System Analysis and Design

Software Testing

Prof. Rania Elgohary
rania.elgohary@cis.asu.edu.eg
Dr. Yasmine Afify
yasmine.afify@cis.asu.edu.eg

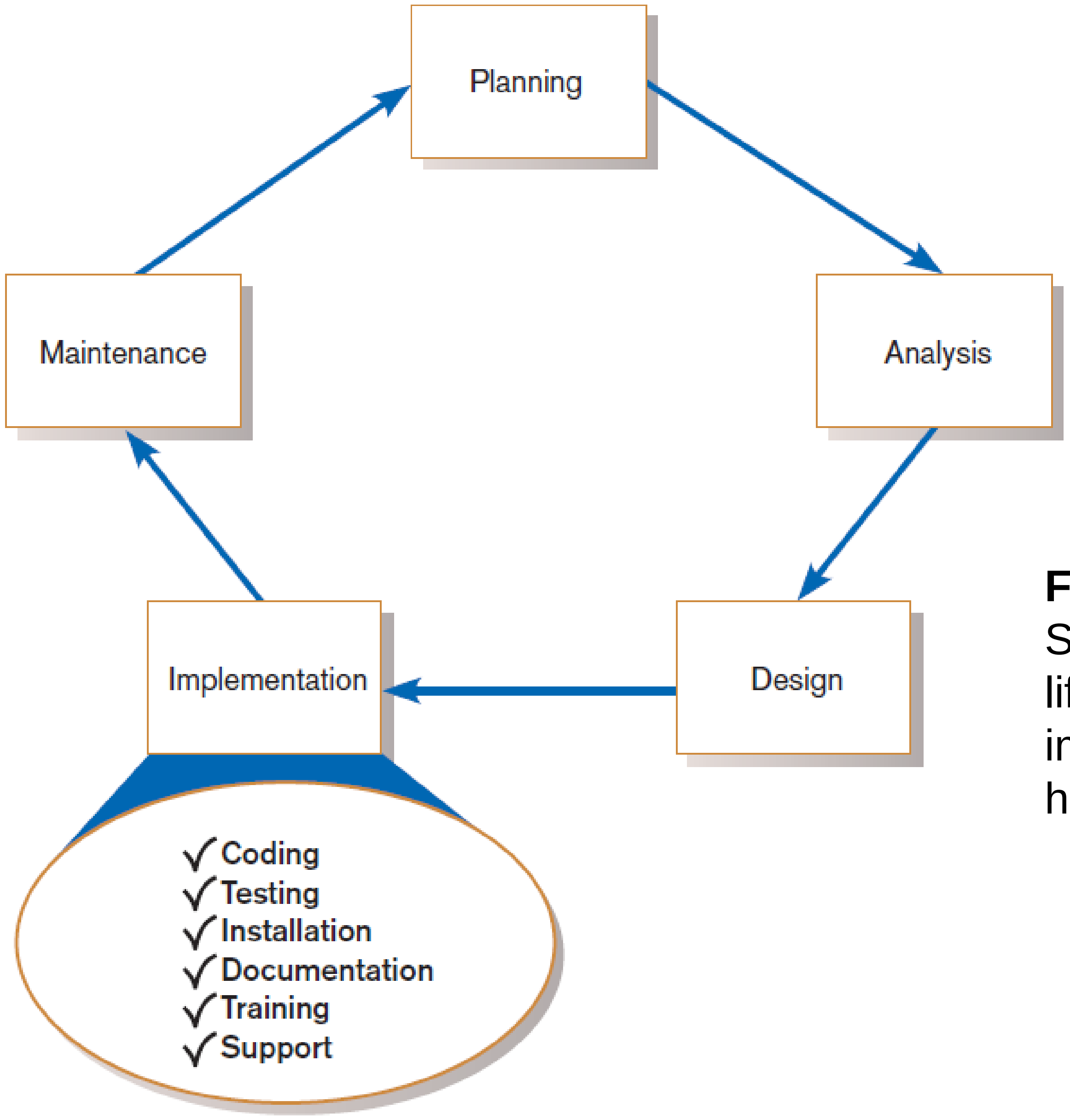
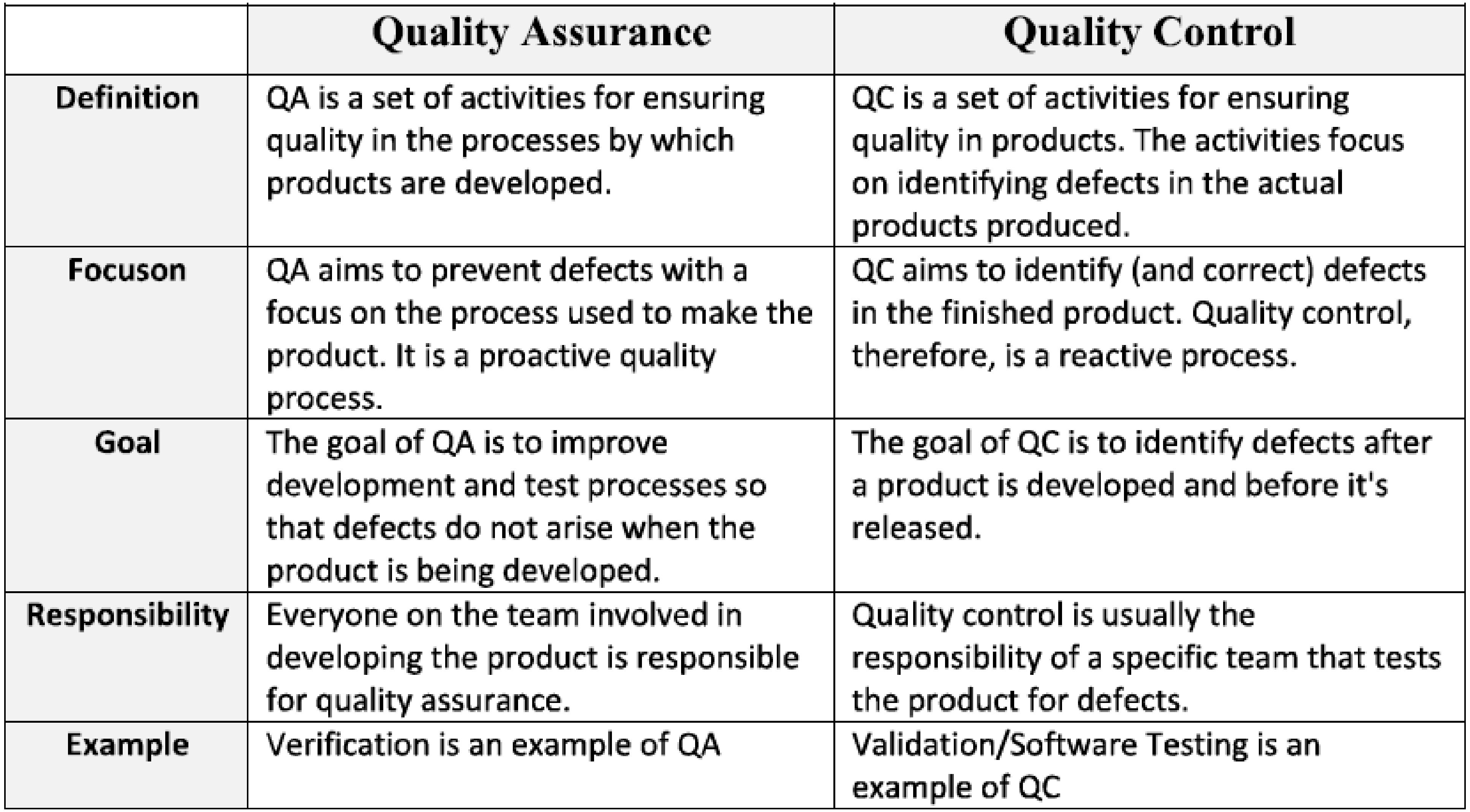


FIGURE 13-1
Systems development life cycle with the implementation phase highlighted

Questions…

- Should I test my own code, or should somebody else?
- Which code of my project should I test the most/least?
- Can I test all possible inputs to see whether something works?
- How do I know if I've tested well/enough?
- What constitutes a good or bad test case method?
- Is it good or bad if a test case fails?
- What if a test case itself has a bug in it?

Difference between QA and QC

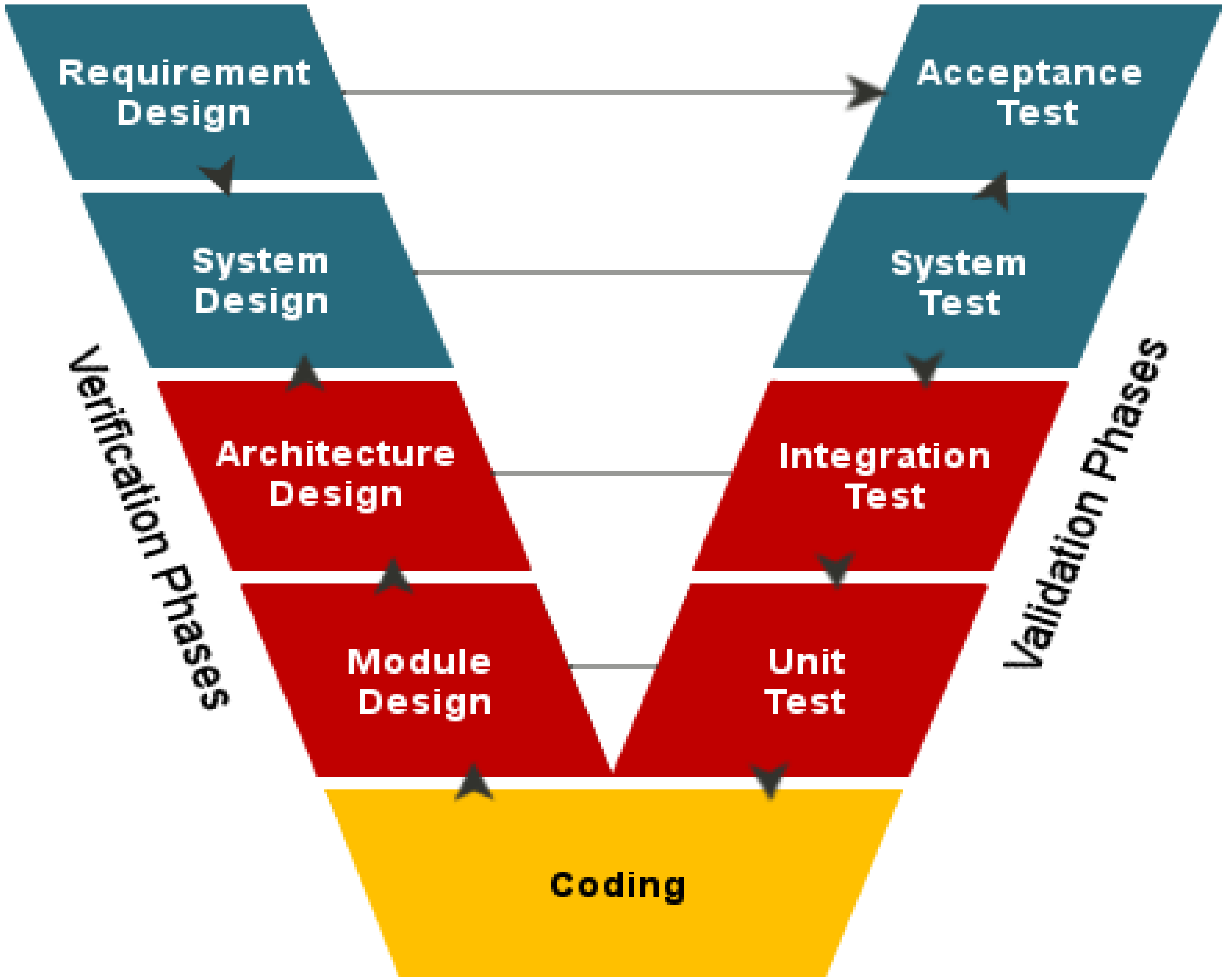


	Quality Assurance	Quality Control
Definition	QA is a set of activities for ensuring quality in the processes by which products are developed.	QC is a set of activities for ensuring quality in products. The activities focus on identifying defects in the actual products produced.
Focuson	QA aims to prevent defects with a focus on the process used to make the product. It is a proactive quality process.	QC aims to identify (and correct) defects in the finished product. Quality control, therefore, is a reactive process.
Goal	The goal of QA is to improve development and test processes so that defects do not arise when the product is being developed.	The goal of QC is to identify defects after a product is developed and before it's released.
Responsibility	Everyone on the team involved in developing the product is responsible for quality assurance.	Quality control is usually the responsibility of a specific team that tests the product for defects.
Example	Verification is an example of QA	Validation/Software Testing is an example of QC

Testing goals

1. To demonstrate to the customer that the software **meets its requirements**. This means that there should be at least one test for every requirement in the requirements document.
2. To **discover situations** in which the behavior of the software is incorrect, undesirable or does not conform to its specification. It is concerned with rooting out undesirable system behavior such as system crashes, unwanted interactions with other systems, incorrect computations and data corruption.

V-Model

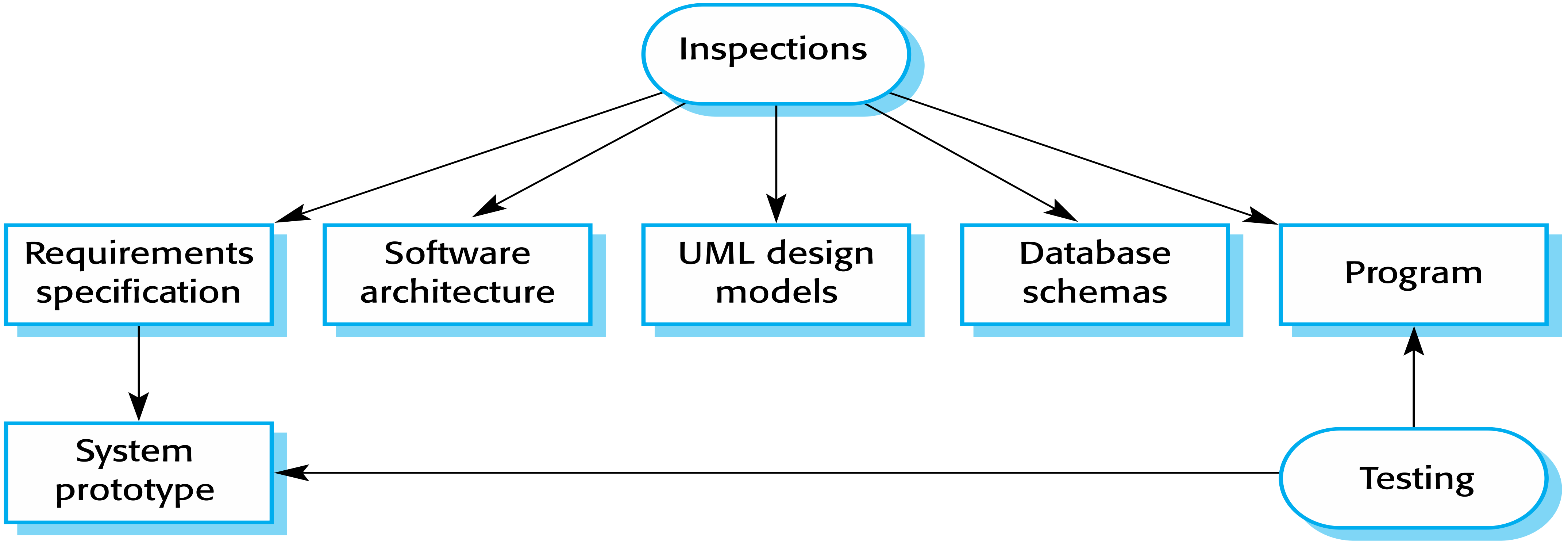

Requirement Design
Acceptance Test
System Design
System Test
Architecture Design
Integration Test
Module Design
Unit Test
Coding
Verification Phases
Validation Phases

Verification vs. Validation

- **Verification:** static process of verifying documents, design and code to determine they satisfy the conditions imposed at the start of that phase.

- **Validation:** dynamic process to ensure that the software satisfies the customer real requirements and expectations. Test data is entered, processed and the output is verified against what the user wants.

Static Testing Artifacts



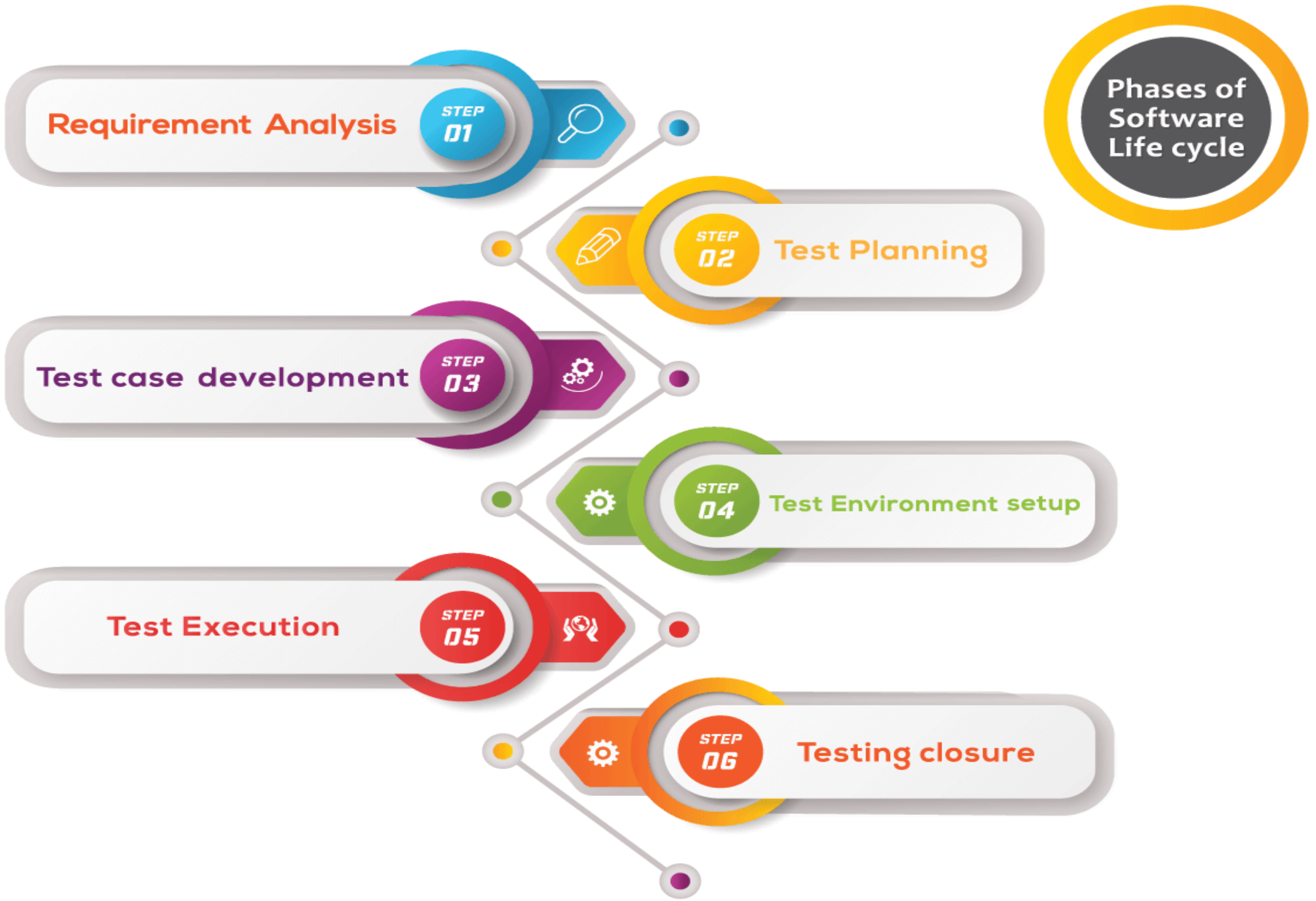
Static Testing Types

- **Peer Review.** An author asks a peer to read, comment, and critique his work. If the work artifact is code, the reviewer will read the code.
- **Walkthrough**. An author presents the work artifact to others. If the work artifact is code, the author walks through the code, explaining what each piece does.
- **Inspection**. An author requests the services of a moderator, scribe, reader/reviewers in a formal meeting. The moderator books the room, sends out the material. The reviewers read the material before the meeting. During the meeting, the reader/reviewers take turns reading the work artifact out loud. The scribe takes notes of issues the reader/reviewers discovered.

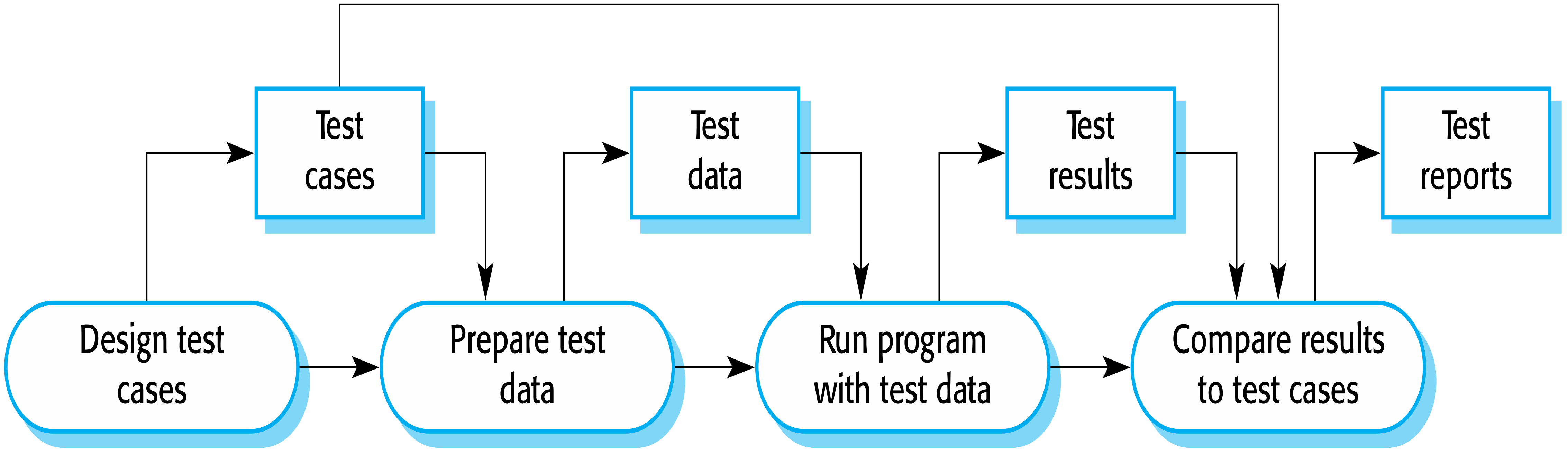
Dynamic Testing

- Testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use.
- When you test software, you execute a program using artificial data (test data).
- You check the results of the test run for errors, anomalies or information about the program’s non-functional attributes.
- Can reveal the presence of errors NOT their absence. Showing that the system is free of errors requires exhaustive testing which is almost always impossible.

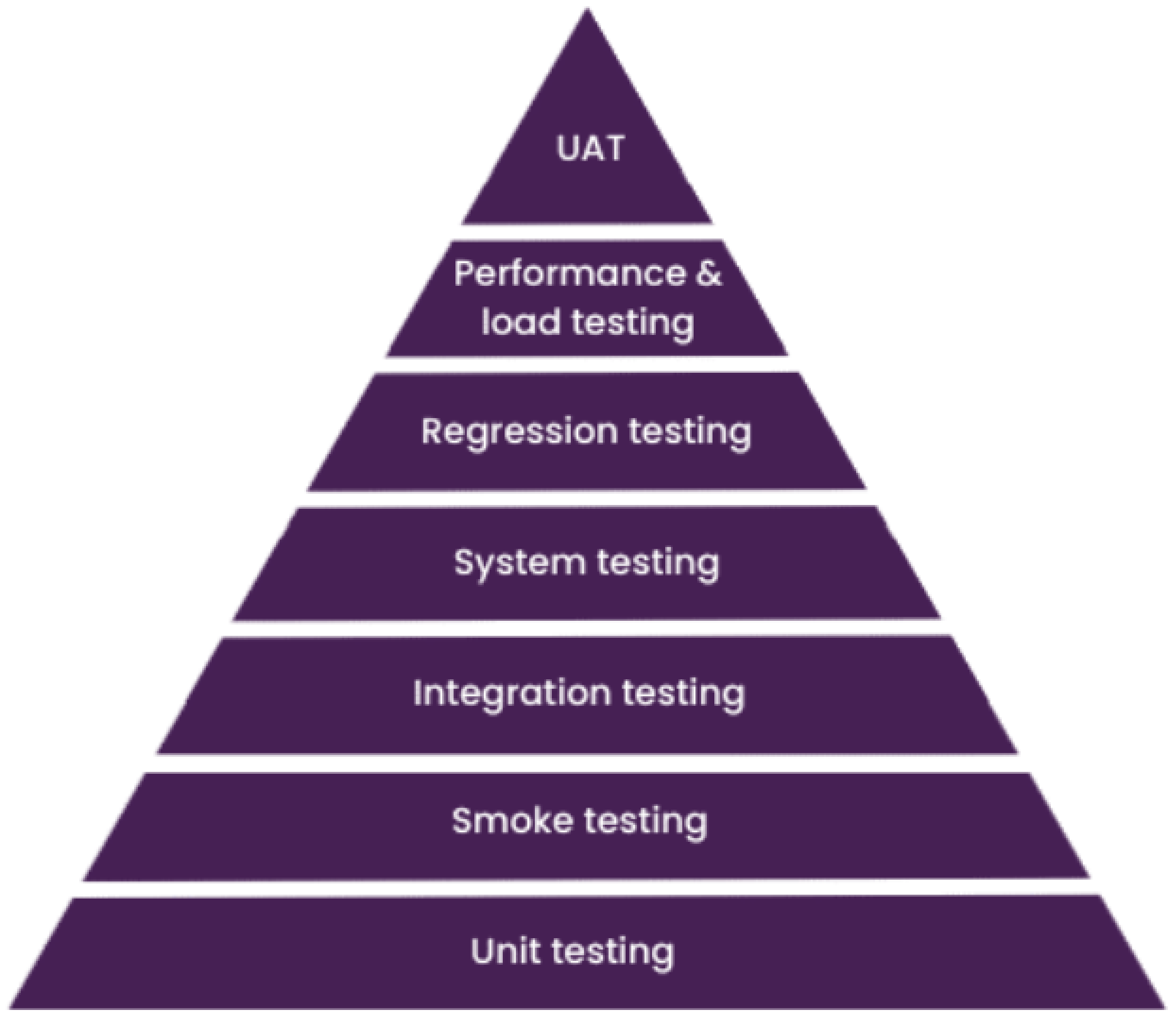
STLC (Software Testing Life Cycle)



Functional/Dynamic Testing



Functional Testing



Component/Unit Testing

- Component testing is the process of testing individual components in isolation.
- Carried out by the team developing the system.
- It is a defect testing process.
- Components may be:
    - Individual functions or methods within an object
    - Object classes with several attributes and methods

Advantages of Unit Testing

- The earlier a problem is identified, the fewer compound errors occur.
- Fixing problems early is usually cheaper than fixing them later in development.
- Easier debugging processes.
- Developers can quickly make changes to the codebase.
- Developers can reuse code and migrate it to new projects.

Smoke Testing

- A type of software testing that comprises of a non-exhaustive set of tests that aim at ensuring that the most critical/important functions work.
- It is executed "before" any detailed functional or regression tests are executed on the software build.
- The purpose is to reject a badly broken application so that the Quality team does not waste time installing and testing the software application.
- The result of this testing is used to decide if a build is stable enough to proceed with further testing.
- Conducted by developers or testers.

Integration testing

- A level of software testing where individual units are combined and tested as a group.
- The purpose of this level of testing is to expose faults in the interaction between integrated units.
- Subfunctions, when combined, may not produce the desired major function.
- Interfacing errors not detected in unit testing may appear.
- Timing problems (in real-time systems) are not detectable by unit testing.
- It is assumed that component tests on the individual objects within the component have been completed.

Integration Testing

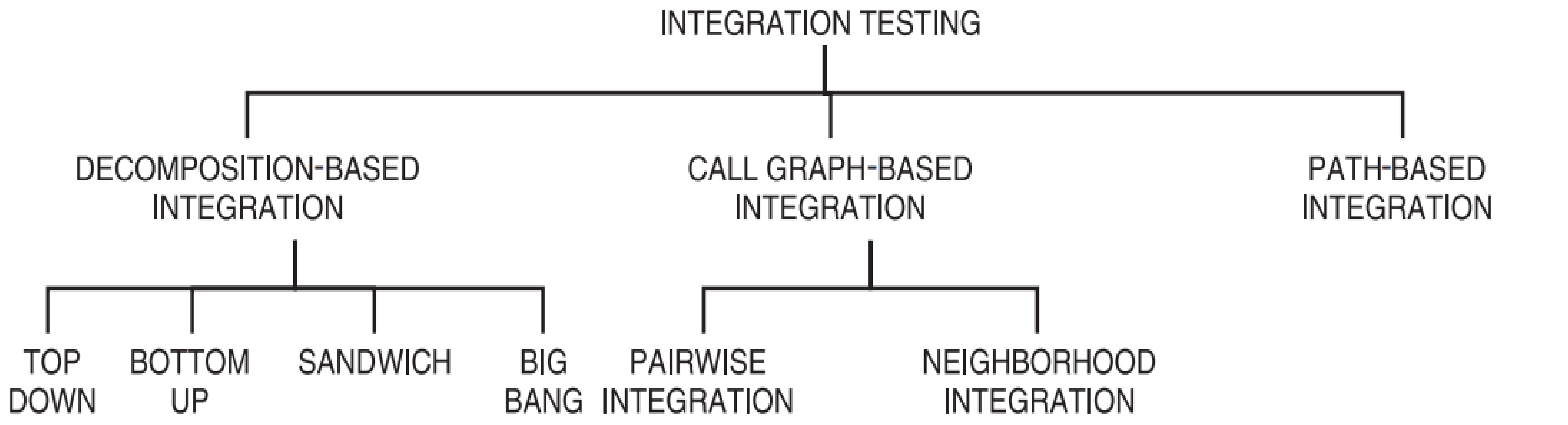


FIGURE 7.4

Decomposition-based Integration Top-down Integration Approach

- It begins with the main program, i.e., the root of the tree. Any lower-level unit that is called by the main program appears as a "stub." A stub is a piece of throw-away code that emulates a called unit.
- No. of Required Stubs = (No. of Nodes – 1)
- Once all of the stubs for the main program have been provided, we test the main program as if it were a standalone unit.
- When we are convinced that the main program logic is correct, we gradually replace the stubs with the actual code.

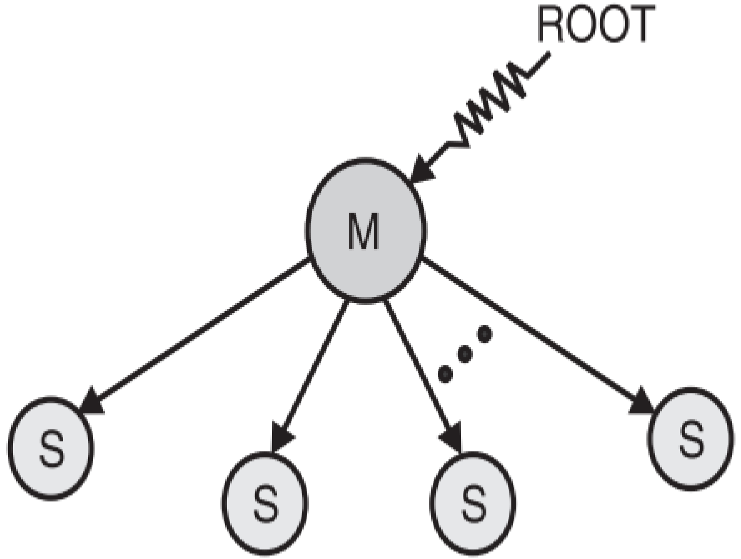


FIGURE 7.5 Stubs.

Decomposition-based Integration Bottom-up Integration Approach

- In bottom-up integration, we start with the leaves of the decomposition tree and test them with specially coded drivers.
- No. of drivers required =

  (No. of nodes – No. of leaf nodes)

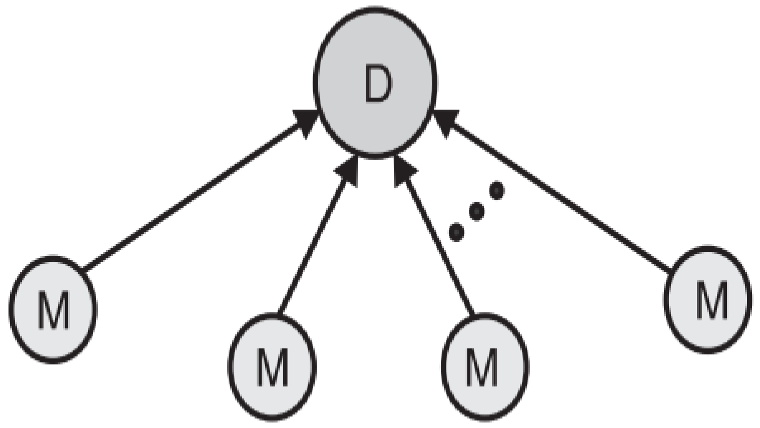
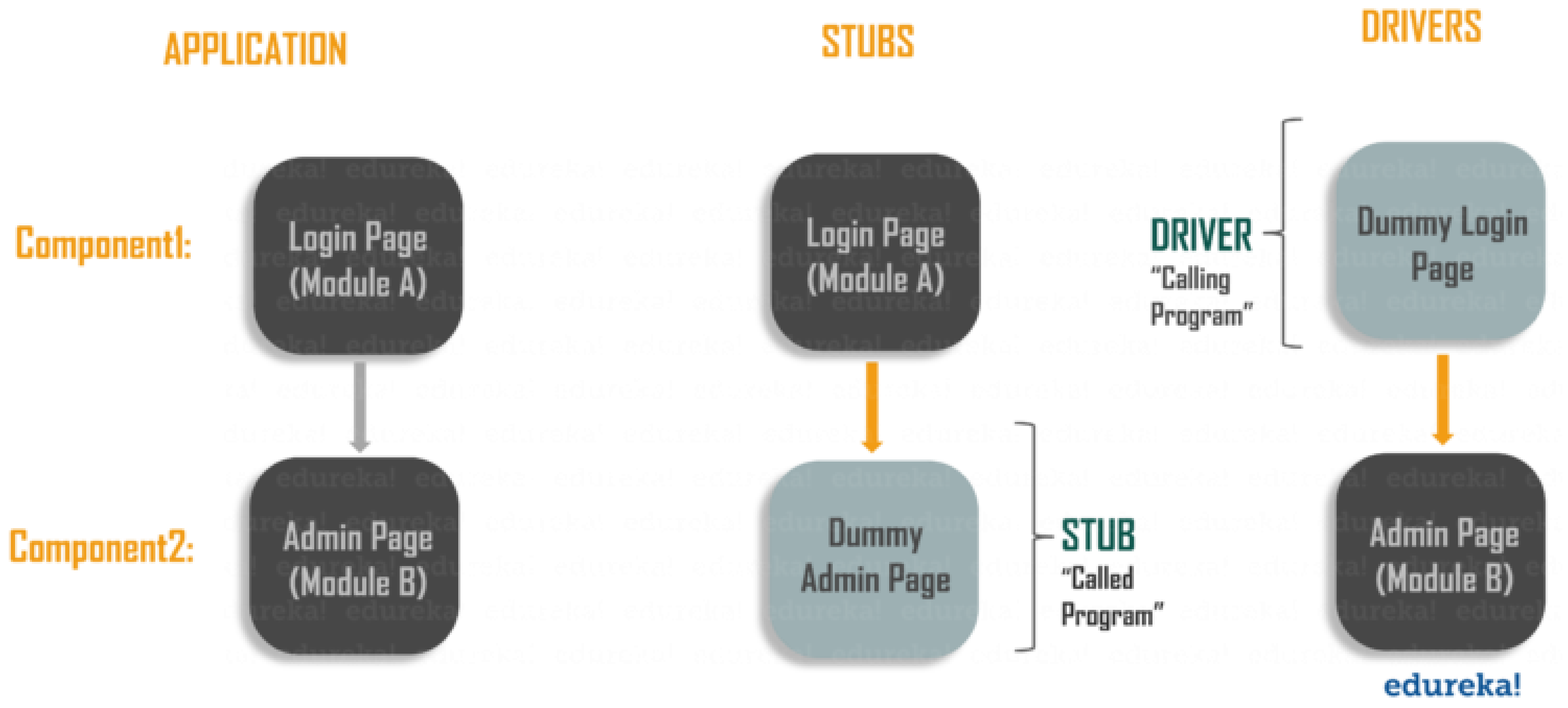


FIGURE 7.6 Drivers.


APPLICATION
STUBS
DRIVERS
Component1:
Component2:
Login Page (Module A)
Admin Page (Module B)
Login Page (Module A)
Dummy Admin Page
STUB "Called Program"
DRIVER "Calling Program"
Dummy Login Page
Admin Page (Module B)
edureka!

Decomposition-based Integration
Big-bang Integration

- Instead of integrating component by component and testing, this approach waits until all the components arrive, and **one round** of integration testing is done.
- It reduces testing effort and removes duplication in testing for the multi-step component integrations.
- Big-bang integration is ideal for a product where the interfaces are stable with fewer number of defects.

System testing

- System testing focuses on a complete, integrated system to evaluate compliance with specified requirements. Tests are made on characteristics that are only present when the entire system is run.
- The purpose of a system test is to evaluate the end-to-end system specifications.
- System testing is the only testing phase that tests both functional and non-functional aspects (quality factors) of the product.
- Test product behaviour in a holistic, complete, and realistic environment.
- Customer scenarios and usage patterns serve as the basis for system testing.

Regression testing

- All tests are re-run every time a change is made to the program.
- Regression testing is testing the system to check that changes have not ‘broken’ previously working code.
- In a manual testing process, regression testing is expensive but, with automated testing, it is simple and straightforward.

User acceptance testing (UAT)

- UAT is a stage in the testing process in which users provide input and advice on system testing.
- UAT is essential, even when comprehensive system and release testing have been carried out.
- Important because users have different perspective than the developers. Moreover, the influences from the user’s working environment have a major effect on the reliability, performance, usability and robustness of a system. These cannot be replicated in a testing environment.

User acceptance testing (UAT)

- The quality team has a meeting with the client, with "UAT test cases" which are the basic scenarios the client should run himself.
- The client will then give feedback: bugs or approval and a sign off that "UAT has passed successfully".
- This is a very crucial activity done for all projects in all IT companies and the quality team is responsible for managing it.

Testing levels

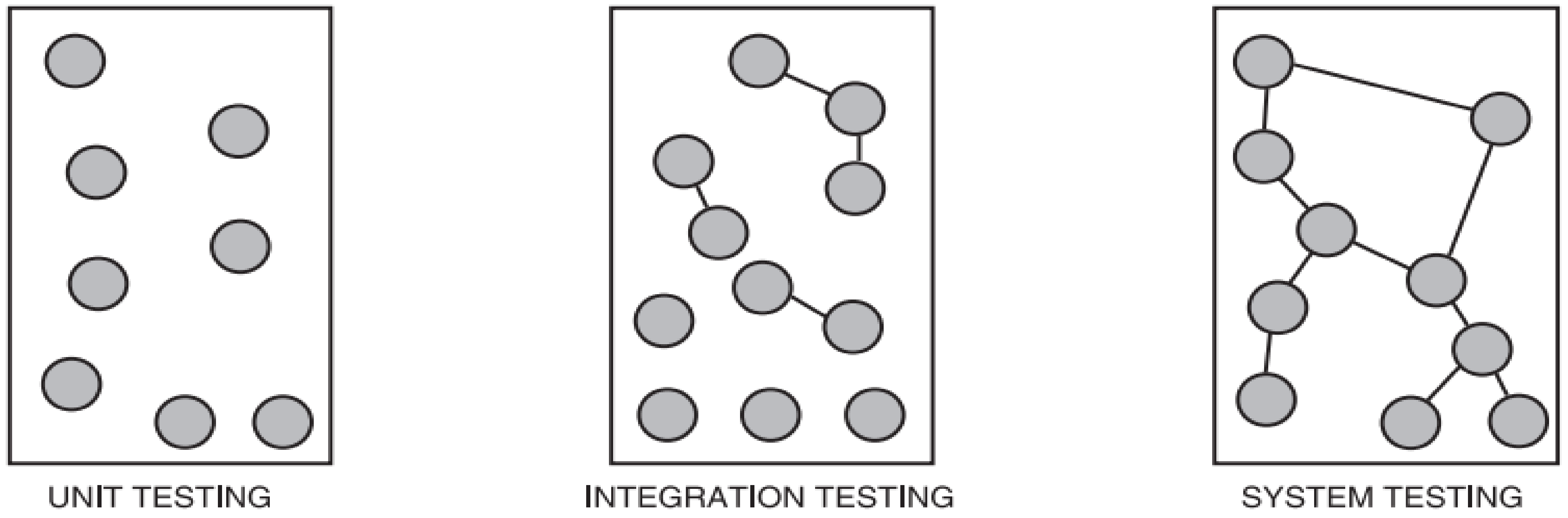


FIGURE 7.1 Levels of Testing.

Test Cycle Closure

- Testing team **meet** , **discuss** and **analyze** testing **artifacts.**
- Taking **lessons** from the **current** test **cycle** to remove the process **bottlenecks** for future test cycles and share **best practices** for any similar projects in future.

- Prepare **test metrics** based on the above parameters
- Prepare **Test closure report**

- **Deliverables**:
  - Test Closure report
  - Test metrics

Documentation

- Testing documentation involves the documentation of artifacts that should be developed before or during the testing of software.
- Documentation for software testing helps in estimating the testing effort required, test coverage, requirement tracking/tracing, etc.
- Some of the commonly used documented artifacts related to software testing:
  1. Test Plan
  2. Test Case
  3. Requirements Traceability Matrix

Master Test Plan

- A master test plan is developed during the analysis phase.
- During the design phase, the unit, integration and system test plans are developed.

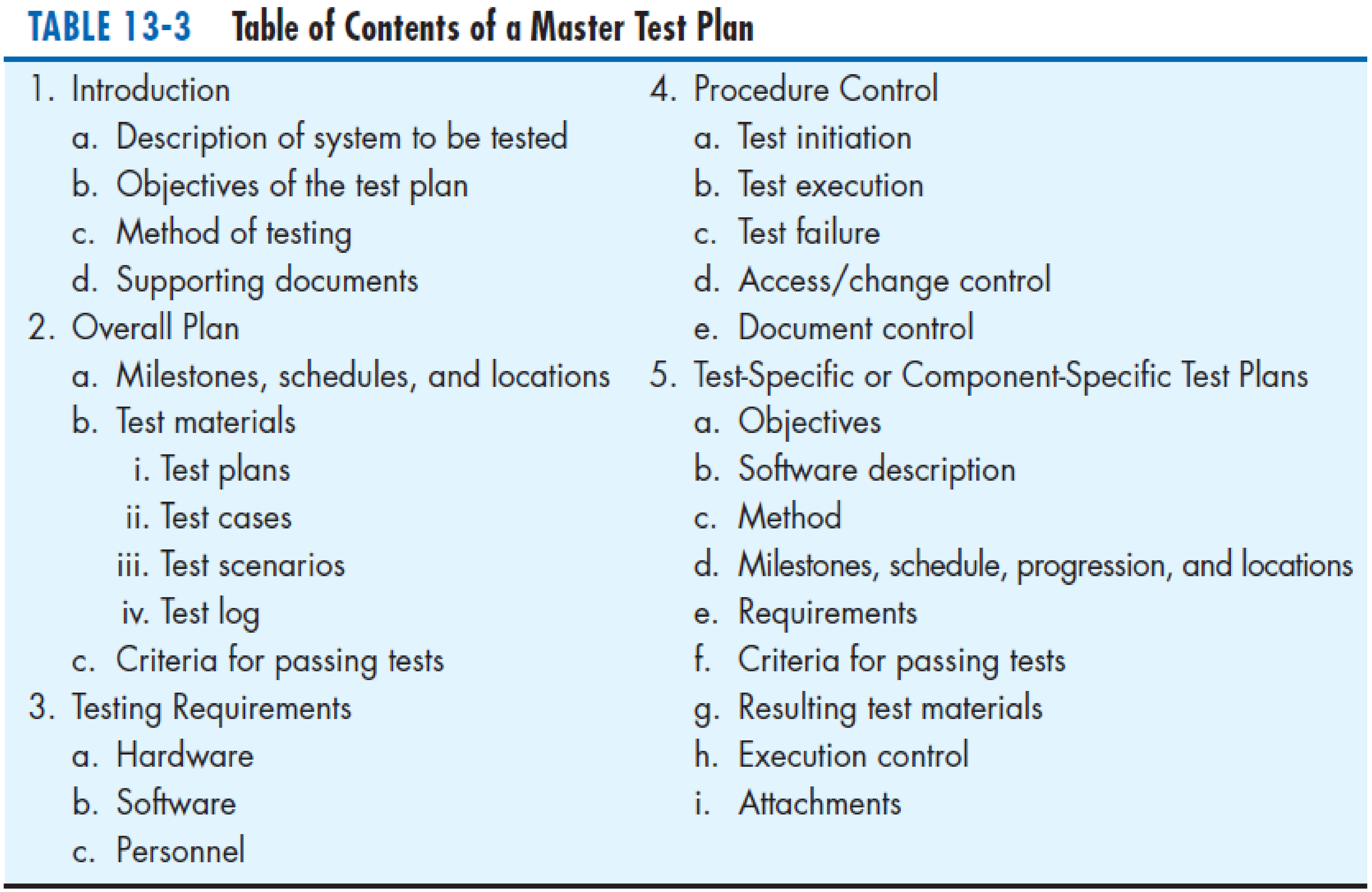


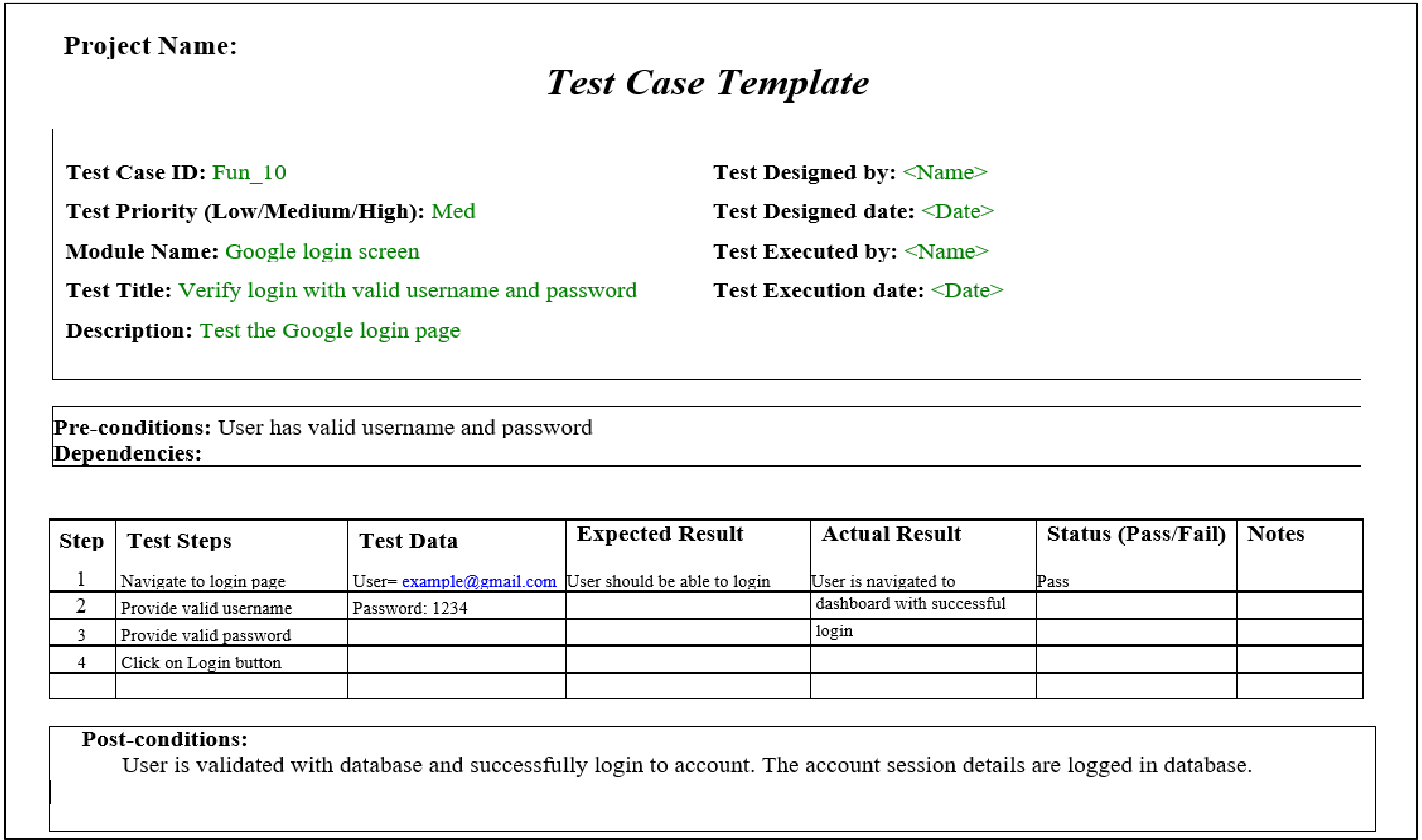
TABLE 13-3 Table of Contents of a Master Test Plan

1. Introduction
   a. Description of system to be tested
   b. Objectives of the test plan
   c. Method of testing
   d. Supporting documents
2. Overall Plan
   a. Milestones, schedules, and locations
   b. Test materials
      i. Test plans
      ii. Test cases
      iii. Test scenarios
      iv. Test log
   c. Criteria for passing tests
3. Testing Requirements
   a. Hardware
   b. Software
   c. Personnel
4. Procedure Control
   a. Test initiation
   b. Test execution
   c. Test failure
   d. Access/change control
   e. Document control
5. Test-Specific or Component-Specific Test Plans
   a. Objectives
   b. Software description
   c. Method
   d. Milestones, schedule, progression, and locations
   e. Requirements
   f. Criteria for passing tests
   g. Resulting test materials
   h. Execution control
   i. Attachments

Test Case

- Test cases involve a set conditions and inputs that can be used while performing testing tasks.
- Its main intent is to ensure whether a software passes or fails in terms of its functionality and other aspects.
- Test cases are written to keep track of the testing coverage of a software.
- There are many types of test cases such as functional, negative, logical test cases, physical test cases, UI test cases, etc.

Test Case Template



Project Name:

Test Case Template

Test Case ID: Fun_10

Test Priority (Low/Medium/High): Med

Module Name: Google login screen

Test Title: Verify login with valid username and password

Description: Test the Google login page

Test Designed by: <Name>

Test Designed date: <Date>

Test Executed by: <Name>

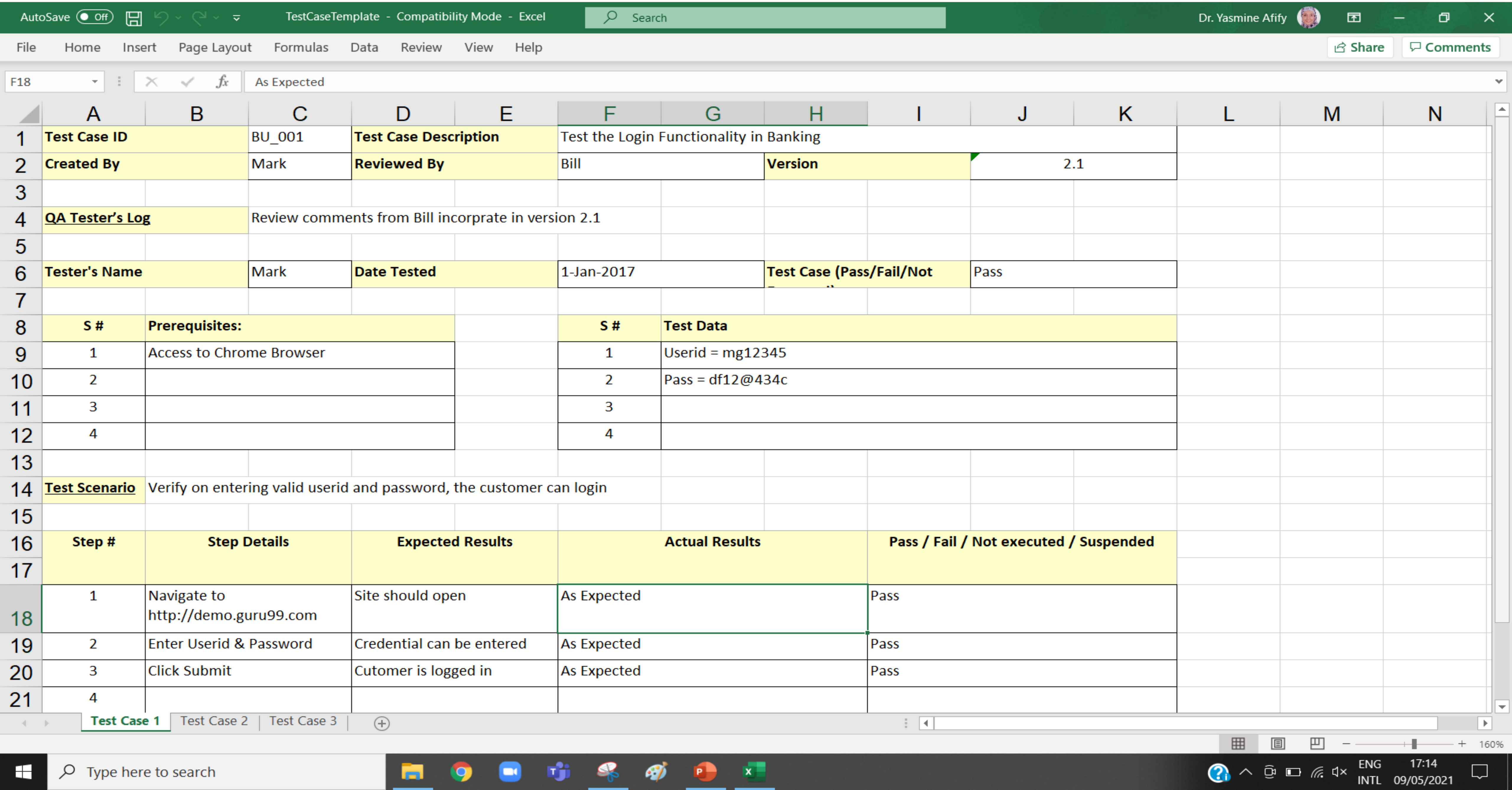
Test Execution date: <Date>

Pre-conditions: User has valid username and password
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to login page	User= example@gmail.com	User should be able to login	User is navigated to	Pass	
2	Provide valid username	Password: 1234		dashboard with successful		
3	Provide valid password			login		
4	Click on Login button					

Post-conditions:
User is validated with database and successfully login to account. The account session details are logged in database.

Test Case Template



AutoSave Off
TestCaseTemplate - Compatibility Mode - Excel
Search
Dr. Yasmine Afify
File Home Insert Page Layout Formulas Data Review View Help
Share Comments
F18
As Expected
Test Case 1 Test Case 2 Test Case 3
160%
Type here to search
ENG INTL 17:14 09/05/2021

Test Case ID	BU_001	**Test Case Description**	Test the Login Functionality in Banking		
Created By	Mark	**Reviewed By**	Bill	**Version**	2.1

QA Tester's Log Review comments from Bill incorprate in version 2.1

Tester's Name	Mark	**Date Tested**	1-Jan-2017	**Test Case (Pass/Fail/Not**	Pass

S #	Prerequisites:
1	Access to Chrome Browser
2	
3	
4	

S #	Test Data
1	Userid = mg12345
2	Pass = df12@434c
3	
4	

Test Scenario Verify on entering valid userid and password, the customer can login

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://demo.guru99.com	Site should open	As Expected	Pass
2	Enter Userid & Password	Credential can be entered	As Expected	Pass
3	Click Submit	Cutomer is logged in	As Expected	Pass
4				

Bug Report Template

ID number	#123
Name	CART - Unable to add new item to my cart
Reporter	Mike A
Submit Date	03/04/2016
Summary	When my cart contains one item, I am unable to add a second item via the add to cart button on a product page
URL	www.example.com/product/abc
Screenshot	www.example.com/screenshot123
Platform	Macintosh
Operating System	OS X 10.12.0
Browser	Chrome 53
Severity	Major
Assigned to	/
Priority	High

Description
When my cart contains one item, I am unable to add a second item via the add to cart button on a product page

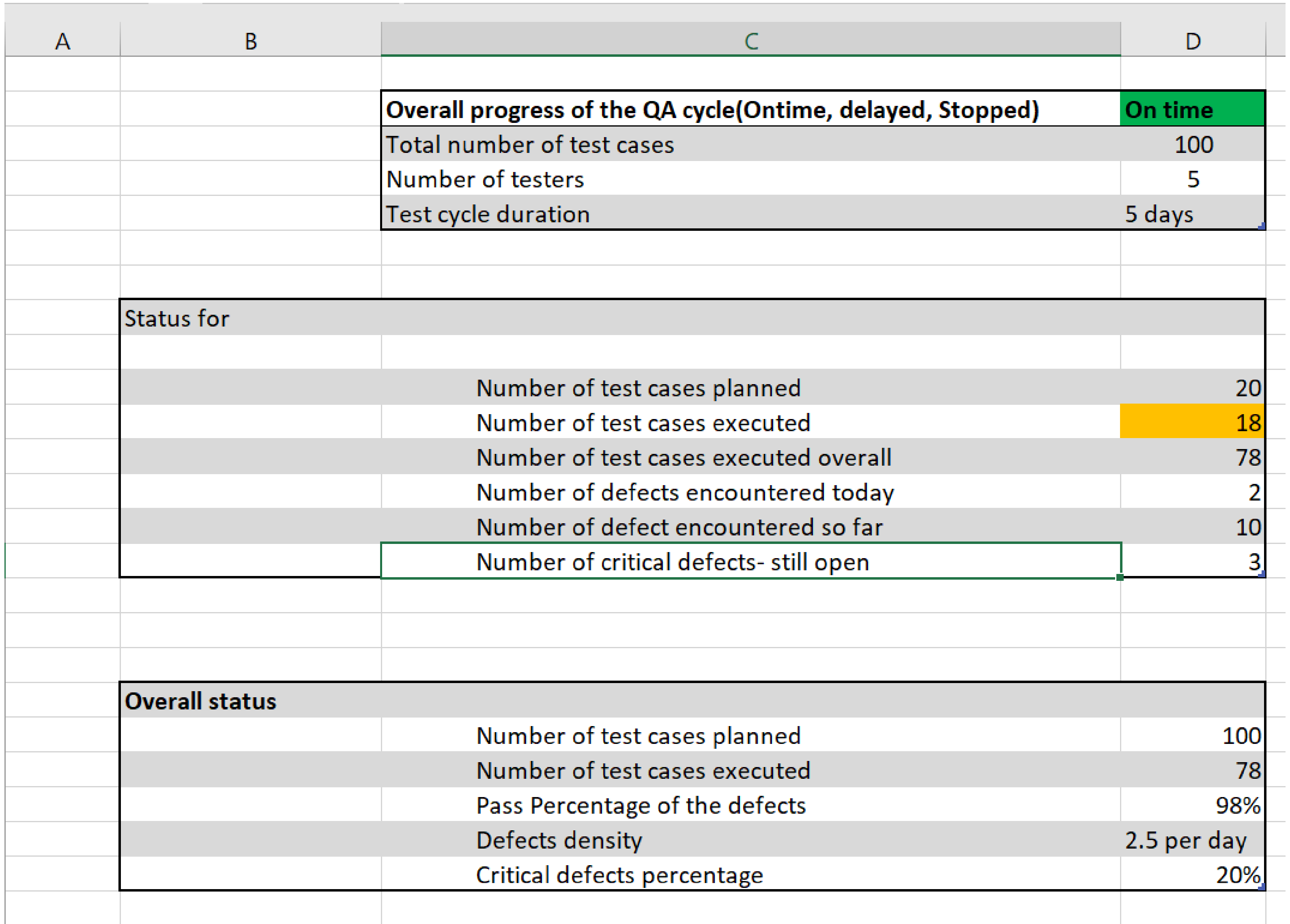
Steps to reproduce
> add one item to cart
> go to product abc via the search bar
> add new item to cart via "add to cart" button (see screenshot)
> go to cart

Expected result
The cart should contain 2 items

Actual result
The cart contains only 1 item

Notes

Sample Status/Progress Report



A	B	C	D
		Overall progress of the QA cycle(Ontime, delayed, Stopped)	**On time**
		Total number of test cases	100
		Number of testers	5
		Test cycle duration	5 days
	Status for		
		Number of test cases planned	20
		Number of test cases executed	18
		Number of test cases executed overall	78
		Number of defects encountered today	2
		Number of defect encountered so far	10
		Number of critical defects- still open	3
	Overall status		
		Number of test cases planned	100
		Number of test cases executed	78
		Pass Percentage of the defects	98%
		Defects density	2.5 per day
		Critical defects percentage	20%

Test Closure Report

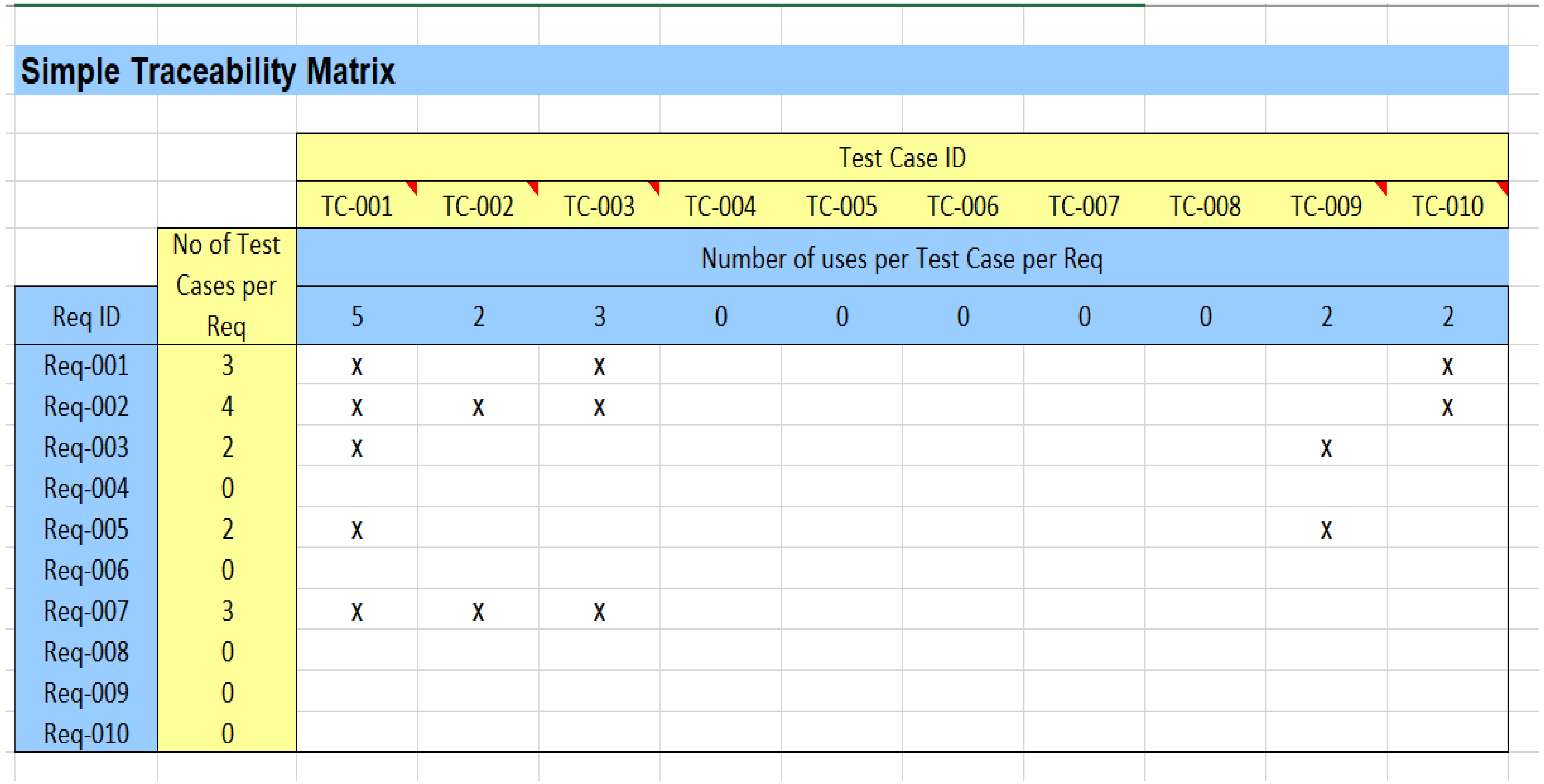
- It is a **report** that is created once the **testing phase** is **successfully completed** by meeting **exit criteria** defined for the project.
- It is a **document** that gives a **summary** of all the **tests conducted**.
- It also gives a **detailed analysis** of the **bugs removed** and **errors found**.
- It also presents the list of **known issues**.
- It is **created** by **test Lead**, **reviewed** by various **stake holders** like **test architect**, **test manager**, **business analyst**, **project manager** and finally approved by **clients**.

Requirements Traceability Matrix (RTM)

- **Requirements Traceability Matrix** (RTM) is a document that connects requirements throughout the validation process.
- Used by the validation team to ensure that requirements are not lost during the validation project.

- It shows the **relationship** between requirements and test cases to ask such question as:
  - Which test cases may be affected by change of requirement?
  - How much coverage of requirements has been achieved?
  - Are any requirements overly complex and need too many test cases?

RTM

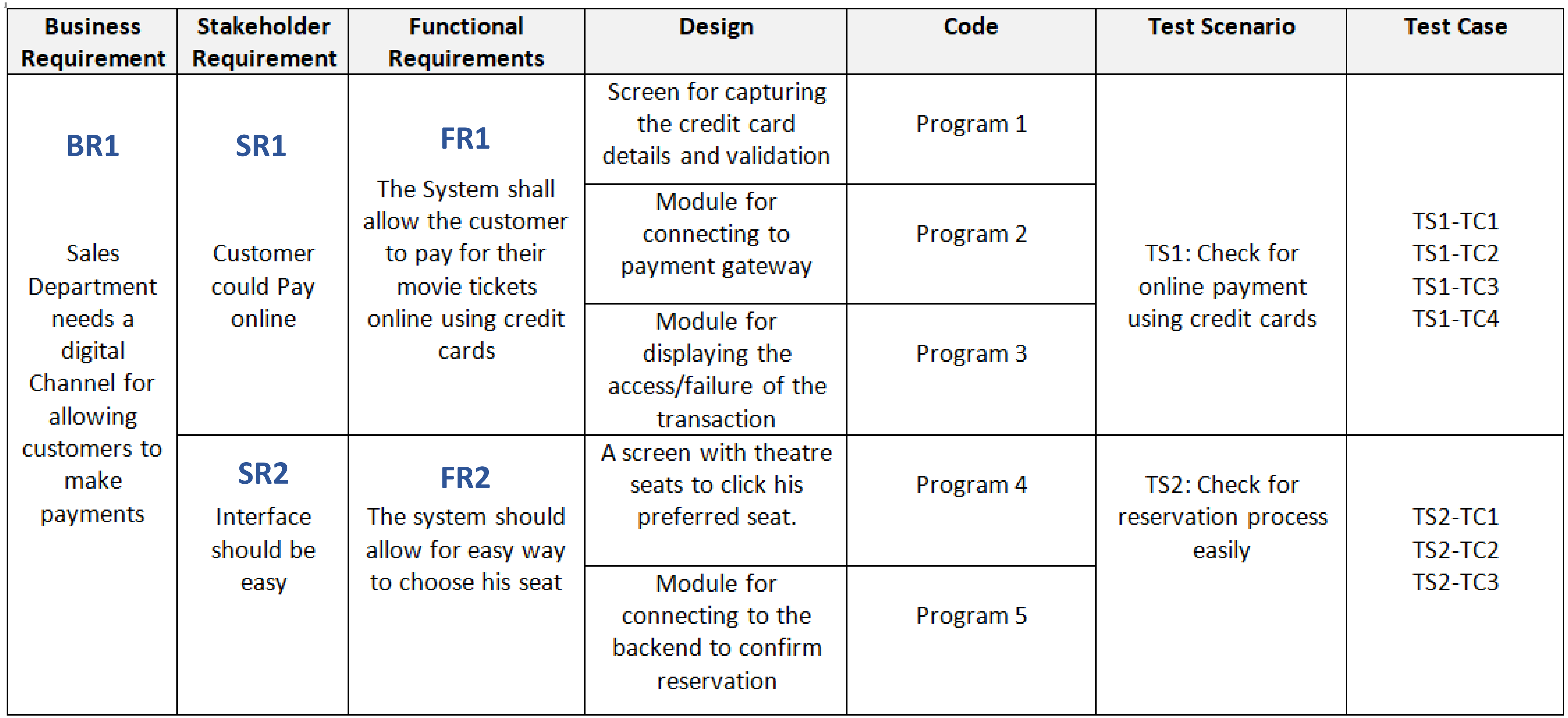


Simple Traceability Matrix

		Test Case ID									
		TC-001	TC-002	TC-003	TC-004	TC-005	TC-006	TC-007	TC-008	TC-009	TC-010
	No of Test Cases per Req	Number of uses per Test Case per Req									
Req ID		5	2	3	0	0	0	0	0	2	2
Req-001	3	X		X							X
Req-002	4	X	X	X							X
Req-003	2	X								X	
Req-004	0										
Req-005	2	X								X	
Req-006	0										
Req-007	3	X	X	X							
Req-008	0										
Req-009	0										
Req-010	0										

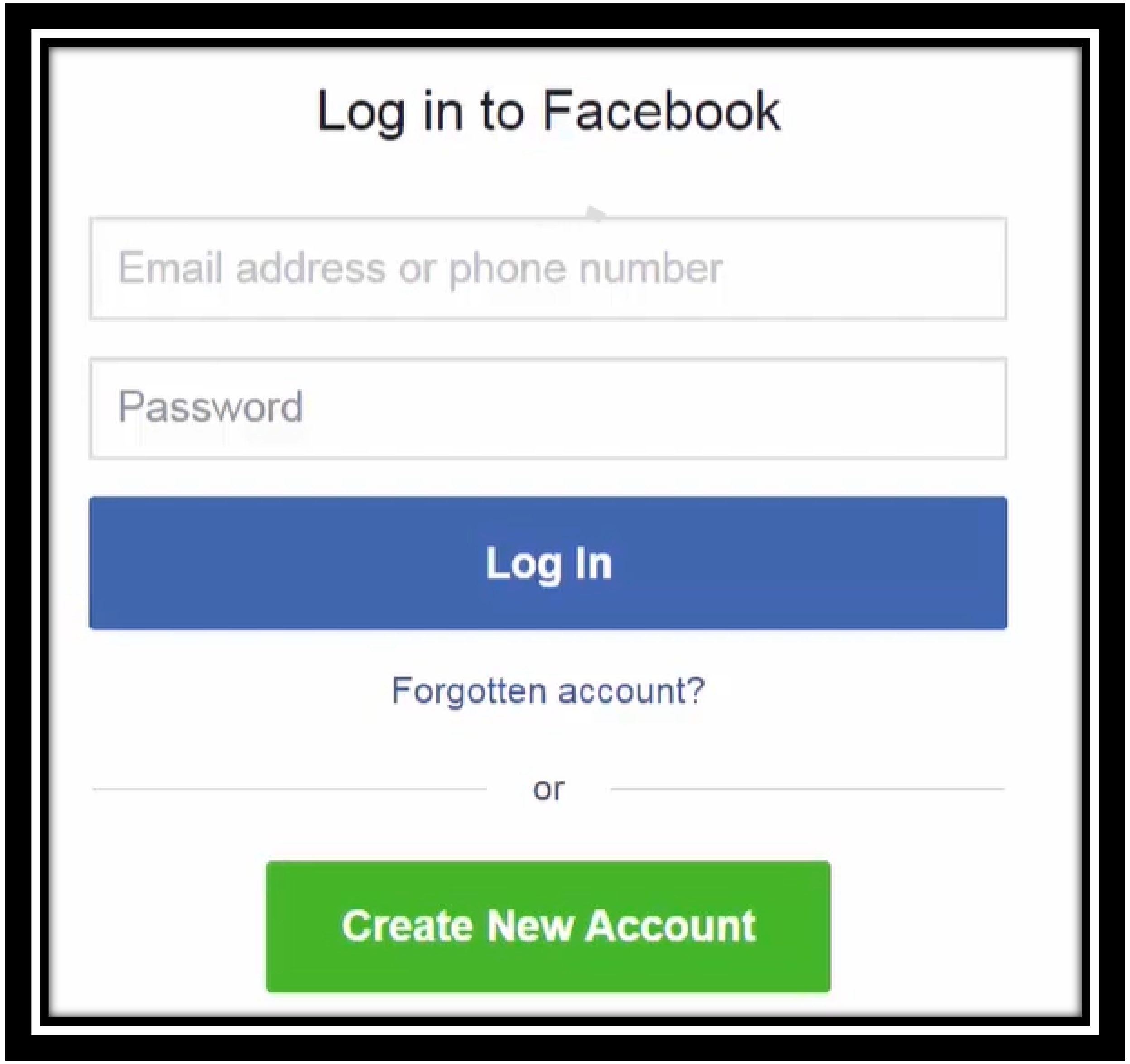
RTM

- It is a matrix is used to trace requirements. It provides forward and backward traceability.

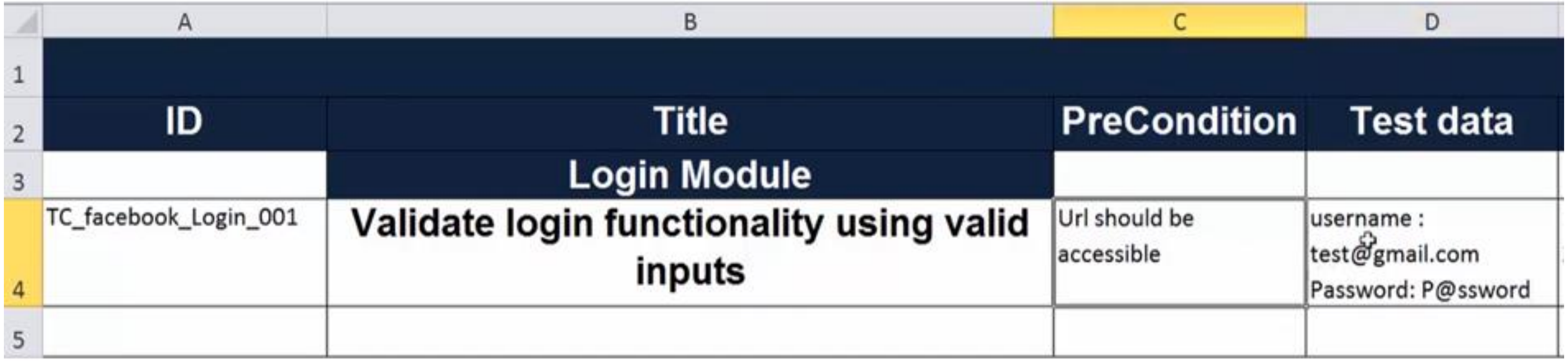


Business Requirement	Stakeholder Requirement	Functional Requirements	Design	Code	Test Scenario	Test Case
BR1 Sales Department needs a digital Channel for allowing customers to make payments	**SR1** Customer could Pay online	**FR1** The System shall allow the customer to pay for their movie tickets online using credit cards	Screen for capturing the credit card details and validation	Program 1	TS1: Check for online payment using credit cards	TS1-TC1 TS1-TC2 TS1-TC3 TS1-TC4
			Module for connecting to payment gateway	Program 2		
			Module for displaying the access/failure of the transaction	Program 3		
	SR2 Interface should be easy	**FR2** The system should allow for easy way to choose his seat	A screen with theatre seats to click his preferred seat.	Program 4	TS2: Check for reservation process easily	TS2-TC1 TS2-TC2 TS2-TC3
			Module for connecting to the backend to confirm reservation	Program 5		

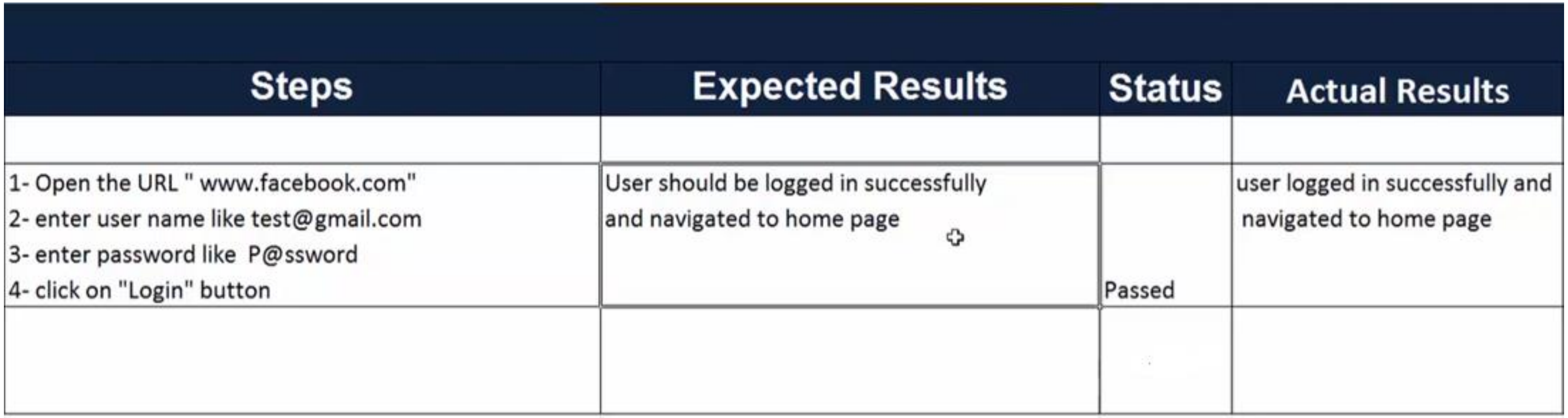
Practical Example
Test Case Writing



TEST CASE

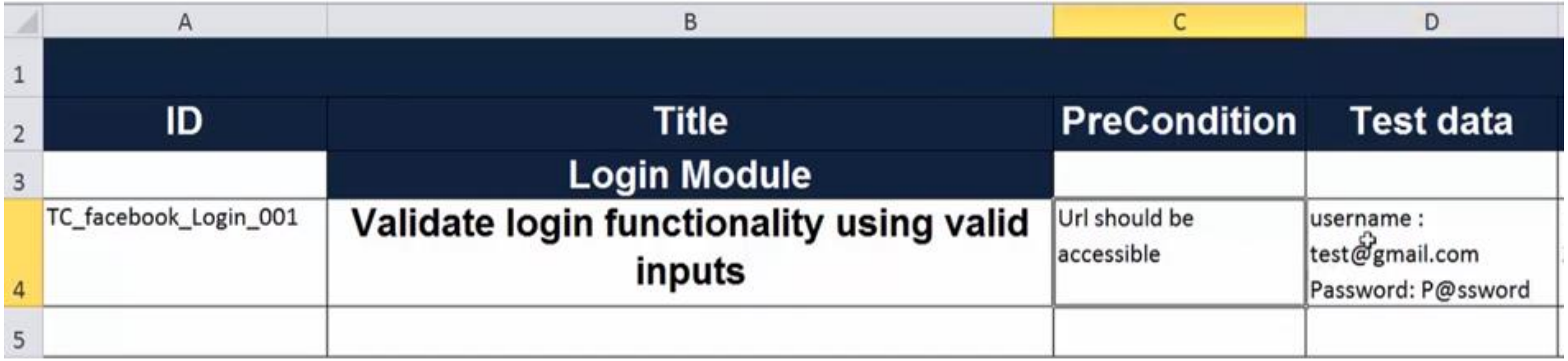


ID	Title	PreCondition	Test data
	Login Module		
TC_facebook_Login_001	Validate login functionality using valid inputs	Url should be accessible	username : test@gmail.com Password: P@ssword

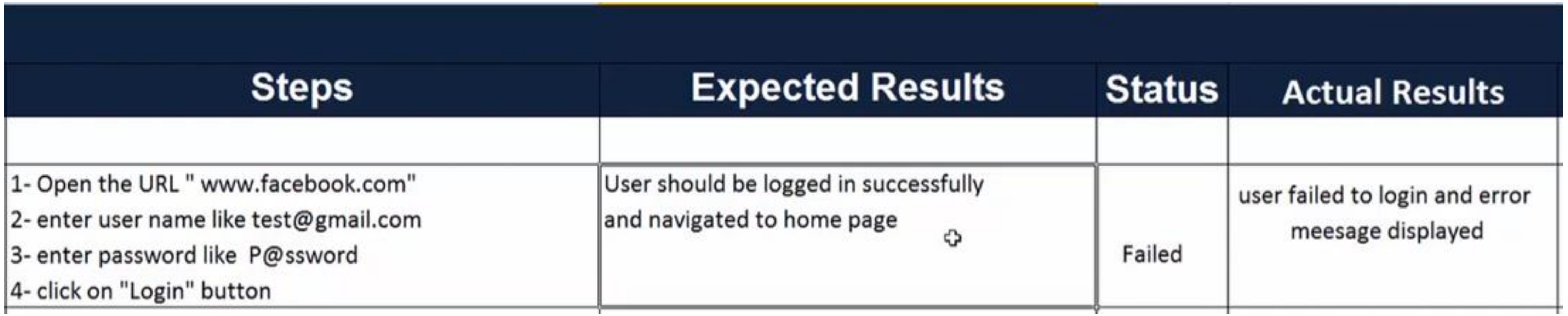


Steps	Expected Results	Status	Actual Results
1- Open the URL " www.facebook.com" 2- enter user name like test@gmail.com 3- enter password like P@ssword 4- click on "Login" button	User should be logged in successfully and navigated to home page	Passed	user logged in successfully and navigated to home page

TEST CASE

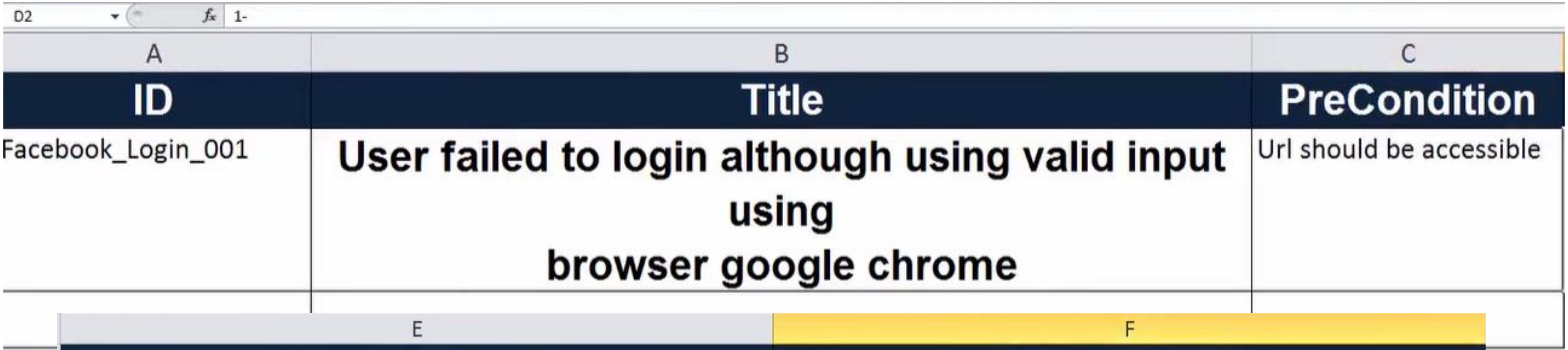


	A	B	C	D
1				
2	ID	Title	PreCondition	Test data
3		Login Module		
4	TC_facebook_Login_001	Validate login functionality using valid inputs	Url should be accessible	username : test@gmail.com Password: P@ssword
5				



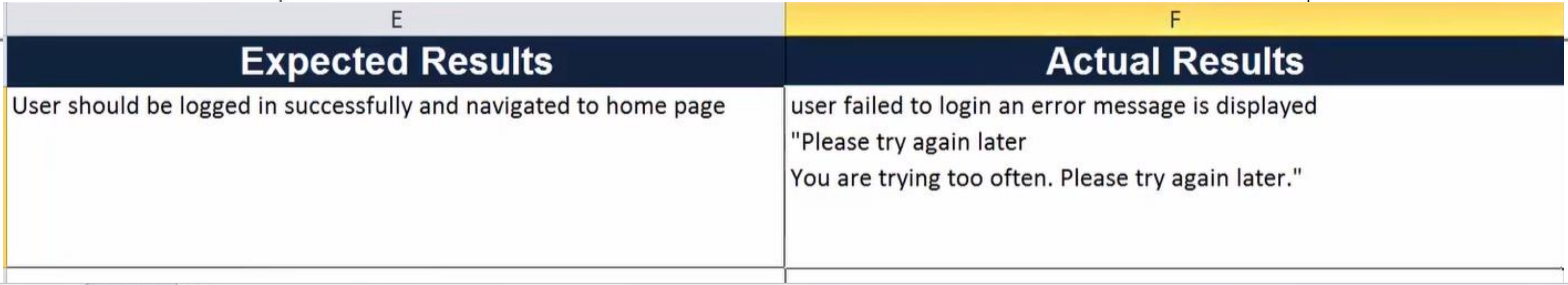
Steps	Expected Results	Status	Actual Results
1- Open the URL " www.facebook.com" 2- enter user name like test@gmail.com 3- enter password like P@ssword 4- click on "Login" button	User should be logged in successfully and navigated to home page	Failed	user failed to login and error meesage displayed

BUG REPORT

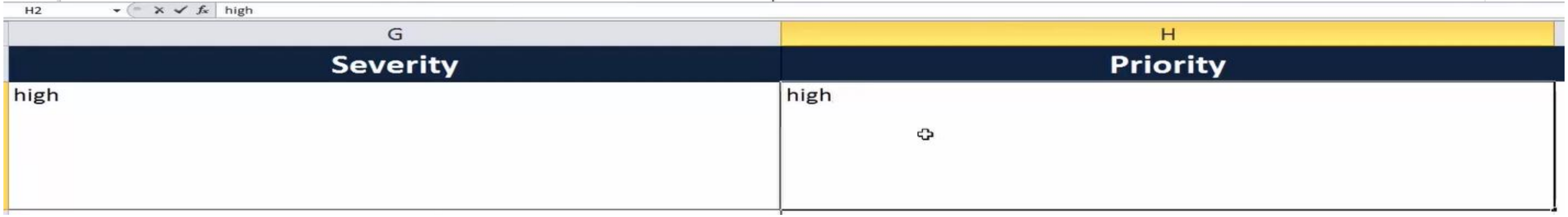


D2 1-

ID	Title	PreCondition
Facebook_Login_001	User failed to login although using valid input using browser google chrome	Url should be accessible



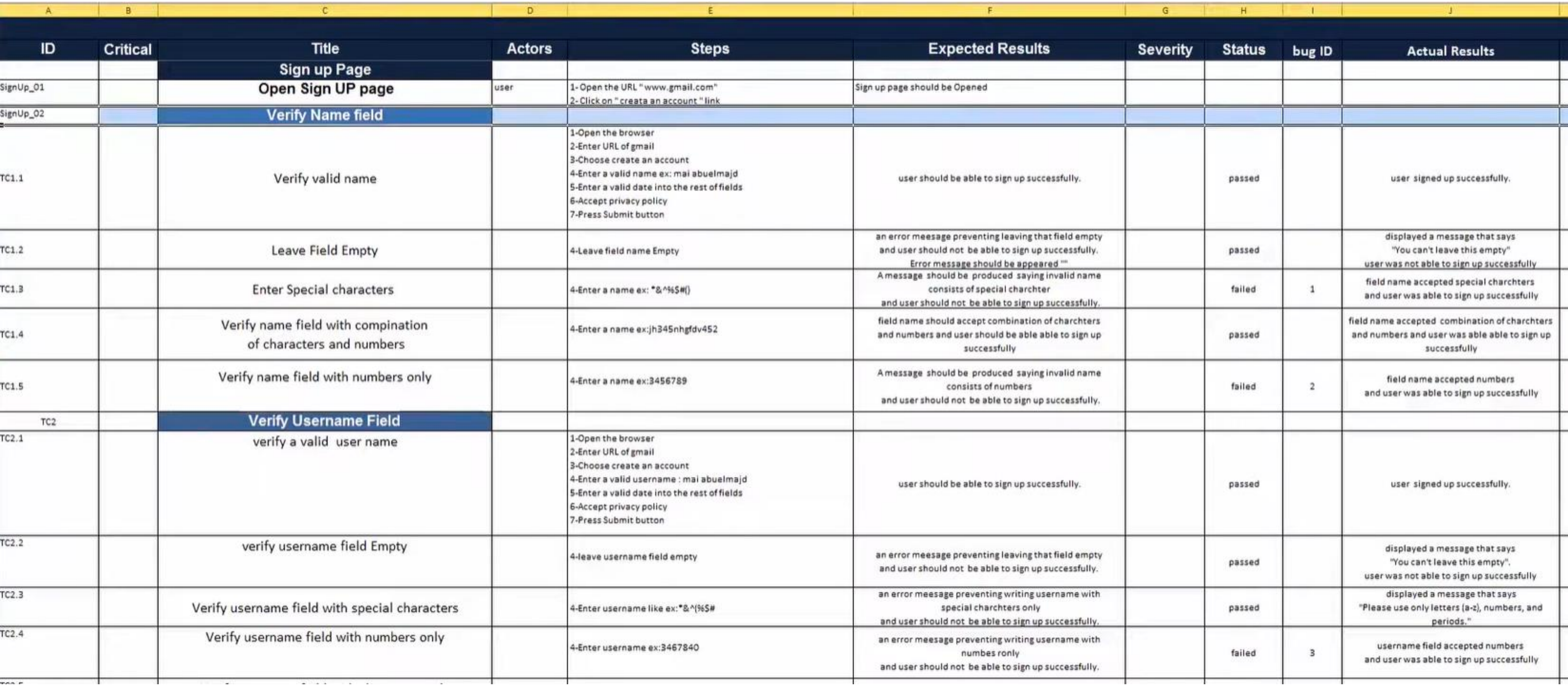
Expected Results	Actual Results
User should be logged in successfully and navigated to home page	user failed to login an error message is displayed "Please try again later You are trying too often. Please try again later."



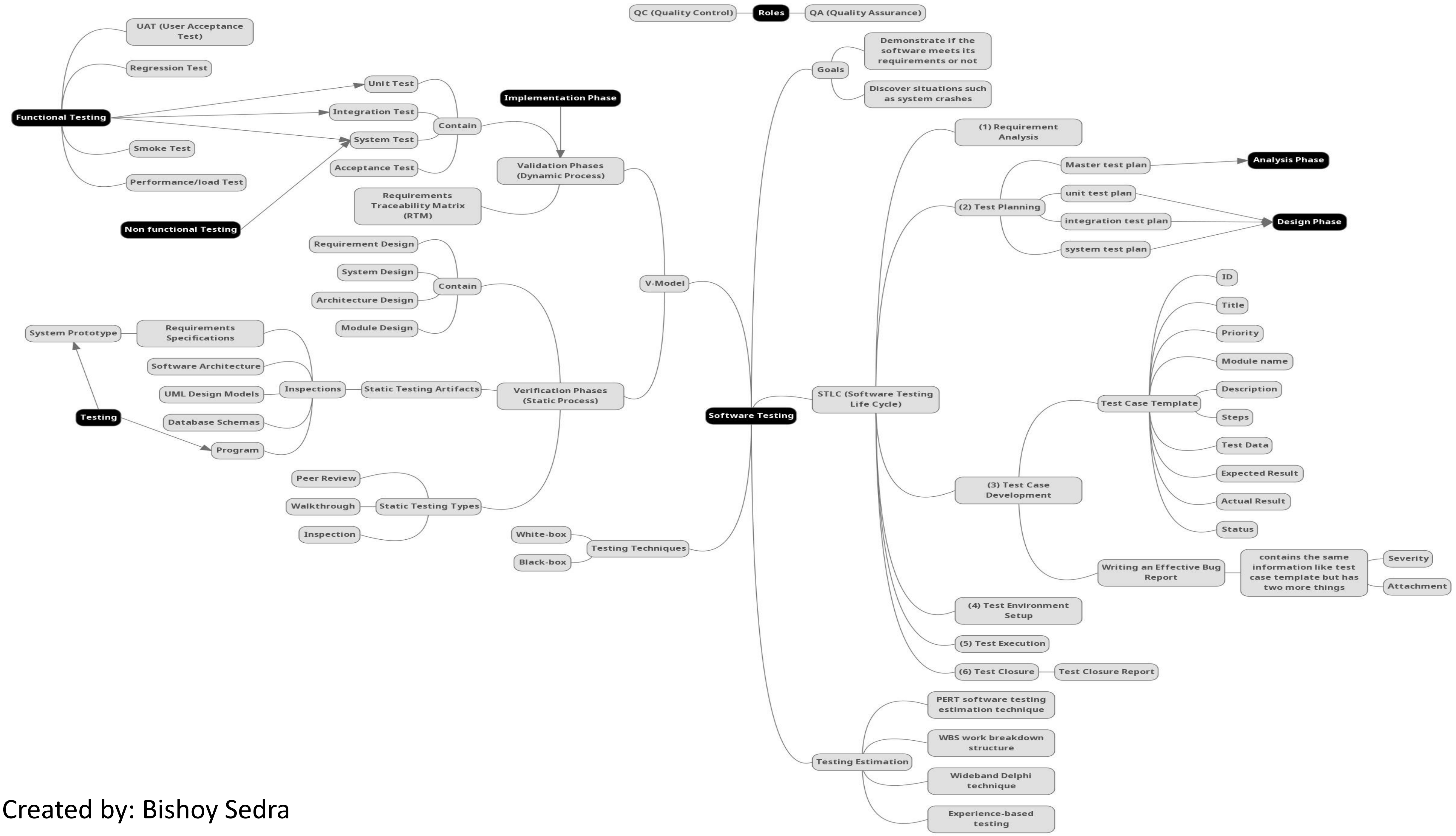
H2 high

Severity	Priority
high	high

Example Test Cases for Sign-up Page



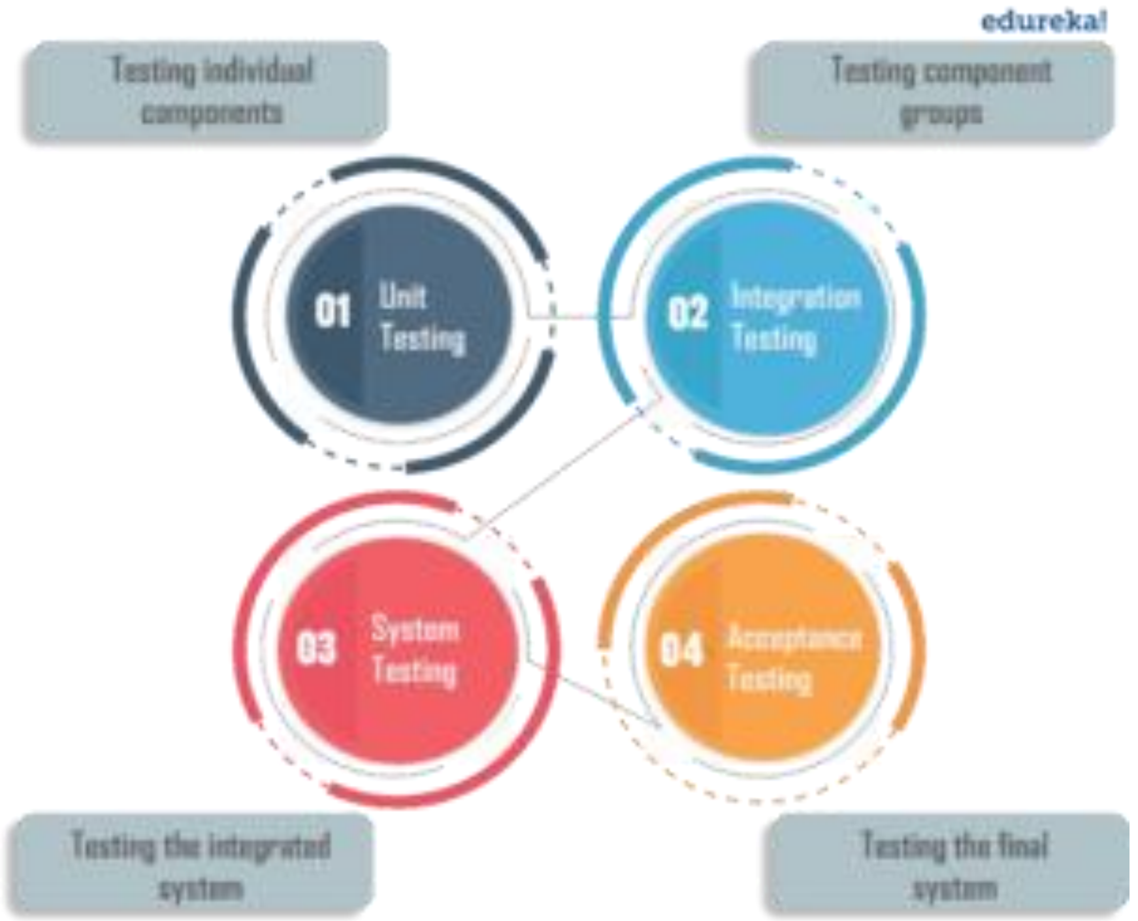
A	B	C	D	E	F	G	H	I	J
ID	Critical	Title	Actors	Steps	Expected Results	Severity	Status	bug ID	Actual Results
		Sign up Page							
SignUp_01		Open Sign UP page	user	1- Open the URL " www.gmail.com" 2- Click on " creata an account " link	Sign up page should be Opened				
SignUp_02		Verify Name field							
TC1.1		Verify valid name		1-Open the browser 2-Enter URL of gmail 3-Choose create an account 4-Enter a valid name ex: mai abuelmajd 5-Enter a valid date into the rest of fields 6-Accept privacy policy 7-Press Submit button	user should be able to sign up successfully.		passed		user signed up successfully.
TC1.2		Leave Field Empty		4-Leave field name Empty	an error meesage preventing leaving that field empty and user should not be able to sign up successfully. Error message should be appeared ""		passed		displayed a message that says "You can't leave this empty" user was not able to sign up successfully
TC1.3		Enter Special characters		4-Enter a name ex: *&^%$#()	A message should be produced saying invalid name consists of special charchter and user should not be able to sign up successfully.		failed	1	field name accepted special charchters and user was able to sign up successfully
TC1.4		Verify name field with compination of characters and numbers		4-Enter a name ex:jh345nhgfdv452	field name should accept combination of charchters and numbers and user should be able able to sign up successfully		passed		field name accepted combination of charchters and numbers and user was able able to sign up successfully
TC1.5		Verify name field with numbers only		4-Enter a name ex:3456789	A message should be produced saying invalid name consists of numbers and user should not be able to sign up successfully.		failed	2	field name accepted numbers and user was able to sign up successfully
TC2		Verify Username Field							
TC2.1		verify a valid user name		1-Open the browser 2-Enter URL of gmail 3-Choose create an account 4-Enter a valid username : mai abuelmajd 5-Enter a valid date into the rest of fields 6-Accept privacy policy 7-Press Submit button	user should be able to sign up successfully.		passed		user signed up successfully.
TC2.2		verify username field Empty		4-leave username field empty	an error meesage preventing leaving that field empty and user should not be able to sign up successfully.		passed		displayed a message that says "You can't leave this empty". user was not able to sign up successfully
TC2.3		Verify username field with special characters		4-Enter username like ex:*&^(%$#	an error meesage preventing writing username with special charchters only and user should not be able to sign up successfully.		passed		displayed a message that says "Please use only letters (a-z), numbers, and periods."
TC2.4		Verify username field with numbers only		4-Enter username ex:3467840	an error meesage preventing writing username with numbes ronly and user should not be able to sign up successfully.		failed	3	username field accepted numbers and user was able to sign up successfully



Created by: Bishoy Sedra

References

- https://estb.org.eg/English/Pages/Resources/ISTQBSyllabi.aspx
- https://www.testorigen.com/writing-an-effective-bug-report-for-a-better-software-development-process/
- https://www.edureka.co/blog/what-is-integration-testing-a-simple-guide-on-how-to-perform-integration-testing/


edureka!
Testing individual components
Testing component groups
01 Unit Testing
02 Integration Testing
03 System Testing
04 Acceptance Testing
Testing the integrated system
Testing the final system