1. **Overview:**

Process the client provided data which carries Customer Id, Contract Id, Geo Zone, Team Code, Project Code, Build Duration from the data set (either by csv, xml or flat file) and print the final result in the report

1. **Roles & Responsibilities:**

|  |  |
| --- | --- |
| Requirement Analysis | Anup |
| Testable Item/Data Identification | Anup |
| Test Plan Design | Anup |
| Manual Test Case Design | Anup |
| Automation Test Case Design | Anup |

1. **Requirement:**

|  |
| --- |
| **Given a string with the following data (it includes multiple lines):**  2343225,2345,us\_east,RedTeam,ProjectApple,3445s  1223456,2345,us\_west,BlueTeam,ProjectBanana,2211s  3244332,2346,eu\_west,YellowTeam3,ProjectCarrot,4322s  1233456,2345,us\_west,BlueTeam,ProjectDate,2221s  3244132,2346,eu\_west,YellowTeam3,ProjectEgg,4122s  **The data is organized into columns delimited by a comma (,) in the following order:**  customerId,contractId,geozone,teamcode,projectcode,buildduration  **The first line of data would then be interpreted as**  2343225 is the customerId  2345 is the contractId  us\_east is the geozone  RedTeam is the teamcode  ProjectApple is the projectcode  3445s is the buildduration    **Write an application using Java that would consume the entire multiline string as the input**  **and produce the following report as the output**  The number of unique customerId for each contractId  The number of unique customerId for each geozone  The average buildduration for each geozone  The list of unique customerId for each geozone |

1. **Test Strategy:**

* Identify the test data and filter out what to be tested and automated based on the requirement
* Write the code the referring to the data sheet to split out the data and organize it with a systematic manner

1. **Test Scenario:**

* Validate all the requirements as given in the requirement (.CSV, xlsx or flat file data format)
* Validate with valid inputs (assert data which are in scope)
* Validate with invalid inputs (assert data which are out of scope)

1. **Test Cases:**

* **TC001 :-** Identify and verify the test\_unique\_customer\_ids\_by\_contract
  + - **Expected Result:** Test Case Pass
* **TC002 :-** Identify and verify the test\_unique\_customer\_ids\_by\_geozone
  + - **Expected Result:** Test Case Pass
* **TC003 :-** Identify and verify the test\_average\_build\_duration\_by\_geozone
  + - **Expected Result:** Test Case Pass
* **TC004 :-** Identify and verify the test\_unique\_customer\_ids\_list\_by\_geozone
  + - **Expected Result:** Test Case Pass
* **TC005 :-** invalid input test\_unique\_invalid\_customer\_ids\_list\_by\_geozone
  + - **Expected Result:** Test Case Fail

1. **Ad-Hoc/Edge Test Cases:**

* Process some invalid data file formats:
  + - .done file format
    - .pl file format
    - .mf file format
      * System will throw error if above file formats are used

1. **Automation:**

* **Language:** Python
* **Framework:** Pytest Unit-Test Framwork
* **Reporting:** pytest-html
* **Time Frame of Execution:** Max: 0.35s Min in 0.28s

1. **Integration Testing:**

* Client data placed in the correct directory to process through Pytest Automation
* Code picks up the data from the directory
* Validate data with as per the test cases written in Python Automation
* Generate the pass/report in html

1. **Defect handling:**

* Identify the root-cause analysis of the failed test cases
* Provide accurate reproducible steps to recreate the bug
* Attach required screenshot to the bug ticket (JIRA)

1. **Exit Criteria:**

* All the requirements should be covered
* All test cases should be executed and complete status
* All the defects should be closed, with QA evidence document attached
* Demo session to business stake holders what has tested and review of test results
* Business Sign off for code release to UAT (Ready for Client Testing)