Event Management App – Test Plan

📌 Application Overview

This is a RESTful API for managing events and bookings. Key features include:

- Create an Event
- List Events
- Create a Booking
- Cancel a Booking

🎯 Testing Objectives

- Validate API endpoints for correct functionality and data.
- Catch regressions early via automation.
- Verify request/response structure and logic.
- Confirm business rules: capacity handling, booking limits, cancellation rules.

📂 Test Categories

Category	Description
Functional	Verify core features work as expected
Regression	Re-run critical tests to detect breakages
Boundary	Test edge values for fields or logic
Negative	Handle invalid inputs or paths gracefully
Contract	Ensure API response schema is consistent

🧪 Test Scenarios

#	Test Case Description	Type	Should Automate?	Priority
01	Create a valid event	Functional	✅ Yes	High
02	Create an event with missing fields	Negative	✅ Yes	Medium
03	Create an event with invalid date range	Negative	✅ Yes	Medium
04	List all events	Functional	✅ Yes	High
05	Verify event list includes newly created event	Functional	✅ Yes	High
06	Create booking with valid event reference	Functional	✅ Yes	High
07	Book an already full event	Business	✅ Yes	Medium
08	Book an event with invalid event reference	Negative	✅ Yes	Medium
09	Verify available capacity reduces after booking	Functional	✅ Yes	High
10	Create duplicate booking	Business	❌ No	Medium
11	Cancel booking with valid reference	Functional	✅ Yes	High
12	Cancel a booking twice	Negative	❌ No	Low
13	Cancel booking with invalid reference	Negative	✅ Yes	Medium
14	Validate event response schema and types	Contract	✅ Yes	Medium
15	Booking response includes correct fee and event details	Functional	✅ Yes	High
16	Missing header	Security	✅ Yes	High
17	Validate event list - available capacity should not exceed total capacity	Business	✅ Yes	High

⚙️ Test Implementation

- TestNG + Maven + RestAssured + ExtentReports

Sample TestNG Command:

- `mvn test (To do a simple run)`

OR

- `mvn test -DbookingType=REGULAR -Dsurefire.suiteXmlFiles=testng.xml (To override inputs at runtime) (sample: BookingTypes - VIP, REGULAR)`

📊 Reporting

- **ExtentReports**: Generated in `/target/ExtentReport/`
- **Log4j**: Logs output to `eventmanager` logger

📦 Test Data Strategy

- Dynamic event IDs and booking references are saved to:
  - `EventId.txt`
  - `bookingId.txt`
- Configurable fields (e.g. `userId`, `bookingType`) read from `config.properties`
- Data-driven tests utilizing:
  - `@DataProvider` to test multiple inputs for a single test case making a test method reusable and dynamic.
  - `@parameter` to inject a single value into the test, typically for environment or config-based inputs and **control execution** externally (CI/CD) so that data can be overridden at runtime if necessary.

🧹 Maintenance Notes

- Review regression suite after each major feature addition
- Keep test data dynamic to avoid conflicts
- Prioritize test coverage for high-impact features