CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

ONLINE MOVIE TICKET BOOKING SYSTEM TESTING REPORT

Submitted By

Rahul Saxena
UID: 24MCI10204
24MAM 3-B

Submitted To

Mrs. Disha Handa
EID: E11162

Abstract

This project focuses on the testing and quality assurance of a web-based Online Ticket Booking System, designed to provide users with a seamless experience for browsing movies, booking tickets, and making payments online. The system is divided into two primary modules: the User Module, which allows end-users to interact with the platform, and the Admin Module, which enables administrators to manage movies, users, and view system analytics.

To ensure the reliability and correctness of the application, both manual and automated testing approaches were applied. Manual testing helped in validating functional requirements, identifying usability issues, and simulating real-user scenarios, while automated testing using Selenium WebDriver allowed efficient execution of repetitive test cases, improving test accuracy and saving time.

Aim

The aim of this project is to thoroughly test the functionality, usability, and reliability of an Online Ticket Booking System through both manual and automated testing methods. The testing process focuses on identifying potential bugs, ensuring that all features behave as expected under various scenarios, and validating that the system can handle both user and admin activities effectively.

By applying manual testing, we aim to simulate real-world user interactions and explore edge cases, while automated testing with Selenium will help verify repetitive and critical functionalities with speed and precision. This dual testing approach ensures that the application is not only functionally accurate but also robust, efficient, and ready for deployment in a real-world environment.

The ultimate goal is to deliver a high-quality, user-friendly, and secure web application that allows users to easily book tickets and enables admins to manage movie content, user profiles, and sales analytics smoothly.

Objective

- To verify the functional correctness of all features in the Online Ticket Booking System, ensuring that each component behaves as expected under normal and edge-case scenarios.
- To test both the User and Admin modules through a series of comprehensive and well-structured test cases that cover all major user journeys, including login, booking, payment, movie management, and analytics.
- To implement a hybrid testing approach by combining manual testing (for exploratory, usability, and error-handling scenarios) with automated testing using Selenium (for repetitive, time-consuming, and regression tasks) to achieve thorough test coverage and improved testing efficiency.
- To identify and document any bugs, inconsistencies, or performance issues encountered during testing and ensure they are resolved before the final deployment of the system.
- To ensure the system meets user expectations, provides a smooth and secure experience, and is ready for deployment in a real-world production environment.
- To evaluate the system’s scalability, maintainability, and responsiveness, especially under various loads, to guarantee long-term reliability.

Technology Used

Frontend:
The front-end of the application is developed using **HTML**, **CSS**, and **JavaScript**.

- **HTML (HyperText Markup Language)** is used to structure the content of the web pages.
- **CSS (Cascading Style Sheets)** is applied for styling elements, ensuring the application is visually appealing and responsive across various devices.
- **JavaScript** is utilized to implement client-side logic, enhance interactivity, and provide dynamic behavior such as real-time content updates, form validation, and interactive UI elements.

Backend:
The backend is responsible for handling business logic, server-side processing, and communication with the database. (Specify your choice here, e.g., **Node.js**, **PHP**, **Python with Flask/Django**, or **Java with Spring Boot**).

- It manages user authentication, processes ticket bookings, stores transaction data, and provides admin controls like movie management and analytics.

Database:
The application uses a database management system like **MySQL** or **MongoDB** to store and retrieve persistent data.

- **MySQL**, a relational database, is ideal for structured data with defined schemas (tables for users, movies, bookings, etc.).
- Alternatively, **MongoDB**, a NoSQL database, can be used for more flexible, document-oriented data storage.

Testing Tool:
Selenium WebDriver is used for performing **automated testing** of the web application.

- It allows the simulation of user actions such as clicking buttons, filling out forms, and navigating through pages to ensure features work as expected.
- Automated scripts are written in a language like Java, Python, or JavaScript, and executed to repeatedly validate core functionalities with speed and accuracy.

Browser:
Testing is primarily conducted using modern browsers like **Google Chrome** or **Mozilla Firefox**.

- These browsers ensure compatibility, support for the latest web standards, and provide developer tools for debugging and testing purposes.

IDE (Integrated Development Environment):
The development and testing work is carried out in an IDE such as **Visual Studio Code**, **Eclipse**, or **IntelliJ IDEA**.

- These tools offer features like code highlighting, version control integration, extension support, and debugging capabilities that enhance the productivity of developers and testers.

Website Tested: "http://localhost/OnlineBooking/"

Test Cases

- **Test Case ID: TC001**

  Module Name: User Login Functionality

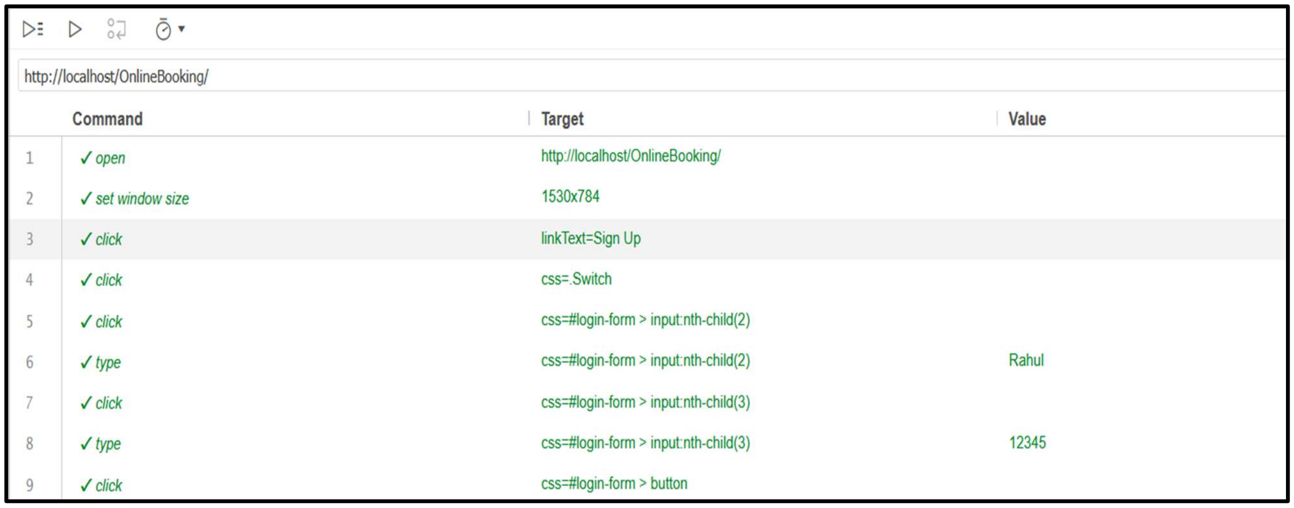
  Pre-requisites: User account must already exist in the system.

  Objective: To verify that user can log in with valid credentials.

  Test Case: Enter valid email and password, click login.

Expected Output: User is redirected to the homepage/dashboard.

Test Result



http://localhost/OnlineBooking/

	Command	Target	Value
1	✓ open	http://localhost/OnlineBooking/	
2	✓ set window size	1530x784	
3	✓ click	linkText=Sign Up	
4	✓ click	css=.Switch	
5	✓ click	css=#login-form > input:nth-child(2)	
6	✓ type	css=#login-form > input:nth-child(2)	Rahul
7	✓ click	css=#login-form > input:nth-child(3)	
8	✓ type	css=#login-form > input:nth-child(3)	12345
9	✓ click	css=#login-form > button	

- **Test Case ID:** TC002

  Module Name: Invalid Login Attempt

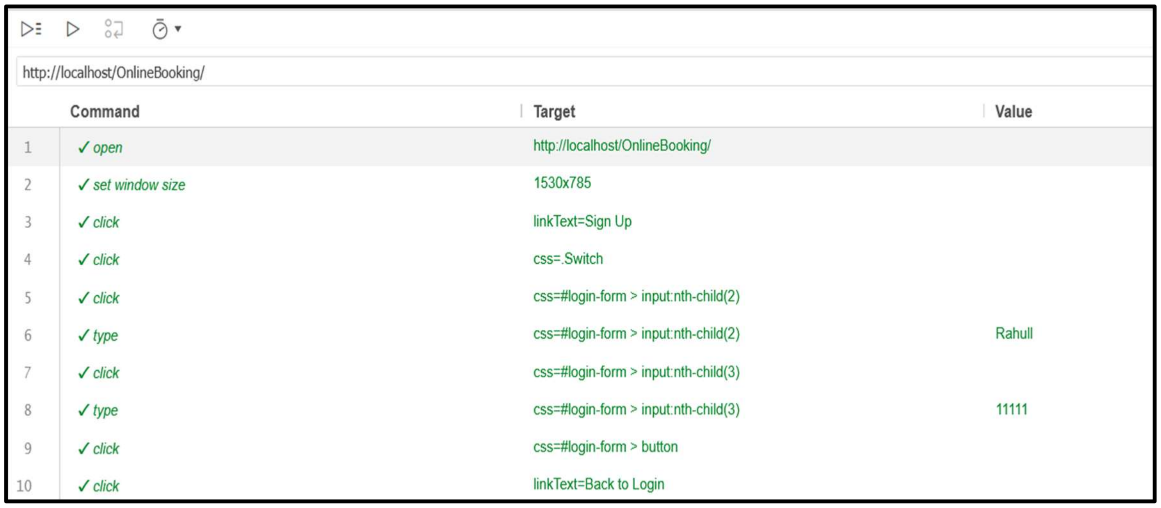
  Pre-requisites: Use incorrect password for existing user.

  Objective: Ensure login fails with wrong credentials.

  Test Case: Enter valid email and wrong password, click login.

  Expected Output: Display error message "Invalid username or password."

  Test Result



http://localhost/OnlineBooking/

	Command	Target	Value
1	✓ open	http://localhost/OnlineBooking/	
2	✓ set window size	1530x785	
3	✓ click	linkText=Sign Up	
4	✓ click	css=.Switch	
5	✓ click	css=#login-form > input:nth-child(2)	
6	✓ type	css=#login-form > input:nth-child(2)	Rahull
7	✓ click	css=#login-form > input:nth-child(3)	
8	✓ type	css=#login-form > input:nth-child(3)	11111
9	✓ click	css=#login-form > button	
10	✓ click	linkText=Back to Login	

- **Test Case ID:** TC003

  Module Name: View Available Movies.

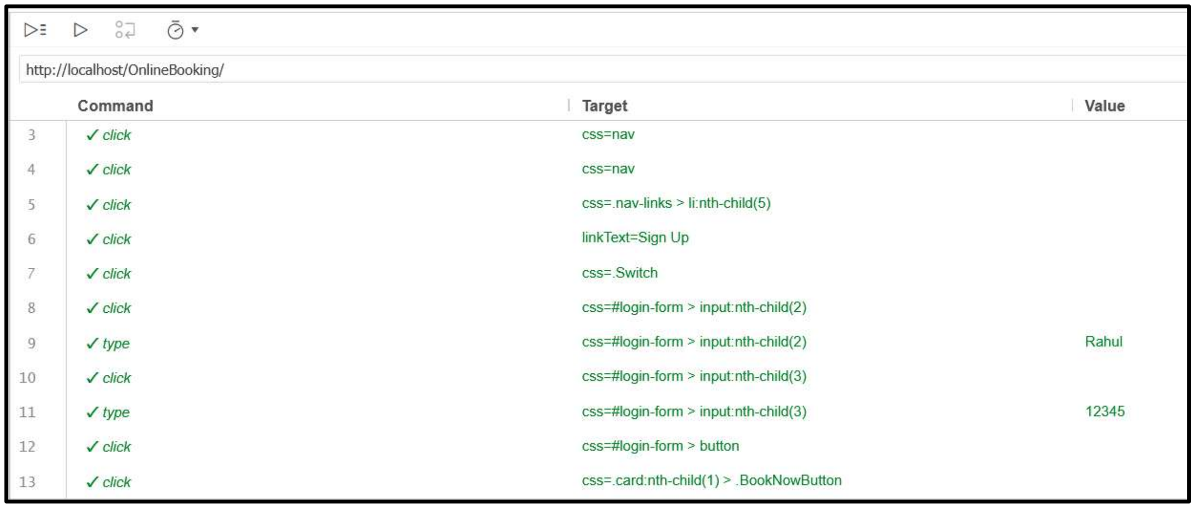
  Pre-requisites: User must be logged in.

  Objective: To check if all available movies are listed for the user.

  Test Case: Navigate to the movies page.

  Expected Output: List of movies with title, genre, and show timings displayed.

Test Result



http://localhost/OnlineBooking/

	Command	Target	Value
3	✓ click	css=nav	
4	✓ click	css=nav	
5	✓ click	css=.nav-links > li:nth-child(5)	
6	✓ click	linkText=Sign Up	
7	✓ click	css=.Switch	
8	✓ click	css=#login-form > input:nth-child(2)	
9	✓ type	css=#login-form > input:nth-child(2)	Rahul
10	✓ click	css=#login-form > input:nth-child(3)	
11	✓ type	css=#login-form > input:nth-child(3)	12345
12	✓ click	css=#login-form > button	
13	✓ click	css=.card:nth-child(1) > .BookNowButton	

➢ **Test Case ID:** TC004

Module Name: Book Ticket and Make Payment.

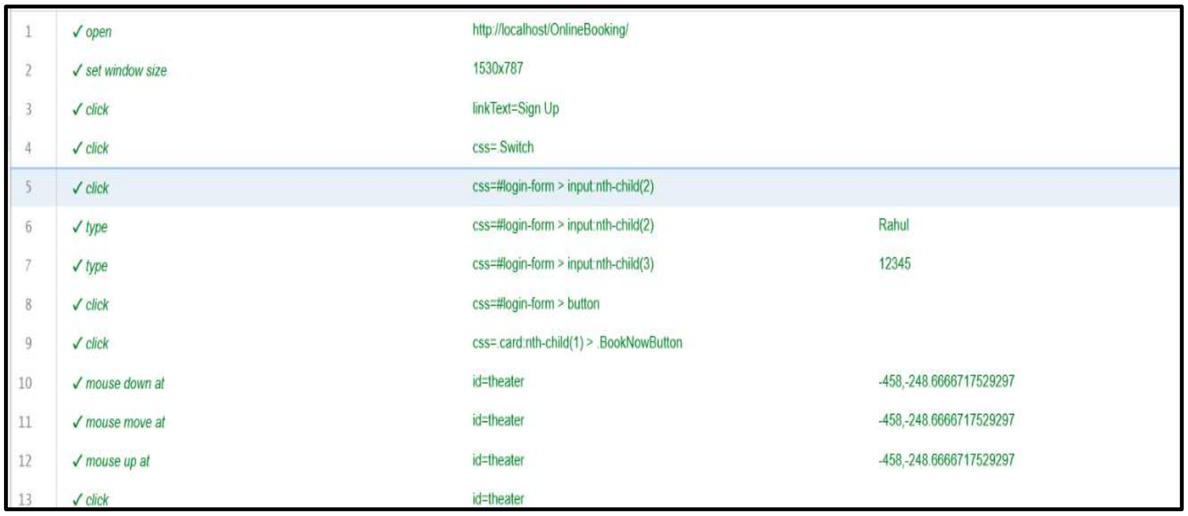
Pre-requisites: User must be logged in; movies must be available.

Objective: To test the booking and payment process.

Test Case: Select movie, choose seat, proceed to payment.

Expected Output: Payment successful and ticket added to user's wallet.

Test Result



1	✓ open	http://localhost/OnlineBooking/	
2	✓ set window size	1530x787	
3	✓ click	linkText=Sign Up	
4	✓ click	css=.Switch	
5	✓ click	css=#login-form > input:nth-child(2)	
6	✓ type	css=#login-form > input:nth-child(2)	Rahul
7	✓ type	css=#login-form > input:nth-child(3)	12345
8	✓ click	css=#login-form > button	
9	✓ click	css=.card:nth-child(1) > .BookNowButton	
10	✓ mouse down at	id=theater	-458,-248.6666717529297
11	✓ mouse move at	id=theater	-458,-248.6666717529297
12	✓ mouse up at	id=theater	-458,-248.6666717529297
13	✓ click	id=theater	

➢ **Test Case ID:** TC005

Module Name: View Booked Ticket in “Recent Booking Tab”.

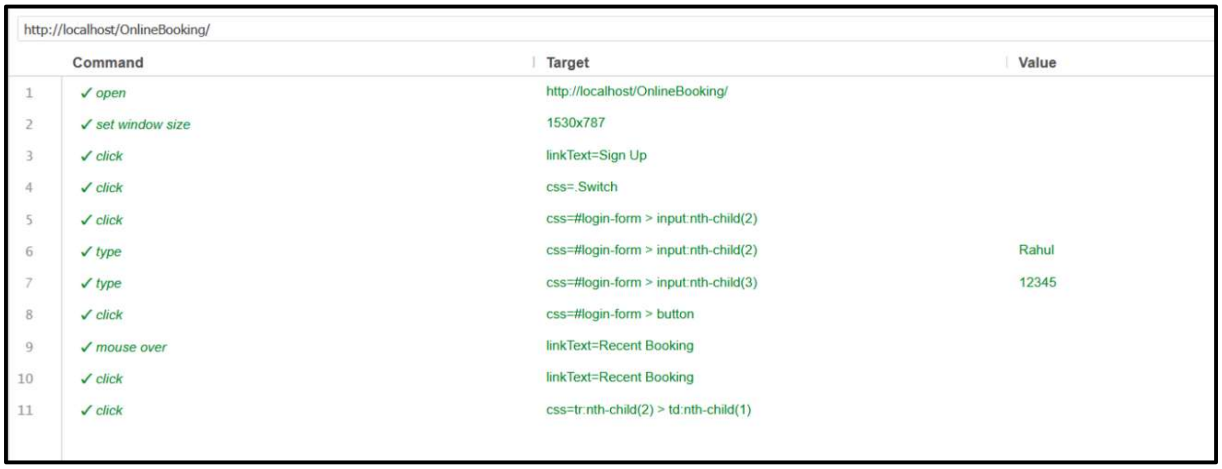
Pre-requisites: Ensure booked tickets are displayed in the tab.

Objective: Navigate to wallet after booking.

Test Case: Enter valid email and wrong password, click login.

Expected Output: Booked ticket details appear in Recent Booking Tab.

Test Result:



http://localhost/OnlineBooking/

	Command	Target	Value
1	✓ open	http://localhost/OnlineBooking/	
2	✓ set window size	1530x787	
3	✓ click	linkText=Sign Up	
4	✓ click	css=.Switch	
5	✓ click	css=#login-form > input:nth-child(2)	
6	✓ type	css=#login-form > input:nth-child(2)	Rahul
7	✓ type	css=#login-form > input:nth-child(3)	12345
8	✓ click	css=#login-form > button	
9	✓ mouse over	linkText=Recent Booking	
10	✓ click	linkText=Recent Booking	
11	✓ click	css=tr:nth-child(2) > td:nth-child(1)	

➢ **Test Case ID:** TC006

Module Name: Admin Login

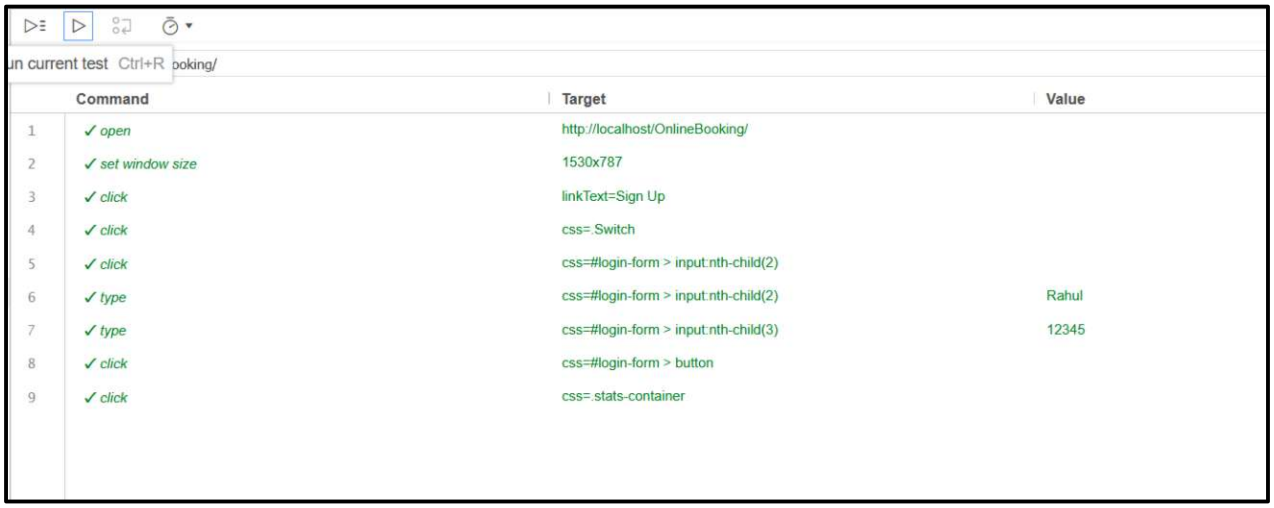
Pre-requisites: Admin account credentials must exist.

Objective: Validate admin login functionality.

Test Case: Enter admin username and password, click login.

Expected Output: Redirected to admin dashboard.

Test Result:



	Command	Target	Value
1	✓ open	http://localhost/OnlineBooking/	
2	✓ set window size	1530x787	
3	✓ click	linkText=Sign Up	
4	✓ click	css=.Switch	
5	✓ click	css=#login-form > input:nth-child(2)	
6	✓ type	css=#login-form > input:nth-child(2)	Rahul
7	✓ type	css=#login-form > input:nth-child(3)	12345
8	✓ click	css=#login-form > button	
9	✓ click	css=.stats-container	

➢ **Test Case ID:** TC007

Module Name: Admin - Add New Movie

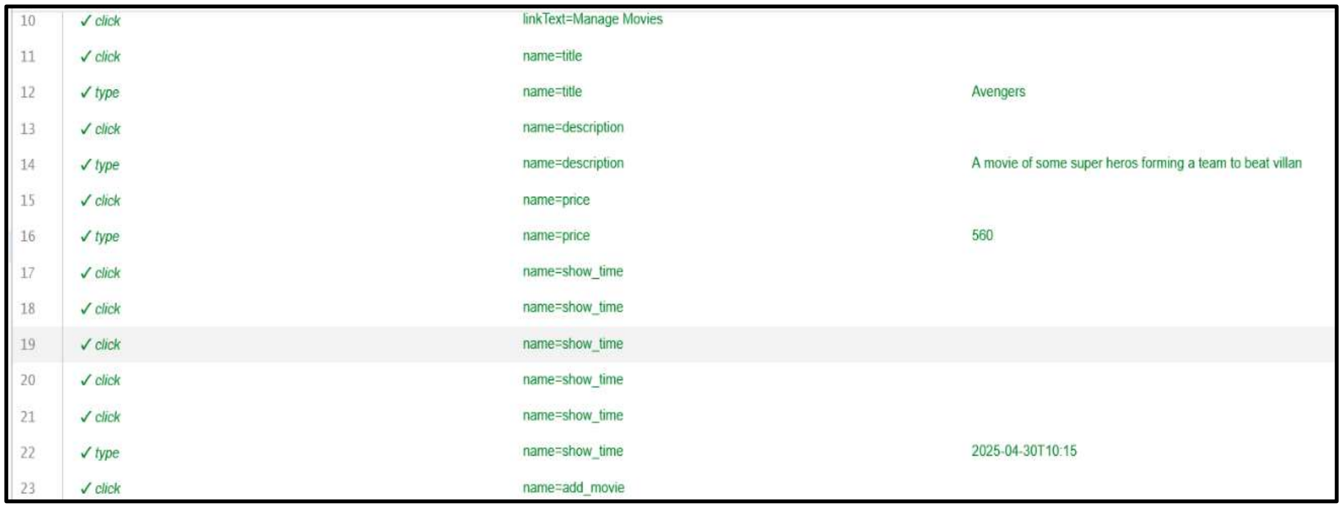
Pre-requisites: Admin must be logged in.

Objective: Ensure admin can add a new movie.

Test Case: Fill in movie details and click "Add Movie".

Expected Output: Movie successfully added and reflected in user movie list.

Test Result



10	✓ click	linkText=Manage Movies	
11	✓ click	name=title	
12	✓ type	name=title	Avengers
13	✓ click	name=description	
14	✓ type	name=description	A movie of some super heros forming a team to beat villan
15	✓ click	name=price	
16	✓ type	name=price	560
17	✓ click	name=show_time	
18	✓ click	name=show_time	
19	✓ click	name=show_time	
20	✓ click	name=show_time	
21	✓ click	name=show_time	
22	✓ type	name=show_time	2025-04-30T10:15
23	✓ click	name=add_movie	

➢ **Test Case ID:** TC008

Module Name: Admin - View User Profiles

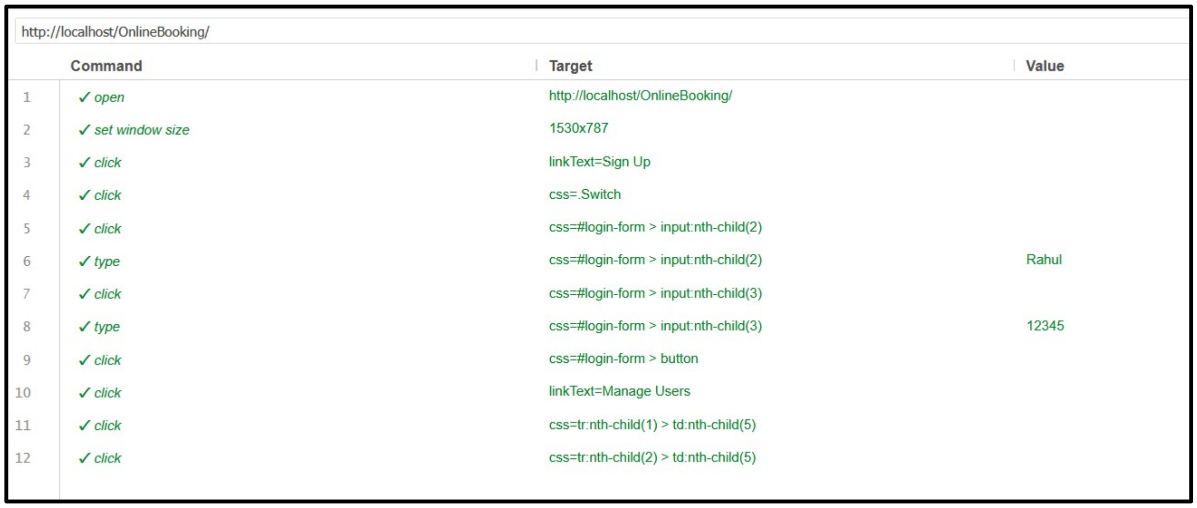
Pre-requisites: Admin must be logged in; users exist.

Objective: Verify that admin can view details of any user.

Test Case: Navigate to users section and click on a user profile.

Expected Output: Full profile details of selected user are shown.

Test Result:



http://localhost/OnlineBooking/

	Command	Target	Value
1	✓ open	http://localhost/OnlineBooking/	
2	✓ set window size	1530x787	
3	✓ click	linkText=Sign Up	
4	✓ click	css=.Switch	
5	✓ click	css=#login-form > input:nth-child(2)	
6	✓ type	css=#login-form > input:nth-child(2)	Rahul
7	✓ click	css=#login-form > input:nth-child(3)	
8	✓ type	css=#login-form > input:nth-child(3)	12345
9	✓ click	css=#login-form > button	
10	✓ click	linkText=Manage Users	
11	✓ click	css=tr:nth-child(1) > td:nth-child(5)	
12	✓ click	css=tr:nth-child(2) > td:nth-child(5)	

➢ **Test Case ID:** TC009

Module Name: Admin - Delete Movie

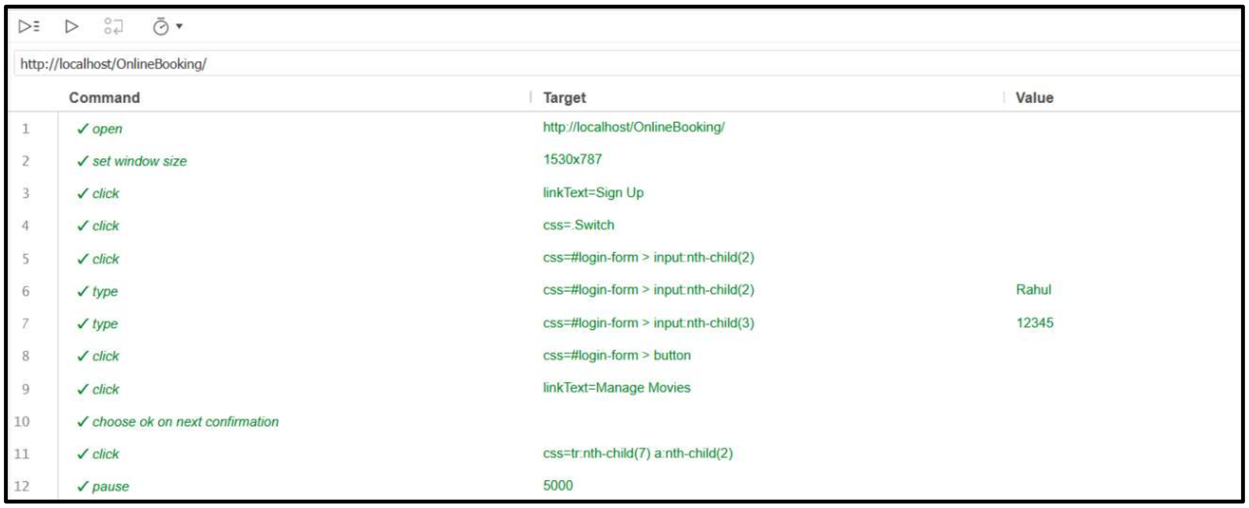
Pre-requisites: Admin must be logged in; movies exist in the database.

Objective: Test if admin can delete movies.

Test Case: Select a movie and click "Delete".

Expected Output: Movie removed from database and no longer visible to users.

Test Result



http://localhost/OnlineBooking/

	Command	Target	Value
1	✓ open	http://localhost/OnlineBooking/	
2	✓ set window size	1530x787	
3	✓ click	linkText=Sign Up	
4	✓ click	css=.Switch	
5	✓ click	css=#login-form > input:nth-child(2)	
6	✓ type	css=#login-form > input:nth-child(2)	Rahul
7	✓ type	css=#login-form > input:nth-child(3)	12345
8	✓ click	css=#login-form > button	
9	✓ click	linkText=Manage Movies	
10	✓ choose ok on next confirmation		
11	✓ click	css=tr:nth-child(7) a:nth-child(2)	
12	✓ pause	5000	

➢ **Test Case ID:** TC010

Module Name: View Sales Analytics

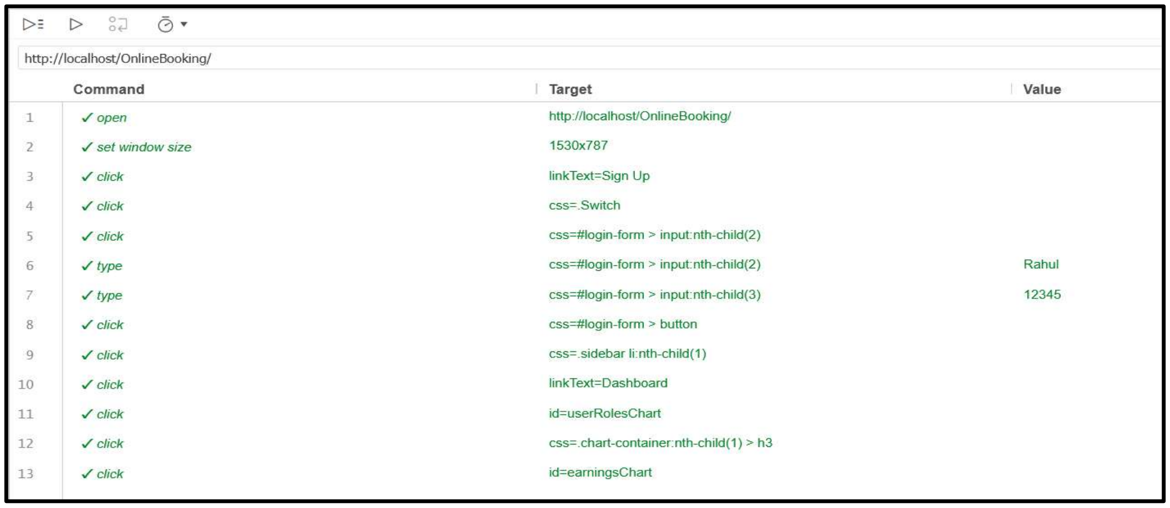
Pre-requisites: Admin must be logged in; tickets must be booked by users.

Objective: To ensure sales and analytics are correctly displayed.

Test Case: Navigate to analytics section.

Expected Output: Dashboard shows total sales, total users, and other analytics.

Test Result



http://localhost/OnlineBooking/

	Command	Target	Value
1	✓ open	http://localhost/OnlineBooking/	
2	✓ set window size	1530x787	
3	✓ click	linkText=Sign Up	
4	✓ click	css=.Switch	
5	✓ click	css=#login-form > input:nth-child(2)	
6	✓ type	css=#login-form > input:nth-child(2)	Rahul
7	✓ type	css=#login-form > input:nth-child(3)	12345
8	✓ click	css=#login-form > button	
9	✓ click	css=.sidebar li:nth-child(1)	
10	✓ click	linkText=Dashboard	
11	✓ click	id=userRolesChart	
12	✓ click	css=.chart-container:nth-child(1) > h3	
13	✓ click	id=earningsChart	

Conclusion

The Online Ticket Booking System project was successfully developed and tested using a combination of manual and automated testing techniques, offering a deep understanding of both practical testing methods and the functionality of a real-world web application.
Through manual testing, we were able to explore the system in detail, validating critical functionalities such as user login, movie browsing, ticket booking, payment processing, and admin operations. This allowed us to

simulate actual user behavior and identify issues that may not be immediately apparent through automation—such as UI inconsistencies, unexpected inputs, or edge-case scenarios.

On the other hand, the use of Selenium for automated testing provided significant advantages in terms of speed, accuracy, and efficiency. Repetitive tasks, such as logging in, checking movie lists, and verifying successful ticket booking, were automated to reduce human effort and ensure consistency across multiple test executions. Screenshots captured during these automated tests serve as evidence of test completion and help track test results visually.

Combining these two testing approaches allowed us to maximize test coverage and gain confidence in the reliability and robustness of the system. It also highlighted the importance of having a balanced testing strategy, where manual testing focuses on exploratory and UI-related scenarios, while automation handles routine, repetitive, and regression cases.