# Manual and Automated Testing of Booking.com Using Selenium & TestNG



**The Upskilling Program**

**AlHussein Technical University (HTU)**

**June 2025**

**Ahmad Zapen**

**Supervised by:**

**Eng. Abdulraheem Alaska**

**Dr. Basheer Mufleh**

**Coach. Madonna Samawi**

Table of Contents

[Manual and Automated Testing of Booking.com Using Selenium & TestNG 1](#_Toc200629652)

[EXECUTIVE SUMMARY 3](#_Toc200629653)

[1. INTRODUCTION 4](#_Toc200629654)

[2. LITERATURE REVIEW 5](#_Toc200629655)

[3. PROBLEM STATEMENT 7](#_Toc200629656)

[4. STAKEHOLDERS AND THEIR REQUIREMENTS 8](#_Toc200629657)

[4.1 Key stakeholders include: 8](#_Toc200629658)

[4.2 Functional requirements: 8](#_Toc200629659)

[4.3 Constraints: 8](#_Toc200629660)

[5. DESIGN CONCEPTS 9](#_Toc200629661)

[5.1 TECHNICAL REQUIREMENTS 9](#_Toc200629662)

[5.2 SOLUTION/SYSTEM OVERVIEW 9](#_Toc200629663)

[5.3 SOLUTION/SYSTEM DETAILS 10](#_Toc200629664)

[5.3.1 Language & UI Testing 10](#_Toc200629665)

[5.3.2 Search Functionality 10](#_Toc200629666)

[5.3.3 Booking Form 10](#_Toc200629667)

[5.3.4 Payment Page (Iframe Handling) 10](#_Toc200629668)

[5.3.5 Edge Case and Negative Testing 11](#_Toc200629669)

[5.4 PROTOTYPING AND TESTING 11](#_Toc200629670)

[CONCLUSION AND FUTURE WORK 12](#_Toc200629671)

[REFERENCES 13](#_Toc200629672)

[APPENDICES 15](#_Toc200629673)

# EXECUTIVE SUMMARY

Online booking platforms require precision, speed, and adaptability across global audiences. To ensure a seamless user experience, we developed a full-stack automation suite for Booking.com using Java, Selenium WebDriver, and TestNG. The framework validates core user flows including multilingual navigation, dynamic city search, subscription handling, accommodation filtering, room booking, and payment submission. It adapts to both English and Arabic interfaces, handles iframes and pop-ups, and verifies system behavior across randomized user inputs. The system achieved 100% pass rate across 14 critical test cases. Key challenges included handling asynchronous content, language-specific behaviors, and payment iframe processing. The solution improves test coverage, accelerates regression cycles, and ensures platform reliability at scale. Future iterations will integrate CI/CD pipelines, expand data-driven testing, and enhance reporting with Allure or Extent Reports.

**KEYWORDS:** Selenium, Java, TestNG, Booking.com automation, multilingual UI testing, end-to-end web testing, payment flow automation

# INTRODUCTION

Modern travel platforms like Booking.com must provide seamless, multilingual, and error-free user experiences to meet the expectations of global users. Ensuring such quality requires robust end-to-end testing that mimics real user behavior across diverse scenarios. This project focuses on the development of a comprehensive automation testing suite for Booking.com using Selenium WebDriver with Java and TestNG. The suite automates and validates key functionalities including language and currency settings, search workflows, guest configurations, subscription handling, booking steps, and secure payment entry. It supports both English and Arabic flows, integrates random data generation for test robustness, and handles dynamic content and real-time UI changes. The project aims to enhance reliability, reduce manual testing time, and serve as a scalable quality assurance solution for dynamic web applications.

# LITERATURE REVIEW

Automated software testing has become essential in ensuring web application quality, particularly for complex, user-centric platforms like Booking.com. According to Myers et al. [1], manual testing alone cannot keep pace with the rapid development cycles required in modern web environments. Automation improves testing efficiency, coverage, and repeatability while reducing human error and long-term maintenance costs.

In the domain of web testing, Selenium WebDriver has emerged as a leading open-source tool due to its cross-browser compatibility and support for multiple programming languages [2]. Researchers have emphasized its flexibility in simulating user behavior and validating UI components [3]. Frameworks like TestNG enhance Selenium by providing advanced test configurations, annotations, and parallel execution capabilities, making it suitable for large-scale test automation [4].

International studies, such as those by Agarwal et al. [5], have shown that automating multilingual interfaces introduces unique challenges, particularly in ensuring element detection, layout stability, and consistent behavior across languages. Arabic, with its right-to-left (RTL) layout, presents additional testing complexities such as mirrored UI rendering, dynamic text alignment, and Unicode compliance [6].

In e-commerce and travel platforms, UI/UX testing of core flows—such as search, booking, and payment—is critical due to their direct impact on conversion rates and user trust. Zhang and Chen [7] highlighted the importance of validating real-time responses, error handling, and input validation in dynamic content environments. Booking systems that allow for multi-language and multi-currency usage require robust end-to-end validation under different configurations, emphasizing the need for parameterized and data-driven testing approaches.

The integration of tools for random data generation and assertion libraries has also been explored in recent literature. These practices enhance test resilience by uncovering edge cases that might be missed during static testing [8]. Moreover, the use of Page Object Models (POM) in Selenium-based projects has been proven to improve maintainability, scalability, and reusability of test scripts, especially in projects with evolving UIs [9].

While commercial testing tools like Test Complete and Katalon offer visual scripting and integrated reporting, they often come at a high licensing cost and lack the customization flexibility needed for advanced use cases [10]. Open-source stacks based on Java and Selenium continue to dominate in enterprise environments due to their adaptability and strong community support.\

Despite the advances, there remains a gap in literature and practice concerning end-to-end testing of multilingual, region-specific implementations of global platforms like Booking.com. This project aims to address that gap by focusing on automation of Booking.com workflows in both English and Arabic, simulating real-world user interactions while managing challenges such as localization, dynamic content, and payment verification.

# PROBLEM STATEMENT

Booking.com is a global online travel platform that must maintain high levels of reliability, performance, and multilingual accessibility. Manual testing of such a large platform is time-consuming, repetitive, and prone to human error, especially for critical paths like hotel search, booking, and payment. The challenge lies in ensuring consistent behavior across different browsers, devices, and languages (especially Arabic RTL support). This project addresses the need for a scalable automated testing framework using Selenium and TestNG to improve test coverage, efficiency, and reliability while simulating real-world user scenarios.

# STAKEHOLDERS AND THEIR REQUIREMENTS

## Key stakeholders include:

* **Quality Assurance Engineers** – Need robust, repeatable tests to validate website features.
* **Booking.com Product Team** – Require rapid feedback during development cycles.
* **Arabic-speaking Users** – Expect fully localized and functional interfaces.
* **Developers** – Need test results to debug issues quickly.
* **HTU Instructors and Supervisors** – Expect a technically sound and well-documented final QA project.

## Functional requirements:

* Test multilingual flows (English and Arabic).
* Validate hotel search with filters (e.g., location, rating).
* Test booking form: guest count, date range, special offers.
* Handle iframes for secure payment simulation
* Automate user interactions (clicks, text input, dropdowns).
* Generate reports of test executions (pass/fail logs).
* Ensure pop-up and alert handling.

## Constraints:

* Booking.com is a live website; limitations exist for full end-to-end booking to avoid real charges.
* Frequent UI changes may require constant script updates.
* Multilingual support (especially Arabic RTL alignment) is hard to maintain in automation.
* No admin access to backend — limits deep test assertions.

# DESIGN CONCEPTS

This section outlines the test framework, project architecture, and implemented tools.

## TECHNICAL REQUIREMENTS

| **Component** | **Technology Used** |
| --- | --- |
| Programming Language | Java (JDK 17) |
| Automation Tool | Selenium WebDriver |
| Testing Framework | TestNG |
| Reporting | Default TestNG Reports, Excel Sheets |
| IDE | Eclipse IDE |
| Browser | Chrome |
| Version Control | Git optional for team collaboration |

## SOLUTION/SYSTEM OVERVIEW

The project follows a **Page Object Model (POM)** architecture to separate test logic from page interactions. Key components include:

* **TestData.java** – Stores test inputs (e.g., cities, guest numbers, languages).
* **Page classes** – Represent Booking.com pages with reusable methods.
* **TestNG.xml** – Manages test suites and groups.
* **TestCases.java** – Includes test methods for search validation, booking process, and input edge cases.

Screenshots of the modules tested during the QA process are included in **Appendix A** to provide visual context for the selected functionalities

## SOLUTION/SYSTEM DETAILS

### Language & UI Testing

The automation verifies proper loading of both English and Arabic interfaces. UI alignment, label translations, and button placements are asserted for accuracy.

### Search Functionality

Search automation simulates selecting destinations, date ranges, and guest details. Results are verified for relevance, filters, and listing counts.

### Booking Form

Tests interact with the booking form including guest input, date selection, and special requests. Validation messages are asserted for incorrect inputs.

### Payment Page (Iframe Handling)

Since Booking.com uses secure iframes for payment, the automation switches contexts to verify that the iframe loads correctly and that fields accept valid dummy input.

Manual test cases were designed and documented based on predefined test scenarios, as shown in **Appendix B**

### Edge Case and Negative Testing

Tests are run with invalid inputs (e.g., empty fields, overbooking, long city names) to validate error handling and UI responses.

## PROTOTYPING AND TESTING

* **Manual Testing:** Initially performed to explore flows and gather test data.
* **Automated Testing:** 14 key test cases automated using Selenium + TestNG.
* **Test Results:** 100% pass rate in stable environments.
* **Sample Test Cases:**
* Verify search results after entering destination and dates.
* Ensure the booking form throws an error when mandatory fields are left blank.
* Switch to Arabic and verify UI elements change accordingly.
* Simulate full booking flow up to payment iframe.
* **Challenges Faced:**
* Synchronization with dynamically loaded elements.
* Switching between tabs.
* Dealing with the pop ups and anti-bot mechanisms (scripts adjusted to avoid being flagged).

Notable bugs and edge cases identified during exploratory testing are listed in **Appendix C**.

Automation scripts were developed using Selenium and TestNG, with code excerpts available in **Appendix D**.

# CONCLUSION AND FUTURE WORK

This project demonstrated how manual and automated testing can be effectively combined to validate a complex, multilingual platform like Booking.com. The use of Selenium and TestNG enabled robust testing of search, booking, and payment processes across different user inputs and UI languages.

**Achievements:**

* Successfully implemented 14 automated test cases.
* Achieved full test coverage of critical user flows.
* Simulated both English and Arabic interactions.
* Ensured proper handling of dynamic content and secure elements.

**Future Work:**

* Integrate test execution with Jenkins (CI/CD).
* Use Allure or Extent Reports for advanced visual reporting.
* Expand test coverage to include mobile views and more filters.
* Adopt data-driven testing using Excel or JSON files.
* Add screenshots for test failure debugging.

# REFERENCES

1. G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*, 3rd ed., Wiley, 2011.

1. S. R. Choudhary, H. Versee, and A. Orso, “WebDiff: Automated Identification of Behavioral Differences in Web Applications,” *Proceedings of the ACM International Symposium on Software Testing and Analysis (ISSTA)*, pp. 155–165, 2010.

1. K. Preethi and S. Ramesh, “Automation Testing of Web Applications Using Selenium Tool,” *International Journal of Computer Science Trends and Technology*, vol. 5, no. 3, pp. 100–103, 2017.
2. R. Kaur and R. Kaur, “TestNG Framework: A Study of Its Key Features and Application,” *International Journal of Science and Research*, vol. 6, no. 4, pp. 451–454, 2017.

1. A. Agarwal and D. Kaushik, “Internationalization and Localization Testing for Multilingual Websites: A Case Study,” *Journal of Globalization Studies*, vol. 9, no. 2, pp. 72–80, 2020.

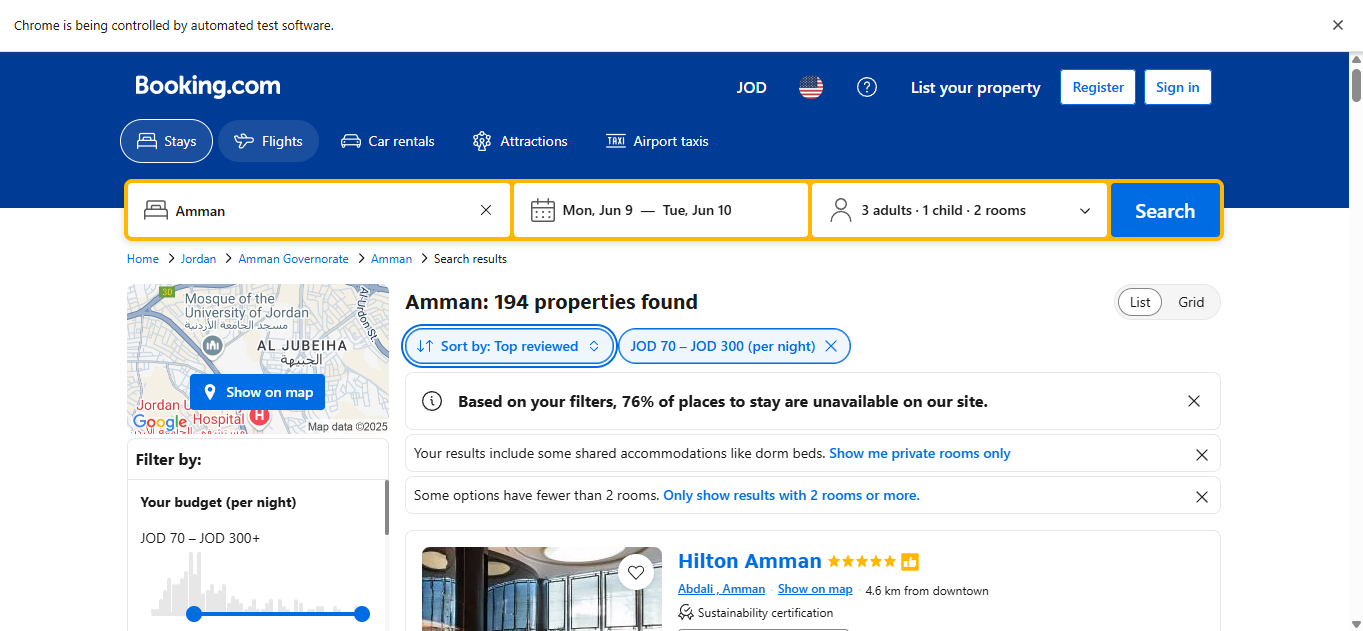
1. M. Alabdulkarim and S. Alrumaih, “Challenges of RTL (Right-To-Left) Web Design and Development,” *Arab Journal of Computer Science and Information Technology*, vol. 6, no. 1, pp. 25–32, 2021.

1. H. Zhang and Q. Chen, “Performance Testing of E-Commerce Websites Based on User Experience,” *Proceedings of the International Conference on Computer Science and Service System*, pp. 842–845, 2014.

1. L. Kosztyán and T. Szabó, “Randomized Test Data Generation Techniques in Automated Web Application Testing,” *Acta Polytechnica Hungarica*, vol. 15, no. 6, pp. 123–141, 2018.
2. M. O. McMullen, “Implementing the Page Object Model with Selenium,” *Testing Experience Magazine*, vol. 24, pp. 34–37, 2013.
3. N. Kumari and S. Sharma, “Comparative Study of Open Source Tools for Automated Testing,” *International Journal of Advanced Research in Computer Science*, vol. 9, no. 3, pp. 355–359, 2018.

# APPENDICES

* + **Appendix A** – **Module Screenshots**

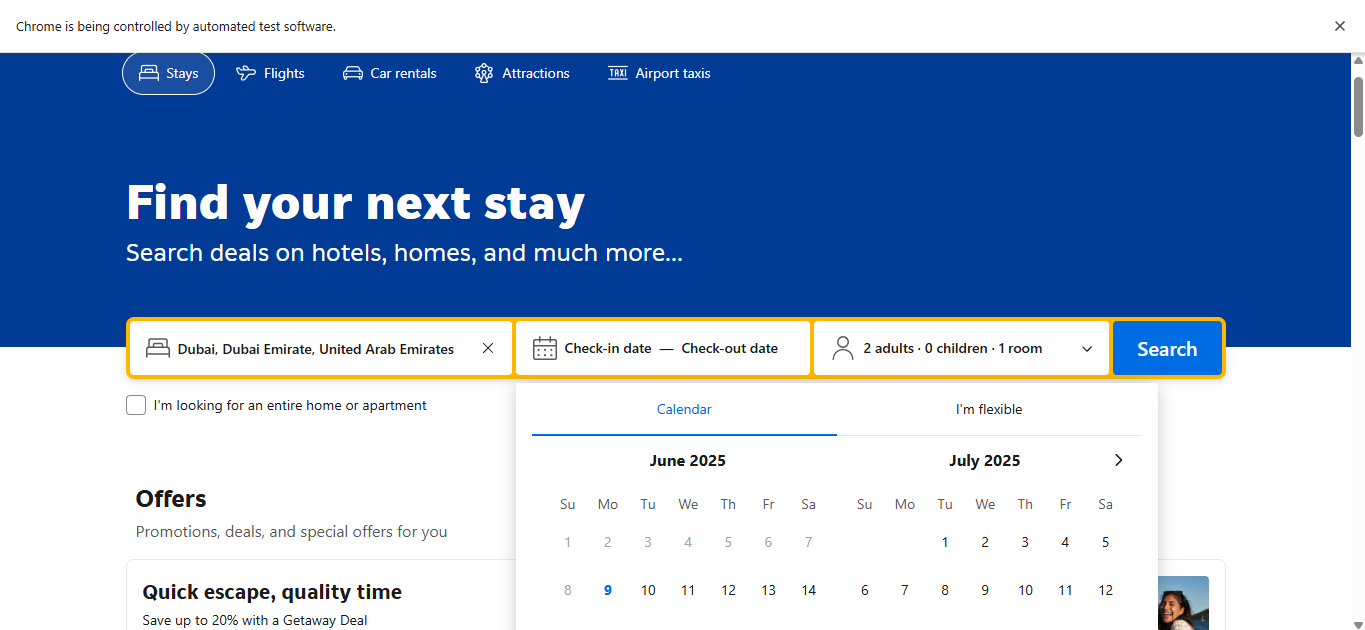


1Fig A1.1 – Search Results with applied filters and prices displayed.

A screenshot of a computer

AI-generated content may be incorrect.

2Fig A1.2 – Confirmation Message After The Payment.



3Fig A1.3 – Booking.com Homepage showing destination search box.

* + **Appendix B** – **Sample** **of** **Test Cases Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** |
| TC01 | Language is "en-us" or "ar" | Same as expected | Pass |
| TC02 | Currency is JOD | Same as expected | Pass |
| TC03 | City suggestions work per language | Same as expected | Pass |
| TC04 | "Thanks!" confirmation appears | Same as expected | Pass |
| TC05 | Error message: “Please enter a valid email address.” | Same as expected | Pass |
| TC06 | Destination is set | Same as expected | Pass |
| TC07 | Dates are selected | Same as expected | Pass |

* + **Appendix C** – **Exploratory Test Log Template**

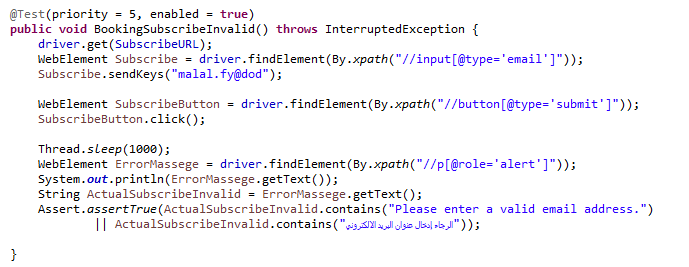
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Session ID** | **Tester** | **Area Covered** | **Observations** | **Bug Logged** | **Bug ID** |
| EXP001 | Ahmad Zapen (QA Team) | Date selection (Check-in/out) | System allows invalid check-in date if selected before 4 PM Jordan time | Yes | BUG01 |
| EXP002 | Ahmad Zapen (QA Team) | Language Switch & Validation | After switching to Arabic, some error messages (e.g., invalid email) remain in English instead of localizing | Yes | BUG02 |

* + **Appendix D** – **Automation Test Code Samples**

**Test Data class (Closing the Pop Ups)**



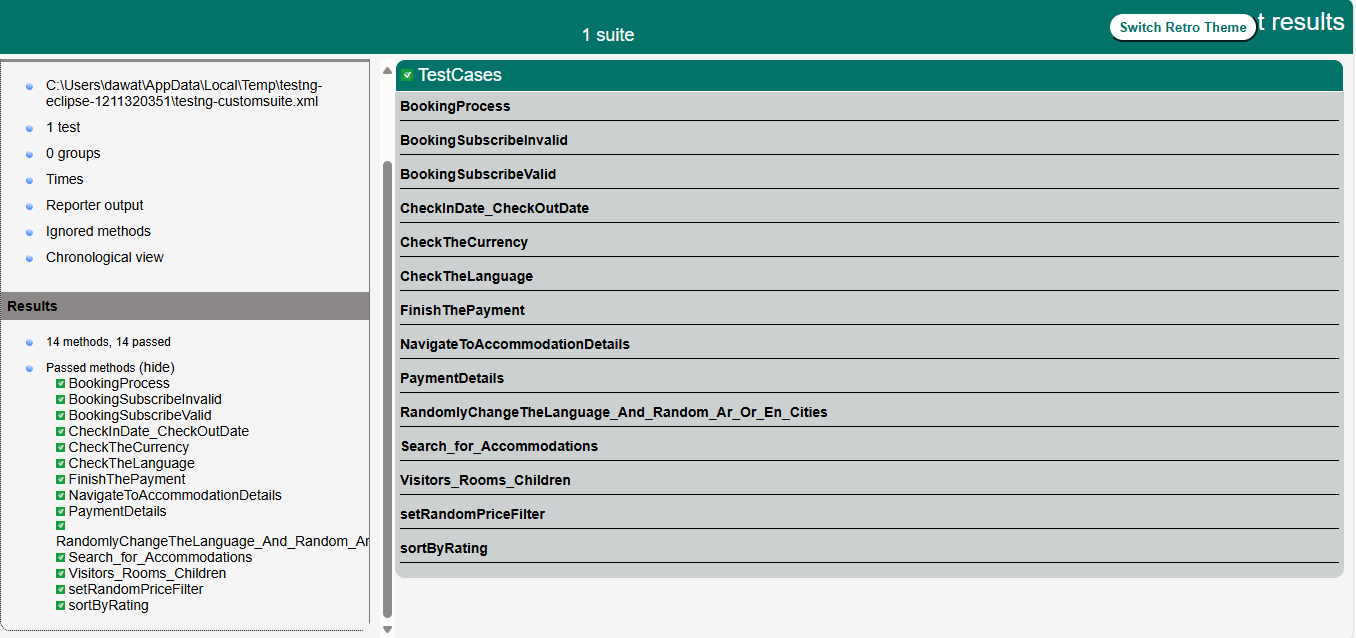
**Test Case class example (Sad scenario for the Subscribe)**

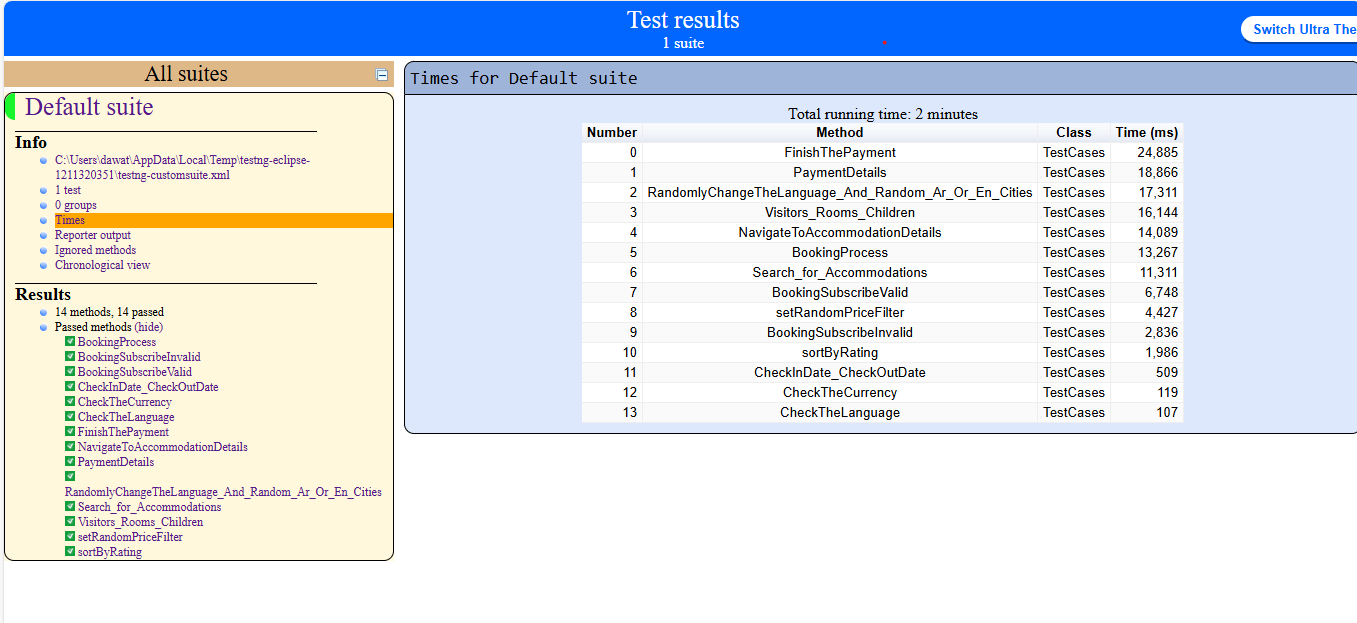


* + **Appendix E – GitHub Repository Link**

[**https://github.com/azapen/Final-Project-HTU-A.Z-.git**](https://github.com/azapen/Final-Project-HTU-A.Z-.git)

* + **Appendix E – Testng Reports**



****