TDD Trainer guide – Angular/React

# General delivery plan

This section explains generally how the course can be delivered over two days.

|  |  |  |
| --- | --- | --- |
| **DAY 1** | | |
| 09:15 | Introductions + Setup | Introductions, goals, assumptions, grouping learners, confirm local environment setup |
| 10:00 | Why TDD? What’s the problem? | TDD mindset, problems with test-after, benefits of test-first |
| 10:45 | *Break* |  |
| 11:00 | TDD Lifecycle & Writing First Test | Red/Green/Refactor, test structure, source of test data.  QuickLab 1: Counter or Stepper component |
| 12:30 | *Lunch* |  |
| 13:30 | Writing Good Unit Tests  Snapshot + Interaction Testing | Test anatomy (AAA), CUT and dependents, test qualities.  Testing component inputs/outputs, DOM events. |
| 14:15 | QuickLabs | QuickLab 2: Form with validation  QuickLab 3: Modal or Alert component |
| 15:15 | *Break* |  |
| 15:30 | Mocks, Spies & Isolation | Test doubles, verification testing, isolation strategy.  QuickLab 4: FileLoader with stubbed API |
| 17:00 | *Finish* |  |
| **DAY 2** | | |
| 09:30 | Recap  Mocking Functions & Services | Mocks vs stubs, jest.fn(), mocking services.  QuickLab 5: UserCard or SaveButton test with mocks |
| 10:45 | *Break* |  |
| 11:00 | Async Testing | waitFor, findBy, fixture.whenStable.  QuickLab 6: UserList async fetch and state check |
| 12:30 | *Lunch* |  |
| 13:30 | Routing + Custom Logic Testing | MemoryRouter, RouterTestingModule, hooks/services.  QuickLab 7: Route-driven component with logic separation |
| 15:15 | Break |  |
| 15:30 | Untestable Code, Contracts | What makes code untestable, how to design for testability. |
| 15:45 | Project | Mini Project: Combine all concepts |
| 16:45 | Wrap-up + Feedback | TDD takeaways, final thoughts, feedback session |
| 17:00 | *Finish* |  |

# Notes on delivering this course

This course is designed to be as practical as possible. Delegates will have varying levels of familiarity with JavaScript testing and may come with different amounts of experience using React or Angular. It is useful to spend time early on day one understanding who’s using which framework and how confident they are writing tests.

This will help when assigning pairs or small groups for the QuickLabs and later activities.

The ideal group size for practical work is 2–3 learners, particularly for pair programming and lab work. Larger groups of 4–5 can be used during review sessions or when discussing testing strategies more broadly.

Where possible, try to group learners by framework to reduce context-switching. You may need to be flexible depending on group size and experience.

Learners should already have access to pre-prepared boilerplate codebases for both React and Angular. These include working examples, a Jest setup, and space for new components.

Make sure everyone has their environment working properly by the end of the first lab — you’ll save time later by sorting setup issues early.

Learners will need Node.js, a text editor like VS Code, and basic familiarity with Git and the terminal.

QuickLabs follow most major topics in the slide deck and should be treated as the core learning activity.

The trainer should always demo first, then release learners to work in pairs or small groups. Encourage learners to take a test-first approach, even if it’s unfamiliar. Avoid jumping in to correct code. Instead, ask what they’re trying to test and whether their test clearly expresses that intent.

A mini-project runs toward the end of Day 2. This should be introduced as a realistic build scenario, encouraging learners to plan their test cases before implementation. Aim for each learner or pair to finish the day with a working, tested feature.

This course uses a show-then-do format and works best when learners are actively encouraged to talk, question, and test assumptions. Testing is a mindset as much as a skill — the best outcomes come when learners feel confident enough to experiment and critique their own code.