Coder by Testingscool

Learning plan

60 hours
10 weeks of: Instructor-led session | Case studies | 1:1s | Reviews | Self-study | Video lessons
Code examples provided in: pseudocode or Java or Typescript

Module	Learning goals

Promoting value & goals of test automation	• Identify and set automation goals
	  ○ Describe what an automation goal looks like
	  ○ Align automation goals with testing strategy goals
	  ○ Consider how automation might help in a given context
	  ○ Formulate a list of automation goals

Test management strategies	• Determine what plan to create
	  ○ Use different techniques to discover explicit automation opportunities - Front-end, API Services
	  ○ Consider what options are available to achieve a strategy's goals
	  ○ Choosing an option to help achieve strategy goals

Tooling	• Identify tool requirements
	  ○ List requirements to consider when building or selecting a tool for automation - API, UI, A11Y, Performance, Security, etc.
	  ○ Ability to highlight which tools can or cannot be used in context

From requirements to Test automation design	• Decide what automated checks to create
	  ○ Break down system behavior to understand how features are implemented
	  ○ Point out risks that might impact different parts of an implementation
	  ○ Prioritize risks to create automated tests for
	• Determine what layer the automated check should be on
	  ○ Determine if testability allows tests to be automated on an identified system layer
	  ○ Decide what layer to create automated checks on

Anatomy of Test Automation in CI/CD projects	• Collect and analyze failure information for further examination. • Enumerate potential causes of a test/build failure. • Scrutinize errors within a build to uncover the core problem. • Identify the reason behind a failed automated check in the process. ○ Detail different factors that could lead to the failure of an automated check.

🗣️ @testingscool
Get in touch: ✉️ mihai@testingscool.ro ; 🙂 : Mihai Oprean ; 📞 : +40753638006

	○ Investigate the root cause of a failing automated check. ○ Decide on the appropriate course of action when a check fails. • Provide an outline of steps to address diverse failures detected by automated checks. ○ Determine the necessary actions to resolve a failing automated check.
Implement a solution to rectify a failing automated check.	• Identify flakey tests and provide alternative solutions • List reasons of inconsistent test results • Provide alternative approaches to create reliable test results
Hands-on XUnit Patterns Frameworks: Junit, TestNG, AssertJ, Jest, Mocha	• List what state, actions and assertions the check will carry out ○ Outline the anatomy of an automated check ○ Describe different types of state to set up for an automated check ○ Describe how automated checks can interact with a system under test
Assertion patterns and strategies Browser automation tools: • MochaJS • Jest • Jasmine • Puppeteer (Node Library) • Cypress • Playwright (Node Library) API Tools • Postman • RestAssured • Jest	• Describe ways in which assertions can be codified in an automated check ○ Decide what state, actions and assertions need to be created for an automated check • Listen and react to situations that indicate maintenance is required ○ List reasons why maintenance might be required for automation ○ Determine what maintenance steps are required for automated checks ○ Implement changes to improve automated checks • Evaluate checks to make sure they are still adding value and delete/update ones that don't ○ Describe why deleting automated checks is sometimes required ○ Articulate different characteristics that can be used to judge automated checks ○ Critique existing automated checks ○ Modify or delete automated checks that no longer deliver value
Reporting	• Identify and list key metrics to inform stakeholders on the output of test automation • Determine approaches on how to report on results for better decision making process

🗣️ @testingscool
Get in touch: ✉️ mihai@testingscool.ro ; 🙂 : Mihai Oprean ; 📞 : +40753638006

Prerequisites

- Code syntax understanding
- Experience operating with IDE (IntelliJ or Eclipse)
- Prior hands-on experience with Object Oriented Programming

Courses recommended as prerequisite for Coder:

- Javascript
  - Basic JavaScript
  - Object Oriented Programming
- Java
  - Java programing by TestAutomation University
  - Java for testers