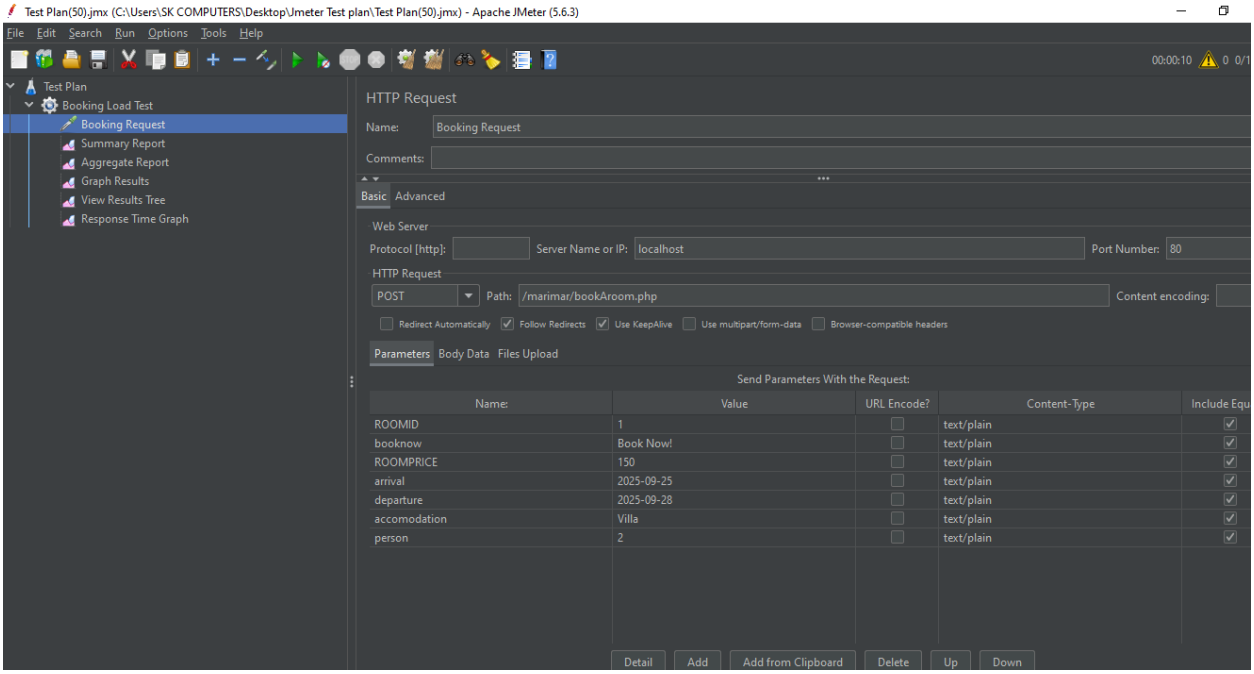


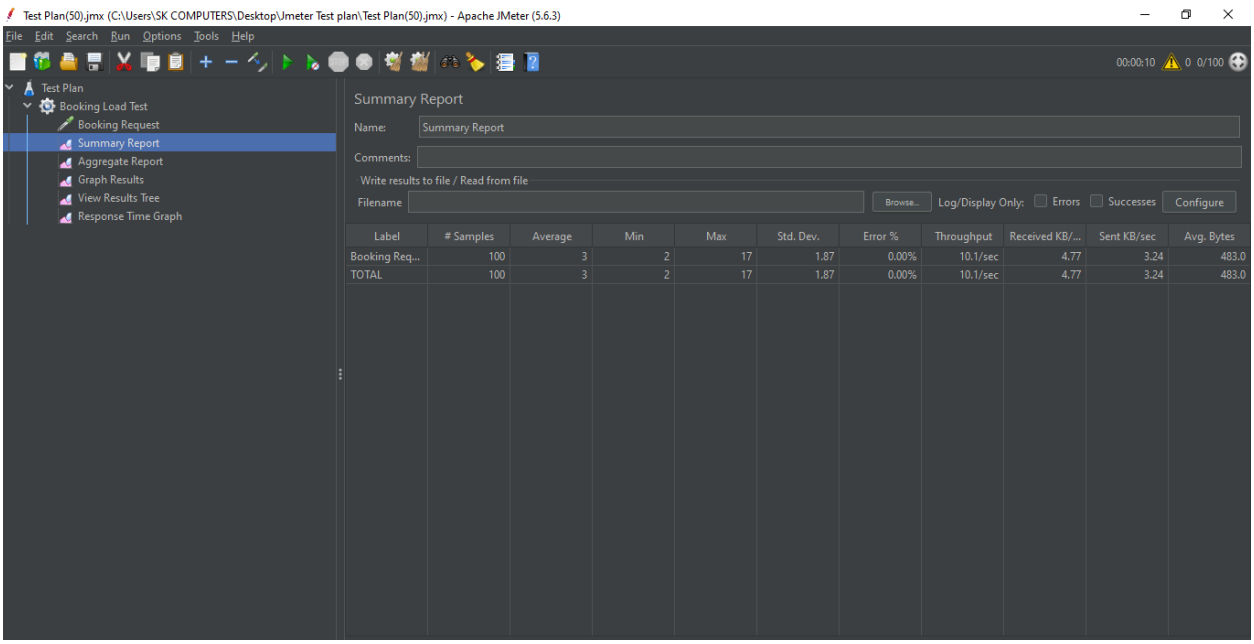
Metrics	Value
Users	50 requests were sent
Average	The average response time is a very good 53 milliseconds.
Min/ Max	The minimum response time was 13 ms, while the maximum was 382 ms. This maximum time is still well within an acceptable range.
Std.dev	109.67 ms. The standard deviation is high compared to the average, indicating that there was a large variance in response times among the requests.
Error %	All requests succeeded.
Throughput	The server processed 1.2 requests per second.
Received KB/sec	2.41 Data received from server per second.

Sent KB/sec	1.63 Data sent to server per second.
Avg.bytes	Average size of each response.

100 concurrent users:







Metrics	Value
Users	100 requests were sent
Average	Extremely low average response time, showing that the server is processing a high volume of booking requests very quickly
Min/ Max	The minimum response time was 2 ms, while the maximum was 17 ms. This indicates a high level of consistency and no significant slowdowns under this load.
Std.dev	1.87 ms very low standard deviation
Error %	All requests succeeded.
Throughput	Server processed 10.1 booking requests per second
Received KB/sec	4.77 Data received from server per second.
Sent KB/sec	3.24 Data sent to server per second.
Avg.bytes	483 Average size of each response.

Interpretation of Booking function:

- Speed and Consistency: The average response time is incredibly low, at just 3-5 milliseconds across all test scenarios. This shows that the function is extremely fast at processing booking requests, even when handling 100 concurrent users. The low standard deviation and narrow range between Min and Max times indicate that the performance is highly consistent and reliable.
- Scalability: The throughput increases directly in proportion to the number of users. As the user load grew from 10 to 100, the number of requests processed per second increased

tenfold (from 1.1/sec to 10.1/sec). This is a textbook example of a scalable application that can handle increased demand efficiently.

- Reliability: The 0% error rate is the most critical finding. It confirms that the booking function is highly stable and can perform the database-intensive task of creating a reservation without any failures, even under high stress.

Bug report

Defect ID	D_marimar_V1_001
Defect Description	The search room function allows customer to choose previous date for the check in date and checked out date.
Steps	1. Open xampp server and run both Apache and MySQL 2. Open PHPMyAdmin from the browser 3. After set up the database, browse as "localhost/marimar" 4. Navigate to the "Rooms" page and set the search field data and click on "Book Now" button
Date raised	16-09-2025
Defected by	Biyanga
Assigned to	Kaveesha
Closed date	16-09-2025
Status	Open
Severity	High
Priority	High

Defect ID	D_marimar_V1_002
Defect Description	"`Fatal error: __autoload() is no longer supported, use spl_autoload_register() instead`" error displays after run 'localhost/marimar/' on the web browser.
Steps	1. Open xampp server and run both Apache and MySQL 2. Open PHPMyAdmin from the browser 3. After set up the database, browse as "localhost/marimar"
Date raised	16-09-2025
Defected by	Biyanga
Fixed by	Prabash
Closed date	16-09-2025
Status	Closed
Severity	Critical
Priority	High

Defect ID	D_marimar_V1_003
Defect Description	"`Fatal error: Uncaught Error: Call to undefined function get_magic_quotes_gpc()` on line 14 of `database.php`" error displays after run 'localhost/marimar/' on the web browser.
Steps	1. Open xampp server and run both Apache and MySQL 2. Open PHPMyAdmin from the browser 3. After set up the database, browse as "localhost/marimar"
Date raised	16-09-2025
Defected by	Biyanga
Fixed by	Prabash
Closed date	17-09-2025
Status	Closed
Severity	Critical
Priority	High

Defect ID	D_marimar_V1_004
Defect Description	"`Warning: Trying to access array offset on value of type null`" error message displays on the "Rooms" page
Steps	1. Open xampp server and run both Apache and MySQL 2. Open PHPMyAdmin from the browser 3. After set up the database, browse as "localhost/marimar" 4. Navigate to the "Rooms" page 5. Each room detail card display an error message with the details and images
Date raised	16-09-2025
Defected by	Biyanga
Fixed by	Kaveesha
Closed date	17-09-2025
Status	Closed
Severity	Medium
Priority	Low

Defect ID	D_marimar_V1_005
Defect Description	There is a contact form but there was no database table to store them
Steps	1. Open xampp server and run both Apache and MySQL 2. Open PHPMyAdmin from the browser 3. After set up the database, browse as "localhost/marimar" 4. Navigate to the "Contact" page 5. Type message and send
Date raised	17-09-2025
Defected by	Biyanga
Fixed by	Biyanga
Closed date	17-09-2025
Status	Closed
Severity	High
Priority	High

Conclusion

The overall conclusion for the Marimar hotel reservation system is that the project is a success from a software testing perspective. The testing process found and fixed several important issues, making the system more stable and reliable. Performance tests showed that both the room search and booking functions are very fast and can handle a large number of users without any errors. This proves that the application is ready for use and can handle real-world traffic effectively.