7. Customizations

- **Pass Args in Pytest Cmd line :**
  - Use the pytest_addoption hook function in conftest.py to define a new option.
  - Then use 'pytestconfig fixture' in a fixture of your own to grab the name.
  - We can also use pytestconfig from a test to avoid having to write your own fixture.
  - Useful when we are trying to do some tests based on diff configurations.
  - For e.g. in prod and QA, which have diff URL we can pass in cmd line arg to identify between prod and QA.

- **Configuring pytest.ini:**
  - Ref: https://docs.pytest.org/en/stable/customize.html#config-file-formats
  - Pytest related configs are put in this file.
  - Placed in the root/main folder of our project.
  - Primary pytest configuration file that allows you to change default behavior.
  - You can get help on command line options and values in INI-style configurations files by using the general help option: pytest -h
  - Also refer: https://docs.pytest.org/en/stable/reference/reference.html#ini-options-ref
  - E.g. # content of pytest.ini
  - [pytest]
  - addopts = --maxfail=4 -rf # exit after 4 failures, report fail info
  - You can change default behavior of pytest, e.g. change test module name to work with something like check_something.py.

7. Customizations

- **Data Provider with Pytest:**
  - Use pytest parameterize to pass in data to test, after reading from a file.
  - E.g. write a module utils.py which has function get_data() and pass this function get_data() to pytest.mark.parametrize.

- **Using Configurations files:**
  - Config files can be done in diff formats - like json, yaml, or ini files.
  - Write function for reading config file and reusing it from multiple places.
  - For designing framework:
    - Config files in config folder.
    - Config parser in utils folder.
    - have functions to read the configs and return values.
  - Write config parser in OOP way to control diff configs passing from Cmdline options.