Date:
Lab Session No.: 12

Task 12: Make a report on Cucumber and CLion.

Aim: Report on Cucumber and CLion.

Cucumber:

- Cucumber is a testing tool that supports Behavior Driven Development (BDD) framework. It defines application behavior using simple English text, defined by a language called Gherkin.
- Cucumber allows automation functional validation that is easily read and understood. Cucumber was initially implemented in Ruby and then extended to Java framework. Both the tools support native JUnit.
- How it Works:

- It is a very simple notion, but what we need in order to get this concept implemented. The answer is, Behavior Driven Development (BDD) Framework. Cucumber is one such open source tool, which supports behavior driven development. To be more precise, Cucumber can be defined as a testing framework, driven by plain English text. It serves as documentation, automated tests, and a development aid – all in one.
- **Feature files:** They are the essential part of cucumber which is used to write test automation steps or acceptance tests. This can be used as the live document. The steps are the application specification. All the feature files end with .feature extension.
- **Feature :** This gives information about the high-level business functionality (Refer to the previous example) and the purpose of Application under test. Everybody should be able to understand the intent of feature file by reading the first Feature step. This part is basically kept brief.
- **Scenario** : Basically, a scenario represents a particular functionality which is under test. By seeing the scenario user should be able to understand the intent behind the scenario and what the test is all about. Each scenario should follow given, when and then format. This language is called as "gherkin".
  1. **Given:** As mentioned above, given specifies the pre-conditions. It is basically a known state.
  2. **When**: This is used when some action is to be performed. As in above example, we have seen when the user tries to log in using username and password, it becomes an action**.**
  3. **Then:** The expected outcome or result should be placed here. For Instance: verify the login is successful, successful page navigation.
  4. **Background:** Whenever any step is required to perform in each scenario then those steps need to be placed in Background. For Instance: If a user needs to clear database before each scenario then those steps can be put in a background.
  5. **And**: And is used to combine two or more same type of action.

- **Scenario outline** :Scenario outlines are used when the same test has to be performed with different data set. Let's take the same example. We have to test login functionality with multiple different sets of username and password.
- **Tags :**
  Cucumber by default runs all scenarios in all the feature files. In real time projects, there could be hundreds of feature file which are not required to run at all times.
  For instance: Feature files related to smoke test need not run all the time. So if you mention a tag as smokeless in each feature file which is related to smoke test and runs cucumber test with @SmokeTest tag. Cucumber will run only those feature files specific to given tags. Please follow the below example. You can specify multiple tags in one feature file.
- **JUnit Runner** :
  To run the specific feature file cucumber uses standard JUnit Runner and specify tags in @Cucumber. Options. Multiple tags can be given by using comma separate. Here you can specify the path of the report and type of report you want to generate.
- **Cucumber Report** :
  Cucumber generates its own HTML format. However, better reporting can be done using Jenkins or bamboo tool. Details of reporting are covered in next topic of cucumber.
- **Advantages :**
  1. It is helpful to involve business stakeholders who can't easily read code
  2. Cucumber Testing tool focuses on end-user experience
  3. Style of writing tests allow for easier reuse of code in the tests
  4. Quick and easy set up and execution
  5. Cucumber test tool is an efficient tool for testing.
- **Disadvantages :**
  1. Retrofitting can be time-consuming.
  2. Time Overhead.
  3. Structuring all your feature files, scenarios and executable specifications requires some careful planning.
  4. There needs to be a good amount of communication between the person writing the feature files and the person developing the automation code.

CLion:

- CLion 2020.3 brings significant, eagerly anticipated improvements to key parts of the development process – code analysis, running and debugging applications, and unit testing. For embedded projects, CLion now comes with initial MISRA C and C++ support. And Qt users can benefit from IDE features tuned specifically for working with Qt code.
- CLion is more than just an editor as it offers a powerful debugger and dynamic analysis tools to investigate and solve problems with ease, built-in Google Test, Boost.Test and Catch for unit testing, many popular VCS supported out of the box and more.
- **Built-in tools and integrations**:

  1. **Run and Debug** :
     Build, Run and Debug your application and unit tests locally or remotely in CLion. Use the debugger UI with GDB or LLDB as a backend.
  2. **Dynamic analysis** :
     Use Valgrind Memcheck and Google Sanitizers integration to detect memory errors, data races and undefined behaviour issues. Analyze the performance of your application with the CPU Profiler integration.
  3. **CMake support** :

CMake is a famous cross-platform build system, widely used for C and C++ projects. Benefit from the CLion's smart CMake support with code generation, completion and automatic target updates.

4. **Unit testing** :
   CLion supports the Google Test, Boost.Test and Catch frameworks and provides a built-in test runner together with a powerful UI to investigate test results. It can even generate code for you as you write tests (for Google Test).
5. **Embedded Development** :
   Develop for microcontrollers in CLion and benefit from various on-chip debugging options, Peripheral View for ARM devices, and STM32CubeMX integration.
6. **VCS integration and local history** :
   CLion provides a unified interface for most popular VCS including Subversion, Git, GitHub, Mercurial, CVS, and Perforce. Meanwhile, local history will save you from unexpected accidents.

- **Advantages** :

1. It's surprisingly easy to start a new project in CLion, and files can be added to the project in one click. CLion works with CMake, Gradle and compilation database project models. If you use a different type of project, the IDE will help you import to CMake.
2. With an IDE that analyzes the context and understands your project, you can code faster than you think. Try smart completion, formatting and helpful views with code insight.
3. Find your way through the code with instant navigation to a symbol, class or file. Inspect the calls or types hierarchy and easily search everywhere for nearly everything .
4. Save time on unnecessary typing while CLion generates code for you: from getters/setters to more complicated templates. Use refactorings to improve and clean up your code at the speed of thought.

Result:
Successfully created report on Cucumber and CLion .

MARKS:_____ **STAFF SIGNATURE: _____**