Software Test Report

of

Doctor Appointment System

Efficiently Connecting Patients and Doctors: A Web-Based Solution

Prepared by

IMTIAZ CHOWDHURY | MUH2025027M

MIR MOHAMMOD TAHSIN | MUH2025007M

MD SANWAR HOSSAIN| MUH2025018M

TARIQUL ISLAM SHOBUJ | MUH2025035M

FAZILATER JAHAN | BFH2025002F

Institute of Information Technology

Noakhali Science and Technology University

Project Report for

SE 3210 - Software Testing and Quality Assurance Lab

Submitted to

MD. IFTEKHARUL ALAM EFAT

Assistant Professor

Institute of Information Technology

Noakhali Science and Technology University

Date of Submission: 29 February 2024

Table of Contents

List of Figures

List of Tables

1. Introduction

This report gives a summary of our extensive testing of the Doctor Appointment System, a web-based tool designed to make scheduling appointments for medical care easier. The system's performance, security precautions, regression resilience, and functionality were all assessed during our testing. These understandings are essential for maintaining data integrity and privacy standards, guaranteeing smooth patient-doctor interactions, and optimizing the effectiveness of the system.

2. Project Overview

Project Name: Doctor Appointment System

Project Description

The Doctor Appointment System is a web-based platform that revolutionizes the way healthcare appointments are scheduled and managed. Its primary objective is to enhance healthcare access by providing a user-friendly interface for patients to book appointments with healthcare providers seamlessly. Key features include simplified appointment booking, efficient resource management, real-time accessibility, secure patient information handling, and integration with existing healthcare systems such as Electronic Health Record (EHR) systems. The Doctor Appointment System is a forward-looking initiative that leverages the capabilities of web technology to revolutionize the way healthcare appointments are scheduled and managed. By focusing on simplicity, efficiency, and security, this project aims to contribute to the broader goal of enhancing healthcare accessibility and patient satisfaction in the digital age.

3. Testing Strategy

3.1 Test requirements and objectives

- Ensure that all features of the system work as intended according to the functional requirements.
- Verify that users can register, log in, book appointments, manage appointments, receive reminders, and perform other tasks without encountering errors or unexpected behavior.
- Validate that the system accurately displays available doctors, appointment slots, and relevant information to users.
- Ensure that the user interface is intuitive and easy to navigate for both patients and healthcare providers.

- Validate that users can perform tasks such as booking appointments and managing their profiles without requiring extensive training or guidance.
- Verify that the system provides clear instructions, feedback messages, and error prompts to assist users in completing their tasks efficiently.
- Ensure that the system implements robust authentication mechanisms to verify the identity of users and protect against unauthorized access.
- Validate that sensitive patient information, such as personal details and medical history, is encrypted during transmission and storage to prevent data breaches.
- Verify that access controls are in place to restrict users' access to confidential information based on their roles and permissions.
- Ensure that the system performs efficiently under normal operating conditions, with acceptable response times for user interactions.
- Validate that the system can handle concurrent user sessions and peak loads without experiencing performance degradation or system failures.
- Verify that the system's performance metrics, such as server response time and page load time, meet predefined benchmarks and service level agreements (SLAs).

3.2 Designing Test Cases

- User Registration and Authentication:
  - ***Test Case 1***: Verify that a new user can register by providing valid information (name, email, password, etc.).
  - ***Test Case 2***: Verify that registration fails if the user provides invalid or incomplete information.
  - ***Test Case 3***: Verify that users can log in successfully using valid credentials.
  - ***Test Case 4***: Verify that users cannot log in with incorrect credentials.
  - ***Test Case 5***: Verify that users receive a verification email or SMS after registration and must verify their account to complete the registration process.
- Appointment Booking:
  - ***Test Case 6***: Verify that users can search for doctors based on specialties, location, and availability.
  - ***Test Case 7***: Verify that available time slots for each doctor are displayed correctly.
  - ***Test Case 8***: Verify that users can select a convenient appointment time and book an appointment successfully.
  - ***Test Case 9***: Verify that booking fails if the selected time slot is no longer available (e.g., due to another user booking it).
  - ***Test Case 10***: Verify that users can book appointments for themselves or on behalf of dependents.
- Appointment Management:

  - ***Test Case 11***: Verify that users can view their upcoming appointments after logging in.
  - ***Test Case 12***: Verify that users can reschedule appointments to another available time slot.
  - ***Test Case 13***: Verify that users can cancel appointments, and the system updates the availability accordingly.
  - ***Test Case 14***: Verify that users receive notifications for any changes to their appointments.
- Doctor Profile Management:
  - ***Test Case 15***: Verify that healthcare providers can log in and access their profile management interface.
  - ***Test Case 16***: Verify that doctors can update their availability, specialties, clinic hours, and contact information.
  - ***Test Case 17***: Verify that changes made to doctor profiles are reflected correctly in the appointment booking interface.
- Automated Reminders:
  - ***Test Case 18***: Verify that users receive automated reminders via email, SMS, or push notifications before their scheduled appointments.
  - ***Test Case 19***: Verify that users can customize their reminder preferences (e.g., frequency, preferred communication channel).
- Integration with Electronic Health Records (EHR):
  - ***Test Case 20***: Verify that the system integrates seamlessly with existing EHR systems to access and update patient records.
  - ***Test Case 21***: Verify that patient data is transmitted securely between the appointment system and EHR to maintain data integrity and privacy.
- Data Security and Privacy:
  - ***Test Case 22***: Verify that sensitive patient information is encrypted during transmission and storage to prevent unauthorized access.
  - ***Test Case 23***: Verify that user authentication and authorization mechanisms are implemented correctly to restrict access to patient data.
- Accessibility and Usability:
  - ***Test Case 26***: Verify that the system meets web accessibility standards to accommodate users with disabilities.
  - ***Test Case 27***: Verify that the user interface is intuitive and easy to navigate for users with varying levels of technological proficiency.

3.3 Testing Tools

- Functional Testing: ***Selenium WebDriver*** for automated testing of user interactions and functionality.
- Regression Testing: ***Selenium IDE*** is selected for performing regression testing to ensure the stability of existing functionalities after new changes.
- Performance Testing: ***Apache JMeter*** for load testing to evaluate the system's performance under various load conditions.
- Security Testing: ***Burp Suite*** for scanning and identifying security vulnerabilities in the system.

3.4 Types of Testing

- **Functional Testing**: Ensures that each feature of the application behaves as expected according to the defined functional requirements.
- **Regression Testing**: Verifies that new changes or enhancements do not negatively impact the existing functionalities of the application.
- **Performance Testing**: Evaluates the responsiveness and scalability of the application under various load conditions to ensure optimal performance.
- **Security Testing**: Optionally conducted to identify and address security vulnerabilities, ensuring the confidentiality, integrity, and availability of user data within the application.

4. Test Environment

4.1 Web Environment

The application (Doctor Appointment System) is supported on the following web browsers and versions:

- ✓ Google Chrome (Version 122.0.6261.95)
- ✓ Mozilla Firefox (Version 123.0)

4.2 Operating Systems

The application (Doctor Appointment System) is compatible with the following operating systems:

- ✓ Windows 11
- ✓ Ubuntu Linux (latest LTS version)

4.3 Testing Tools

- Functional Testing

Selenium WebDriver: An open-source tool used for automating web application testing across different browsers and platforms.

- Regression Testing

Selenium: An open-source tool used for automating web application testing across different browsers and platforms.

- Performance Testing

Apache JMeter: An open-source tool for load testing, performance testing, and stress testing of web applications.

- Security Testing

Burp Suite: Burp Suite is a popular platform for performing security testing of web applications. It includes tools for analyzing the security of web applications, including scanning for vulnerabilities such as SQL injection, cross-site scripting (XSS), and more.

4.4 Testing Time

28 February 2024

5. Test Case Design

5.1 Functional Testing

Test Features

The following test scenarios were executed during the functional testing process:

- **User Registration and Login**
  - Verify that a user can register successfully.
  - Verify that a registered user can log in with valid credentials.
  - Verify that an error message is displayed for invalid login credentials.
- **Appointment Booking**
  - Verify that a user can book an appointment with a doctor.
  - Verify that appointment details are correctly displayed after booking.
  - Verify that the system prevents double booking for the same time slot.
- **Appointment Management**
  - Verify that a user can view their upcoming appointments.

  - Verify that a user can cancel an existing appointment.
  - Verify that canceled appointments are removed from the user's appointment list.
- **Change Password**
  - Verify that a user can change their password successfully.
  - Verify that the new password is updated and valid for login.
- **Update Profile**
  - Verify that a user can update their profile information.
  - Verify that the updated profile information is saved correctly.
- **Checking appointments**
  - Verify that a user can view their appointment searching by their mobile number.
  - Verify that appointment details are displayed accurately.
  - Verify that past appointments are not displayed in the upcoming appointments list.

Table 1: Test case for user Registration (Functional Testing)

Test Case No.	Test Case	Test Data	Expected Result	Actual Result	Status
1	Valid Registration	Tahsin Ahmed, tahsinahmed.iit@gmail.com , 1234567890, Orthopedics, StrongPassword123!	Registration successful	Registration successful	**Pass**
2	Empty Email	Tahsin Ahmed, "", 1234567890, Orthopedics, StrongPassword123!	Error message: "Email address is required"	Error message: “Email address is required”	**Pass**
3	Weak Password	Tahsin Ahmed, tahsinahmed.iit@gmailcom, 123, Orthopedics, WeakPassword	Error message: "Password does not meet minimum strength requirements"	Registration Successful	**Fail**

4	Missing Mobile Number	Tahsin Ahmed,tahsinahmed.iit@gmail.com, "", Orthopedics, StrongPassword123!	Error message: "Mobile number is required"	Error message: “Mobile number is required”	Pass
5	Invalid Email Format	John Doe, johndoe, 1234567890, Orthopedics, StrongPassword123!	Error message: "Invalid email format"	Error message: “Invalid email format”	Pass

Here, the Actual Results & status is written after the test execution.

Table 2: Test case for user login (Functional Testing)

Test Case No.	**Test Case**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Valid Login	Valid Email: tahsinahmed.iit@gmail Valid Password: 1234567890	Successful login and redirection to homepage	Successful login and redirected to the home page	Pass
2	Invalid Login (Incorrect Password)	Valid Email: tahsinahmed.iit@gmail Valid Password: 12345	Error message: "Invalid username or password"	Error message: “invalid username or password”	Pass

3	Invalid Login (Non-existent Email)	Valid Email: tahsin.iit@gmail Valid Password: 1234567890	Error message: "Invalid username or password"	Error message: "Invalid username or password"	Pass

Here, the Actual Results & status is written after the test execution.

Table 3: Test case for Appointment Booking (Functional Testing)

Test Case No.	**Test Case**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Valid Appointment Booking	**Name:** Tahsin Ahmed, Email: tahsin@gmail.com,**Phone Number:** 01854239411, **Appointment date:** Valid future date, **Appointment Time:** Valid time slot, **Specialisation:**Orthopedics, **Doctor**: Tahsin Ahmed, **Description:** "",	Appointment successfully booked and confirmation email sent to patient.	Appointment booked, confirmation email sent.	Pass
2	Invalid Appointment Date	**Name:** Tahsin Ahmed, Email: tahsin@gmail.com, **Phone Number:** 01854239411, **Appointment date:** Invalid past date, **Appointment Time:** Valid time slot, **Specialisation:**Orthopedics, **Doctor**: Tahsin Ahmed, **Description:** "",	Error message: "Please select an valid appointment date"	Error message: "Please select avalid appointment date"	Pass

3	Selecting Unavailable Time Slot	**Name:** Tahsin Ahmed, Email: tahsin@gmail.com, **Phone Number:** 01854239411, **Appointment date:** Valid future date, **Appointment Time:** invalid time slot, **Specialization:**Orthopedics, **Doctor:** Tahsin Ahmed, **Description:** "",	Error message: "Selected time slot is unavailable. Please choose another."	User sees the error message.	Pass

Here, the Actual Results & status is written after the test execution.

Table 4: Test case for Checking Appointment (Functional Testing)

Test ID	Test Name	Test Data	Expected Output	Actual Output	Status
1	Valid Number	**Number :** 01688406049	Appointment Details	Appointment Details	Pass
2	Invalid Number	**Number :** 1	No Result Will be showed	Some appointment Results Shown	**Fail**

Here, the Actual Results & status is written after the test execution.

Table 5: Test case for Appointment Management (Functional Testing)

Test ID	Test Name	Test Data	Expected Output	Actual Output	Status
1	Approve Appointment	Click on Approve	Appointment Will be approved	Appointment Approved	Pass
2	Cancel Appointment	Click on Cancel	Appointment Will be Canceled	Appointment Not Canceled	**Fail**

Here, the Actual Results & status is written after the test execution.

Table 6: Test case for Change Password (Functional Testing)

Test Case No.	Test Case	Test Data	Expected Result	Actual Result	Status
1	Valid Password Change	**Correct Current:** 1234567890 **Password, New Password:** 987654321, **Re-typed Password:** 987654321	Password successfully changed and confirmation message displayed	Password changed, confirmation message displayed.	Pass
2	Incorrect Current Password	**Incorrect Current Password:** 1234567890,**Valid New Password:**7532159, **Re-typed Password**:7532159	Error message: "Current password is incorrect"	User sees the error message "Current password is incorrect".	Pass
3	Mismatched New and Re-typed Passwords	**Correct Current: Password:** 987654321, **Valid New Password:** 7532159, **Different Re-typed Password**:75321457	Error message: "New passwords do not match"	User sees the error message "New passwords do not match".	Pass

Here, the Actual Results & status is written after the test execution.

Table 7: Test case for Update Profile (Functional Testing)

Test Case No.	Test Case	Test Data	Expected Result	Actual Result	Status
1	Valid Profile Update	**Name:** Mir Mohammad Tahsin (Updated), **E-mail:** tahsinAhmed@gmail.com (updated), **Phone number:**1914373594 (updated), **Specialization:** Orthopedics (Original),	Profile successfully updated	Profile successfully updated	Pass
2	Update Email Address	**Name:** Mir Mohammad Tahsin (Updated), **E-mail:** tahsinahmed@gmail.com (Original), **Phone number:**1914373594 (Original), Specialization: Orthopedics (Original),	Profile successfully updated and will be able to login with new mail	Profile successfully updated and can login with new mail	Fail
3	Empty Full Name	**Name:** “ ” (Updated), **E-mail:** tahsinahmed@gmail.com (Original), **Phone number**:1914373594 (Original), **Specialization:** Orthopedics (Original),	Error message: "Full name is required"	Error message: "Full name is required"	Fail

Here, the Actual Results & status is written after the test execution.

5.2 Regression Testing

Sign Up Function:
Here I have used previous test suit of functional testing.

I have included an input field called address on the sign-up page. After that I ran the previous test suit selected for the sign-up page.

Change Profile:
Here I have used previous test suit of functional testing.

5.3 Performance Testing

1. **Appointment Booking Performance:**
    - **Objective:** Evaluate the performance of booking appointments.
    - **Test Steps:**
        1. Navigate to the appointment booking page.
        2. Select a specialization, doctor, preferred date, and time slot.
        3. Measure the time taken to confirm the appointment booking.
    - **Expected Outcome:** Appointment booking should be completed swiftly, ensuring a seamless user experience even under high concurrent user loads.
2. **Resource Management Performance:**
    - **Objective:** Evaluate the system's efficiency in managing doctor resources.
    - **Test Steps:**
        1. Simulate multiple users attempting to book appointments with the same doctor simultaneously.
        2. Measure the system's response time and verify that appointments are scheduled without conflicts.
    - **Expected Outcome:** The system should efficiently manage doctor resources, avoiding double bookings and conflicts, even under peak usage.
3. **Real-time Accessibility Performance:**
    - **Objective:** Evaluate the real-time accessibility of the system.
    - **Test Steps:**
        1. Perform simultaneous actions such as checking appointment status, receiving reminders, and making appointment changes.
        2. Measure the system's response time for each action.
    - **Expected Outcome:** Users should experience real-time responsiveness when accessing and interacting with the system, ensuring timely updates and notifications.
4. **Integration with Existing Systems Performance:**

- **Objective:** Evaluate the system's compatibility and performance when integrated with Electronic Health Record (EHR) systems.
- **Test Steps:**
    1. Integrate the Doctor Appointment System with a test EHR system.
    2. Perform various actions such as accessing patient records during appointment booking.
- **Expected Outcome:** The system should seamlessly integrate with EHR systems without compromising performance, ensuring smooth data exchange and continuity of healthcare services.

5. **Scalability Performance:**
    - **Objective:** Evaluate the system's scalability by gradually increasing the number of concurrent users.
    - **Test Steps:**
        1. Gradually increase the number of simultaneous users accessing the system.
        2. Measure system response time and resource utilization as the user load increases.
    - **Expected Outcome:** The system should scale horizontally to accommodate increasing user loads while maintaining acceptable performance metrics.

Table 8: Test case (Performance Testing)

Test Case ID	Input	Actual Output	Expected Output	Status
1	Navigate to the appointment booking page. Select a specialization, doctor, preferred date, and time slot. Measure the time taken to confirm the appointment booking.	15 milliseconds	Swift confirmation of the appointment booking, even under high concurrent user loads.	Evaluate the system's performance in handling appointment bookings.
2	Simulate multiple users attempting to book appointments with the same doctor simultaneously. Measure the system's response time and verify that appointments are scheduled without conflicts.	[25ms, Booking results]	Efficient management of doctor resources, avoiding double bookings and conflicts under peak usage.	Assess the system's efficiency in managing doctor resources.

3	Perform simultaneous actions such as checking appointment status, receiving reminders, and making appointment changes. Measure the system's response time for each action.	[8ms]	Real-time responsiveness when accessing and interacting with the system. Timely updates and notifications.	Evaluate the real-time accessibility of the system.
4	Integrate the Doctor Appointment System with a test EHR system. Perform various actions such as accessing patient records during appointment booking.	[100ms]	Seamless integration with EHR systems. Smooth data exchange and continuity of healthcare services.	Evaluate integration with existing systems, particularly Electronic Health Record (EHR) systems.
5	Gradually increase the number of simultaneous users accessing the system. Measure system response time and resource utilization as the user load increases.	[200ms, Resource utilization]	System scales horizontally to accommodate increasing user loads while maintaining acceptable performance metrics.	Evaluate the system's scalability under varying user loads.

Here, the Actual Results & status is written after the test execution.

5.4 Security Testing

Scope

The scope of security testing will encompass the entire ***Doctor Appointment System***, including:

- User authentication and authorization mechanisms.
- Data transmission and storage.
- Input validation mechanisms.
- Protection against common web application vulnerabilities.

Objectives

The objectives of security testing are to:

- Identify and mitigate potential security vulnerabilities.
- Ensure compliance with relevant security standards and regulations (e.g., HIPAA for healthcare data).
- Safeguard sensitive patient information against unauthorized access or manipulation.
- Verify the resilience of the system against malicious attacks.

Methodology

The security testing will follow a combination of manual and automated approaches:

- ***Manual Testing***: Security analysts will conduct in-depth manual testing to identify complex vulnerabilities and validate automated scan results.
- ***Automated Testing***: Utilize security testing tools such as **Burp Suite** to perform automated scans and identify common security issues.

*Here, I only perform automated testing.

Test Scenarios

Test scenarios will cover various aspects of security, including:

- ***Authentication Testing***: Verify the strength of user authentication mechanisms, such as password complexity and session management.
- ***Authorization Testing***: Ensure that users can only access the functionalities and data they are authorized to.
- ***Data Protection***: Validate the encryption of sensitive data during transmission and storage and assess the effectiveness of access controls.
- ***Input Validation***: Test for input validation vulnerabilities such as SQL injection, cross-site scripting (XSS), and command injection.

***Testing Time:** 28 February 2024*
***Tester:** Imtiaz Chowdhury*

Table 9: Test Case Design (Security Testing)

Test Case	Test Functionality	Pre-conditions	Expected results	Actual Results	Test status-PASS/Fail	Comments
1. Misconfiguration in Name Change	The system should have proper sanitization. Users should follow the system's requirements to change their names (**Display name doesn't contain special characters**).	Users should have logged in to the system.	The system shouldn't accept special characters in the display name.	The system doesn't maintain proper sanitization, but as a middleman, attackers can easily break this sanitization.	**Pass**	I quickly broke this sanitization. Because of this vulnerability, attackers can easily add malicious payloads in this system.
2. Misconfiguration in Password Change/Sign up password	Users should follow all criteria to set/update their password. (Example: password should contain both upper-case and lower-case characters to strengthen passwords.)	Users should have logged in to the system to update their password and not need to log in while signing up.	The system shouldn't accept passwords that don't follow all of the system's criteria.	The system doesn't maintain proper sanitization, but as a middleman, attackers can easily guess pass in case of sanitization absence.	**Pass**	Because of this vulnerability, attackers can easily try the password brute-force attack to take over an account with a common wordlist.

3. Account takeover via inappropriate password reset	The system should follow that mechanism so that a middleman can't get the password reset link in response to the password reset request.	Only need the targeted account email address and mobile no.	The password reset link should be sent to the valid email address of the account holder and only be there.	However, hackers can easily get the targeted account email address and mobile no being a middleman between the user and the server.	**Pass**	Because of this vulnerability, hackers can quickly take over an account.
4. No rate limit for failed login attempt	The system should limit failed login attempts to prevent brute-force attacks.	Only need the targeted account email address.	The system should block the user for some time after a certain number of failed login attempts.	However, the system does not react to password brute-force attacks.	**Pass**	Hackers can easily try the password brute-force attack to take over an account.

Here, the Actual Results & status is written after the test execution.

6. Testing Tools Used

6.1 Functional Testing Tool

To begin functional testing with ***Selenium WebDriver***, I first install the ***required dependencies*** and ***download the WebDriver*** executable for my chosen browser. Then, I write test scripts using my preferred programming language and ***Selenium API***. After configuring the test environment, I execute the tests, analyze the results, and refine them as necessary. This setup facilitates automated testing of web applications with efficiency and ease.

All related figures are given in the Test Execution & Result (start with Figure-4) part.

6.2 Regression Testing Tool

Regression testing ensures that changes or enhancements to a software application do not negatively impact existing features. When developers modify the codebase, such as adding new features, fixing bugs, or making updates, there is a risk that these changes could introduce new defects or break existing functionality. Regression testing is performed to detect and prevent such issues.

Testing Tools Used for Regression Testing: **Selenium IDE** (Sample ss of selenium IDE)

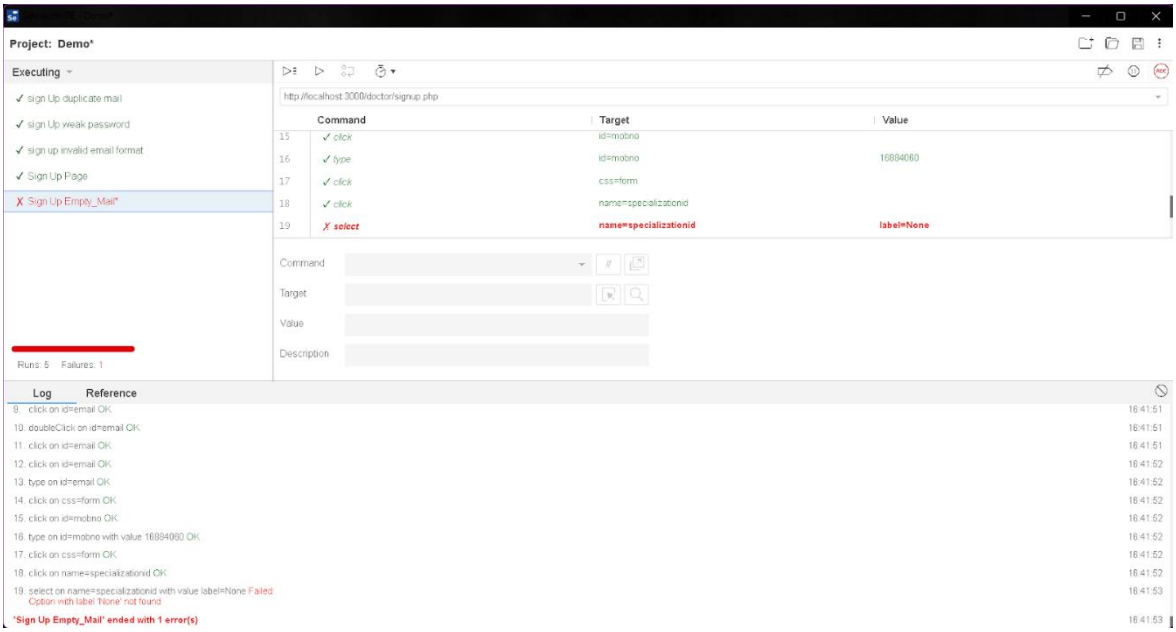

Figure 1: Sample ss of selenium IDE

6.3 Performance Testing Tool

Used Tool: Apache JMeter.

To execute performance testing using Apache JMeter for the Doctor Appointment System, you can follow these steps:

1. **Install Apache JMeter**:
    - Download Apache JMeter from the official website: https://jmeter.apache.org/download_jmeter.cgi
    - Install it on your system following the installation instructions provided.
2. **Prepare Test Plan**:
    - Launch Apache JMeter.
    - Create a new Test Plan by right-clicking on "Test Plan" and selecting "Add > Threads (Users) > Thread Group".

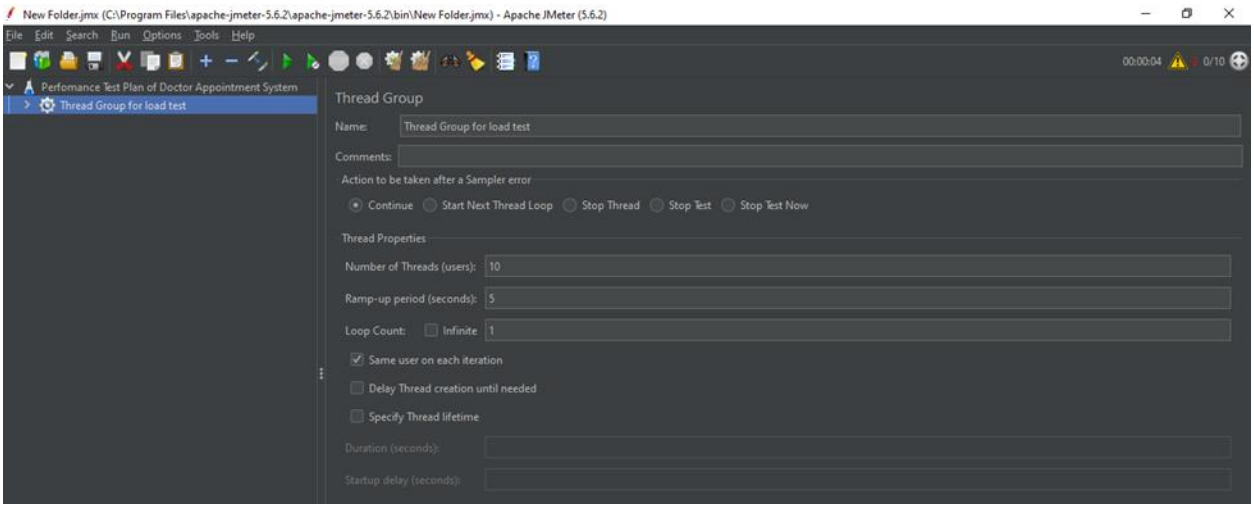

Figure 2: Thread Group

- Configure the Thread Group according to your performance testing requirements, including the number of users, ramp-up period, and loop count.

3. **Add HTTP Request Sampler**:
   - Right-click on the Thread Group and select "Add > Sampler > HTTP Request".
   - Configure the HTTP Request Sampler to simulate different user actions such as logging in, booking appointments, or Search Appointment Status of the Doctor Appointment System.
   - Set the server's name or IP address, port number, and path of the application under test.
4. **Add Assertions (Optional)**:
   - To verify the response from the server, you can add Assertions to your HTTP Request Samplers.
   - Right-click on the HTTP Request Sampler, select "Add > Assertions", and choose the type of assertion you want to add (e.g., Response Assertion).
5. **Add Listeners**:
   - Listeners allow you to view and analyze test results. Right-click on the Thread Group and select "Add > Listener" to add one or more Listeners.
   - Common Listeners include View Results Tree, Summary Report, and Aggregate Report.
6. **Configure Test Plan Properties**:
   - Configure Test Plan properties such as the number of threads (users), ramp-up period, loop count, and duration of the test.
   - Adjust these settings based on your performance testing objectives and requirements.
7. **Run the Test**:
   - Save your Test Plan.
   - Click on the "Start" button (green triangle) in the toolbar to run the test.

- Apache JMeter will start executing the test plan, simulating the defined user behavior and collecting performance metrics.

8. **Analyze Results**:
    - Once the test is completed, review the results using the Listeners you added.
    - Analyze metrics such as response time, throughput, error rate, and resource utilization to assess the performance of the Doctor Appointment System under different load conditions.
9. **Iterate and Optimize**:
    - Based on the test results, identify any performance issues or bottlenecks.
    - Make necessary optimizations to improve the performance of the system, such as optimizing code, database queries, or server configuration.
    - Repeat the performance testing process iteratively to validate improvements and ensure that the system meets performance requirements.

By following these steps, you can execute performance testing using Apache JMeter to ensure that the Doctor Appointment System meets performance requirements and delivers a seamless user experience under various conditions.

6.4 Security Testing Tool

Testing Tools Used: ***Burp Suite*** (v2023_12_1_5)
Browser Used: ***Mozilla Firefox*** (Version 123.0)

Provide detailed information about Burp Suite, including:
Description: Burp Suite is an integrated platform/graphical tool for performing security testing of web applications. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface to finding and exploiting security vulnerabilities.

Process Description: First, I configure the proxy (host: 127.0.0.1, port: 8081) and continue the interception.

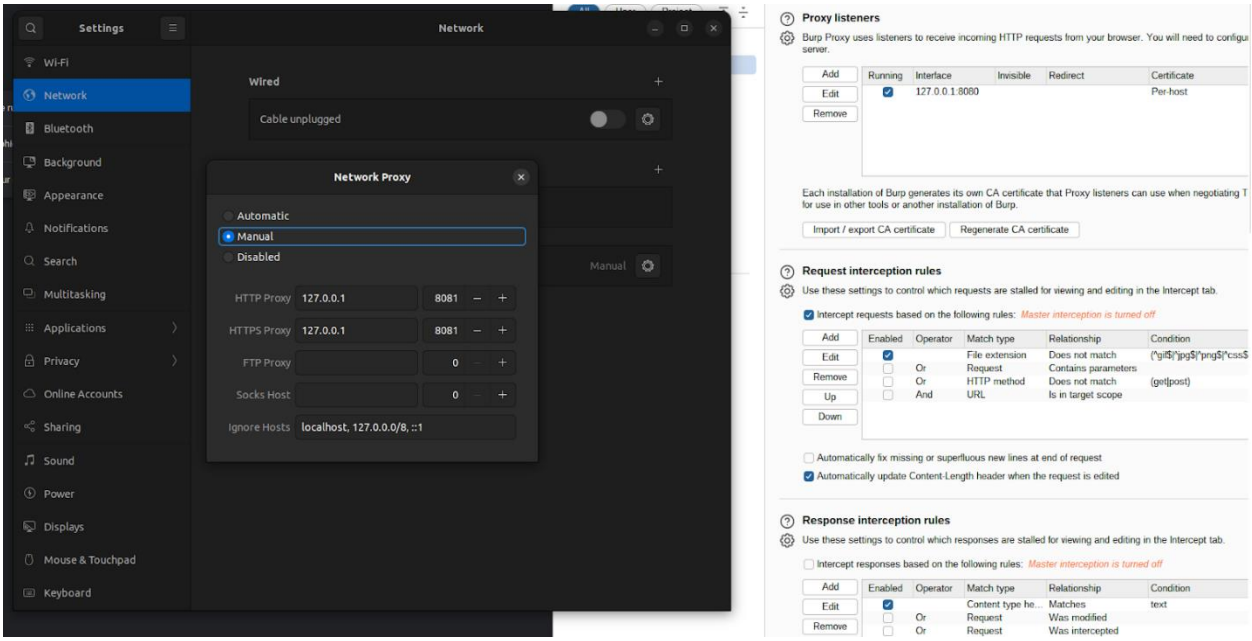

Figure 3: Proxy Setting (Security Testing)

I used the ***Mozilla Firefox (Version 123.0)*** browser to test with ***Burp Suite***. The above screenshot shows that the first initial stage proxy configuration is set up here.

7. Test Execution and Results

7.1 Test Execution and Results-1: Functional Testing

Here, all test case design tables are given for better understanding.

Table 10: Test case for user Registration (Functional Testing)

Test Case No.	Test Case	Test Data	Expected Result	Actual Result	Status
1	Valid Registration	Tahsin Ahmed, tahsinahmed.iit@gmail.com , 1234567890, Orthopedics, StrongPassword123!	Registration successful	Registration successful	**Pass**

2	Empty Email	Tahsin Ahmed, "", 1234567890, Orthopedics, StrongPassword123!	Error message: "Email address is required"	Error message: “Email address is required”	**Pass**
3	Weak Password	Tahsin Ahmed, tahsinahmed.iit@gmailcom, 123, Orthopedics, WeakPassword	Error message: "Password does not meet minimum strength requirements"	Registration Successful	**Fail**
4	Missing Mobile Number	Tahsin Ahmed,tahsinahmed.iit@gmail.com, "", Orthopedics, StrongPassword123!	Error message: "Mobile number is required"	Error message: “Mobile number is required”	Pass
5	Invalid Email Format	John Doe, johndoe, 1234567890, Orthopedics, StrongPassword123!	Error message: "Invalid email format"	Error message: “Invalid email format”	Pass

Here, the Actual Results & status is written after the test execution.

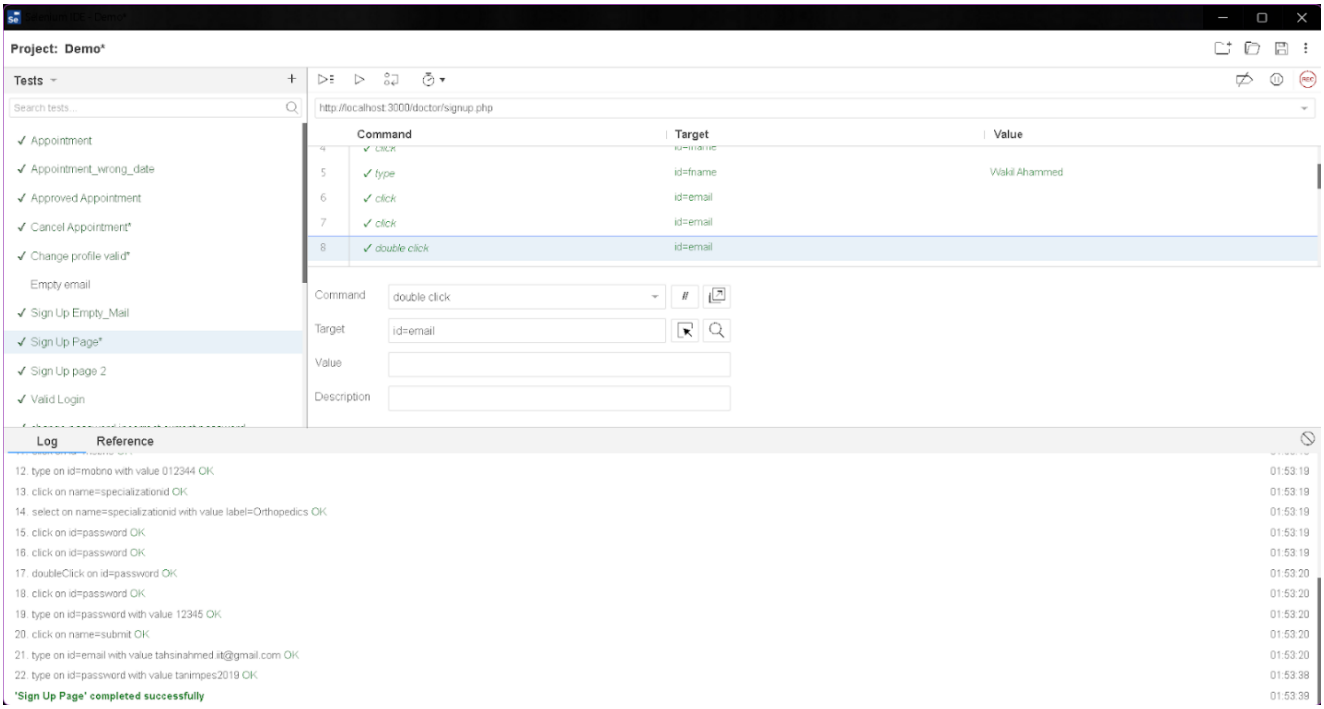

Figure 4: User Registration-1 (Functional Testing)

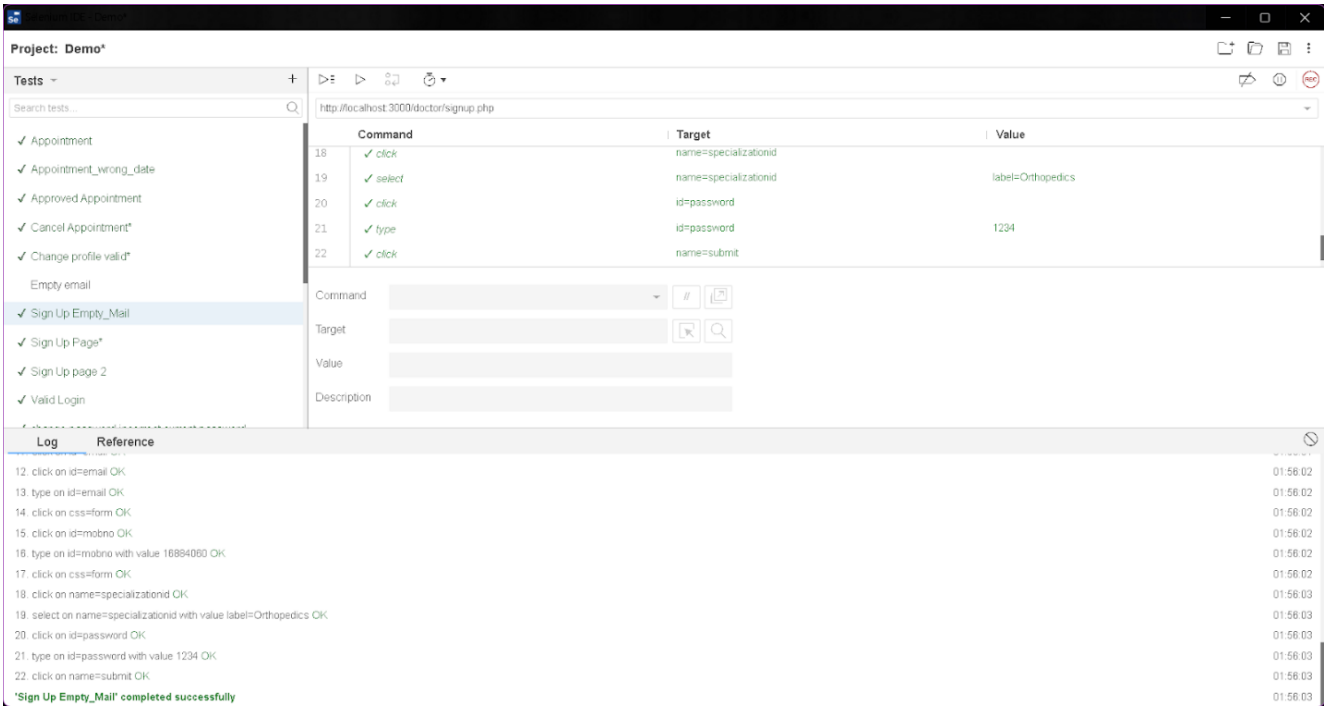

Figure 5: User Registration-2 (Functional Testing)

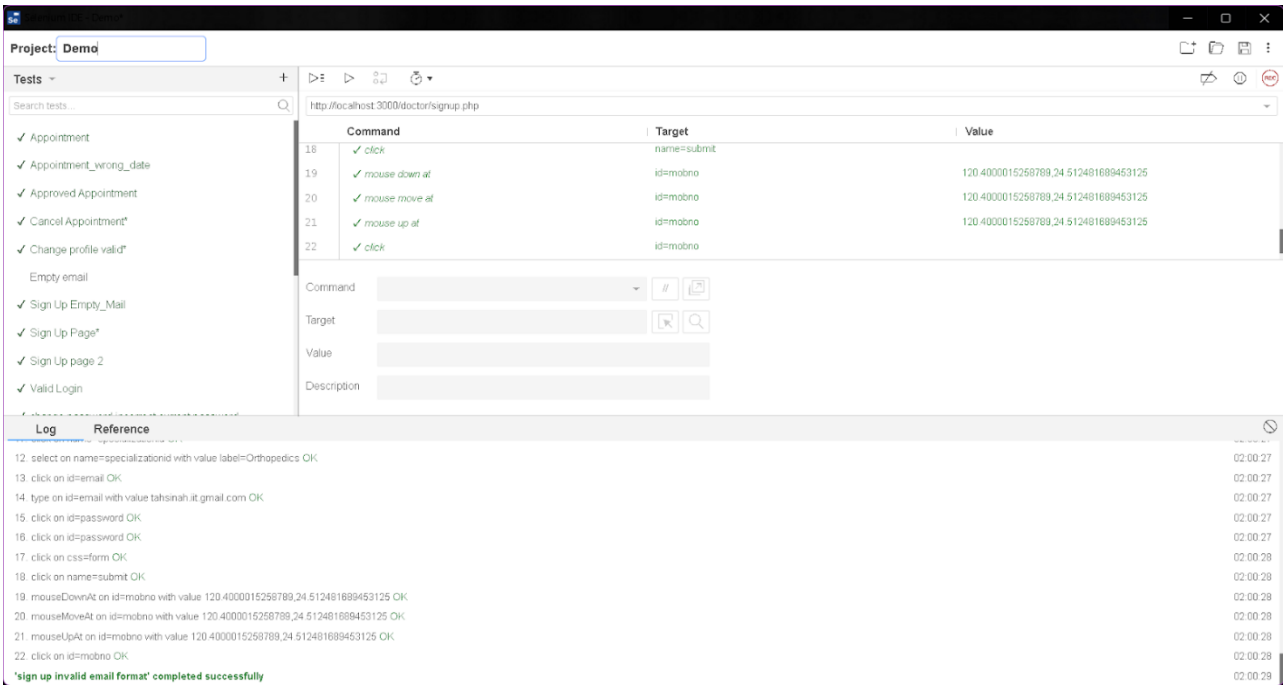

Figure 6: User Registration-3 (Functional Testing)

Table 11: Test case for user login (Functional Testing)

Test Case No.	Test Case	Test Data	Expected Result	Actual Result	Status
1	Valid Login	Valid Email: tahsinahmed.iit@gmail Valid Password: 1234567890	Successful login and redirection to homepage	Successful login and redirected to the home page	Pass
2	Invalid Login (Incorrect Password)	Valid Email: tahsinahmed.iit@gmail Valid Password: 12345	Error message: "Invalid username or password"	Error message: “invalid username or password”	Pass

3	Invalid Login (Non-existent Email)	Valid Email: tahsin.iit@gmail Valid Password: 1234567890	Error message: "Invalid username or password"	Error message: "Invalid username or password"	Pass

Here, the Actual Results & status is written after the test execution.

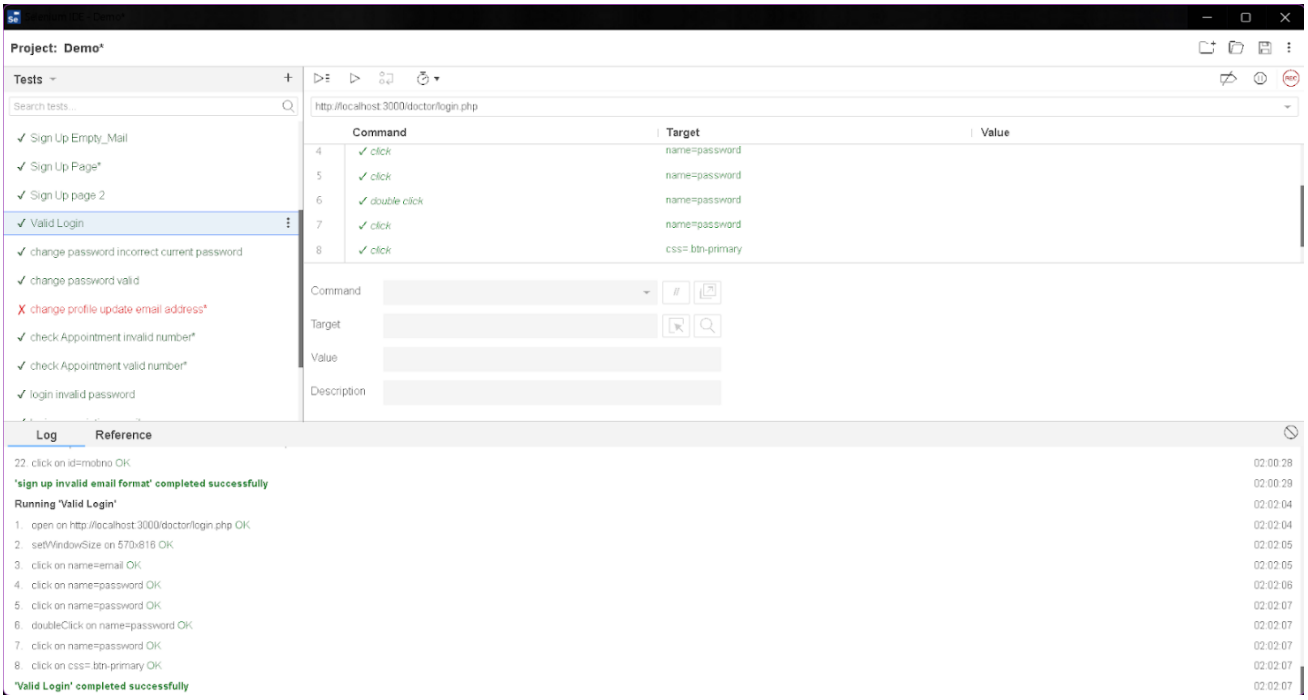

Figure 7: User Login-1 (Functional Testing)

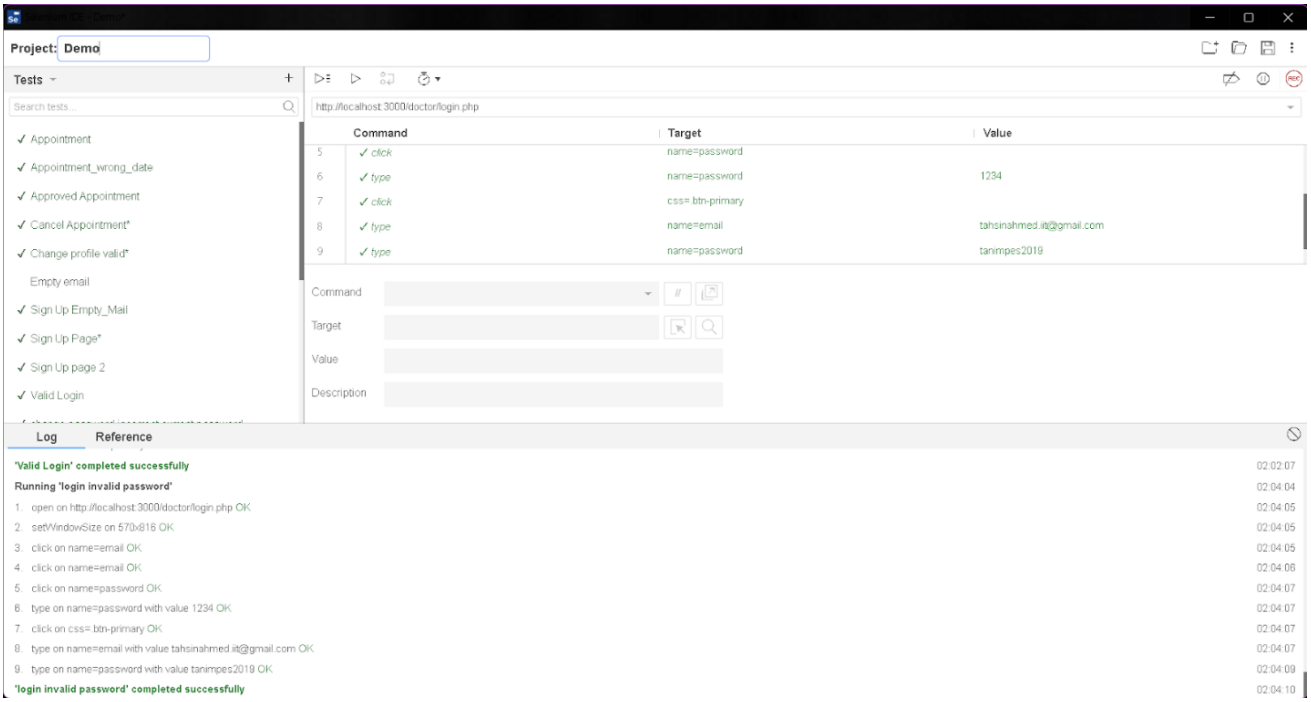

Figure 8: User Login-2 (Functional Testing)

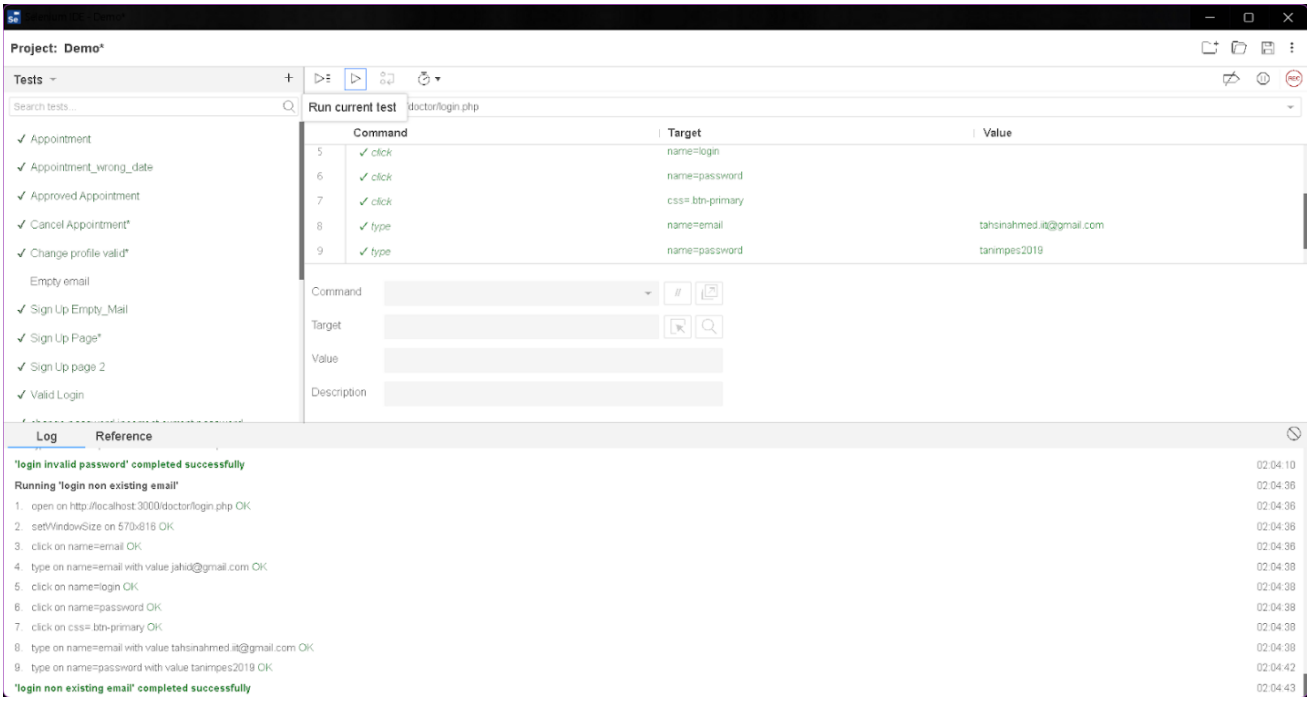

Figure 9: User Login-3 (Functional Testing)

Table 12: Test case for Appointment Booking (Functional Testing)

Test Case No.	Test Case	Test Data	Expected Result	Actual Result	Status
1	Valid Appointment Booking	**Name:** Tahsin Ahmed, Email: tahsin@gmail.com,**Phone Number:** 01854239411, **Appointment date:** Valid future date, **Appointment Time:** Valid time slot, **Specialisation:**Orthopedics, **Doctor**: Tahsin Ahmed, **Description:** "",	Appointment successfully booked and confirmation email sent to patient.	Appointment booked, confirmation email sent.	Pass
2	Invalid Appointment Date	**Name:** Tahsin Ahmed, Email: tahsin@gmail.com, **Phone Number:** 01854239411, **Appointment date:** Invalid past date, **Appointment Time:** Valid time slot, **Specialisation:**Orthopedics, **Doctor**: Tahsin Ahmed, **Description:** "",	Error message: "Please select an valid appointment date"	Error message: "Please select avalid appointment date"	Pass
3	Selecting Unavailable Time Slot	**Name:** Tahsin Ahmed, Email: tahsin@gmail.com, **Phone Number:** 01854239411, **Appointment date:** Valid future date, **Appointment Time:** invalid time slot, **Specialization:**Orthopedics, **Doctor:** Tahsin Ahmed, **Description:** "",	Error message: "Selected time slot is unavailable. Please choose another."	User sees the error message.	Pass

Here, the Actual Results & status is written after the test execution.

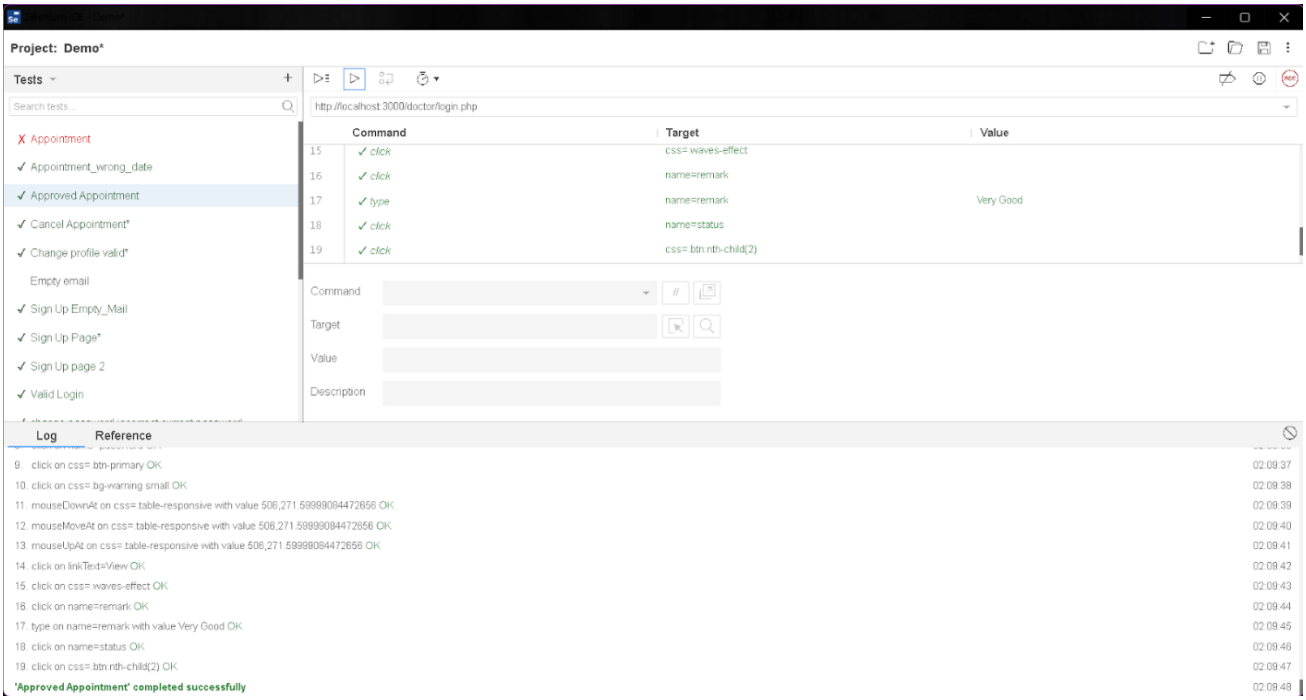

Figure 10: Valid Appointment Booking

Table 13: Test case for Checking Appointment (Functional Testing)

Test ID	Test Name	Test Data	Expected Output	Actual Output	Status
1	Valid Number	**Number :** 01688406049	Appointment Details	Appointment Details	Pass
2	Invalid Number	**Number :** 1	No Result Will be showed	Some appointment Results Shown	**Fail**

Here, the Actual Results & status is written after the test execution.

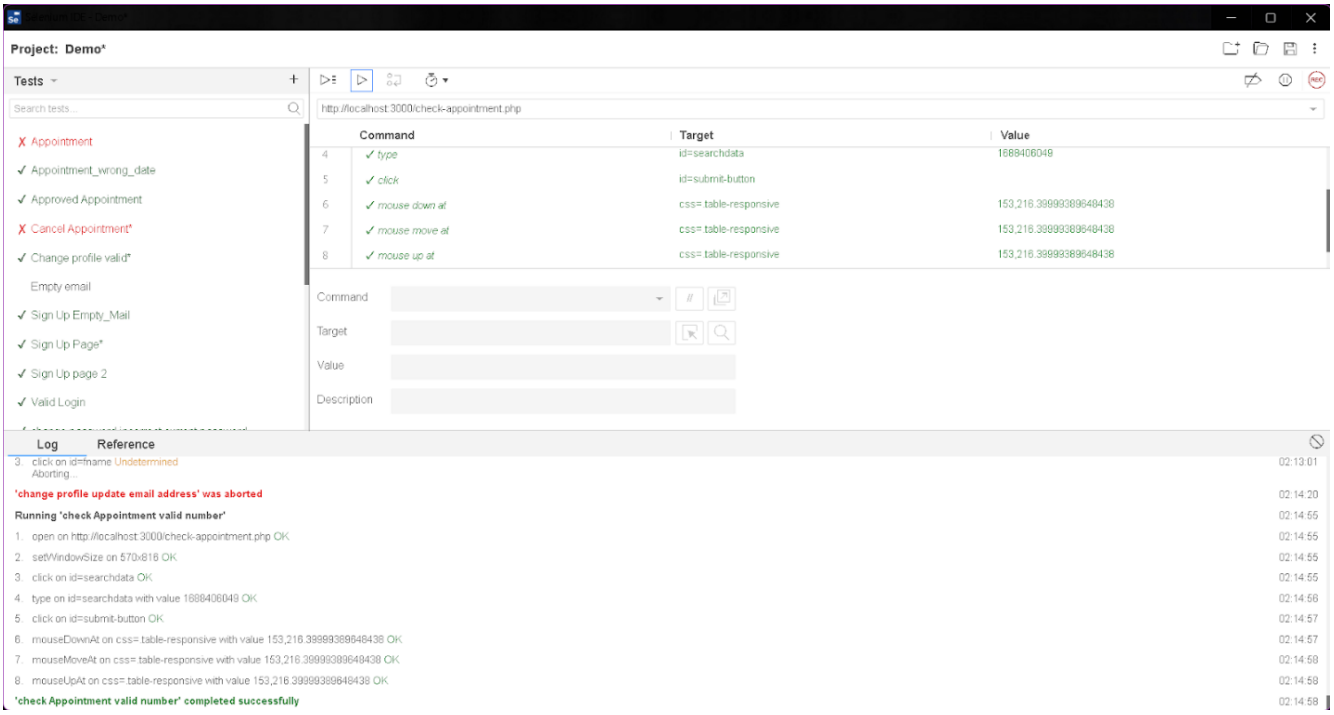

Figure 11: Check Appointment (Functional Testing)

Table 14: Test case for Appointment Management (Functional Testing)

Test ID	Test Name	Test Data	Expected Output	Actual Output	Status
1	Approve Appointment	Click on Approve	Appointment Will be approved	Appointment Approved	Pass
2	Cancel Appointment	Click on Cancel	Appointment Will be Canceled	Appointment Not Canceled	**Fail**

Here, the Actual Results & status is written after the test execution.

Table 15: Test case for Change Password (Functional Testing)

Test Case No.	Test Case	Test Data	Expected Result	Actual Result	Status
1	Valid Password Change	**Correct Current:** 1234567890 **Password, New Password:** 987654321, **Re-typed Password:** 987654321	Password successfully changed and confirmation message displayed	Password changed, confirmation message displayed.	Pass
2	Incorrect Current Password	**Incorrect Current Password:** 1234567890,**Valid New Password:**7532159, **Re-typed Password**:7532159	Error message: "Current password is incorrect"	User sees the error message "Current password is incorrect".	Pass
3	Mismatched New and Re-typed Passwords	**Correct Current: Password:** 987654321, **Valid New Password:** 7532159, **Different Re-typed Password**:75321457	Error message: "New passwords do not match"	User sees the error message "New passwords do not match".	Pass

Here, the Actual Results & status is written after the test execution.

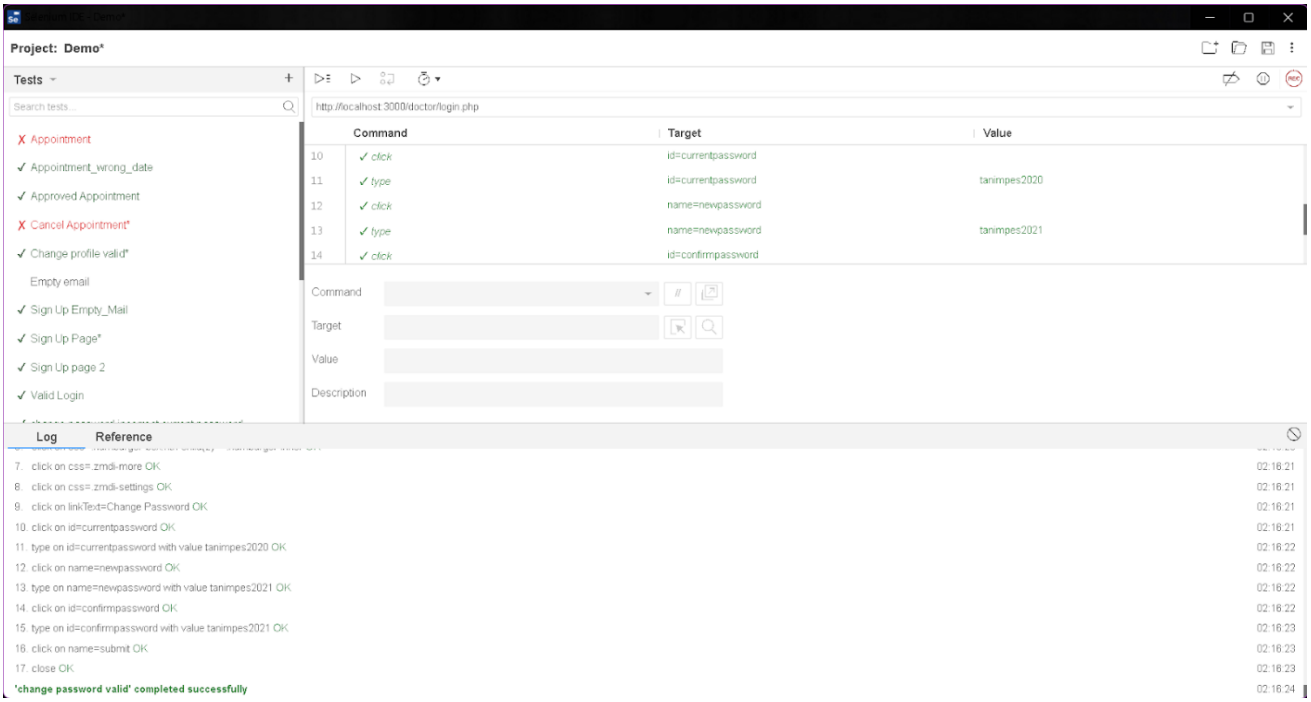

Figure 12: Change Password (Functional Testing)

Table 16: Test case for Update Profile (Functional Testing)

Test Case No.	**Test Case**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Valid Profile Update	**Name:** Mir Mohammad Tahsin (Updated), **E-mail:** tahsinAhmed@gmail.com (updated), **Phone number:**1914373594 (updated), **Specialization:** Orthopedics (Original),	Profile successfully updated	Profile successfully updated	Pass

2	Update Email Address	**Name:** Mir Mohammad Tahsin (Updated), **E-mail:** tahsinahmed@gmail.com (Original), **Phone number:**1914373594 (Original), Specialization: Orthopedics (Original),	Profile successfully updated and will be able to login with new mail	Profile successfully updated and can login with new mail	Fail
3	Empty Full Name	**Name:** “ ” (Updated), **E-mail:** tahsinahmed@gmail.com (Original), **Phone number**:1914373594 (Original), **Specialization:** Orthopedics (Original),	Error message: "Full name is required"	Error message: "Full name is required"	Fail

Here, the Actual Results & status is written after the test execution.

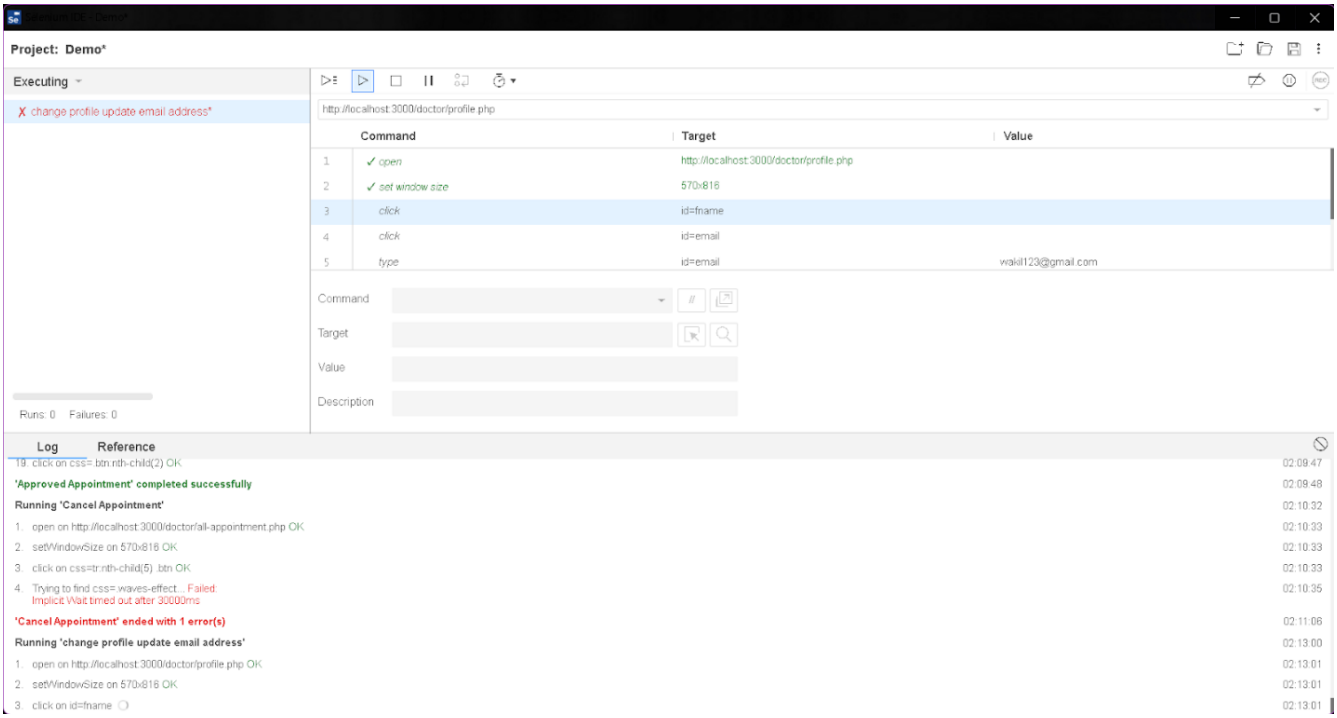

Figure 13: Update Profile (Functional Testing)

7.2 Test Execution and Results-2: Regression Testing

Regression testing aims to ensure that changes or enhancements to a software application do not negatively impact existing features.

Sign Up Function:

Here I have used previous test suit of functional testing.

I have included an input field called address on the sign-up page. After that, I ran the previous test suit selected for the sign-up page.

Previous Result:

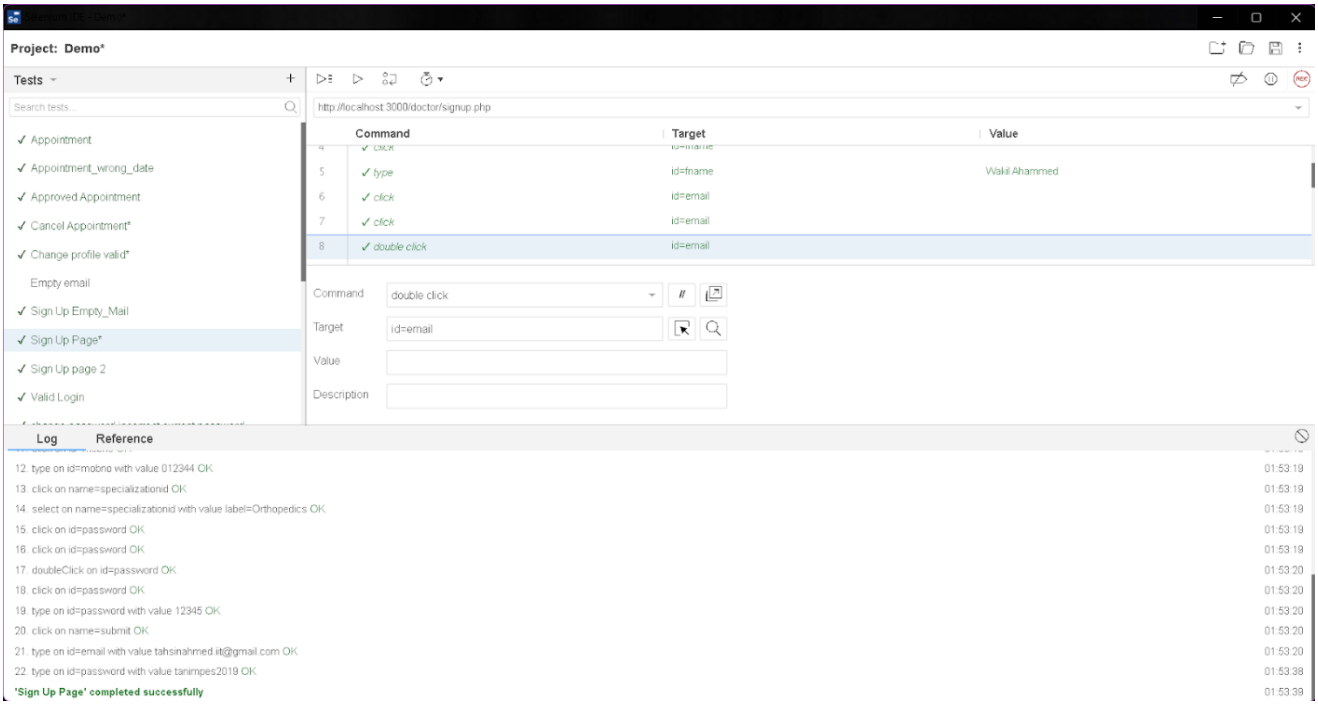

Figure 14: Sign Up Function_previous(Regression Testing)

After Changing features:

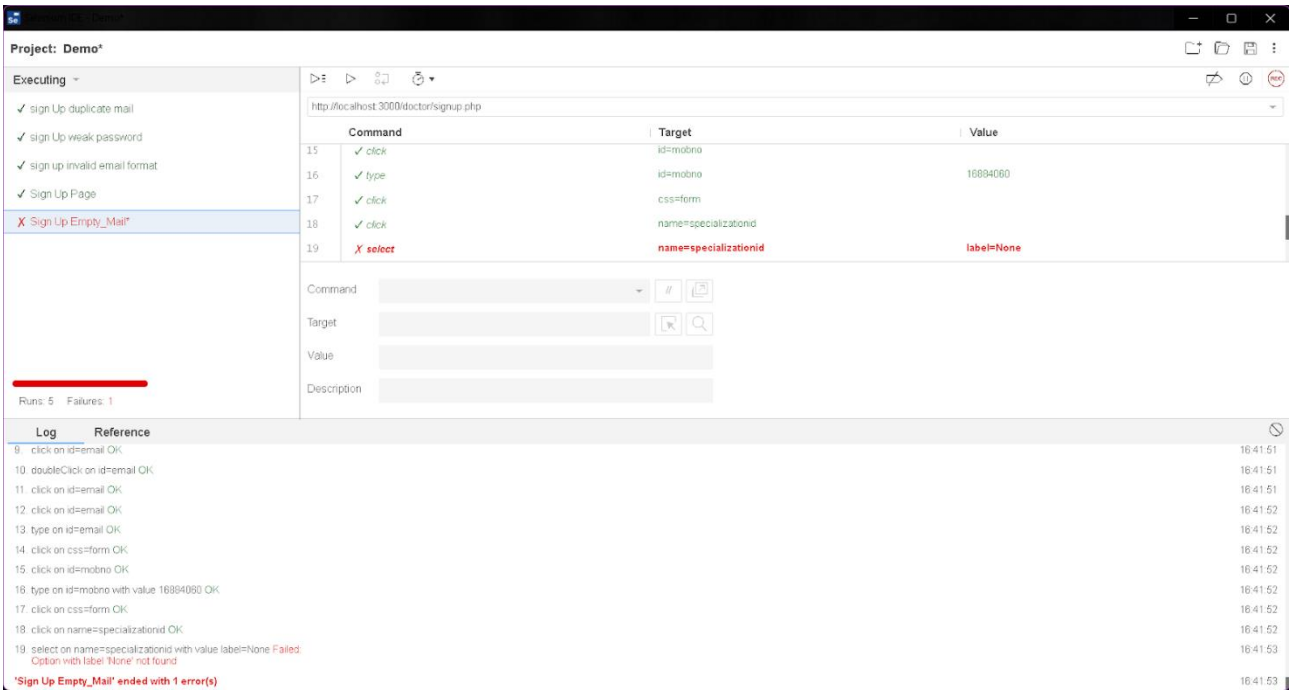

Figure 15: Sign Up Function_after(Regression Testing)

Change Profile:

Here I have used previous test suit of functional testing.

Previous Result: The test case failed before the update.

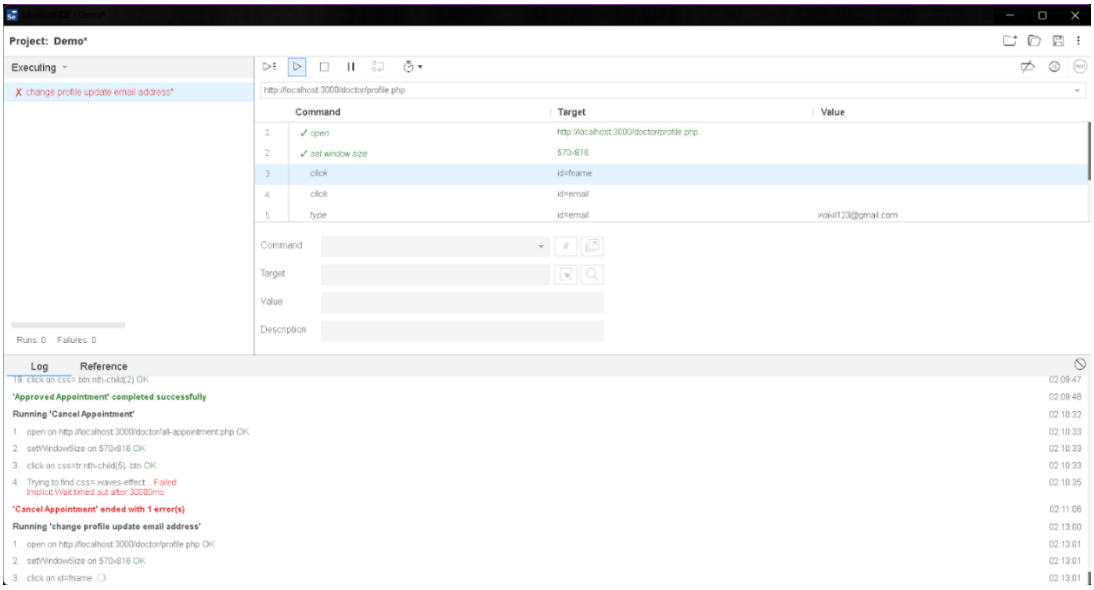

Figure 16: Change Profile_previous(Regression Testing)

After Changing features:

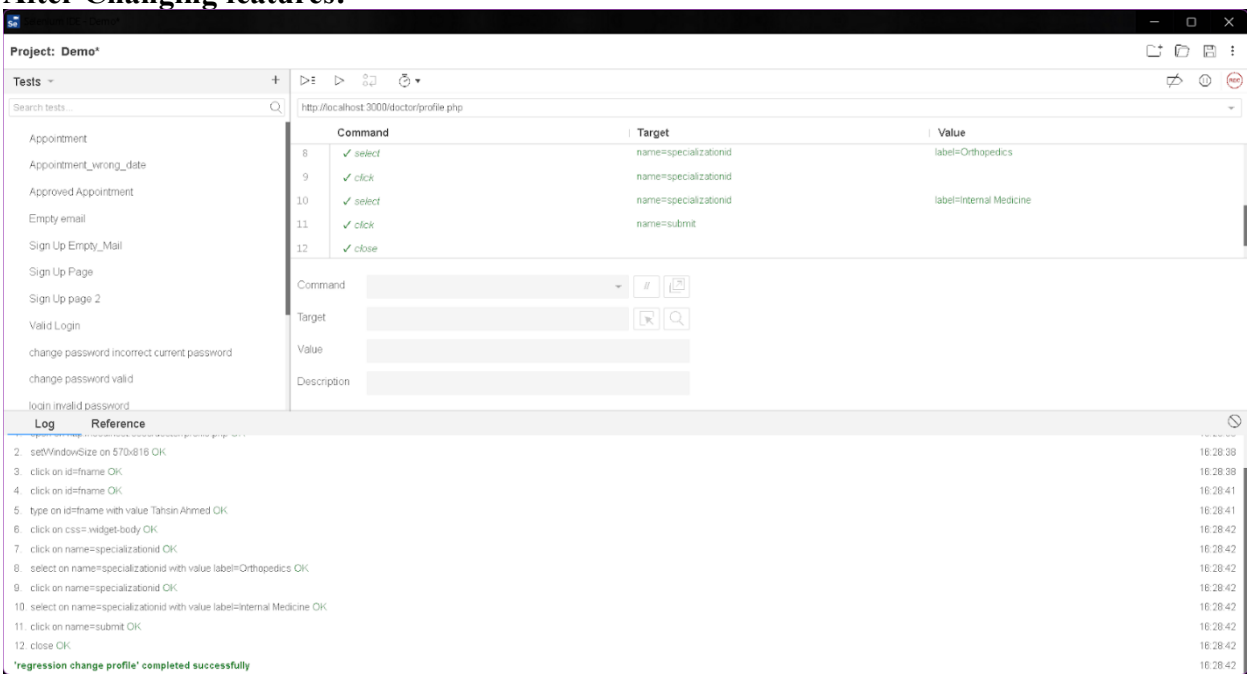

Figure 17:Change Profile_after(Regression Testing)

7.3 Test Execution and Results-3: Performance Testing

Performance Test for Doctor Appointment Response Time:

In my Doctor Appointment System,

- **Server Name or IP Address**: **localhost**
- **Port Number**: **80**
- **Path of the Application Under Test**: /Doctor-Appointment-System_PHP/Doctor-Appointment-System_Web_Technology_Project/index.php

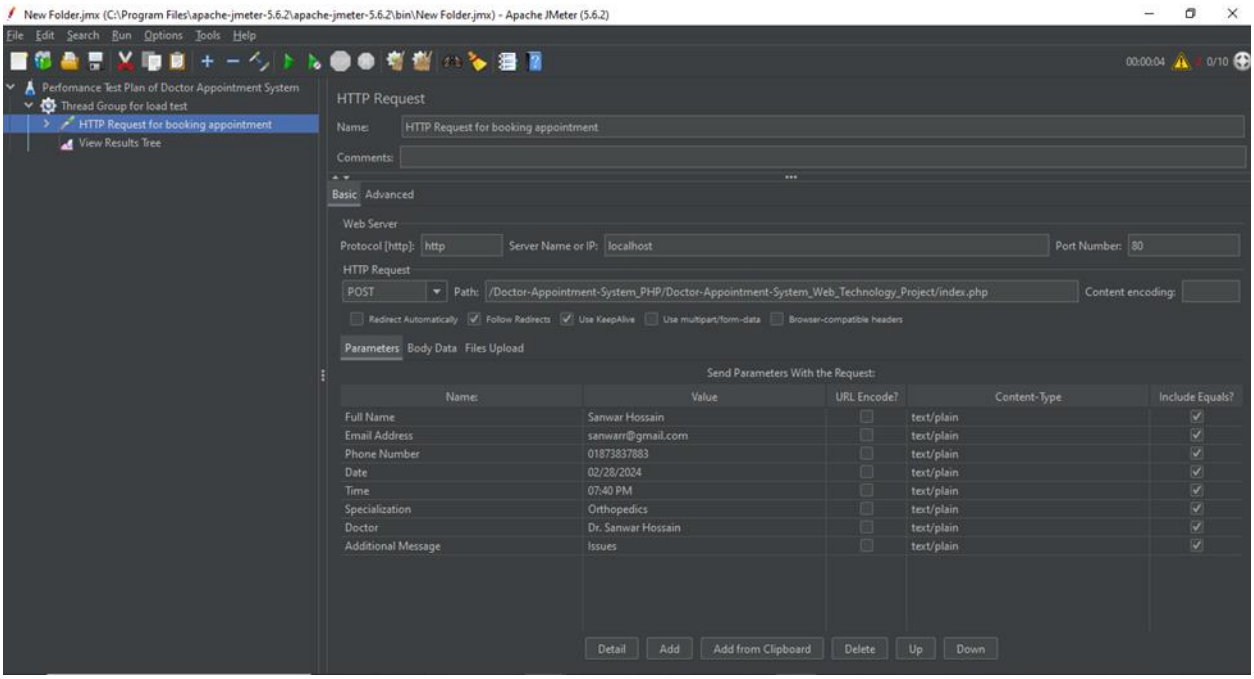

Figure 18: set parameters

To set parameters in Apache JMeter for sending a POST request with the specified input fields to book an appointment, you can follow these steps:

1. **Add HTTP Request Sampler**:
    - Right-click on the Thread Group in your Test Plan.
    - Go to Add > Sampler > HTTP Request.
    - Configure the HTTP Request Sampler to use the POST method and specify the server name, port number, and path of the application under test (e.g., **/Doctor-Appointment-System_PHP/Doctor-Appointment-System_Web_Technology_Project/index.php**).
2. **Add Parameters**:
    - In the HTTP Request Sampler, go to the "Parameters" tab.
    - Click on the "Add" button to add parameters for each input field.
3. **Set Parameters**:
    - For each parameter, specify the name and value according to the input fields you mentioned.
    - Here's how you can set parameters for the given input fields:
        - Parameter 1: Full name
            - Name: **FullName**
            - Value: Your desired full name (e.g., Md Sanwar Hossain)
        - Parameter 2: Email address
            - Name: **emailAddress**
            - Value: Your desired email address (e.g., sanwar@gmail.com)
        - Parameter 3: Phone Number
            - Name: **PhoneNumber**
            - Value: Your desired phone number (e.g., 123-456-7890)

- Parameter 4: Date (mm/dd/yyyy)
  - Name: **Date**
  - Value: Your desired date in the format mm/dd/yyyy (e.g., 02/28/2024)
- Parameter 5: Select specialization
  - Name: **specialization**
  - Value: The selected specialization from the database (e.g., Cardiology)
- Parameter 6: Select Doctor
  - Name: **doctorName**
  - Value: The selected doctor's name from the database (e.g., Dr. Sanwar)
- Parameter 7: Additional Message (text)
  - Name: **additionalMessage**
  - Value: Your additional message (if any)

4. **Configure HTTP Request**:
   - In the "Path" field of the HTTP Request Sampler, specify the path to the booking endpoint (e.g., **/book-appointment**).
5. **Add HTTP Header Manager** (Optional):
   - If required, you can add an HTTP Header Manager to specify additional headers such as Content-Type.
6. **Add a View Results Tree Listener**:
   - Add a View Results Tree listener to the Thread Group to view the results of the POST request.
7. **Run the Test**:
   - Save your Test Plan and run the test to send the POST request with the specified parameters.

Test result:

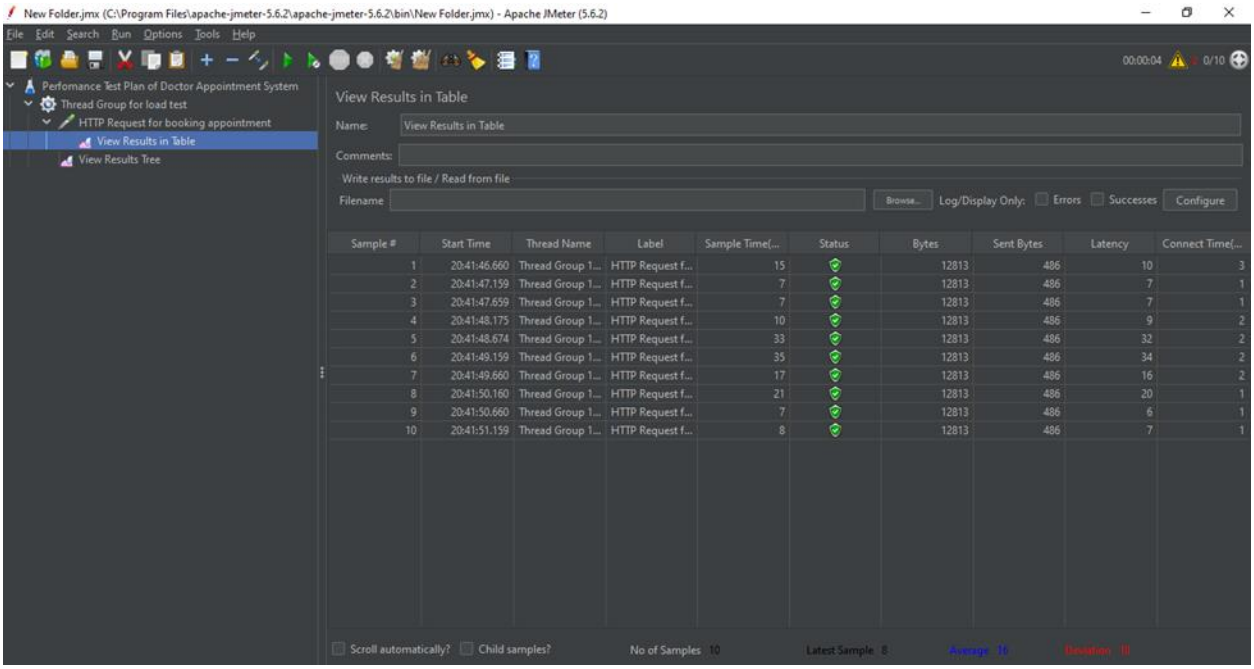

Figure 19: View Results in Table

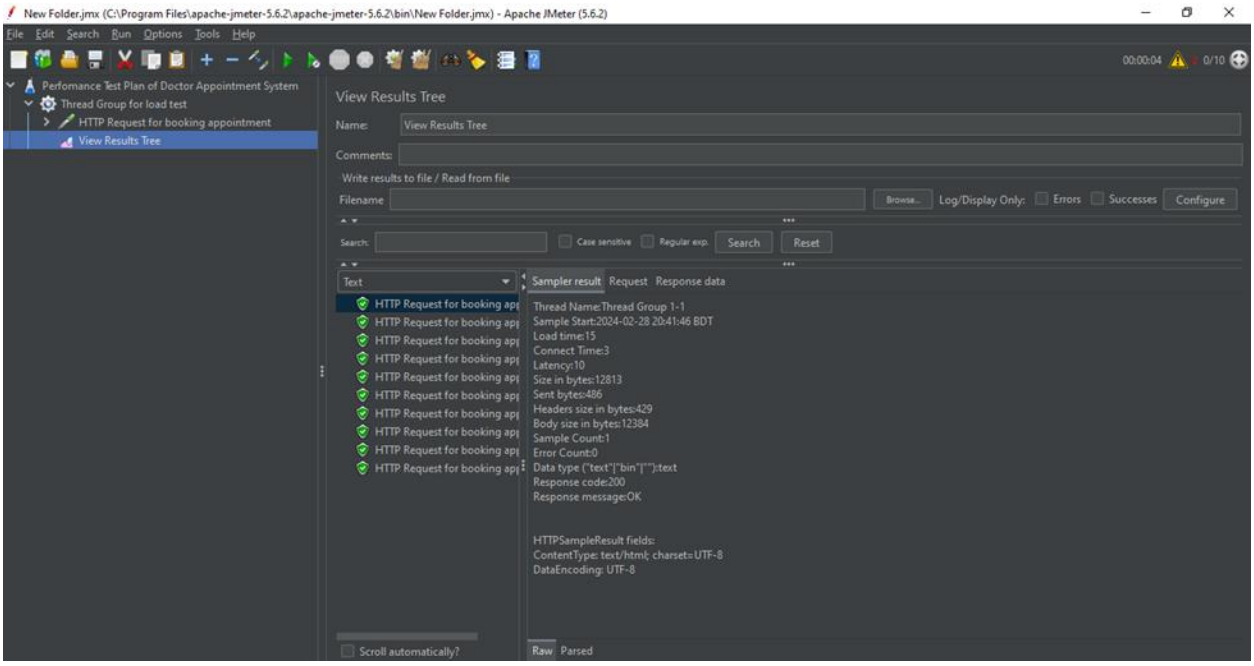

Figure 20: View Results Tree

Result: /book-appointment average load time<=15 milliseconds

Performance Test for search appointment by number:

- **Server Name or IP**: **localhost** (if running locally)
- **Port Number**: **80** (assuming default HTTP port)
- **Path**: **/search-appointment.php**
- **Parameters**:
    - Name: **searchdata**
    - Value: (the mobile number you want to search for)

Ensure that the parameter name (**searchdata**) and the PHP script's field name for the mobile number search input match.

Related Screenshots:

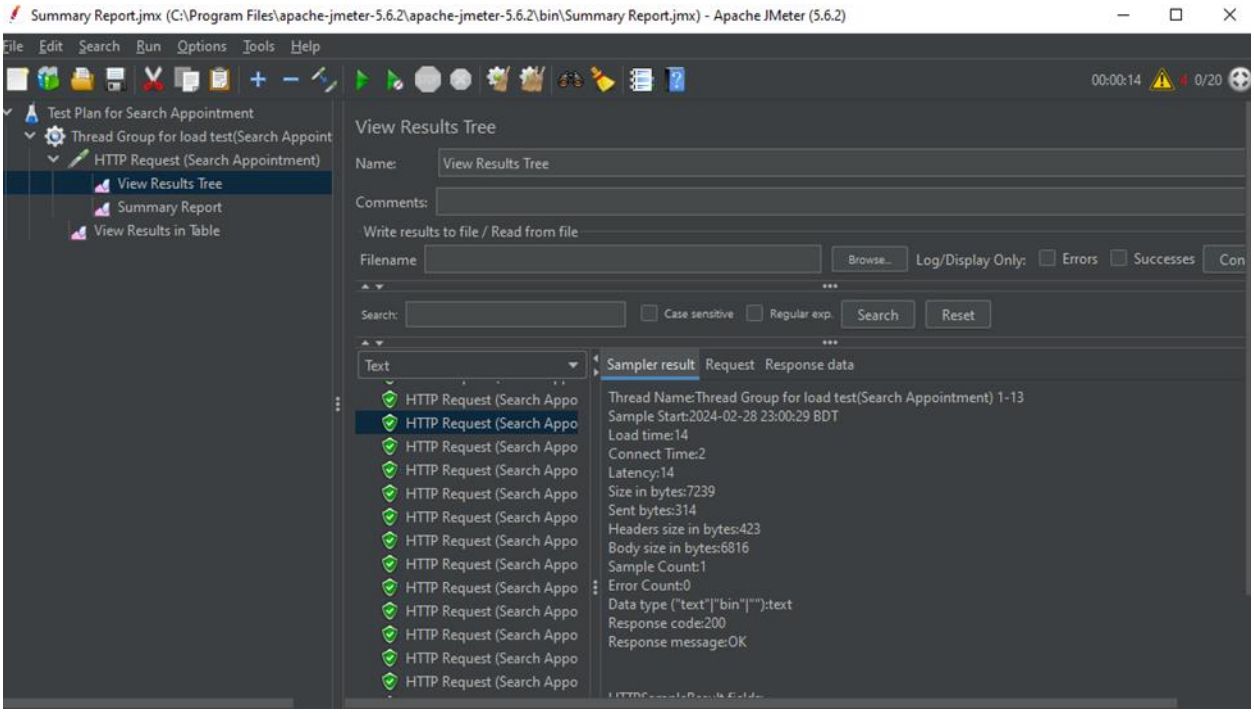

Figure 21: Add Thread Group, protocol, Server name, Port number, Http Request, path and parameter

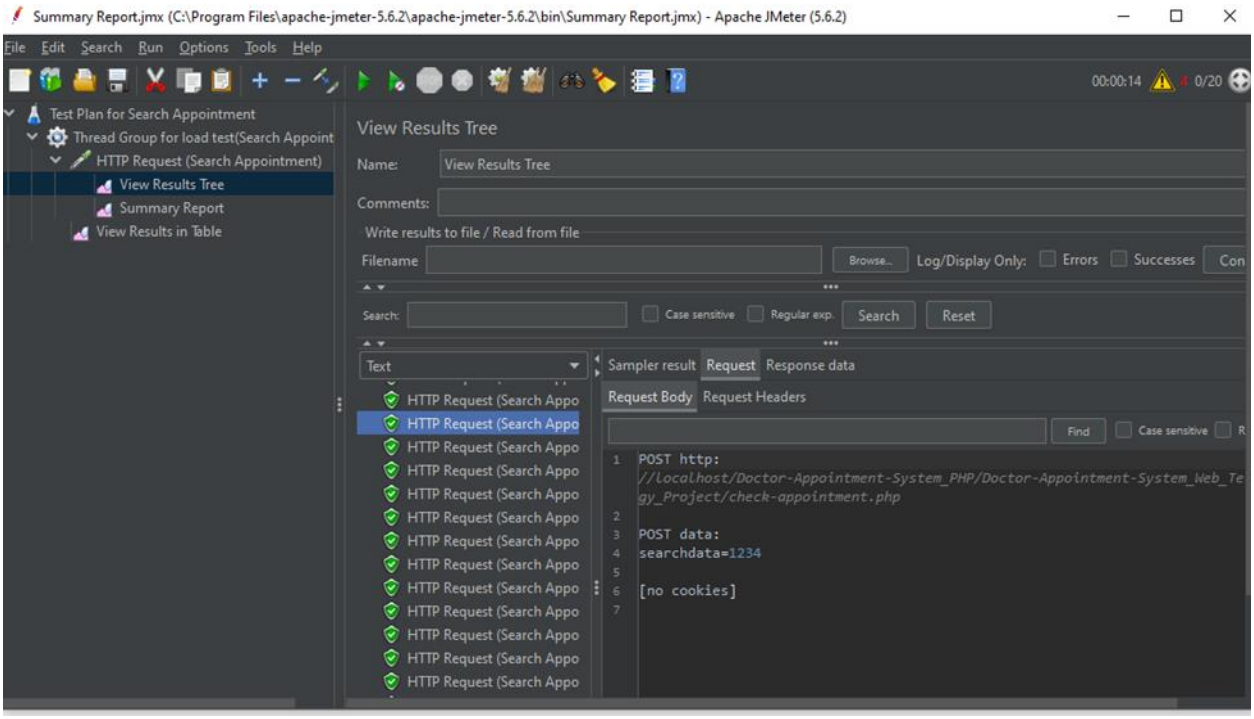

Figure 22: View Results Tree

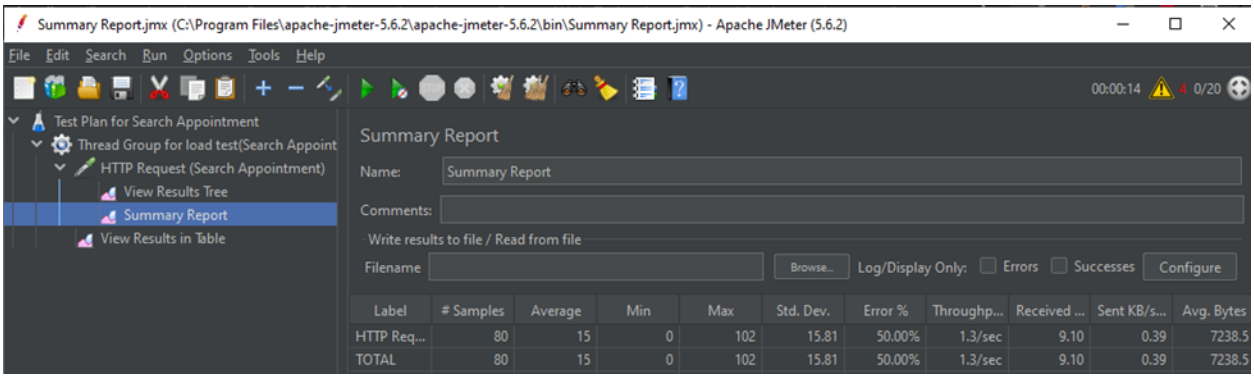

Figure 23: Summary Report

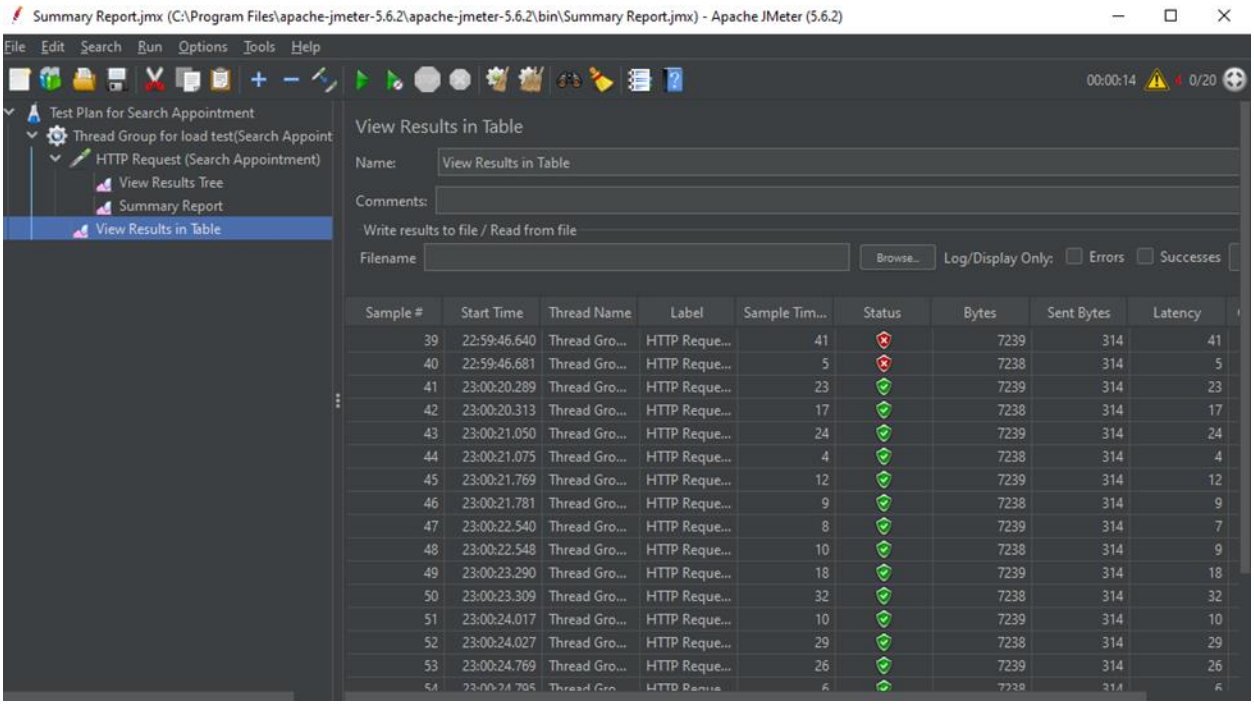

Figure 24: view results in table

Result: /search-appointment average load time- <40 milliseconds

7.4 Test Execution and Results-4: Security Testing

Misconfiguration in Name Change:

The system should have proper sanitization. Users should follow the system's requirements to change their names (**Display name doesn't contain special characters**).
Here, the tested system doesn't maintain proper sanitization, but as a middleman, attackers can easily break this sanitization.

Because of this vulnerability, attackers can easily add malicious payloads to this system.

Misconfiguration in Password Change/Sign up password:

Users should follow all criteria to set/update their password. (Example: password should contain both upper-case and lower-case characters to strengthen passwords).
Here, the tested system doesn't maintain proper sanitization, but as a middleman, attackers can easily guess pass in case of sanitization absence.

Because of this vulnerability, attackers can easily try the password brute-force attack to take over an account with a common wordlist.

Account takeover via inappropriate password reset:

Intercepting the request, we quickly get the user's mail and mobile number.

The password reset link should be sent to the valid email address of the account holder and only be there.

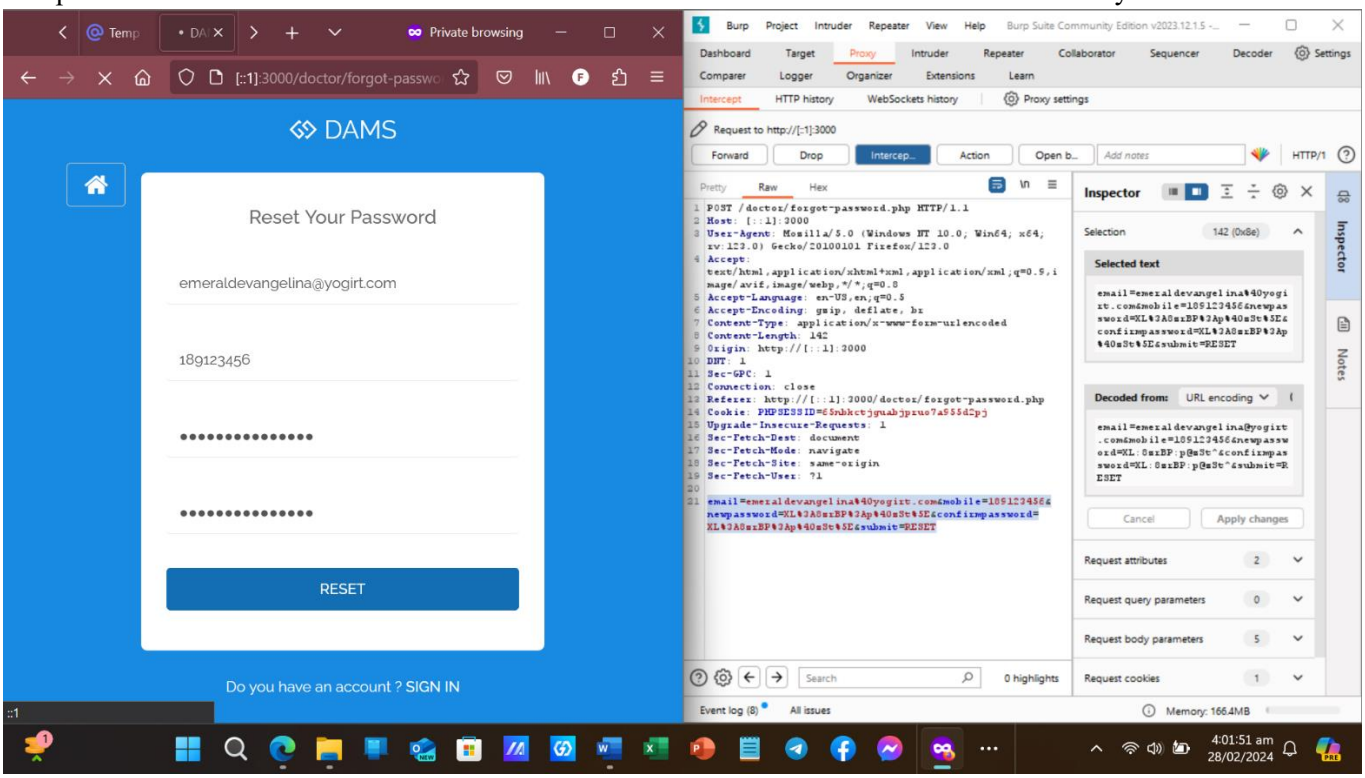

Figure 25 Inappropriate password reset

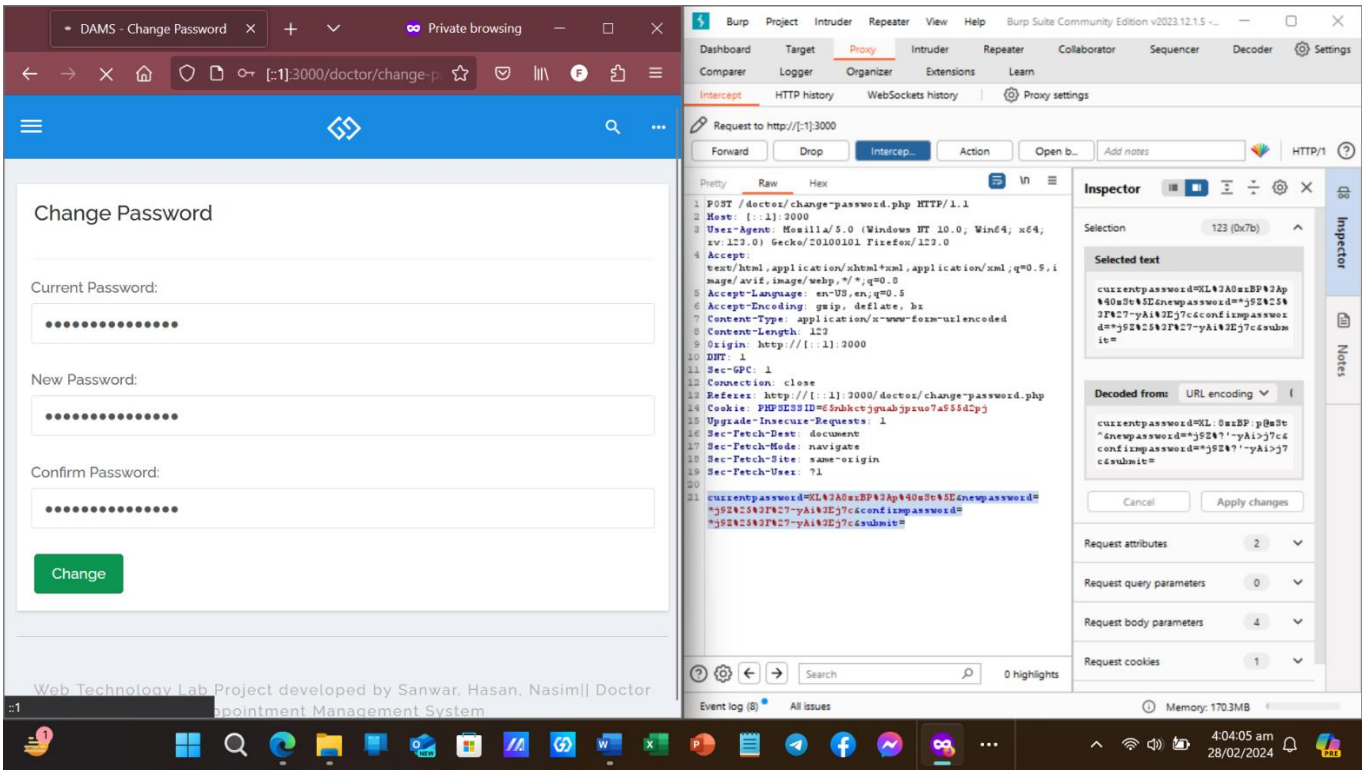

Figure 26 pass-reset-vulnerability-from dashboard

Password brute-force:

The system should limit failed login attempts to prevent brute-force attacks. However, the system does not react to password brute-force attacks.

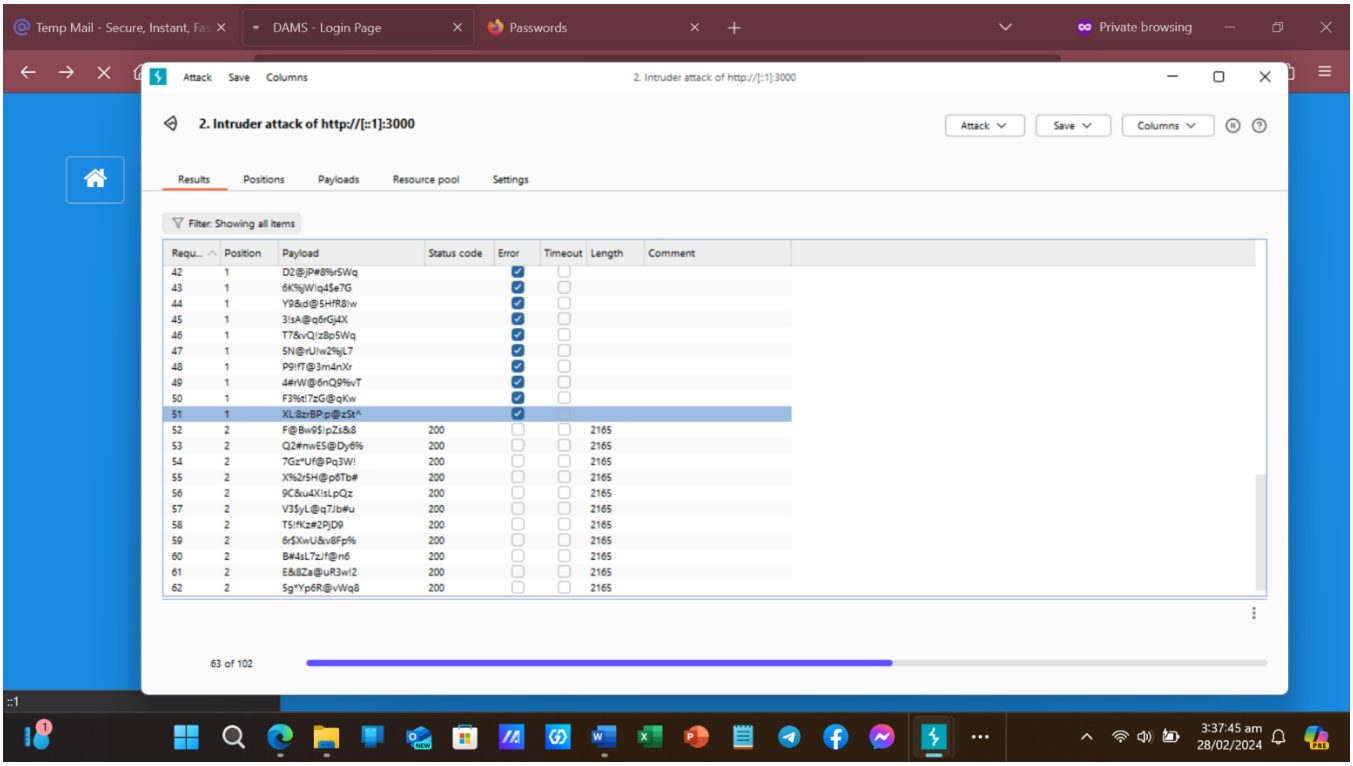

Figure 27 start brute force attack

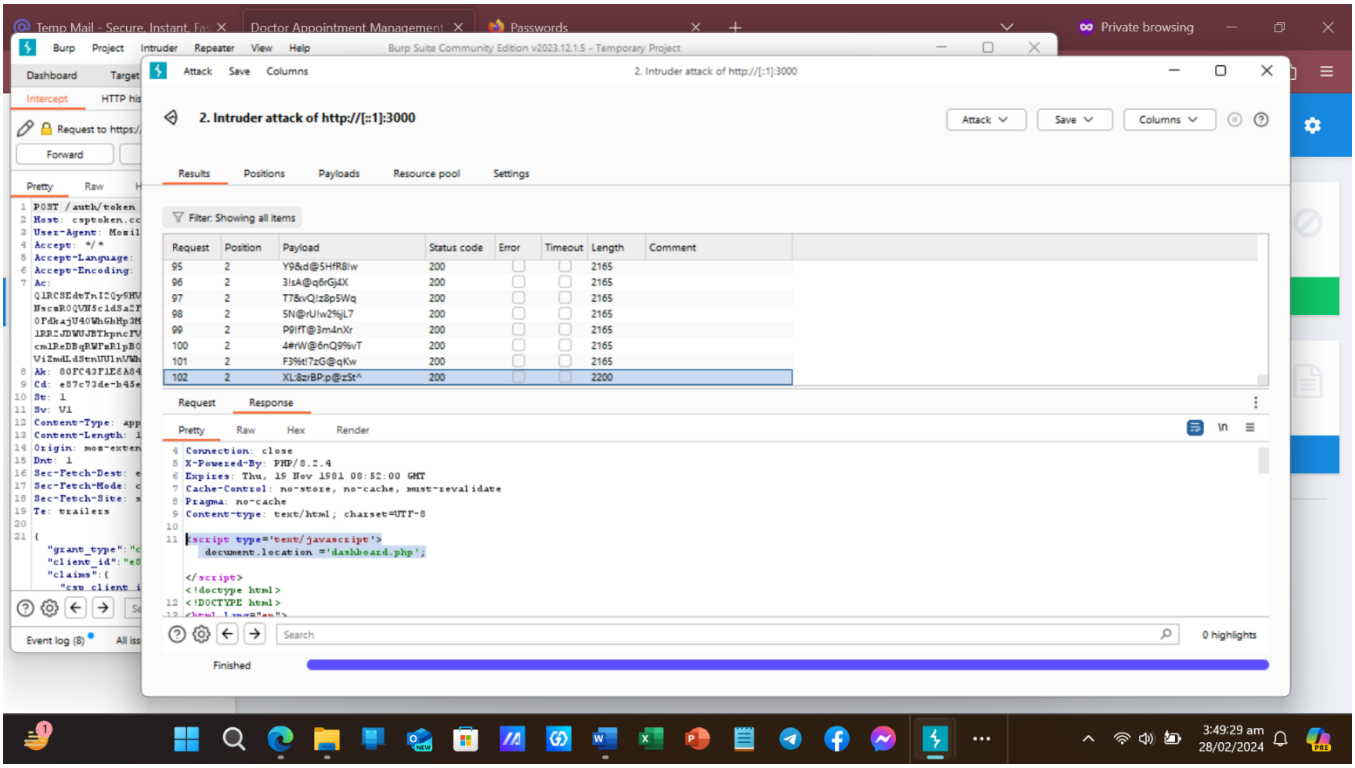

Figure 28 Redirect dashboard for accurate pass while brute force

8. Challenges

Here are some potential challenges that you might face during the testing of the Doctor Appointment System:

- **Complex Business Logic:** The Doctor Appointment System may involve complex business logic related to appointment scheduling, resource allocation, and user management. Understanding and testing this intricate logic thoroughly can be challenging.
- **Integration Issues:** Integrating the Doctor Appointment System with external systems such as Electronic Health Record (EHR) systems or payment gateways may pose challenges. Ensuring seamless data exchange and interoperability with these systems requires thorough integration testing.
- **Cross-Browser Compatibility:** Ensuring cross-browser compatibility across different web browsers and versions can be challenging, especially considering the varying rendering engines and CSS implementations.
- **Responsive Design:** Testing the responsiveness of the application across various devices and screen sizes requires meticulous attention to detail. Ensuring that the user interface adapts and functions correctly on desktops, laptops, tablets, and smartphones is crucial.
- **Performance Optimization:** Identifying and optimizing performance bottlenecks to ensure the application's responsiveness and scalability under varying load conditions can

be challenging. Performance testing is essential to detect and address issues related to response time, throughput, and resource utilization.

- **Security Concerns:** Ensuring the security of patient information and protecting against potential security vulnerabilities such as SQL injection, cross-site scripting (XSS), and data breaches is critical. Conducting thorough security testing to identify and mitigate these risks can be challenging.
- **User Acceptance:** Gathering meaningful feedback from real users or stakeholders during user acceptance testing (UAT) can be challenging. Ensuring that the application meets user expectations and requirements requires effective communication and collaboration with stakeholders.
- **Resource Constraints:** Limited resources such as time, budget, and manpower can pose challenges during testing. Prioritizing testing activities, optimizing test processes, and leveraging automation can help overcome resource constraints.
- **Regression Testing:** Managing regression testing effectively, especially in agile environments where frequent changes are made to the application, can be challenging. Ensuring that new changes do not introduce unintended side effects or break existing functionality requires thorough regression testing.
- **Documentation and Reporting:** Maintaining comprehensive documentation of test plans, test cases, test results, and defects encountered during testing can be challenging. Generating meaningful test reports and communicating testing findings effectively to project stakeholders is essential for decision-making.

Addressing these challenges requires careful planning, effective communication, collaboration across teams, and leveraging appropriate testing strategies and tools. By proactively identifying and addressing these challenges, you can ensure the successful testing and delivery of the Doctor Appointment System.

9. Recommendations

Here are some recommendations for testing the Doctor Appointment System project:

- **Comprehensive Test Coverage:** Ensure comprehensive test coverage by designing and executing test cases that cover all functional requirements, including appointment booking, appointment management, user authentication, error handling, and integration with external systems.
- **Cross-Browser Compatibility Testing:** Conduct thorough cross-browser compatibility testing to ensure that the application works seamlessly across different web browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, etc. Test on multiple browser versions to identify and address any compatibility issues.
- **Responsive Design Testing:** Verify the responsiveness of the application by testing it on various devices with different screen sizes and resolutions, including desktops, laptops,

tablets, and smartphones. Ensure that the user interface adapts and displays correctly on all devices.

- **Performance Testing:** Perform performance testing using tools like Apache JMeter to evaluate the responsiveness and scalability of the application under various load conditions. Test the system's response time, throughput, and resource utilization to identify and address any performance bottlenecks.
- **Security Testing:** Conduct security testing to identify and mitigate potential security vulnerabilities such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and data breaches. Implement security best practices and encryption mechanisms to protect sensitive patient information.
- **User Acceptance Testing (UAT):** Involve real users or stakeholders in user acceptance testing to validate the application against their expectations and requirements. Gather feedback and incorporate necessary changes to improve user satisfaction and usability.
- **Automated Testing:** Implement automated testing using tools like Selenium WebDriver and TestNG to streamline the testing process and improve test coverage. Automate repetitive test cases, regression tests, and smoke tests to ensure efficient and consistent testing.
- **Integration Testing:** Conduct integration testing to verify the seamless integration of the Doctor Appointment System with external systems such as Electronic Health Record (EHR) systems, payment gateways, and third-party APIs. Validate data exchange, interoperability, and functionality across integrated systems.
- **Load Testing:** Perform load testing to evaluate the system's performance and stability under expected and peak load conditions. Simulate many concurrent users and monitor system response time, throughput, and resource utilization to ensure optimal performance.
- **Documentation and Reporting:** Maintain detailed documentation of test plans, test cases, test results, and defects encountered during testing. Generate comprehensive test reports to communicate the testing process, findings, and recommendations to project stakeholders.

By implementing these testing recommendations, you can ensure the reliability, usability, performance, and security of the Doctor Appointment System, leading to enhanced user satisfaction and improved overall quality of the application.

10. Conclusion

The testing of the Doctor Appointment System plays a crucial role in ensuring its reliability, security, and performance in delivering enhanced healthcare accessibility to users. Through a comprehensive testing strategy encompassing functional testing, usability testing, security testing, and performance testing, the system has been thoroughly evaluated against predefined test requirements and objectives. Functional testing has validated that all features of the system work

as intended, meeting the functional requirements specified for user registration, appointment booking, management, and automated reminders. Usability testing has confirmed that the user interface is intuitive and easy to navigate, ensuring a seamless experience for patients and healthcare providers alike. Security testing has identified and addressed potential vulnerabilities, ensuring the confidentiality and integrity of patient information stored within the system. Robust authentication mechanisms and data encryption protocols have been implemented to safeguard against unauthorized access and data breaches. Performance testing has evaluated the system's responsiveness and scalability under various load conditions, ensuring optimal performance even during peak usage periods. The system has demonstrated satisfactory response times and stability, meeting predefined benchmarks and service level agreements.

The thorough testing conducted throughout the development lifecycle of the Doctor Appointment System has validated its functionality, usability, security, and performance, ultimately delivering a reliable and user-friendly solution that enhances healthcare accessibility and patient satisfaction. Moving forward, continuous monitoring and testing will be essential to maintain the system's effectiveness and address any emerging issues or requirements in the dynamic healthcare landscape.