**Test Automation Report: PHPtravels Flight Booking Automation**

**1. Test Overview**

|  |  |
| --- | --- |
| Attribute | Details |
| Test Name | PHPtravels Flight Booking Automation |
| Automation Tool | Playwright (Python Async API) |
| Target Website | <https://phptravels.net/> |
| Scenario | Complete end-to-end flow of selecting a flight, filling booking details, payment via PayPal, downloading booking confirmation, and verification steps |
| Tester | Automated by Playwright |
| Final Status | Passed |

**2. Environment Configuration**

|  |  |
| --- | --- |
| Component | Configuration |
| Operating System | Windows / Linux / macOS |
| Browser Engine | Chromium (non-headless mode) |
| Language | Python 3.10+ |
| Playwright Version | Latest stable release |
| Execution Mode | Slow-mo enabled with visual playback (200ms) |

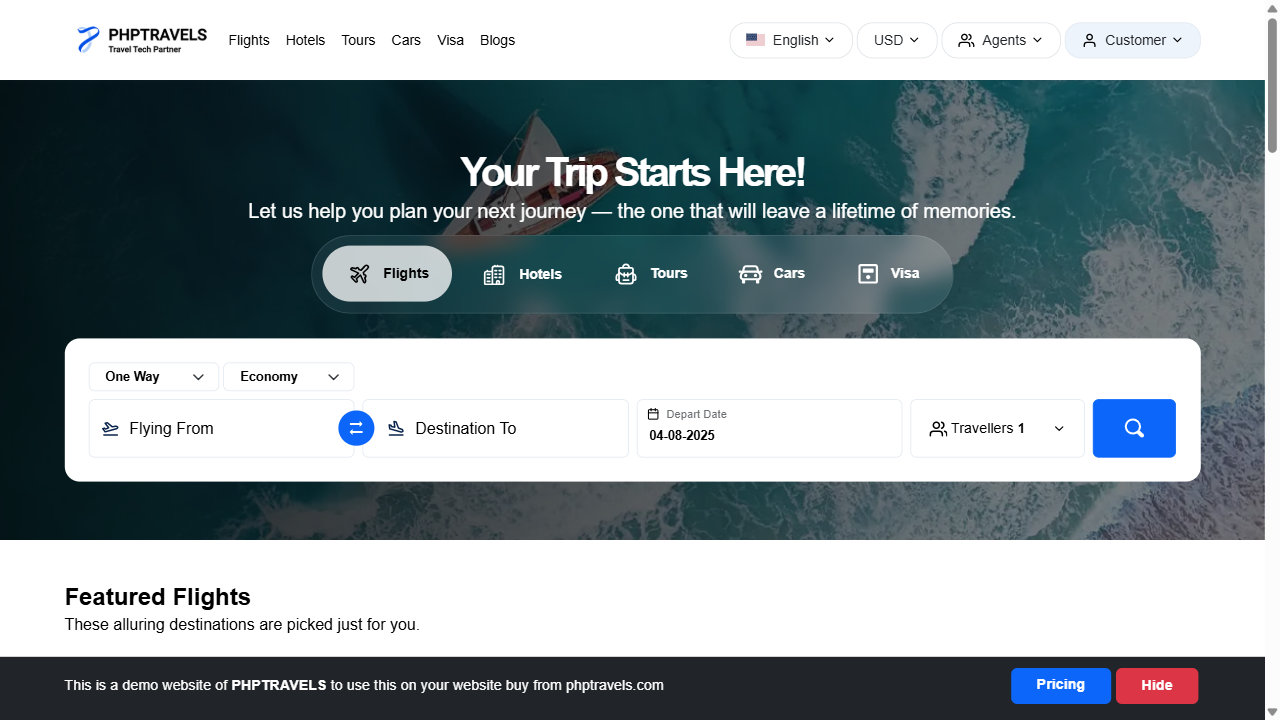
**3. Test Case Breakdown**

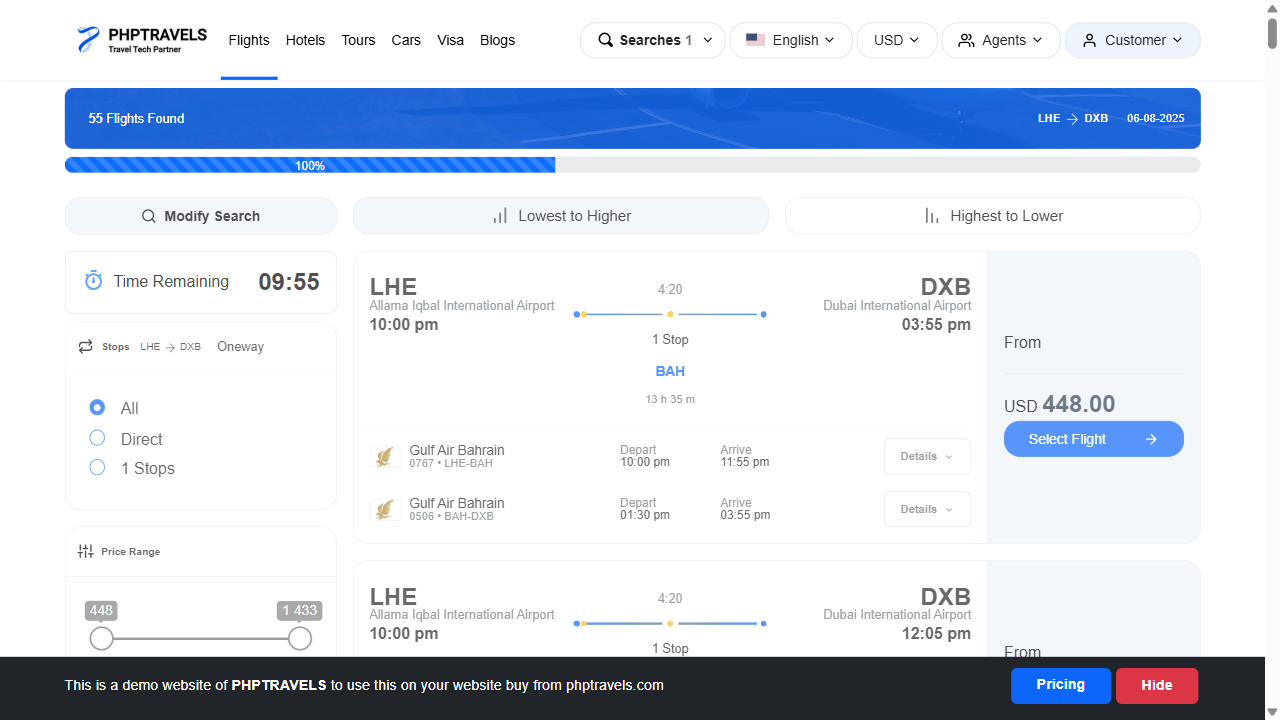
|  |  |  |  |
| --- | --- | --- | --- |
| Step No. | Action | Wait Time | Screenshot Captured |
| 1 | Visit PHPtravels Homepage | Instant | Yes |
| 2 | Click featured flight (LHE → DXB) | 3 seconds | Yes |
| 3 | Select a flight from the list | 3 seconds | Yes |
| 4 | Fill in personal details | 3 seconds/field | Yes |
| 5 | Fill in traveler details | 3 seconds/field | Yes |
| 6 | Tick agreement checkbox | 3 seconds | Yes |
| 7 | Click **Booking Confirm** | 3 seconds | Yes |
| 8 | Wait and then click **Proceed** to Payment Form | 6 seconds | Yes |
| 9 | Click PayPal Payment Option | 3 seconds | Yes |
| 10 | Enter PayPal Email and Password | 3 seconds each | Yes |
| 11 | Submit PayPal Login | 3 seconds | Yes |
| 12 | Click **Download as PDF** | 3 seconds | Yes |
| 13 | Scroll page slowly to visualize booking summary | 30 seconds | Yes |
| 14 | Close browser post wait | 5 seconds | No |

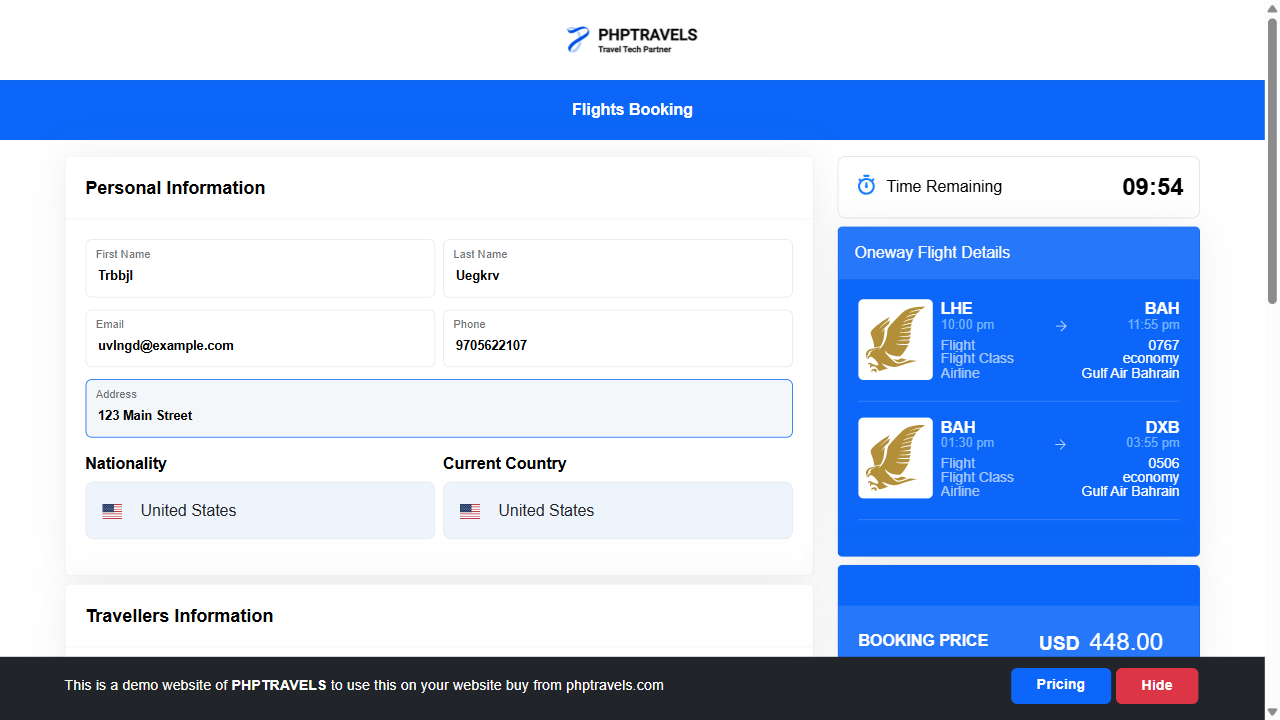
**4. Screenshots & Artifacts**

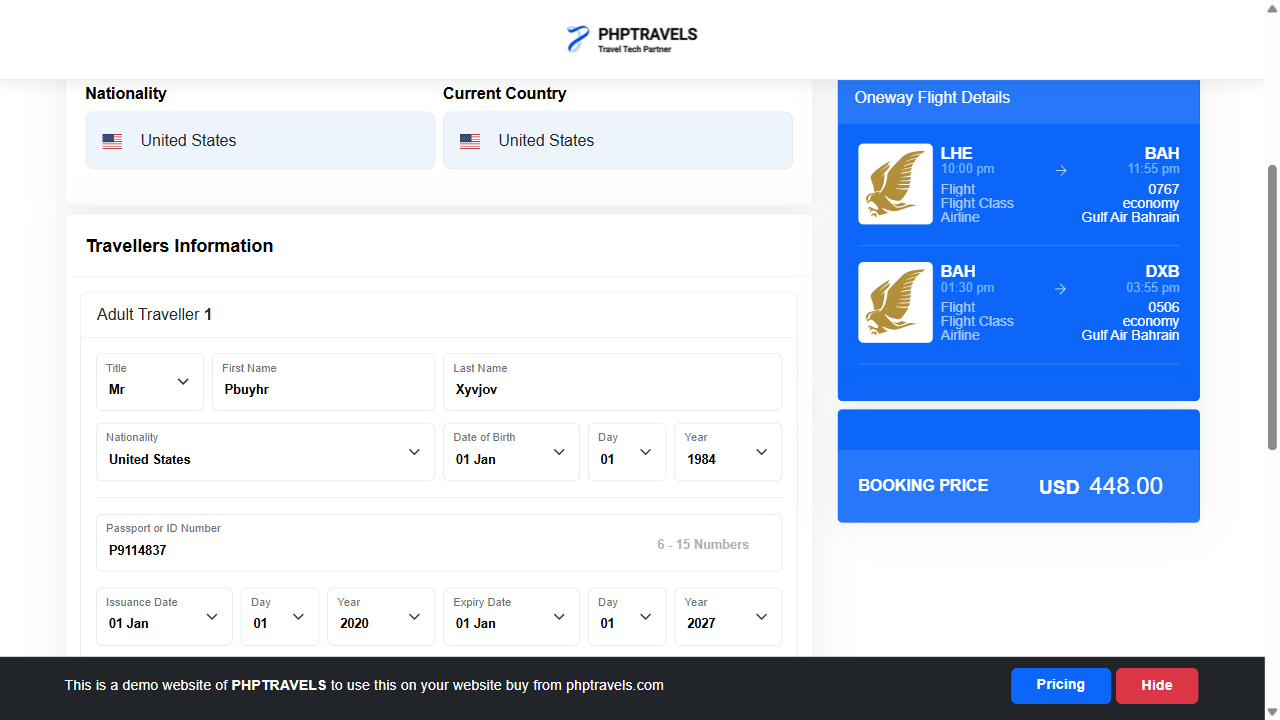
|  |  |
| --- | --- |
| File Name | Description |
| homepage\_visit.png | Screenshot of landing page |
| flight\_selected.png | After selecting the flight |
| form\_filled.png | Personal + Traveler details filled |
| paypal\_login.png | PayPal sandbox login screen |
| payment\_complete.png | Booking success + PDF download section |

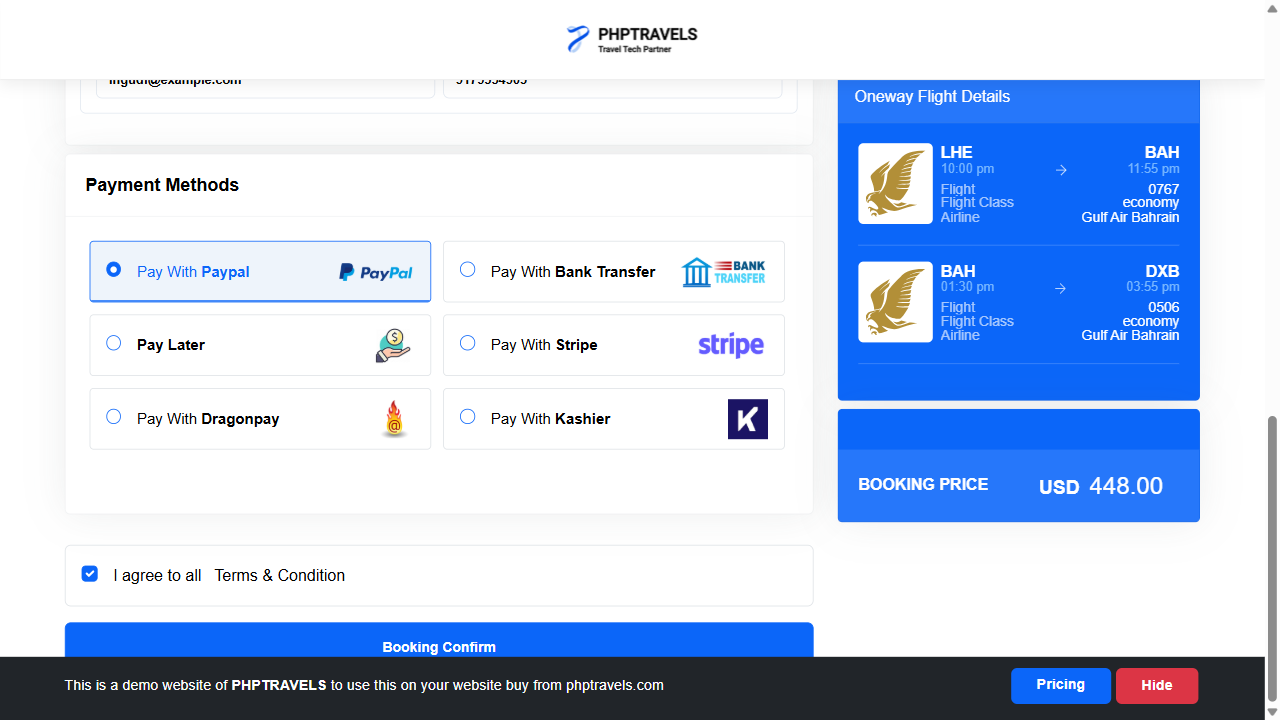
All screenshots were captured using page.screenshot(path=...) in Playwright.

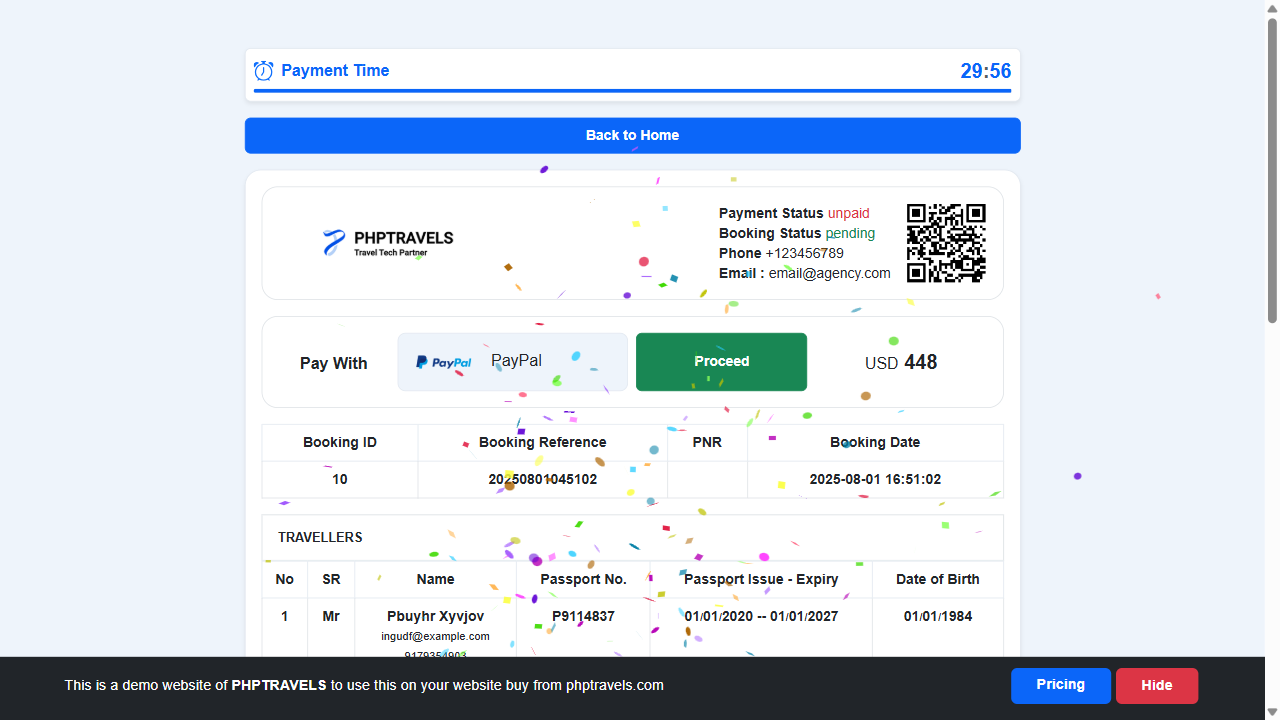


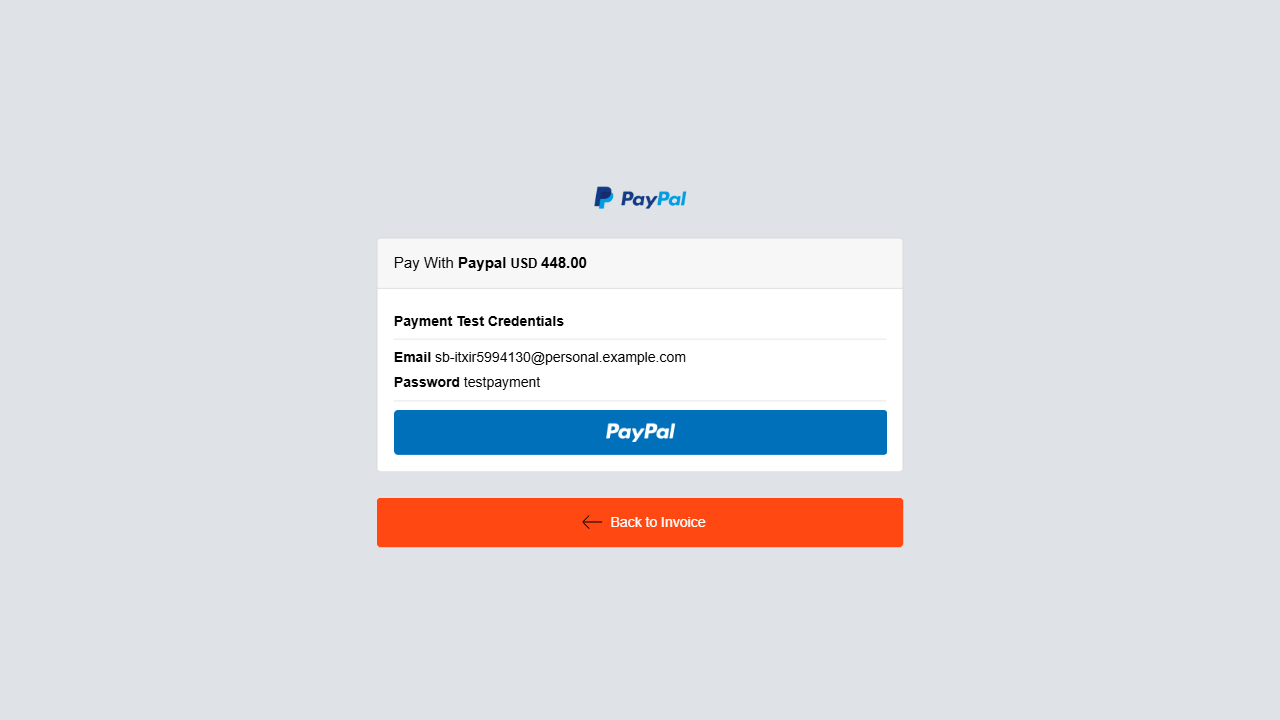


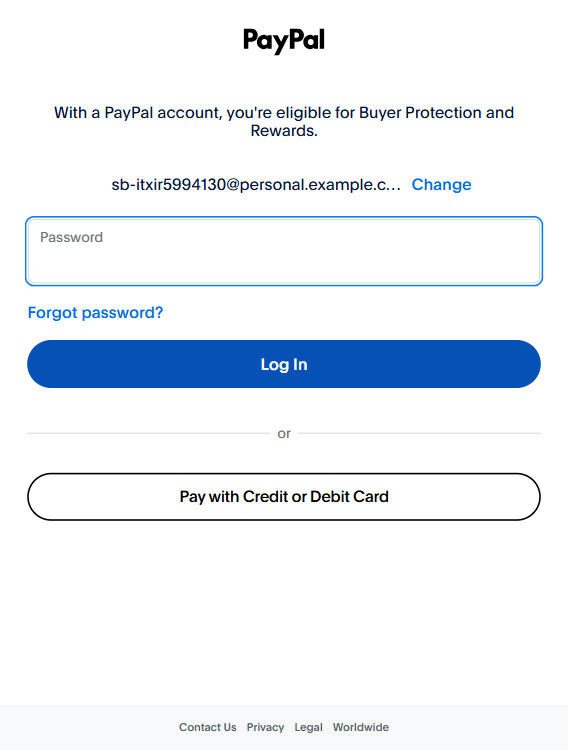


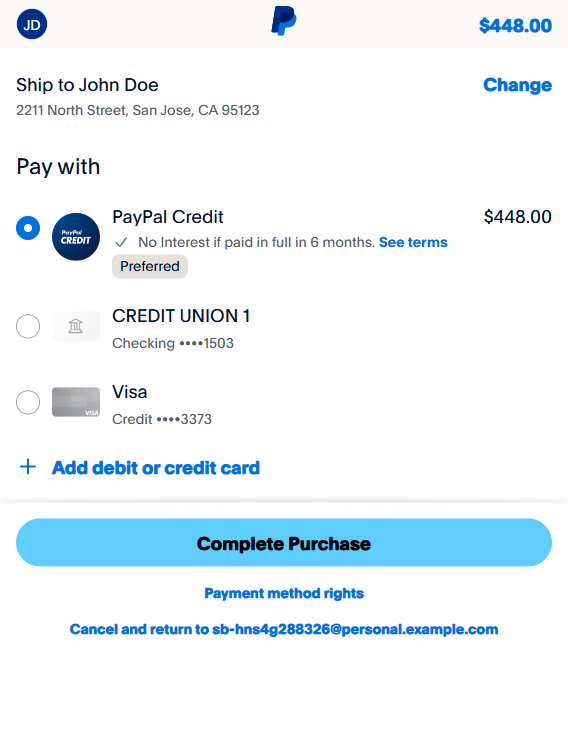
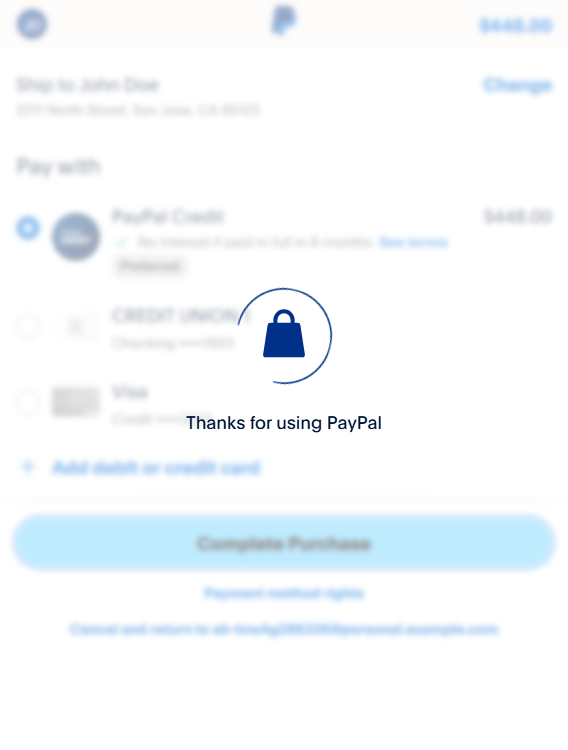


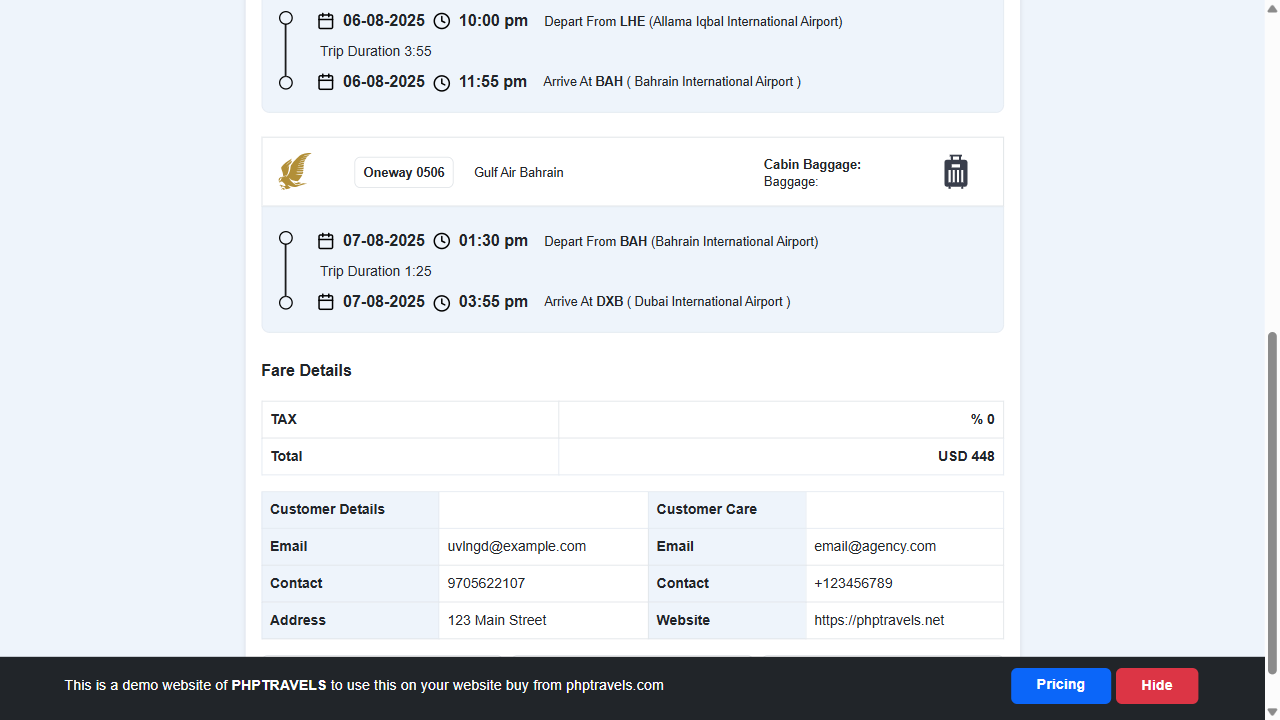










**5. Test Data Summary (Randomized)**

|  |  |  |
| --- | --- | --- |
| Field | Type | Sample Value |
| First Name | Random Name | "Lohena" |
| Last Name | Random Name | "Xaletu" |
| Email | Random Email | "lohena@example.com" |
| Phone | Random Number | "9876543210" |
| Passport Number | Random Passport | "P1234567" |
| Traveler Email | Random Email | "xaletu@example.com" |

**Note:** Randomized fields help prevent data collisions and allow repeated safe test execution.

**6. Result Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Step | Expected Outcome | Actual Result | Status |
| Homepage Visit | Page loads successfully | As expected | Pass |
| Flight Selection | Flight card interaction works | Flight selected successfully | Pass |
| Form Submission | All fields validated and submitted | Form accepted | Pass |
| Booking Confirmation | Redirect to payment page | Redirection successful | Pass |
| PayPal Login | Credentials accepted | Successfully logged in | Pass |
| Payment Completion | Booking confirmation shown, download link | PDF available | Pass |
| Final Scroll & Wait | Booking details visible | Scrolled and verified | Pass |

**7. Attachments**

* 📄 **Screenshots:** Stored locally or to be integrated in test reporting framework.

**8. Theoretical Insight & Best Practices**

1. **Why Playwright?**  
   Playwright is a modern E2E automation library supporting multiple browsers and supports async execution, cross-browser testing, and powerful selectors.
2. **Why Randomized Test Data?**  
   Random inputs simulate real-world variability and avoid form data conflicts or server-side caching issues.
3. **Why Delays & Screenshots?**
   * Static waits ensure elements are rendered before interaction.
   * Screenshots serve as visual evidence for test steps and help in debugging UI inconsistencies.
4. **Why Async Execution?**
   * Async coroutines improve performance and allow non-blocking wait handling.

**9. Suggestions for Production-Grade Automation**

* Implement exception handling with custom logging on failure.
* Export logs and screenshots to a centralized storage or test report.
* Parameterize test data with config files or test case frameworks (e.g., pytest).
* Integrate with CI tools such as GitHub Actions, GitLab CI, or Jenkins for continuous validation.
* Generate HTML/PDF reports using pytest-html, allure, or custom templates.