Course: Test automation with Continuous Integration

1. What is Continuous Integration (CI)?

Continuous Integration (CI) means that developers regularly add (integrate) their code into a shared repository. Each change is automatically built and tested to detect issues early in the development process.

Strengths of CI

1. Early detection of bugs through automated builds and tests.
2. Faster feedback to developers on code changes.
3. Improved code quality and fewer integration issues.
4. Automation reduces human errors in build and testing processes.

Weaknesses of CI

1. Initial setup and maintenance require time and effort.
2. Requires discipline and cultural change (e.g., frequent commits).
3. Flaky (unstable) tests can lead to unreliable build results.

Why it's important?

CI helps catch errors quickly and ensures that new code doesn’t break existing functionality

2. What is test automation?

Test automation means using software tools to run tests on your application automatically, instead of doing it manually. This saves time and helps find bugs early.

How is it connected to CI/CD?

1. CI/CD is about automating the process of building, testing, and delivering software.
2. When a developer writes new code and commits it to a repository,
3. A CI tool automatically runs the automated tests,
4. If all tests pass, a CD tool can deploy the new version of the app.
5. If tests fail, the process stops, and the developer is notified.

Example:

Example:
- You're working on a food delivery app.
- You add a "Track My Order" feature.
- You push code to GitHub.
- GitHub Actions runs automated tests.
- If tests pass → Code can be deployed.
- If tests fail → you’re notified to fix the issue.

Popular Tools: CI/CD Tools: Jenkins, GitHub Actions, GitLab CI/CD, and CircleCI.

Test Automation Tools: Selenium, Cypress, JUnit, TestNG, and Playwright.

3. How to run tests automatically using GitHub Actions?

Steps:

1. Put your code on GitHub.
2. Create a folder: .github/workflows.
3. Inside that folder, create a file (e.g., test.yml).
4. In the file, write instructions to:
    - Run tests on every push or pull request.
    - Set up the environment.
    - Install dependencies.
    - Run the test scripts.

Example:

Once this is set up, GitHub will automatically run your tests every time someone pushes new code or opens a pull request.