Resources for Software Testing

Unit Testing

- VSCode supports JUnit 4/5 for testing in Java and this is the official tutorial link: Java Testing in Visual Studio Code
- `pytest` is the recommended Python framework for unit testing and the corresponding official tutorial: Testing Python in Visual Studio Code

API Testing

Use Postman and Postbot (by Postman): Introducing Postbot, Postman's New AI Assistant | Postman Blog

Functional Testing

Integration Testing (Black-box)

- Selenium (Chrome) Webdrivers for testing webpages whether you are using Java or Python.
    - Setup Instructions for Java: Setup Selenium with Java on Visual Studio Code - Funnel Garden and Selenium with VS Code. This article will describe how to set... | by Harshi Wickramaarachchi | Medium (Free articles available as of writing)
    - Setup and Guide for Python: Selenium Python Tutorial (with Example) | BrowserStack and Selenium Python Tutorial : A Python Automation Testing Guide with examples (lambdatest.com) (Free as of writing)

Regression Testing (White-Box)

If using React for the frontend, one can use Jest to perform Snapshot testing for easily tracking UI changes or regressions and quickly detecting major changes in a website's/component's behaviour, refer this guide for more details: Snapshot Testing · Jest (jestjs.io)

Performance Testing (Black-Box)

- Apache JMeter (Apache JMeter - Apache JMeter™) is the most common software used for load and performance testing HTTP based Applications and is even used

for testing things like JDBC (for Databases), CLI Tools, SMTP (Email), Java Objects etc.
    - In case one wants to dive deep and get started, here is the official user manual: Apache JMeter - User's Manual
- There are also attempts to port the concept into Python using packages like `pymeter` (Welcome to pymeter's documentation! — pymeter Documentation (1.0.x))
- (Companies / the industry likely has its own wrappers/setups/configs around JMeter – For example BlazeMeter Load Testing | Blazemeter by Perforce)

Code Coverage (White-Box)

- For Java, JaCoCo measures code coverage in the form of # of instructions (equivalent to statement coverage) and # of branches (branch coverage) -
    - Configuration instructions: Using the <plugins>...</plugins> part from jacoco.org/jacoco/trunk/doc/examples/build/pom.xml for unit test coverage and jacoco.org/jacoco/trunk/doc/examples/build/pom-it.xml for integration test coverage.
    - Alternatively, adapt this answer: https://stackoverflow.com/a/25485301 and tweak the version numbers. A quick tutorial for usage: How to generate Code Coverage Report using Jacoco in a Java application | Codementor
    - A sample report of their own repo is found here: https://www.jacoco.org/jacoco/trunk/coverage/
- For Python, the packages used are `pytest-cov` and `coverage` (pytest-cov 5.0.0 documentation and Coverage.py — Coverage.py 7.6.1 documentation) and example usage are given here: How to Generate pytest Code Coverage Report | LambdaTest (The default options provide only statement coverage, but there is an option `--cov-branch` to calculate and display branch coverage as well.

Additional AI-Enhanced Testing Tools

- **GitHub Copilot** for code generation during testing: Use GitHub Copilot to automatically generate unit and integration tests. Video tutorials available on the class website.
- **Codeium** and **GPT-4o** for enhancing testing scripts and automating test case generation, especially useful for complex testing scenarios in whitebox testing.
- **GPT-4o**: Capable of generating complex test scenarios based on the description of software functionalities. Supports multiple languages and frameworks, enhancing the versatility in test generation.

- **Testim**: Employs AI to speed up the creation, execution, and maintenance of automated tests. Useful for regression and functional testing, particularly in dynamic and complex web applications.
- **Applitools**: Uses Visual AI to automatically validate the appearance of UI elements across different screens and operating systems. Helps catch visual regressions and UI bugs that are difficult to detect with traditional testing tools.
- **Postman's Postbot**: An AI assistant within Postman that can write and run tests based on user conversations about the intended functionality of APIs. Enhances API testing by generating assertive tests that adapt to changes in API responses.
- **ReTest** (AI-enhanced GUI testing tool): Utilizes AI to learn the correct behavior of the application under test by monitoring user interactions. Automatically generates regression tests and highlights deviations from expected GUI behavior.
- **Mabl**: Provides an end-to-end test automation service using machine learning to auto-heal tests when the application under test evolves. Facilitates integration testing and browser testing, reducing the manual effort in maintaining test suites.

Industry Tools and Practices

- **SonarQube** for in-depth code quality analysis: Commonly used in industry for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities.
    - Setup with GCP: Instructions included on class website for setting up SonarQube on a GCP VM. Setup instructions are borrowed from CS2340.