**System Requirements for PHPTRAVELS Testing**

**1. Functional Requirements**

**1.1 User Authentication**

* User registration and login with valid/invalid credentials.
* Password reset functionality.
* Role-based access control (Customer, Admin, Agent).
* Profile Management
* Session management & security.

**1.2 Hotel Booking System**

* Search functionality with location, price, and rating filters.
* Hotel details page displaying room availability, images, and pricing.
* Booking process with room selection, guest details, and payment validation.
* Cancellation & modification policies.

**1.3 Flight Booking System**

* Flight search with filters (date, price, stops, airline).
* Booking process including seat selection, passenger details, and payment integration.
* E-ticket generation & notifications.

**1.4 Car Rentals**

* Search & compare available cars.
* Booking process & payment validation.
* Rental duration

**1.5 Tours & Activities**

* Tour listings & search filters.
* Booking confirmation process.

**1.6 Payment System**

* Support for multiple payment methods (Credit Card, PayPal, Bank Transfer).
* Payment gateway validation & error handling.

**1.7 API Testing**

* Hotels API: Fetch hotel details & availability.
* Flights API: Retrieve flight data.
* Booking API: Create, modify, and cancel bookings.

**1.8 Admin Panel**

* User & Role Management.
* Inventory & Pricing Updates.
* Reporting & Analytics.

**2. Non-Functional Requirements**

**2.1 Performance Requirements**

* The system must handle concurrent users without degradation.
* API response times should be within acceptable limits (e.g., authentication <400ms, booking <2s).

**2.2 Usability Requirements**

* The UI should be intuitive and user-friendly.
* Compatibility across major browsers (Chrome, Firefox, Edge, Safari).

**2.3 Reliability & Availability**

* The system should ensure high availability with minimal downtime.
* Booking transactions should be processed reliably to avoid payment failures.

**2.4 Compatibility Requirements**

* The application should function correctly on both desktop and mobile devices.
* Cross-browser compatibility must be maintained.

**3. Testing Environment**

* Supported Browsers: Chrome, Firefox, Edge, Safari.
* Supported Devices: Desktop, Mobile, Tablets.
* Database: MySQL or equivalent relational database.
* Backend: PHP-based framework.
* Frontend: HTML, CSS, JavaScript.

**4. Testing Tools & Metrics**

**4.1 Test Coverage**

* Estimated coverage should be above 80% for critical functionalities.
* Example:
  + Car Search: 85%
  + Car Availability: 80%
  + Booking Process: 90%
  + Rental Policies Validation: 75%

**4.2 API Response Time Benchmarks**

|  |  |
| --- | --- |
| **API Type** | **Target Response Time** |
| Authentication | < 400 ms |
| Data Retrieval | < 300 ms |
| Checkout Operations | < 2 sec |

**Current API Response Times (Average 400.7ms)**

* Login API: 310ms
* Register API: 307ms
* Reset Password API: 354ms
* Hotel Search API: 320ms
* Booking API: 438ms
* Payment API: 1s

**5. Testing Risks & Challenges**

* **Browser & Device Compatibility:** Variations in UI behavior across different browsers.
* **Third-Party API Failures:** Downtime or incorrect data from booking and payment APIs.
* **Performance Issues:** Load handling for concurrent searches and bookings.
* **Security Risks:** Weak authentication could allow unauthorized access.
* **Data Integrity Issues:** Booking records may not sync properly between frontend and backend.

**6. Out of Scope**

* 3rd Party Integrations (unless explicitly required).
* UI/UX Design Consistency (unless requested separately).
* Data Migration & Back-end Database Queries.
* Security Testing (to be handled separately).