AGILE INTERMEDIATE 2 DAYS

Certified Scrum Developer (CSD)

The Certified Scrum Developer (CSD) is the foundational certification for software and hardware developers who want to combine solid Agile engineering practices with a working knowledge of the Scrum framework. It covers the principles of iterative and incremental development, test-driven development, continuous integration, and clean code practices within the context of a Scrum team.

Why CSD?

Recognised credential

Certified Scrum Developer (CSD) credential from Scrum Alliance, valid for two years.

Practical skills

Ability to collaborate more effectively on cross-functional Scrum teams.

Practical capability

Applied knowledge of Agile engineering practices within a Scrum context.

Career progression

Foundation to progress to the Advanced Certified Scrum Developer (A-CSD).

Team impact

Practical introduction to TDD, refactoring, CI/CD, and clean code.

What you'll be able to do

- ✓ Understand and apply Lean, Agile, and Scrum principles in a development context
- ✓ Explain the role of Developers within the Scrum framework
- ✓ Apply engineering practices that support iterative and incremental delivery
- ✓ Describe and apply Test-Driven Development (TDD) principles
- ✓ Understand refactoring and its role in maintaining code quality
- ✓ Enhance collaboration between technical and non-technical team members
- ✓ Understand architecture and design principles in an Agile environment
- ✓ Contribute to the definition and maintenance of a Definition of Done

Who this is for

- Developers
- Software Engineers
- QA Engineers
- DevOps Engineers
- Technical Leads
- Scrum team members

Curriculum

01 Lean, Agile, and Scrum for Developers

- Agile Manifesto and the 12 principles from a developer's perspective
- Lean principles: eliminate waste, amplify learning, defer commitment
- The Scrum framework refresher: events, artifacts, accountabilities
- The Developer accountability: what it means to be a Scrum Developer
- Done vs. partially done: the cost of incomplete work

02 Collaboration and Team Dynamics

- Cross-functional team design and T-shaped skills
- Working with Product Owners on backlog refinement and acceptance criteria
- Effective Daily Scrums and collaborative sprint planning
- Knowledge sharing, pair programming, and collective code ownership
- Reducing silos and specialisation bottlenecks

03 Architecture and Design in Agile

- Emergent architecture vs. up-front design
- SOLID principles and their application in iterative development
- Design patterns relevant to Agile development
- Managing technical decisions incrementally
- Just-enough design: avoiding over-engineering and under-engineering

04 Test-Driven Development (TDD)

- The Red-Green-Refactor cycle
- Writing failing tests first: habits and discipline
- Unit testing vs. integration testing vs. acceptance testing
- Test coverage and what meaningful coverage looks like
- TDD in practice: common challenges and how to overcome them

05 Refactoring

- What refactoring is and what it is not
- Recognising code smells: duplication, long methods, large classes
- Common refactoring patterns: extract method, rename, move function
- Refactoring safely with tests as a safety net
- Balancing refactoring with feature delivery in a Sprint

06 Continuous Integration and Delivery

- The principles of Continuous Integration (CI)
- Build pipelines and automated testing in CI
- Continuous Delivery (CD) and Continuous Deployment
- Feature toggles and trunk-based development
- The relationship between CI/CD and the Definition of Done

07 Definition of Done in Practice

- Creating a Definition of Done with the team
- Layered Definitions of Done: team, product, release
- Evolving the Definition of Done over time
- Technical debt and how it accumulates when Done is weak
- The relationship between Done and the Increment