# What Developers Need to Ensure (QA/QC Readiness)

# Frontend (React):

1. **Input Validations:**

**What it means:** Ensuring the user enters only valid data (e.g., no letters in a phone number field).  
**Why it matters:** **Prevents bad data** from reaching the backend (server/database) and helps QA test how different inputs behave.

Example : (Without input validation:

* A user could enter abc in the **Phone Number** field.
* Or leave the **Email** blank.
* Or enter special characters in the **Name** field.

That’s not good. This can confuse the system or even break it!)

1. **Error Messages:**

**What it means:** Messages shown when something goes wrong (e.g., "Password is too short"). (Ensure that proper error messages are shown for invalid inputs)  
**Why it matters:** QA checks if users are getting helpful feedback when mistakes are made.

1. **Browser Console Logs:**

**What it means:** Developers use the console.log () to show outputs/errors in the browser console. (maintain console log)  
**Why it matters:** QA uses these logs to report errors and understand frontend issues better.

1. **Unit Testing of Components:**

**What it means:** Small tests for individual components (like buttons or forms) using testing tools like Jest.  
**Why it matters:** Finds issues early before QA starts testing the full app.

# Backend (Java + Spring Boot + REST APIs):

1. **Logging:**

**What it means:** Server-side logs record errors, actions, or important info using tools like Log4j or SLF4J. (**Logging** means **recording information about what's happening on the server.**)  
**Why it matters:** QA and devs can troubleshoot backend issues easily (Ensure if proper server-side logging is set up.)

1. **Error Codes in APIs:**

**What it means:** API responses should return proper status codes (e.g., 404, 500) and messages.Error codes give **clear signals** about what went wrong and where, so **QA** can understand and report bugs **accurately. (**Implement proper error codes and meaningful messages in the API responses to help us test more accurately.)  
**Why it matters:** QA understands what's going wrong and where, like authentication failures.

1. **Exception Handling:**

**What it means:** The Backend should gracefully catch and handle unexpected situations without crashing.(Make sure the backend handles exceptions properly and returns clear error messages instead of crashing.)  
**Why it matters:** QA can continue testing even if things go wrong; no white screens or server crashes.

1. **Request Validations:**

**What it means:** Backend checks if request data is valid (e.g., price not negative). (You test APIs with **wrong or edge-case data** on purpose (this is called **negative testing**), if validation works, the API should **reject** that data with proper error messages.)  
**Why it matters:** Helps QA ensure incorrect data gets blocked at the server level.

1. **API Documentation:**

**What it means:** Clear documentation (e.g., using Swagger) explaining how each API works.  
**Why it matters:** QA knows exactly how to test API inputs and outputs.

# Database Layer:

1. **Data Integrity:**

**What it means:** Enforcing rules like primary/foreign keys to keep data accurate and connected. You **don't have to check** if players or moves are "real" — the database won’t allow bad data. (Can we enforce foreign keys and constraints so we don’t run into fake bugs or broken test scenarios?)  
**Why it matters:** QA doesn’t have to worry about duplicate or broken records.

1. **DB Validations:**

**What it means:** Rules directly in DB (e.g., "email" should not be null). Can you add database rules (constraints) to make sure fields like email and player ID are always correct? That way, we avoid data issues when testing.  
**Why it matters:** Ensures correctness even if validations fail in the frontend/backend.

Note: **Data Integrity** = The **overall concept** of making sure data is **correct, consistent, and reliable**.

**DB Validation** = Specific **Rules in the database** (such as **NOT NULL** or **UNIQUE**) to make sure data follows certain standards before it’s saved.

1. **Sample Test Data:**

**What it means:** Dummy data in tables (users, orders, etc.) for testing purposes.  
**Why it matters:** QA saves time by not needing to create test data manually.

# **Testing Resources You (QA) Need From Them:**

# FRONT-END:

* Links to working pages/web app (weblinks) - to access and test the UI
* List of completed features/components - To map against test cases
* Any test credentials (if login required) - To log in and test different roles
* Info about input fields, expected values, validations - to verify how the frontend behaves

# Server

* API documentation (Swagger or Postman collection) - To test and verify APIs
* Base URLs + API Paths - For calling APIs
* Expected request/response format - So the tester can verify the API response
* Sample input data you can use to test - Helps create test cases
* Info about error codes or response messages - So the tester knows the expected behavior in negative scenarios

# Database (Data layer)

* Access to test DB or SQL dump - To validate data flow and correctness
* Table Structure (column names, data types) - For understanding DB logic
* Sample test data - For data-based testing
* Any constraints/validations set at the DB level -To check edge cases