TID	Test Case Description	Test Steps	Test Data (Request Body)	Expected Result	Test Result (Pass/Fail)
1	Verify successful booking creation with valid data	1. Send POST request to booking endpoint 2. Use valid request body	{ "firstname": "Abhishek", "lastname": "Sharn	Status 200, response contains bookingid and matching request data	
2	Verify booking with minimum required fields	1. Send POST with only mandatory fields	{ "firstname": "A", "lastname": "B", "totalpric	Status 200, booking created without additionalneeds	
3	Verify booking with maximum length strings	1. Send POST with max length names	{ "firstname": "A...<250 chars>", "lastname":	Status 200, handles long strings	
4	Verify booking with special characters in names	1. Send POST with special chars	{ "firstname": "Jöhn-D'oe", ... }	Status 200, accepts special chars	Pass
5	Verify booking with totalprice as 0	1. Send POST with totalprice=0	{ ..., "totalprice": 0, ... }	Status 200 or proper validation	Pass
6	Verify booking with negative totalprice	1. Send POST with totalprice=-100	{ ..., "totalprice": -100, ... }	Status 400 (bad request)	
8	Verify booking with depositpaid=false	1. Send POST with depositpaid=false	{ ..., "depositpaid": false, ... }	Status 200, accepts false value	Pass
9	Verify booking with invalid depositpaid (string)	1. Send POST with depositpaid="true"	{ ..., "depositpaid": "true", ... }	Status 400 (type mismatch)	Fail
10	Verify booking with same checkin/checkout date	1. Send POST with checkin=checkout	{ "bookingdates": { "checkin": "2024-01-01",	Status 400 (invalid dates)	Fail
11	Verify booking with checkout before checkin	1. Send POST with invalid dates	{ "bookingdates": { "checkin": "2024-01-02",	Status 400 (invalid dates)	Fail
12	Verify booking with invalid date format	1. Send POST with malformed dates	{ "bookingdates": { "checkin": "01-01-2024",	Status 400 (bad format)	Fail
13	Verify booking with far future dates	1. Send POST with dates 10+ years ahead	{ "bookingdates": { "checkin": "2035-01-01",	Status 200 or business validation	Pass
14	Verify booking with past dates	1. Send POST with historical dates	{ "bookingdates": { "checkin": "2000-01-01",	Status 200 or business validation	Pass
15	Verify booking with empty additionalneeds	1. Send POST with empty string	{ ..., "additionalneeds": "" }	Status 200, handles empty string	Pass
16	Verify booking with long additionalneeds	1. Send POST with 500+ char needs	{ ..., "additionalneeds": "A...<500 chars>" }	Status 200 or proper validation	Pass
17	Verify booking with missing firstname	1. Send POST without firstname	{ "lastname": "Sharma", ... }	Status 400 (required field)	Fail
18	Verify booking with null firstname	1. Send POST with firstname=null	{ "firstname": null, ... }	Status 400 (invalid)	
19	Verify booking with missing bookingdates	1. Send POST without bookingdates	{ "firstname": "A", ... }	Status 400 (required)	Fail
20	Verify booking with empty bookingdates	1. Send POST with empty bookingdates	{ "bookingdates": {} }	Status 400 (invalid)	Fail
21	Verify booking with extra fields	1. Send POST with unexpected fields	{ ..., "discount": 10 }	Status 400 (ignores extras) or 400	Pass
22	Verify booking with malformed JSON	1. Send POST with invalid JSON	{ "firstname": "A", ... (missing brace)	Status 400 (bad JSON)	
23	Verify booking with SQL injection attempt	1. Send POST with SQLi payload	{ "firstname": "Robert'); DROP TABLE booki	Status 400 (sanitized)	
24	Verify booking with XSS attempt	1. Send POST with XSS payload	{ "firstname": "<script>alert(1)</script>", ... }	Status 400 (sanitized)	
25	Verify booking with content-type=text/plain	1. Send POST with wrong content-type	Same valid data but wrong header	Status 415 (unsupported media)	
26	Verify booking with no content-type header	1. Send POST without content-type	Same valid data	Status 400 or 415	
27	Verify booking with accept=text/xml	1. Send POST requesting XML response	Valid data + Accept: text/xml	Status 406 (not acceptable) or 200 with JSON	
28	Verify response headers	1. Send valid POST request	Valid data	Headers include content-type=application/json	
29	Verify response time < 1s	1. Send valid POST request	Valid data	Response time < 1000ms	
1	Valid Booking ID	1. Send GET request with a valid booking ID. 2	1	Status: 200 OK Response: Returns booking details (firstname, lastname, dates, etc.) matching the ID.	
2	Invalid Booking ID (Non-existent)	1. Send GET request with an ID that doesn't exist. 2. Check response.	id = 999999	Status: 404 Not Found Response: Error message like "Not Found" or en	Pending
3	Invalid Booking ID (Non-numeric)	1. Send GET request with a non-numeric ID (e.g., st 2. Check response.	id = "abc"	Status: 400 Bad Request or 404 Not Found Response: Error for invalid input.	Pending
4	Missing Booking ID	1. Send GET request without an ID. 2. Check response.	id = (empty)	Status: 405 Method Not Allowed or 404 Not Fou Response: Error for invalid endpoint.	Pending
5	Validate Response Schema	1. Send GET request with valid ID. 2. Verify JSON structure and data types.	id = 1	Status: 200 OK Response: Matches schema (e.g., firstname is str	Pending