Capstone Project – End-to-End Quality Engineering Implementation

A comprehensive assessment designed to evaluate your ability to apply software testing, automation frameworks, database validation, and quality engineering best practices in a simulated real-world environment. This capstone bridges the gap between theoretical knowledge and practical application, preparing you for professional QA and QE roles.

What is the Capstone Project?

The capstone project is a comprehensive, end-to-end quality engineering assessment that simulates a real-world software testing engagement. You will work on a web-based e-commerce application, applying manual testing, test automation, database validation, and quality engineering principles throughout the software testing lifecycle.

Real-World Simulation

Work on an e-commerce application mirroring actual industry projects

Full STLC Coverage

Execute all phases from requirement analysis through test closure

Hands-On Assessment

Demonstrate practical skills through deliverables and execution

Objectives of the Capstone

Primary Goals

Assess your ability to independently plan, design, execute, and automate testing activities for a complete application lifecycle. Demonstrate proficiency in transitioning from manual testing to automation whilst maintaining quality engineering rigour.

Key Objectives

- Apply test design techniques to real requirements
- Execute manual testing with proper documentation
- Design and implement automation frameworks
- Perform database validation and backend testing
- Generate professional test reports and metrics
- Demonstrate quality engineering mindset

Skills and Competencies Evaluated

This capstone assesses your readiness for professional QA, automation, and quality engineering roles by evaluating technical skills, analytical thinking, and quality ownership.

Manual Testing Proficiency

Requirement analysis, test design techniques, test case creation, execution tracking, and defect management with proper severity and priority assignment.

Automation Expertise

Framework design, Selenium WebDriver implementation, TestNG usage, data-driven testing, and Maven build configuration with proper code structure.

Database Validation

SQL query writing, CRUD operations validation, join-based queries, and UI-to-database data verification to ensure backend integrity.

Quality Engineering Mindset

Risk-based testing, shift-left and shift-right approaches, automation-first thinking, and ownership of quality throughout the lifecycle.

Tools and Technologies Used

Selenium WebDriver

Browser automation and web element interaction for functional test automation

TestNG

Test execution framework with annotations, assertions, and reporting capabilities

Maven

Build automation and dependency management for project structure

SQL

Database validation and backend data verification through queries

Java

Programming language for automation scripts with OOP principles

Excel / Sheets

Test case documentation, test data management, and defect tracking

Mapping to Real Project Environments

The capstone project mirrors actual industry practices and project structures, ensuring that skills developed are directly transferable to professional QA and QE roles.

1

Manual Testing

Test planning, scenario design, execution tracking, and defect lifecycle management as practised in real QA teams

2

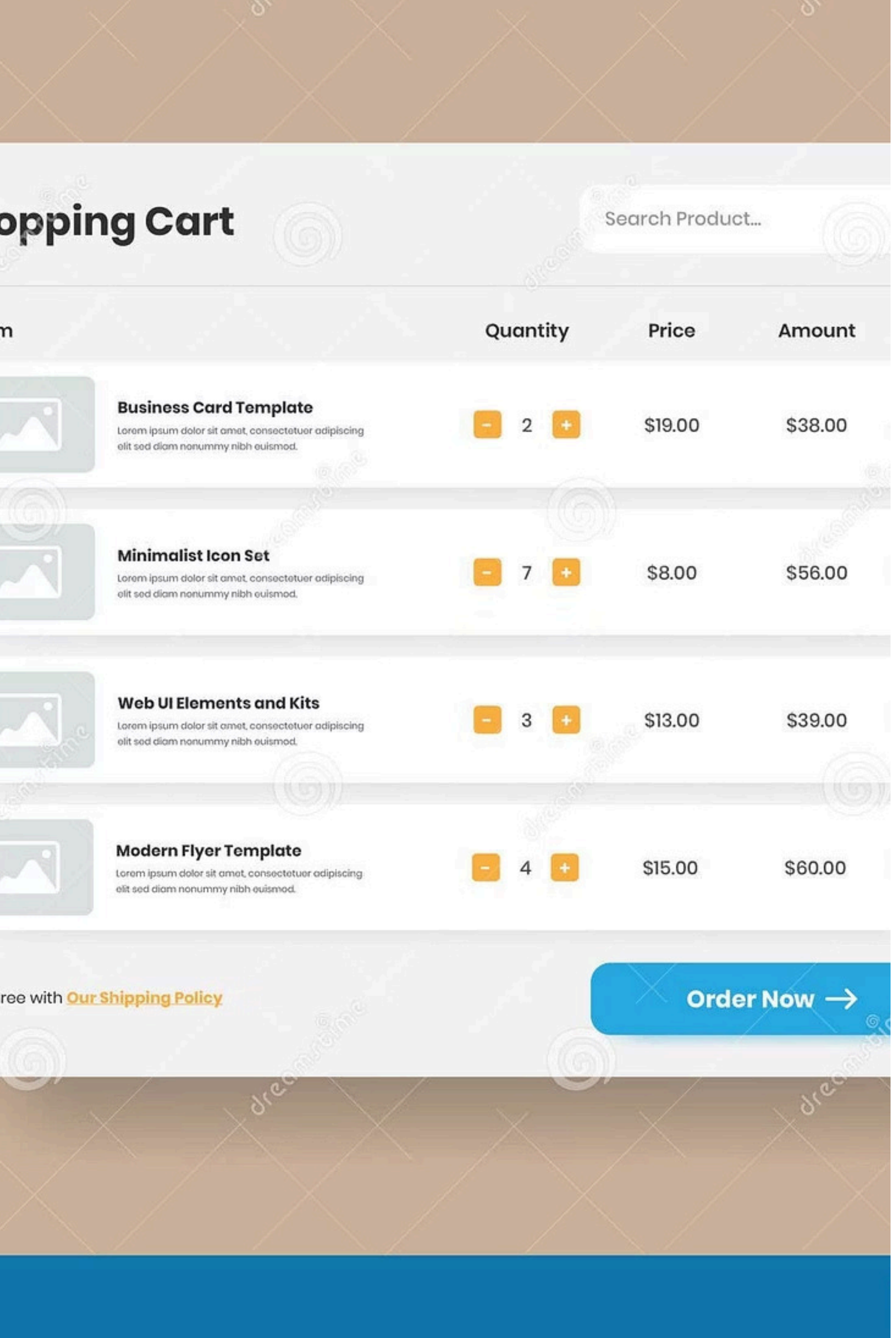
Test Automation

Framework architecture, page object model, and reusable components matching industry automation standards

3

Quality Engineering

Shift-left practices, risk-based approach, and quality ownership throughout the development lifecycle


opping Cart

Search Product...

m	Quantity	Price	Amount
Business Card Template Lorem ipsum dolor sit amet, consectetuer adipiscing elit sed diam nonummy nibh euismod.	- 2 +	$19.00	$38.00
Minimalist Icon Set Lorem ipsum dolor sit amet, consectetuer adipiscing elit sed diam nonummy nibh euismod.	- 7 +	$8.00	$56.00
Web UI Elements and Kits Lorem ipsum dolor sit amet, consectetuer adipiscing elit sed diam nonummy nibh euismod.	- 3 +	$13.00	$39.00
Modern Flyer Template Lorem ipsum dolor sit amet, consectetuer adipiscing elit sed diam nonummy nibh euismod.	- 4 +	$15.00	$60.00

ree with Our Shipping Policy

APPLICATION UNDER TEST

Application Overview

The application under test is a fully functional web-based e-commerce platform designed for retail operations. It simulates a real-world online shopping experience with multiple user workflows, transaction processing, and data management capabilities. The application provides comprehensive coverage of typical e-commerce functionalities, making it ideal for demonstrating end-to-end testing skills.

Business Objectives of the Application

Core Business Goals

The e-commerce application aims to provide customers with a seamless online shopping experience whilst enabling the business to manage inventory, process orders efficiently, and maintain customer relationships. The platform must ensure transaction accuracy, data security, and system reliability.

Customer Acquisition

Attract and onboard new users through intuitive registration

Sales Conversion

Enable smooth product discovery and purchase completion

Customer Retention

Provide order tracking and purchase history for repeat business

Functional Modules

The application comprises six core functional modules that cover the complete user journey from registration through order fulfilment.

User Registration & Login

Account creation with validation, user authentication, password management, and session handling

Product Search & Listing

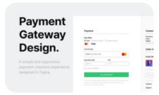
Product catalogue browsing, search functionality with filters, category navigation, and product detail views

Cart Management

Add to cart, quantity updates, item removal, price calculations, and cart persistence across sessions

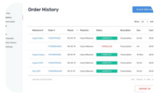
Order Placement

Checkout workflow, shipping address entry, order review, and order confirmation with order ID generation



Payment Processing

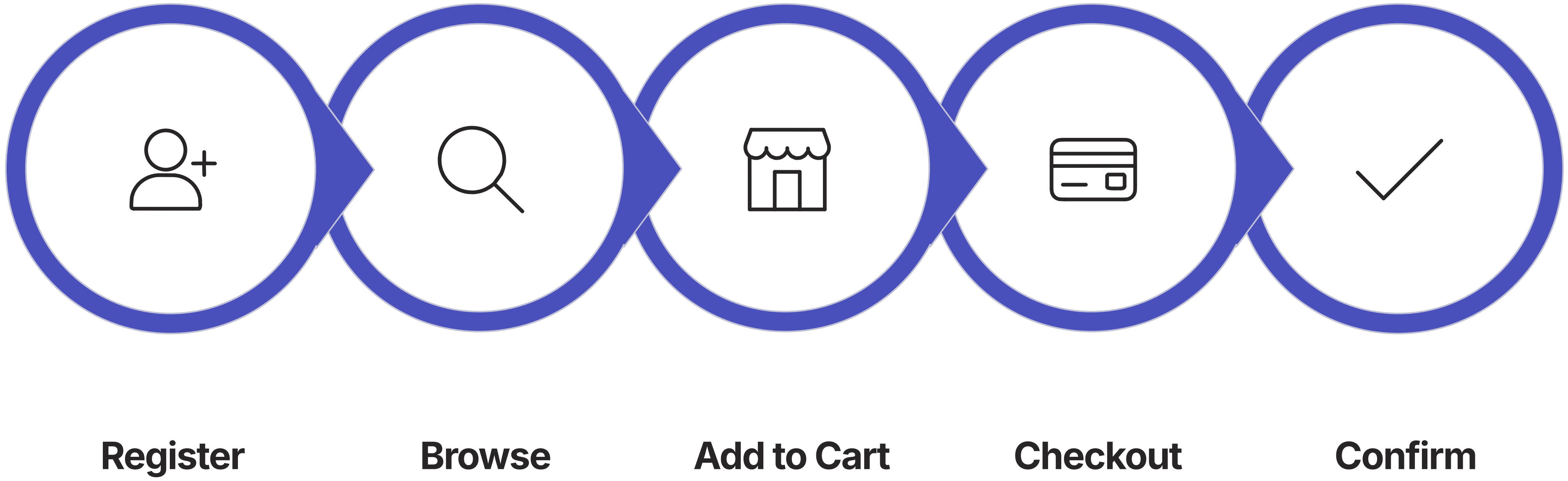
Simulated payment gateway integration with multiple payment methods and transaction status handling



Order History & Tracking

View past orders, track current order status, access order details, and download invoices

Application Workflow Diagram



This high-level workflow illustrates the primary user journey through the application, from initial registration through order tracking. Each step represents a critical testing area requiring both functional validation and automation coverage.

PROJECT SCOPE

In-Scope Activities

The following activities are within the scope of the capstone project and must be completed as part of the assessment.

1

Test Planning

Requirement analysis, test strategy definition, scope identification, and risk assessment

2

Test Design

Test scenario creation, test case writing using design techniques, and test data preparation

3

Manual Execution

Test case execution, status tracking, evidence collection, and defect logging

4

Test Automation

Framework design, script development, data-driven testing, and execution from Maven

5

Database Validation

SQL query writing, CRUD validation, join-based queries, and UI-DB comparison

6

Reporting & Closure

Test metrics analysis, summary report generation, and test closure documentation

Out-of-Scope Activities

Not Included

The following activities are explicitly excluded from the capstone project scope to maintain focus on core QA and automation competencies.

- Performance testing and load testing activities
- Security testing and penetration testing
- Mobile application testing (focus is web-based only)
- API testing and web services validation
- Continuous integration and deployment setup
- Production deployment and release activities
- User acceptance testing coordination
- Test environment setup and infrastructure management

Assumptions and Constraints

Assumptions

- Application is stable and accessible in test environment
- Test data can be created and modified as needed
- All required tools and access are provided
- Functional requirements are documented and available
- Payment gateway is simulated (no real transactions)

Constraints

- Project timeline is fixed per assessment schedule
- Scope is limited to defined functional modules
- Testing is limited to Chrome browser only
- Database access is read-only for validation
- No modification of application source code permitted

PHASE 1

Requirement Analysis & Test Planning

The first phase establishes the foundation for all subsequent testing activities. You will analyse functional requirements, identify testable components, assess risks, and define a comprehensive test strategy that balances manual and automated testing approaches.

Requirement Understanding Approach

01

Requirement Gathering

Review functional specifications, user stories, and business requirements documentation

02

Requirement Analysis

Identify testable requirements, ambiguities, and areas requiring clarification

03

Test Scope Definition

Determine in-scope modules, features, and functionalities for testing coverage

04

Risk Assessment

Identify high-risk areas requiring focused testing attention and early validation

Test Strategy: Manual + Automation

A balanced test strategy combines manual testing for exploratory scenarios and user experience validation with automation for regression testing and repetitive workflows.

Manual Testing Strategy

- Initial exploratory testing of new features
- User experience and usability validation
- Complex business logic scenarios
- Ad-hoc and negative testing
- Payment workflow simulation

Automation Strategy

- End-to-end user journeys (login to checkout)
- Regression test suite for core functionality
- Data-driven testing for multiple data sets
- Cross-module integration scenarios
- Smoke test suite for build verification

Test Levels and Test Types

Test Levels

- **Component Testing:** Individual module validation
- **Integration Testing:** Module interaction verification
- **System Testing:** End-to-end workflow validation
- **Regression Testing:** Impact analysis of changes

Test Types

- **Functional Testing:** Feature behaviour validation
- **UI Testing:** Interface and layout verification
- **Database Testing:** Backend data integrity
- **Negative Testing:** Error handling validation

Phase 1: Mapped Topics

Software Testing Fundamentals

Understanding of testing principles, objectives, and the role of testing in software quality assurance

SDLC vs STLC

Software Development Lifecycle phases and how Software Testing Lifecycle integrates within each phase

QA vs QE

Distinction between Quality Assurance (process focus) and Quality Engineering (product focus) approaches

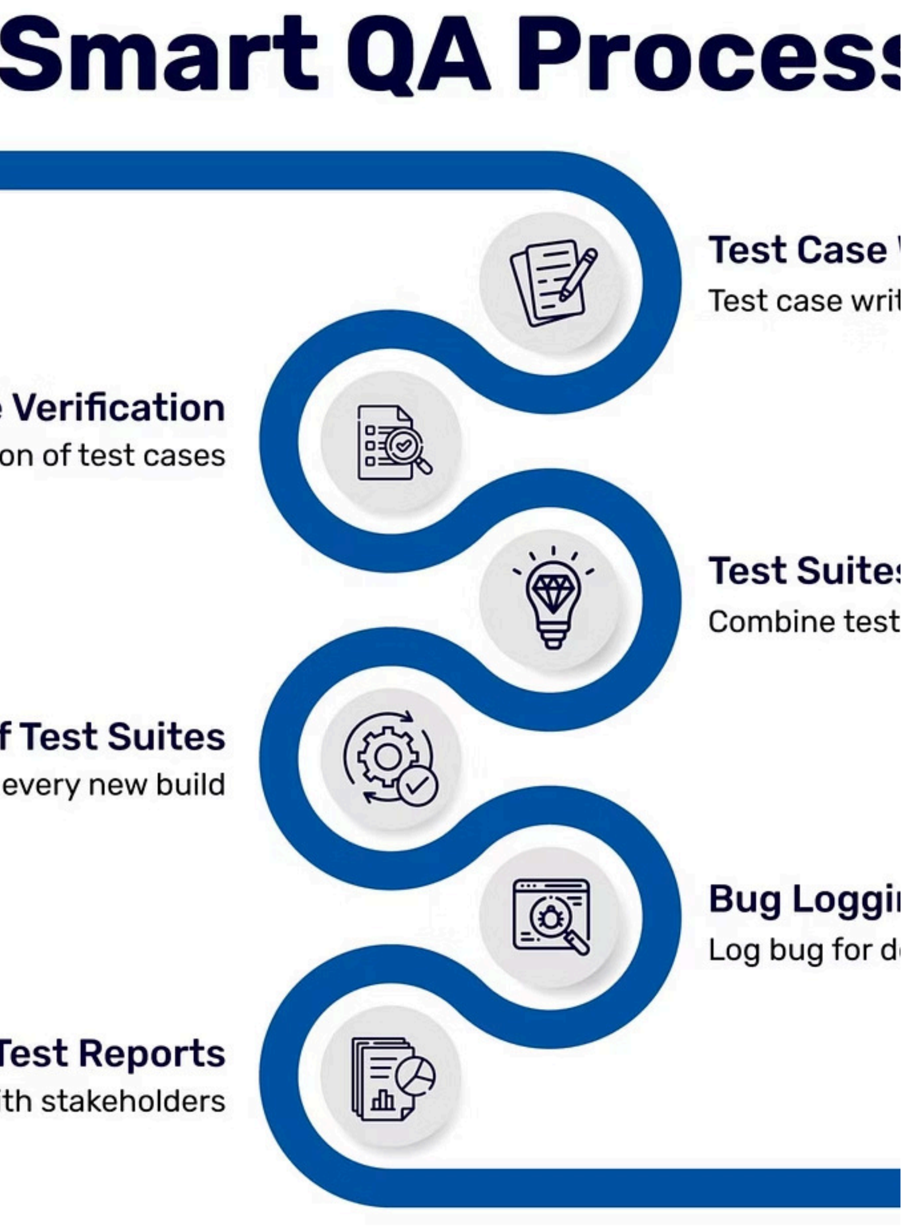
Phase 1: Deliverables

Test Strategy Document

Comprehensive document outlining testing approach, test levels, test types, entry and exit criteria, risk assessment, and resource allocation for the project.

High-Level Test Plan

Project-level test plan including scope, objectives, timelines, test environment details, assumptions, dependencies, and sign-off criteria.



PHASE 2

Test Design & Test Data Preparation

Phase 2 focuses on creating comprehensive test scenarios and detailed test cases using industry-standard test design techniques. You will apply structured approaches to ensure thorough coverage whilst preparing relevant test data sets to support execution.

Test Scenario Identification

Test scenarios represent high-level user actions or system behaviours that need validation. Each scenario encompasses multiple test cases covering positive, negative, and boundary conditions.

User Registration Scenarios

Valid registration, duplicate email handling, password strength validation, mandatory field checks

Product Search Scenarios

Keyword search, filter application, category navigation, search result validation, no results handling

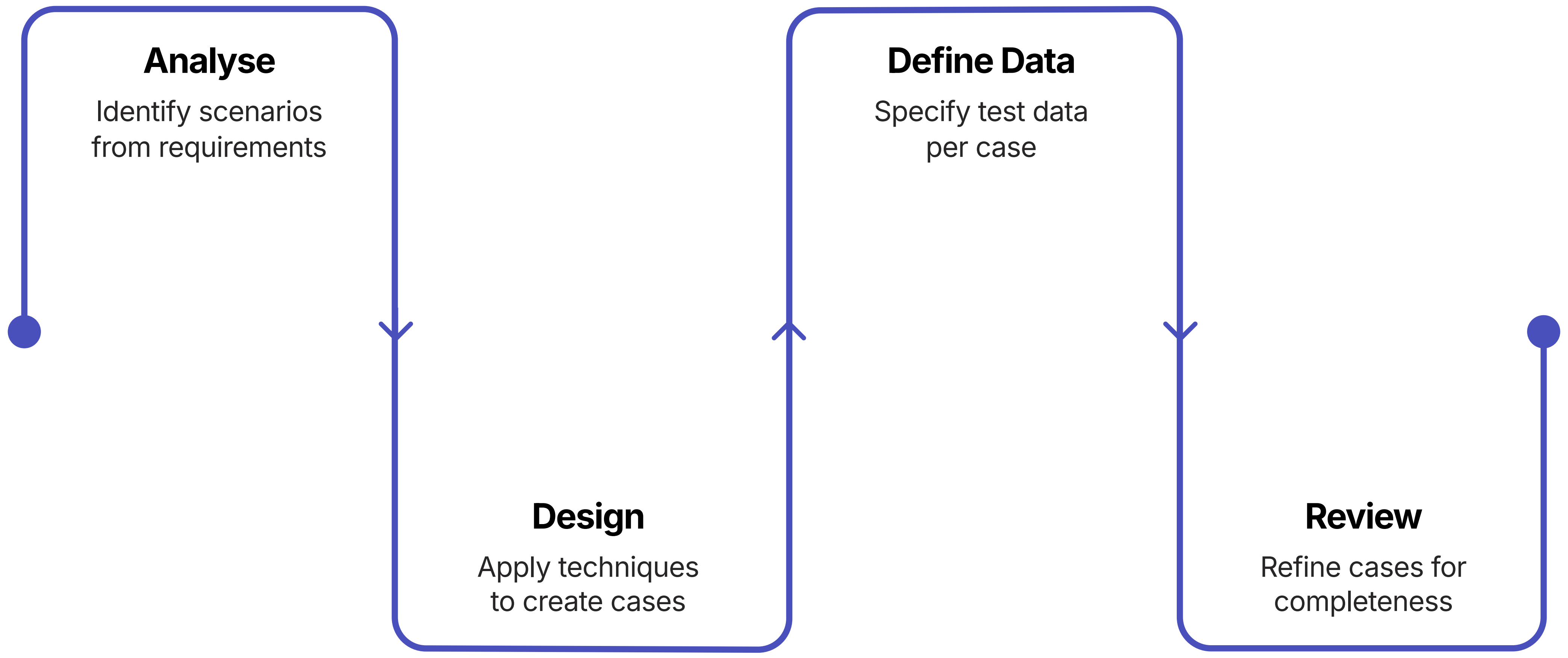
Cart Management Scenarios

Add items, update quantities, remove items, price calculations, cart persistence, empty cart handling

Order Placement Scenarios

Checkout workflow, address validation, order review, payment processing, order confirmation

Test Case Design Approach


Analyse
Identify scenarios from requirements
Design
Apply techniques to create cases
Define Data
Specify test data per case
Review
Refine cases for completeness

Each test case follows a structured format including test case ID, description, preconditions, test steps, test data, expected results, and actual results. This standardised approach ensures consistency and traceability.

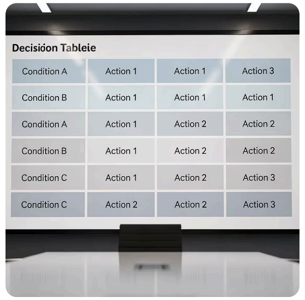
Test Design Techniques Applied

Equivalence Partitioning

Dividing input data into valid and invalid partitions to reduce test cases whilst maintaining coverage

Boundary Value Analysis

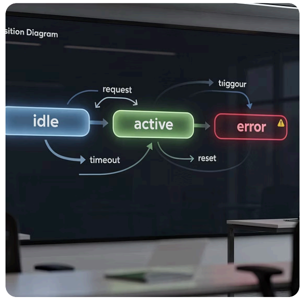
Testing at boundaries of input domains to identify edge case defects


Decision Tableie

Condition A	Action 1	Action 1	Action 3
Condition B	Action 1	Action 1	Action 1
Condition A	Action 1	Action 2	Action 2
Condition B	Action 1	Action 2	Action 2
Condition C	Action 1	Action 2	Action 3
Condition C	Action 2	Action 2	Action 3

Decision Tables

Representing complex business logic with conditions and corresponding actions



State Transition Testing

Validating system behaviour through different states and transitions

Test Design Technique Examples

Equivalence Partitioning Example

Requirement: Age must be 18-65 for registration

- **Valid Partition:** 18 ≤ age ≤ 65
- **Invalid Partition 1:** age < 18
- **Invalid Partition 2:** age > 65

Test Cases: Age = 30 (valid), 15 (invalid), 70 (invalid)

Decision Table Example

Scenario: Discount application

Condition	Case 1	Case 2
Member?	Yes	No
Cart > £50?	Yes	Yes
Discount	15%	5%

Test Data Preparation Strategy

Effective test data ensures comprehensive coverage across positive, negative, and boundary scenarios whilst maintaining data integrity and reusability.

User Data

Valid and invalid user credentials, multiple user profiles with different roles and statuses

Product Data

Various product categories, price ranges, inventory levels, and product attributes

Payment Data

Test payment credentials, multiple payment methods, valid and expired card details

Address Data

Valid shipping addresses, incomplete addresses, special character handling in address fields

Phase 2: Mapped Topics

Test Design Techniques

Application of structured techniques to create effective test cases with optimal coverage

Black Box Testing

Specification-based testing without knowledge of internal code structure or implementation

Test Data Preparation

Creating realistic and comprehensive test data sets to support various testing scenarios

Phase 2: Deliverables

1

Test Scenarios

High-level test scenarios document covering all functional modules with traceability to requirements

2

Test Cases

Detailed test case repository with steps, expected results, and design technique mapping

3

Test Data Sheet

Comprehensive test data set organised by module with valid and invalid data samples

The Importance of Manual testing in QA process

For instructor-led training, please

PHASE 3

Manual Test Execution & Defect Management

Phase 3 involves systematic execution of designed test cases, diligent tracking of test results, evidence collection for validation, and comprehensive defect lifecycle management. This phase demonstrates your ability to execute testing methodically whilst maintaining detailed documentation.

Test Execution Workflow

01

Pre-Execution Setup

Verify test environment readiness, prepare test data, and ensure application accessibility

02

Execute Test Cases

Follow test steps precisely, compare actual vs expected results, capture screenshots as evidence

03

Update Test Status

Mark test cases as Pass, Fail, Blocked, or Skipped with appropriate comments and timestamps

04

Log Defects

Document failures with detailed steps to reproduce, attach evidence, and assign severity/priority

05

Retest & Verify

Execute retesting for fixed defects and perform regression testing for impacted areas

Test Case Status Handling

Status Categories

Proper status tracking provides visibility into test progress and helps identify bottlenecks requiring attention.

- **Pass:** Test case executed successfully with expected results achieved
- **Fail:** Actual results deviate from expected results, defect logged
- **Blocked:** Cannot execute due to environment issues or dependency failure
- **Skipped:** Test case not executed due to scope or priority changes
- **Not Executed:** Test case pending execution in current cycle

Defect Logging Process

A well-documented defect report accelerates resolution by providing developers with clear, reproducible information about the issue.

Defect ID

Unique identifier for tracking

Summary

Concise description of the issue

Description

Detailed explanation with context

Steps to Reproduce

Exact sequence to recreate issue

Expected Result

Correct system behaviour

Actual Result

Observed incorrect behaviour

Severity/Priority

Impact and urgency assessment

Attachments

Screenshots, logs, videos

Environment

Browser, OS, version details

Severity vs Priority Assignment

Severity reflects the technical impact on system functionality, whilst priority indicates the urgency of fixing from a business perspective. These two dimensions work together to guide defect resolution sequencing.

Classification	Severity	Priority
Critical	System crash, data loss, security breach	Must fix immediately, blocks release
High	Major feature not working, workaround exists	Fix before release, high business impact
Medium	Feature partially working, affects some users	Fix in current sprint, moderate impact
Low	Cosmetic issues, minor inconveniences	Fix when time permits, minimal impact

Retesting and Regression Testing

Retesting

Re-executing the specific test case that initially failed to verify the defect has been fixed. Retesting confirms that the exact issue reported has been resolved.

- Execute failed test case after fix deployment
- Verify expected behaviour is now achieved
- Update defect status and test case status

Regression Testing

Testing related functionality to ensure the fix has not introduced new defects or broken existing features. Regression validates system stability after changes.

- Identify potentially impacted modules
- Execute relevant test cases from regression suite
- Verify no new defects have been introduced

Phase 3: Mapped Topics

Test Execution Process

Systematic approach to executing test cases with proper documentation and evidence collection

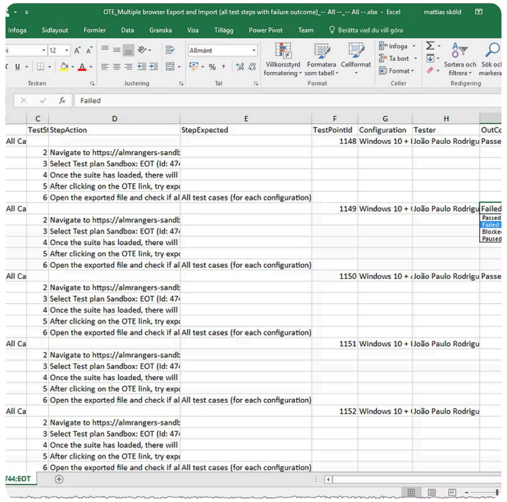
Defect Lifecycle

Understanding defect states from New through Open, Fixed, Retest, Verified, to Closed

Severity vs Priority

Distinguishing between technical impact (severity) and business urgency (priority) for defects

Phase 3: Deliverables

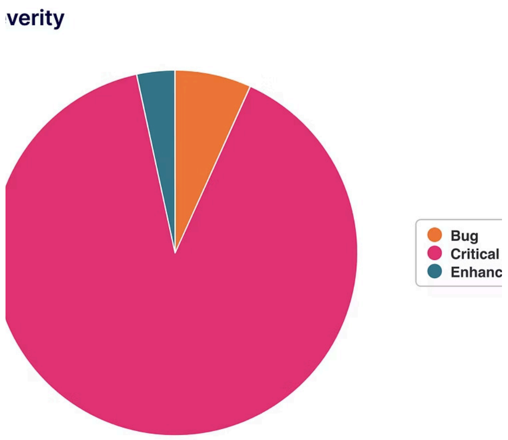


Failed

	C	D	E	F	G	H	
	TestSt	StepAction	StepExpected	TestPointId	Configuration	Tester	OutCo
All Ca				1148	Windows 10 + I	João Paulo Rodrigu	Passe
	2	Navigate to https://almrangers-sandb					
	3	Select Test plan Sandbox: EOT (Id: 47					
	4	Once the suite has loaded, there will					
	5	After clicking on the OTE link, try expo					
	6	Open the exported file and check if al	All test cases (for each configuration)				
All Ca				1149	Windows 10 + (	João Paulo Rodrigu	Failed
	2	Navigate to https://almrangers-sandb					
	3	Select Test plan Sandbox: EOT (Id: 47					
	4	Once the suite has loaded, there will					
	5	After clicking on the OTE link, try expo					
	6	Open the exported file and check if al	All test cases (for each configuration)				
All Ca				1150	Windows 10 + /	João Paulo Rodrigu	Passe
	2	Navigate to https://almrangers-sandb					
	3	Select Test plan Sandbox: EOT (Id: 47					
	4	Once the suite has loaded, there will					
	5	After clicking on the OTE link, try expo					
	6	Open the exported file and check if al	All test cases (for each configuration)				
All Ca				1151	Windows 10 + I	João Paulo Rodrigu	
	2	Navigate to https://almrangers-sandb					
	3	Select Test plan Sandbox: EOT (Id: 47					
	4	Once the suite has loaded, there will					
	5	After clicking on the OTE link, try expo					
	6	Open the exported file and check if al	All test cases (for each configuration)				
All Ca				1152	Windows 10 + (	João Paulo Rodrigu	
	2	Navigate to https://almrangers-sandb					
	3	Select Test plan Sandbox: EOT (Id: 47					
	4	Once the suite has loaded, there will					
	5	After clicking on the OTE link, try expo					
	6	Open the exported file and check if al	All test cases (for each configuration)				

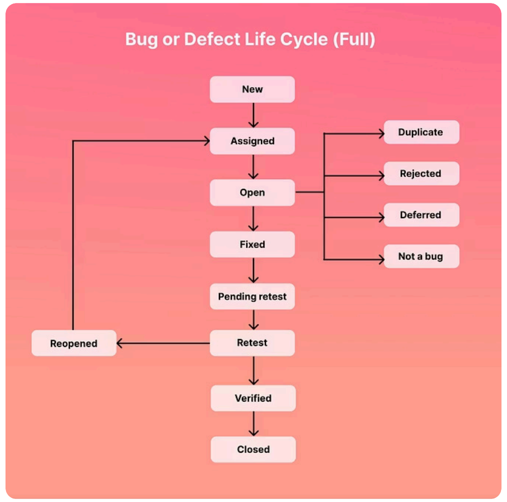
Test Execution Report

Comprehensive report showing test case execution status, pass/fail metrics, and coverage analysis



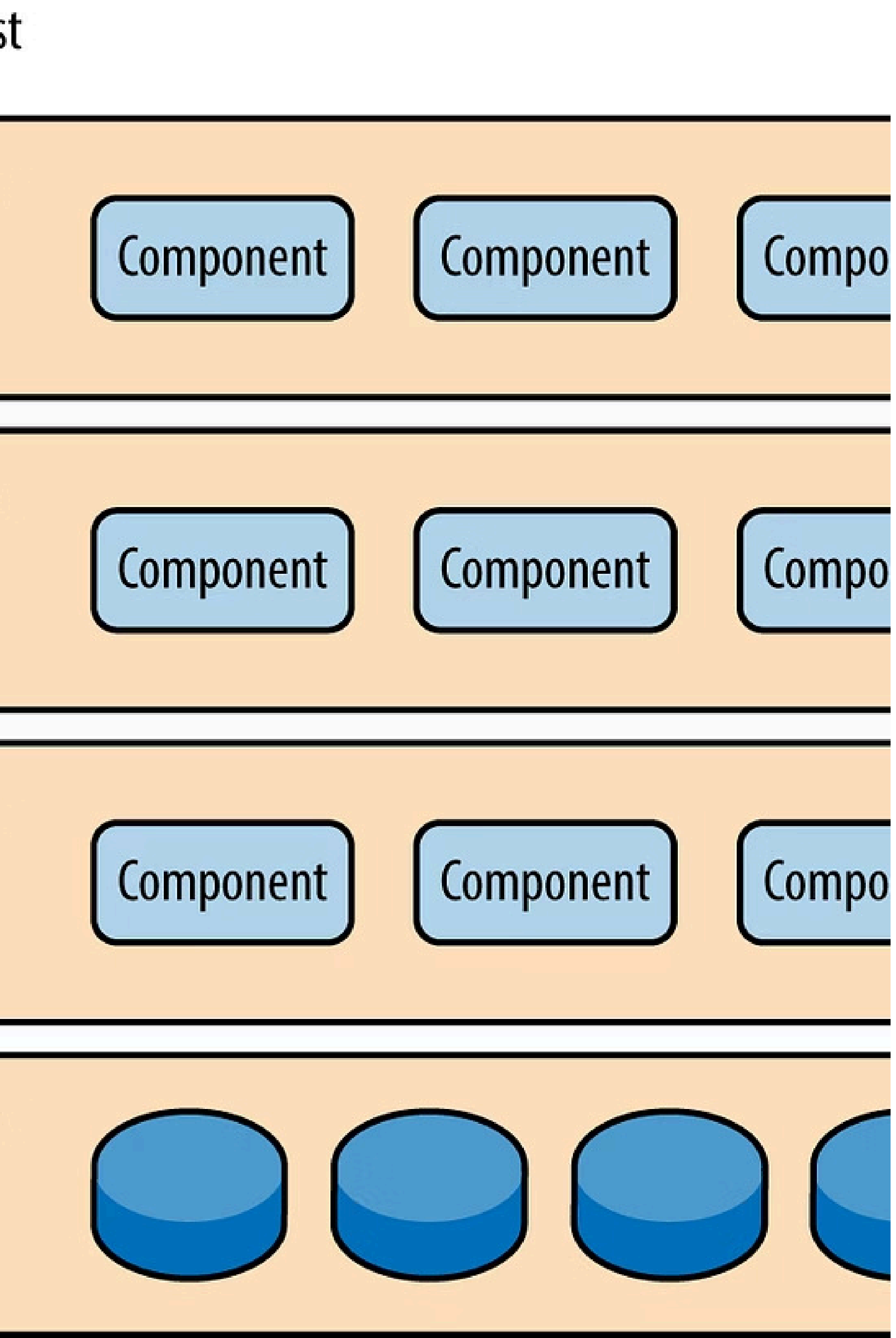
Defect Report

Detailed defect log with all identified issues, severity/priority classifications, and current status



Defect Lifecycle Tracker

Tracking document showing defect progression through lifecycle states with resolution timelines



PHASE 4

Automation Framework Design

Phase 4 marks the transition from manual testing to automation. You will design a robust, scalable automation framework incorporating industry best practices, design patterns, and object-oriented programming principles to create maintainable test automation.

Automation Strategy

A well-defined automation strategy ensures that automation efforts deliver maximum return on investment whilst maintaining code quality and test reliability.

What to Automate

- Stable, repetitive test scenarios
- Regression test suite
- Data-driven scenarios
- Smoke and sanity tests
- End-to-end critical workflows

What Not to Automate

- Exploratory testing scenarios
- One-time use cases
- Frequently changing UI elements
- Complex visual validations
- Tests requiring human judgement

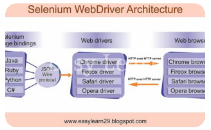
Framework Principles

- Modularity and reusability
- Data-driven capability
- Easy maintenance
- Clear reporting
- Scalable architecture

Framework Structure Overview

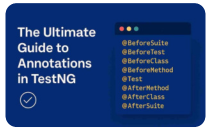
The automation framework follows a layered architecture separating concerns and promoting code reusability.

Tools Used in Framework



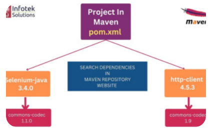
Selenium WebDriver

Core automation tool for browser interaction, web element identification, and user action simulation



TestNG Framework

Test execution framework providing annotations, assertions, parallel execution, and reporting capabilities



Apache Maven

Build automation and dependency management tool providing project structure and execution control

Application of OOP Concepts

Object-oriented programming principles ensure the framework is maintainable, scalable, and follows industry best practices.

Encapsulation

Page Object Model (POM)

Encapsulating web elements and actions within page classes. Each page has its own class containing locators and methods, hiding implementation details from test scripts.

```
public class LoginPage {
    private WebElement username;
    private WebElement password;

    public void login(String user,
                      String pass) {
        // Implementation
    }
}
```

Inheritance

Base Classes

Common functionality inherited by all test and page classes. BasePage contains shared methods; BaseTest contains setup and teardown logic.

```
public class BasePage {
 protected WebDriver driver;
 // Common methods
}

public class LoginPage
 extends BasePage {
 // Page-specific methods
}
```

Abstraction in Framework

Utility Classes

Abstract common operations into reusable utility methods that can be called across the framework.

Common Utilities Include:

- **WaitUtils:** Explicit waits, implicit waits, fluent waits
- **ScreenshotUtils:** Capture screenshots for evidence and failure analysis
- **ExcelUtils:** Read test data from Excel files for data-driven testing
- **ConfigUtils:** Read configuration properties for environment settings
- **DateUtils:** Date formatting and manipulation for dynamic data
- **StringUtils:** String operations and validations

Phase 4: Mapped Topics

Application Architecture Basics

Understanding layered architecture, separation of concerns, and modular design principles

OOP Concepts

Practical application of encapsulation, inheritance, polymorphism, and abstraction in test automation

Framework Structure

Designing scalable framework architecture with clear separation between test, page, data, and utility layers

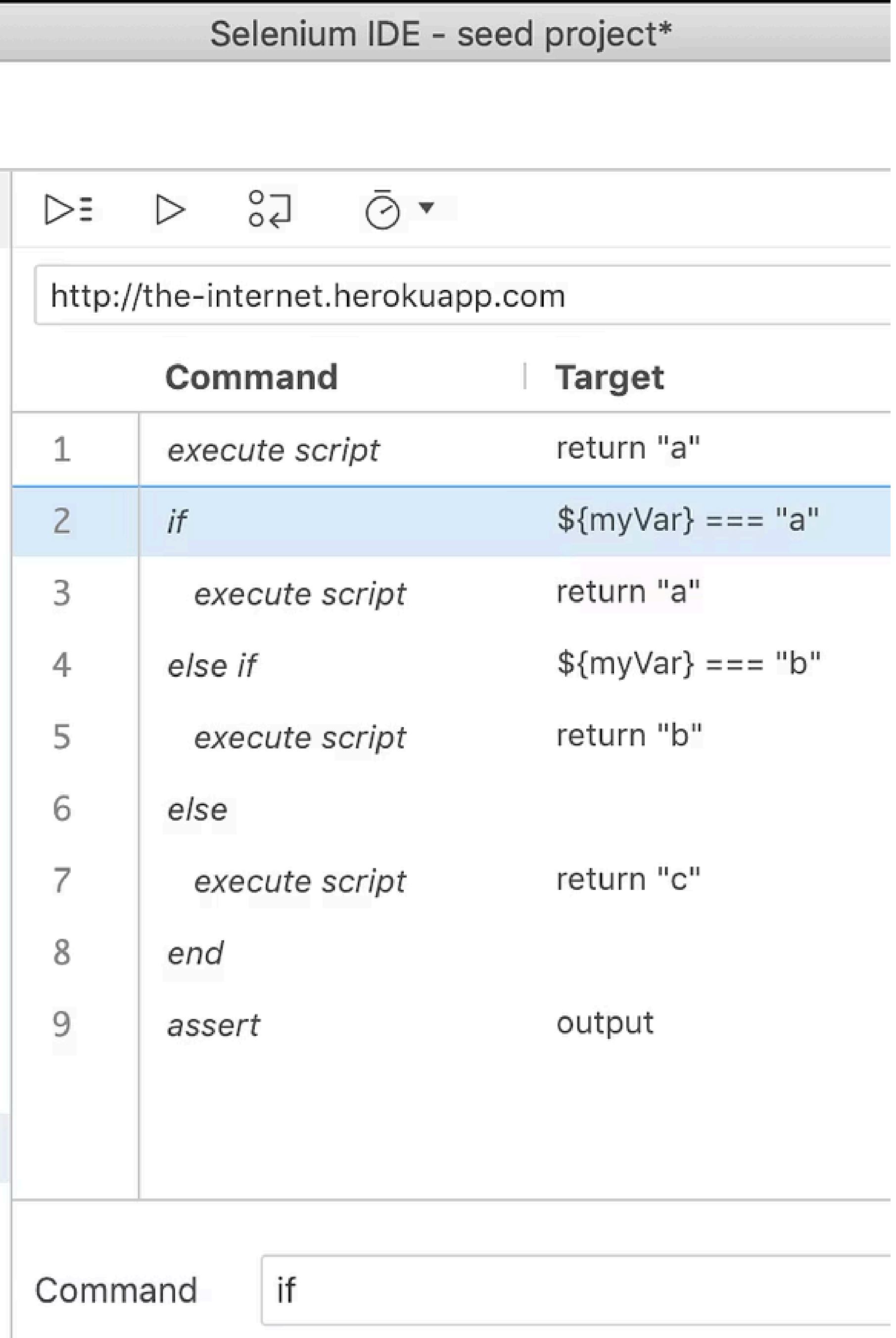
Phase 4: Deliverables

Framework Architecture Diagram

Visual representation of framework structure showing all layers, components, and their interactions with clear labelling

Maven Project Setup

Complete Maven project structure with POM.xml configuration, package organisation, and dependency management



http://the-internet.herokuapp.com

	Command	Target
1	execute script	return "a"
2	if	${myVar} === "a"
3	execute script	return "a"
4	else if	${myVar} === "b"
5	execute script	return "b"
6	else	
7	execute script	return "c"
8	end	
9	assert	output

Command: if

PHASE 5

Automation Script Development

Phase 5 focuses on implementing automation scripts within the designed framework. You will develop page object classes, write test scripts using TestNG, apply robust locator strategies, and handle synchronisation challenges to create reliable automated tests.

Automated Test Scenarios Covered

The automation suite covers critical end-to-end user journeys and key functional workflows across all modules.

1. **User Registration and Login**

   Valid registration, duplicate email validation, successful login, invalid credentials, logout functionality

2. **Product Search and Filtering**

   Keyword search, category filtering, price range filtering, search result validation, no results scenario

3. **End-to-End Purchase Flow**

   Login → Search product → Add to cart → Update quantity → Checkout → Payment → Order confirmation

4. **Cart Operations**

   Add multiple items, update quantities, remove items, price calculation validation, cart persistence

5. **Order History Validation**

   View past orders, verify order details, check order status, validate displayed information accuracy