JavaScript Test-Driven Development Workshop

Workshop Overview

This workshop introduces Test-Driven Development (TDD) using JavaScript and Jest. Participants will learn how to write tests first, implement minimal code to satisfy tests, and refactor safely while maintaining confidence in correctness.

Learning Objectives

1. Understand the Red → Green → Refactor cycle
2. Write unit tests using Jest
3. Implement small increments of functionality
4. Refactor safely using automated tests
5. Design clean and testable functions
6. Model real-world logic with incremental requirements

TDD Cycle

1. Red – Write a failing test describing the desired behavior
2. Green – Write the minimal code required to pass the test
3. Refactor – Improve code structure while keeping tests passing

TDD Best Practices

1. Write the smallest possible test first
2. Implement only what is needed to pass the test
3. Refactor frequently
4. Keep tests readable and focused
5. Avoid implementing future requirements too early
6. Prefer pure functions when possible

Live Demonstration: Password Validator

The instructor will demonstrate TDD by implementing a password validation function step by step. New validation rules will be introduced incrementally, and tests will be written before each implementation change.

Example Validation Rules

1. Password must have a minimum length
2. Password must contain at least one number
3. Password must contain an uppercase letter
4. Code should be refactored after behavior is working

Main Exercise (4–5 Hours): Subscription Billing Engine

You will build the core logic of a subscription billing system using Test-Driven Development. This system calculates charges for customers subscribed to different plans. There is no user interface and no database. Focus only on business logic implemented as functions.

General Requirements

1. Write tests before implementing functionality
2. Work incrementally
3. Refactor regularly
4. Keep functions small and focused
5. Design clear data models
6. Handle invalid input and edge cases explicitly

Part 1 — Base Monthly Subscription Price

1. Calculate monthly price for a subscription plan
2. Validate price values
3. Support simple plans with fixed monthly cost

Part 2 — Multiple Subscription Plans

1. Support different plan tiers (e.g. basic, pro, enterprise)
2. Each plan has different pricing
3. Select correct price based on chosen plan

Part 3 — Billing Period Calculation

1. Calculate cost for multiple billing months
2. Validate billing period
3. Ensure correct multiplication of monthly cost

Part 4 — Discounts

1. Apply percentage discount codes
2. Support fixed amount discounts
3. Prevent negative final price
4. Define rules for combining discounts (or disallow)

Part 5 — Taxes

1. Apply tax rate to final amount
2. Support different tax percentages
3. Define order: discount before or after tax

Part 6 — Proration for Mid-Cycle Changes

1. Upgrade or downgrade subscription mid billing period
2. Charge only for time remaining
3. Handle date-based calculation
4. Define rounding rules

Part 7 — Trial Periods

1. Support free trial days
2. No charge during trial period
3. Billing begins after trial ends

Part 8 — Cancellation Handling

1. Cancel subscription before next billing cycle
2. Determine if refund is required
3. Optional partial refunds depending on policy

Part 9 — Add-ons

1. Allow optional add-on services
2. Each add-on has monthly price
3. Add-ons combine with base plan price

Part 10 — Invoice Generation

1. Return structured billing summary
2. Include base price, discounts, taxes, add-ons, and final total
3. Ensure calculations match breakdown

Optional Extensions

1. Different billing cycles (monthly vs yearly)
2. Usage-based billing (per seat, per API call, etc.)
3. Plan upgrade effective immediately or next cycle
4. Regional tax rules
5. Credit balance system
6. Test data builders for cleaner test setup

Guidelines During the Exercise

1. Always write a failing test first
2. Implement the simplest possible solution
3. Refactor once behavior is correct
4. Keep tests descriptive and readable
5. Focus on modeling business rules clearly