In software testing, **defects** (also known as bugs or faults) are deviations from the expected behaviour of the software. They can arise at various stages of development due to different reasons. Here are the **common types of defects** in software testing:

1. **Functional Defects**
2. **Performance Defects**
3. **Usability Defects**
4. **Compatibility Defects**
5. **Security Defects**
6. **Boundary/Edge Case Defects**
7. **Data Defects**
8. **Logical Defects**
9. **Arithmetic Defects**
10. **Configuration Defects**
11. **Syntax Errors**
12. **Interfacing Defects**

**1. Functional Defects**

* Occur when a feature or function doesn’t work as intended.
* Example: Clicking a "Login" button doesn’t log the user in even with valid credentials.

**Scenario**: E-commerce website – "Add to Cart" button

**Step-by-step**:

1. Open product detail page.
2. Click "Add to Cart" button.
3. **Expected**: Product should be added to the cart and cart count should increase.
4. **Actual**: Nothing happens; cart count remains zero.
5. **Defect**: Functional defect – the core functionality is broken.

**2. Performance Defects**

* Software is too slow or doesn't respond within acceptable time limits.
* Example: A page takes too long to load or the system crashes under load.

**Scenario**: Online banking – loading account statement

**Step-by-step**:

1. Login to your account.
2. Go to "Account Statement" section.
3. Enter the date range: last 6 months.
4. Click "Generate Statement".
5. **Expected**: Results should load within 2-3 seconds.
6. **Actual**: Takes 30+ seconds or times out.
7. **Defect**: Performance defect – poor response time.

**3. Usability Defects**

* Issues that make the software difficult or confusing to use.
* Example: Poor navigation, unclear labels, or non-intuitive UI elements.

**Scenario**: Job portal – Resume upload

**Step-by-step**:

1. Navigate to the "Upload Resume" section.
2. Click "Browse" and upload a resume.
3. No message or progress bar shown.
4. User doesn’t know whether it's uploading or not.
5. **Expected**: A progress bar and success message like "Resume uploaded successfully".
6. **Defect**: Usability defect – poor user feedback.

**4. Compatibility Defects**

* Application behaves differently across different environments or platforms.
* Example: Works fine on Chrome but breaks on Safari.

**Scenario**: Online shopping site – product display issue

**Step-by-step**:

1. Open product list on Google Chrome – products are shown correctly.
2. Open the same page in Safari – product images are not aligned.
3. **Expected**: Same layout across all major browsers.
4. **Defect**: Compatibility defect – different behaviour across browsers.

**5. Security Defects**

* Vulnerabilities that expose the system to threats or unauthorized access.
* Example: A user can access admin features without proper authorization.

**Scenario**: Admin panel access

**Step-by-step**:

1. Open browser, log in as a normal user.
2. Try accessing /admin/dashboard via URL directly.
3. **Expected**: Should be redirected to access denied or login page.
4. **Actual**: Normal user gains access to admin functions.
5. **Defect**: Security defect – unauthorized access.

**6. Boundary/Edge Case Defects**

* Errors occurring at the limits of input values.
* Example: System fails when a field expects 1-100 and gets 0 or 101.

**Scenario**: Registration form – Age input

**Step-by-step**:

1. Age field accepts values between 18 to 60.
2. Enter 17 or 61 and submit the form.
3. **Expected**: Validation error message.
4. **Actual**: Form submits successfully.
5. **Defect**: Boundary defect – edge limits not enforced.

**7. Data Defects**

* Issues related to incorrect or corrupt data handling.
* Example: Saving wrong data in the database or loss of data during transactions.

**Scenario**: Booking system – wrong customer data saved

**Step-by-step**:

1. Fill booking form with Name: John, Email: john@example.com.
2. Submit the form.
3. Open the confirmation page.
4. **Expected**: Same name and email shown.
5. **Actual**: Name is correct, but email shows null or a different user’s email.
6. **Defect**: Data defect – corruption or wrong mapping of data.

**8. Logical Defects**

* Incorrect implementation of business logic.
* Example: Tax calculation is incorrect due to a wrong formula.

**Scenario**: Shopping cart discount logic

**Step-by-step**:

1. Add products worth ₹1000 to cart.
2. Apply coupon code: “SAVE10” (10% discount).
3. **Expected**: ₹100 should be deducted; final price ₹900.
4. **Actual**: Final price shows ₹950.
5. Check code: Developer mistakenly applied 5% instead of 10%.
6. **Defect**: Logical defect – wrong implementation of business logic.

**9. Arithmetic Defects**

* Errors in mathematical calculations or operations.
* Example: Total price shows a wrong sum due to rounding errors.

**Scenario**: Invoice calculation

**Step-by-step**:

1. Item price: ₹99.99, Quantity: 2
2. **Expected**: Total ₹199.98
3. **Actual**: Shows ₹200.00 due to rounding error in code.
4. **Defect**: Arithmetic defect – incorrect calculation or rounding.

**10. Configuration Defects**

* Issues related to software or hardware settings.
* Example: Wrong build or server configuration causes app to crash.

**Scenario**: Application not running in test environment

**Step-by-step**:

1. Deploy app in staging.
2. Open login page.
3. **Expected**: App should load.
4. **Actual**: Error 404 – file not found.
5. Root cause: Wrong environment file used, pointing to non-existent paths.
6. **Defect**: Configuration defect – incorrect setup.

**11. Syntax Errors**

* Mostly caught during development or compilation phase.
* Example: Misspelled variable names, missing semicolons.

**Scenario**: During development, code doesn’t compile

**Step-by-step**:

1. Developer writes JavaScript:

|  |
| --- |
| function greet(name  {  console.log("Hello " + name);  } |

1. Error: Unexpected token – missing closing parenthesis.
2. Code won’t run.
3. Defect: Syntax error – a typo or mistake in code structure.

**12. Interfacing Defects**

* Problems in communication between different modules or systems.
* Example: An API call doesn’t return expected data due to format mismatch.

**Scenario**: Payment gateway integration

**Step-by-step**:

1. Place order and choose "Pay with Razorpay".
2. App sends payment request with wrong parameter user\_name instead of username.
3. Razorpay responds with failure.
4. **Expected**: Payment should succeed.
5. **Actual**: Payment fails due to invalid parameter.
6. **Defect**: Interfacing defect – mismatch between API request structure.