BridgeLabz

Employability Delivered

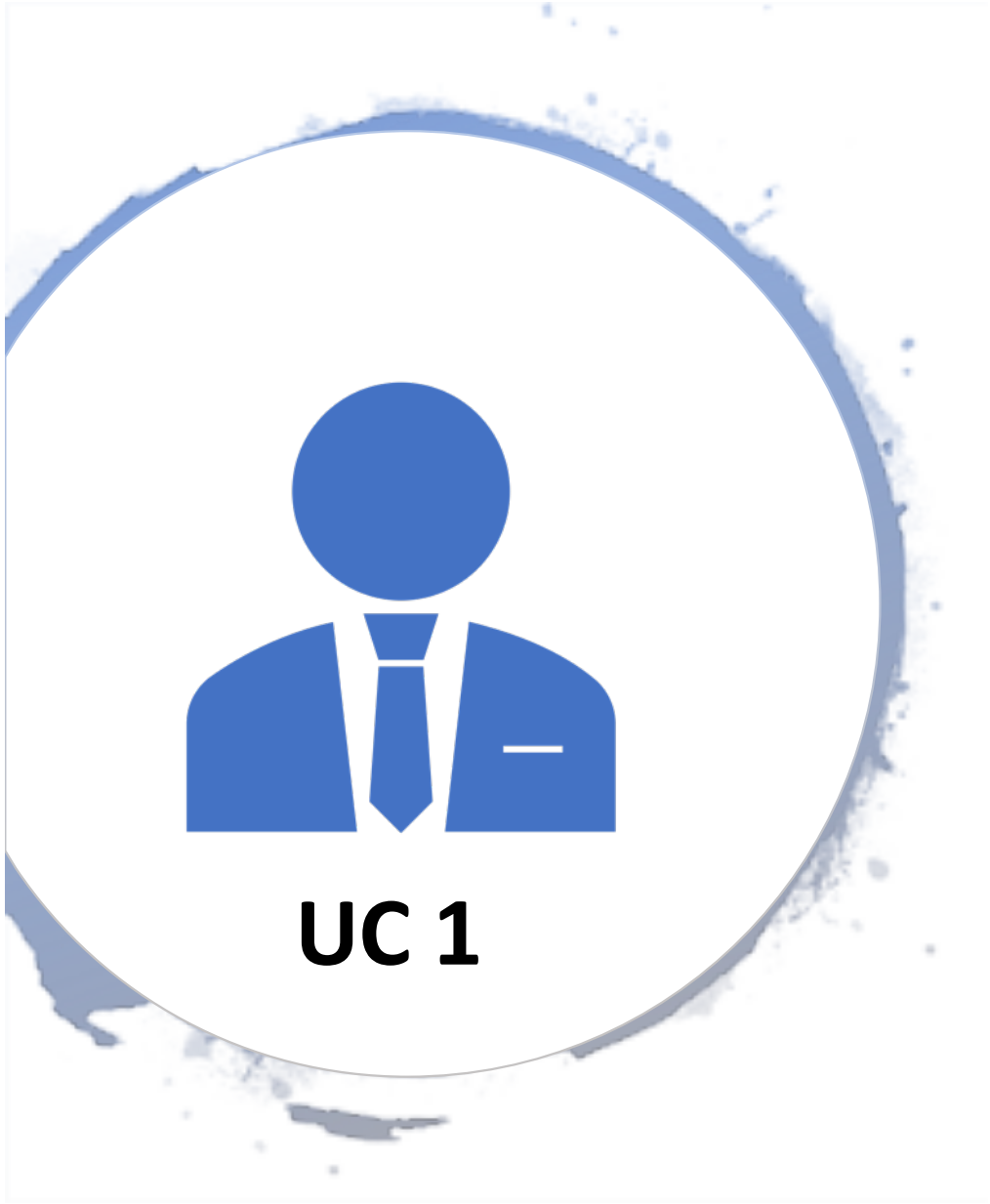
# Census Analyser Problem

Emphasis on

- Libraries – OpenCSV, JSON
- Jar Library
- Generics, Sort and Search
- Design Principles
- Design Patterns - Adapter Pattern, DAO Pattern, Factory Pattern…

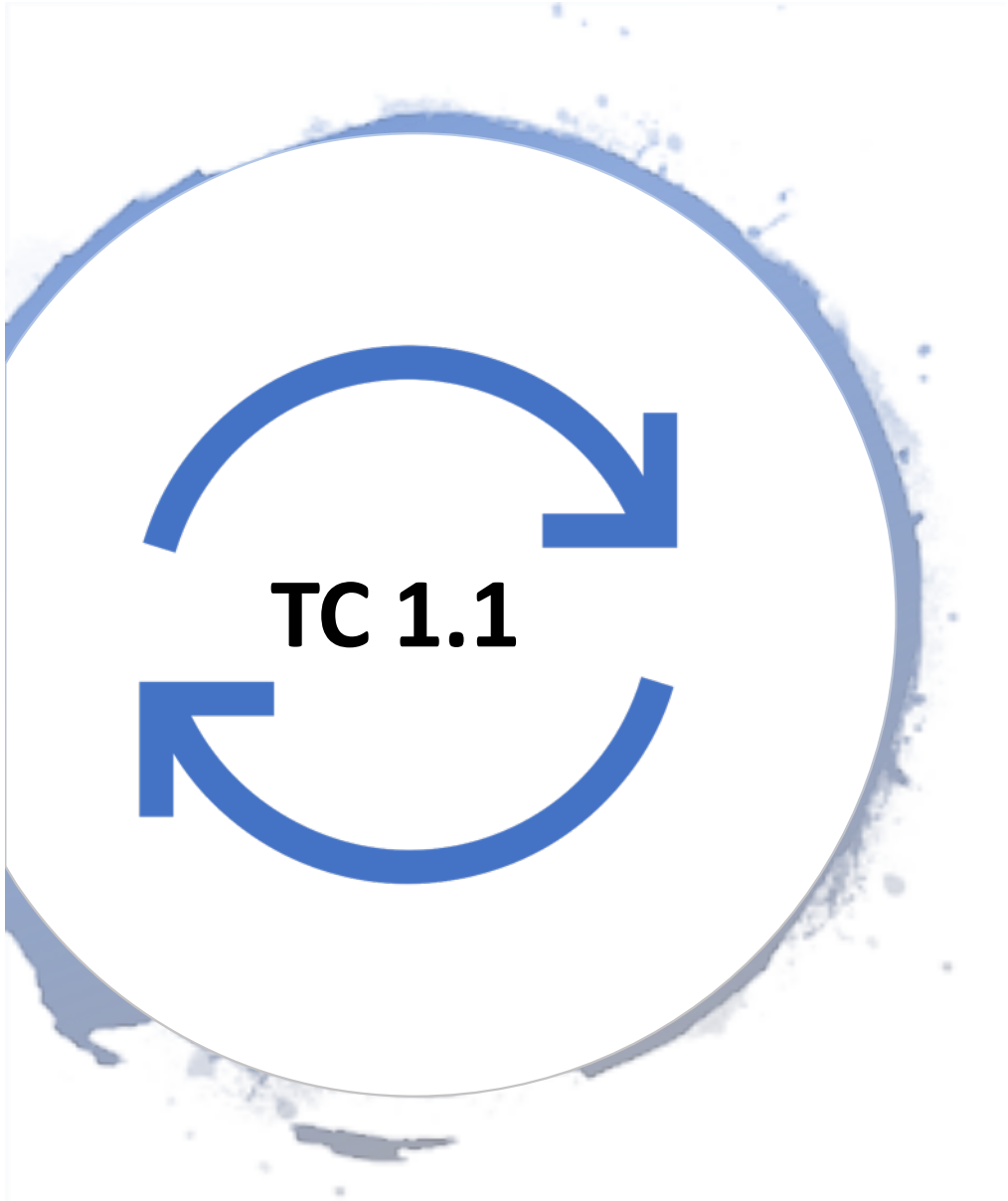
# Rules

- Start with Welcome message in the Main Branch
- Every Use case should have its own commit
- Every Test Case should have its own Commit
- Every Refactor also should have its own commit
- Follow Follow Programming Hygiene with proper and consistent naming and indentation convention
- Follow DRY principle and Refactor Code




# Ability for the analyser to load the Indian States Census Information from a csv file

- Create a StateCensusAnalyser Class to load the State Census CSV Data
- Create CSVStateCensus Class to load the CSV Data
- Use Iterator to load the data
- Check with StateCensusAnalyser to ensure number of record matches

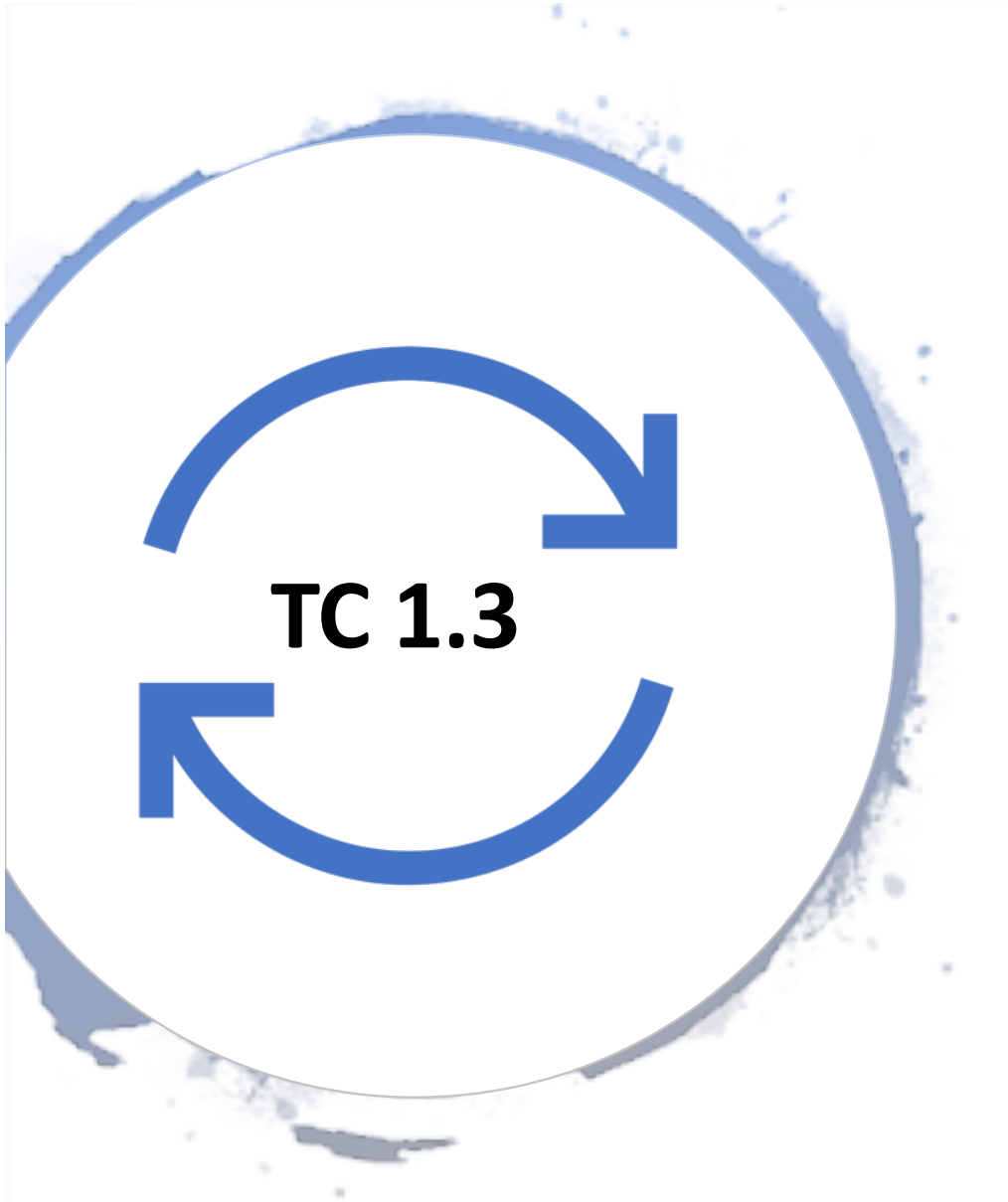



# Given the States Census CSV file, Check to ensure the Number of Record matches

This is a Happy Test Case where the records are checked


# Given the State Census CSV File if incorrect Returns a custom Exception

This is a Sad Test Case to verify if the exception is raised.

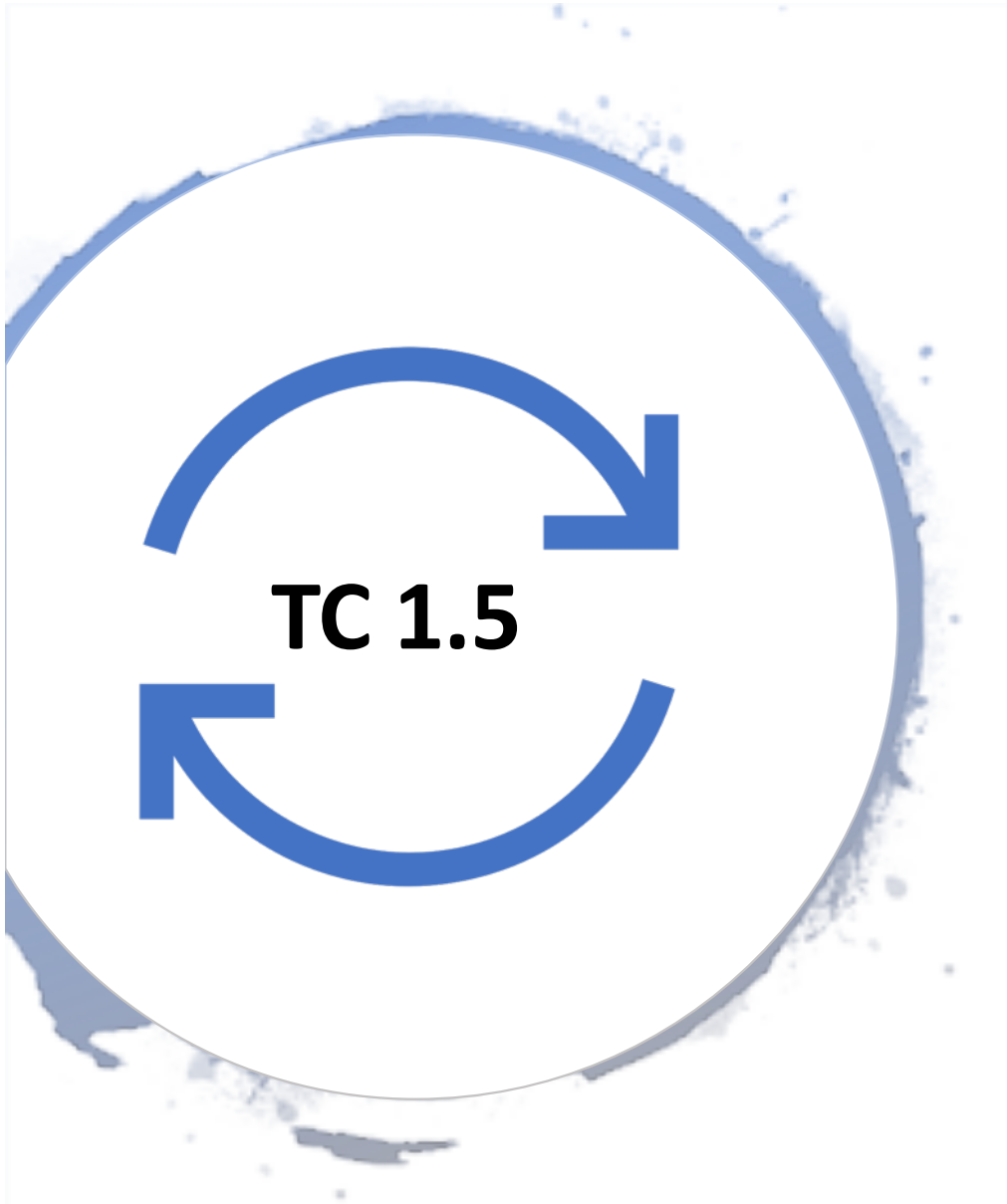



Given the State Census CSV File when correct but type incorrect Returns a custom Exception

This is a Sad Test Case to verify if the type is incorrect then exception is raised.


Given the State Census CSV File when correct but delimiter incorrect Returns a custom Exception

This is a Sad Test Case to verify if the file delimiter is incorrect then exception is raised.

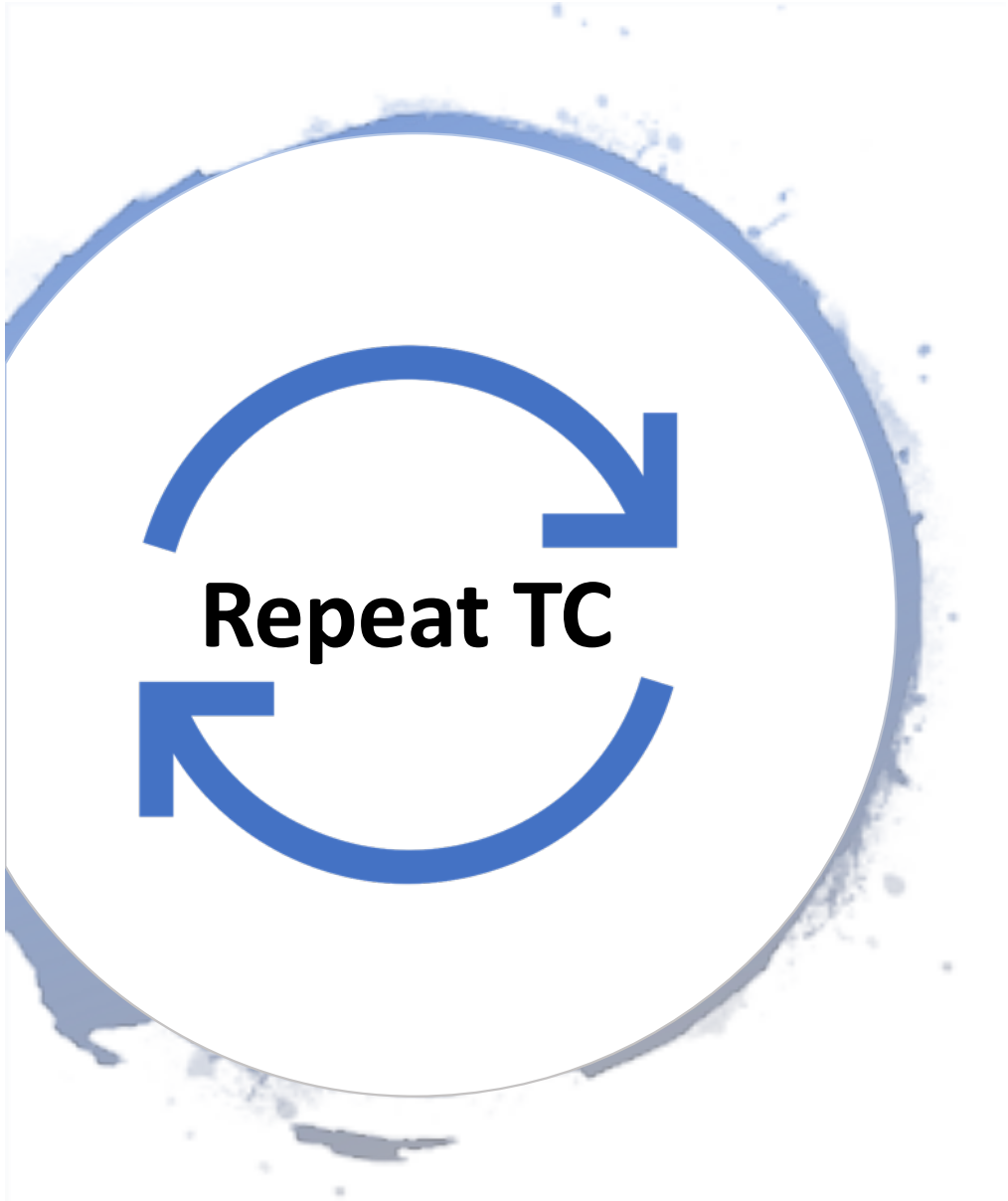



Given the State Census CSV File when correct but csv header incorrect Returns a custom Exception

This is a Sad Test Case to verify if the header is incorrect then exception is raised.


# Ability for the analyser to load the Indian States Code Information from a csv file

- Create CSVStates Class to load the CSV Data
- Use Iterator to load the data
- Check with StateCensusAnalyser to ensure number of record matches

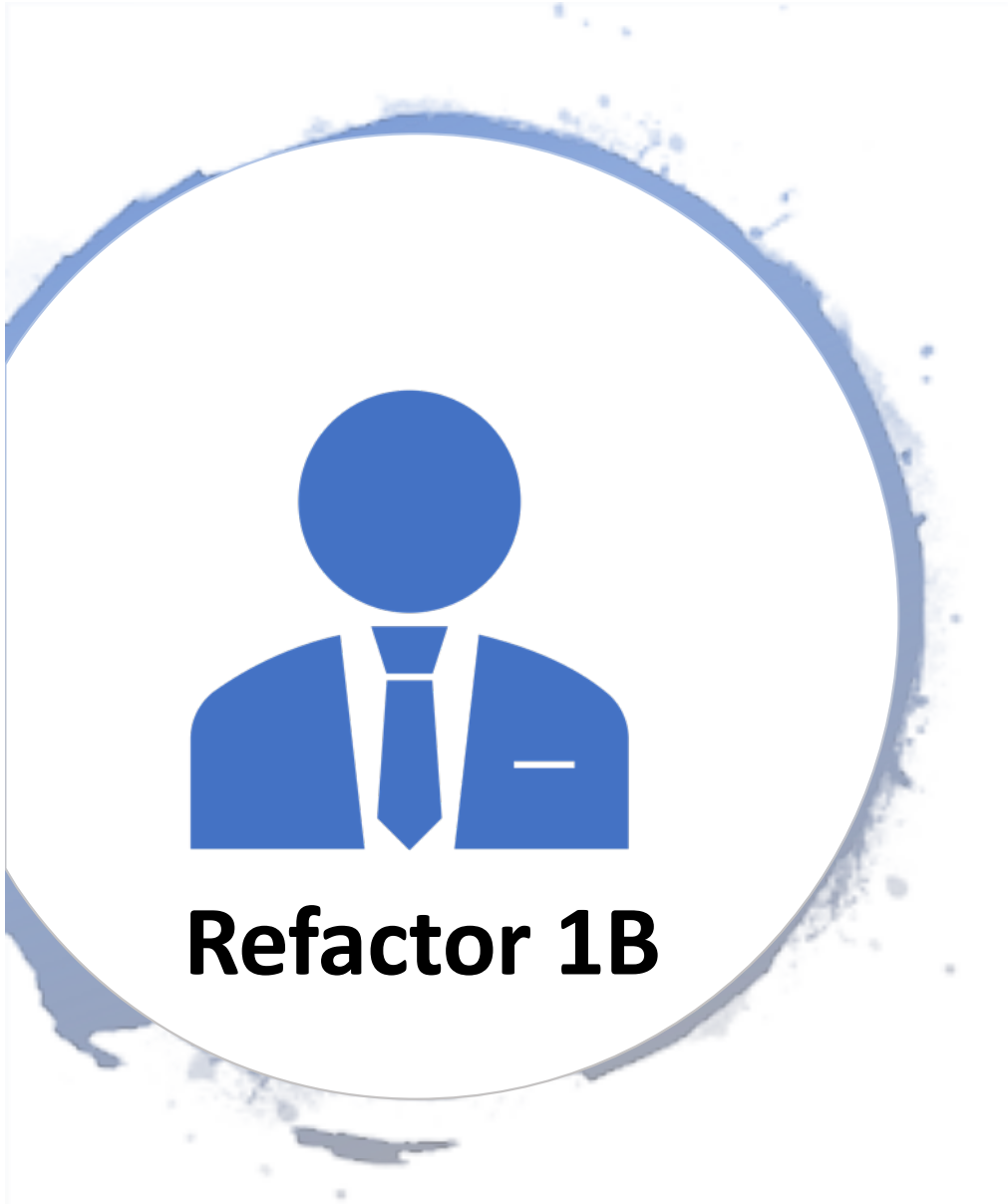



# Like UC 1 repeat all the 5 TC for UC 2

Repeat all the Happy and Sad Test Cases


# Refactor the code as DRY was violated to extract data from OpenCSV

- **Note:**
- Ensure all Test Cases are Refactored and working
- Note typically for refactored code no new test cases are needed




Refactor the code as DRY was violated to get count from CSV Iterator

- **Note:**
- Use Generics Type to extract the count


# Refactored Code to ensure Single Responsibility Principle (SRV) for Analyser and use Delegation Principle to handle CSV Files

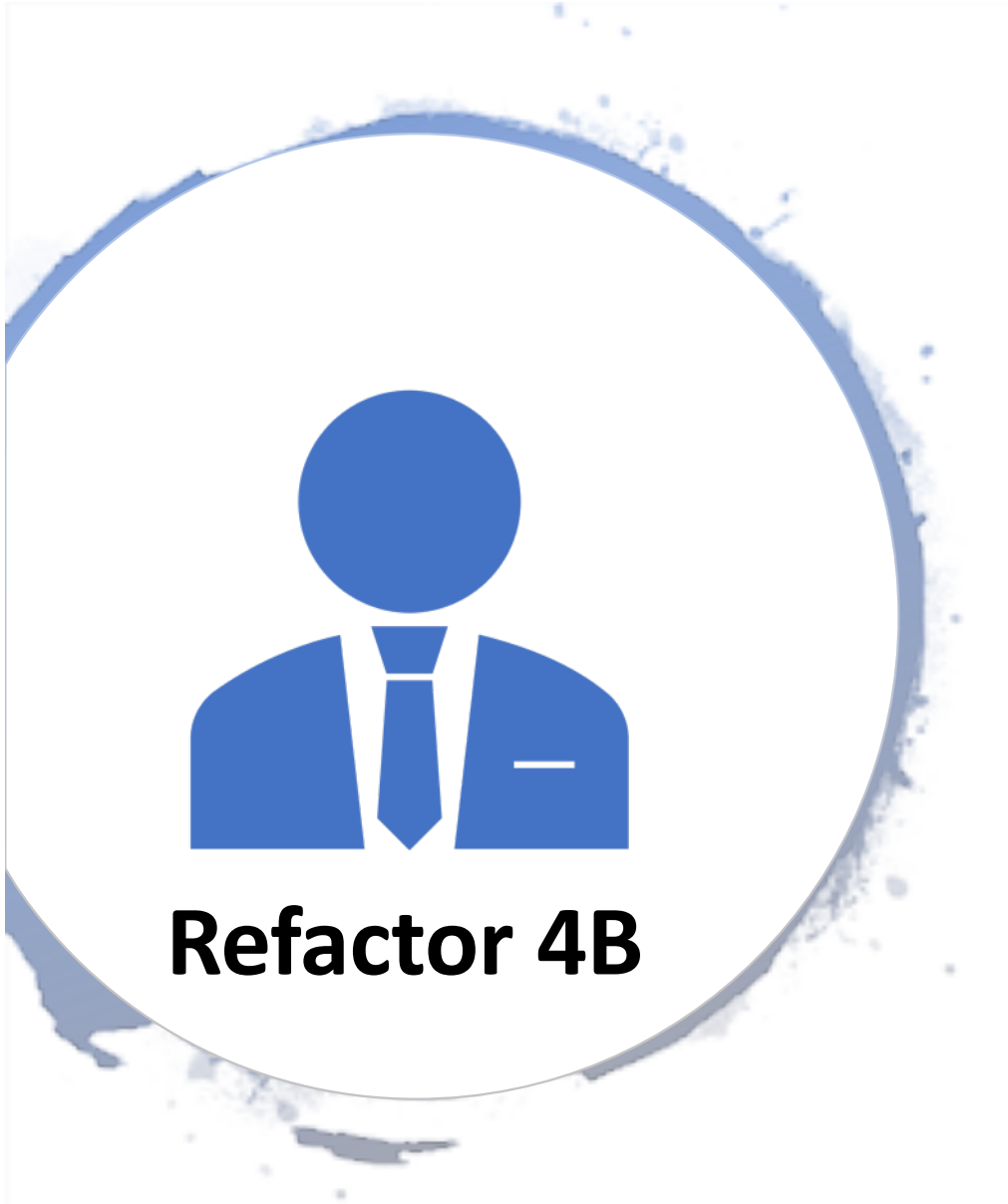
- **Note:**
- Ensure all Test Cases are Refactored and working
- Note typically for refactored code no new test cases are needed


# Refactored Code and Introduced CSV Builder and Factory to ensure Programming for Interface then Implementation

- **Note:**
- Ensure all Test Cases are Refactored and working
- Note typically for refactored code no new test cases are needed

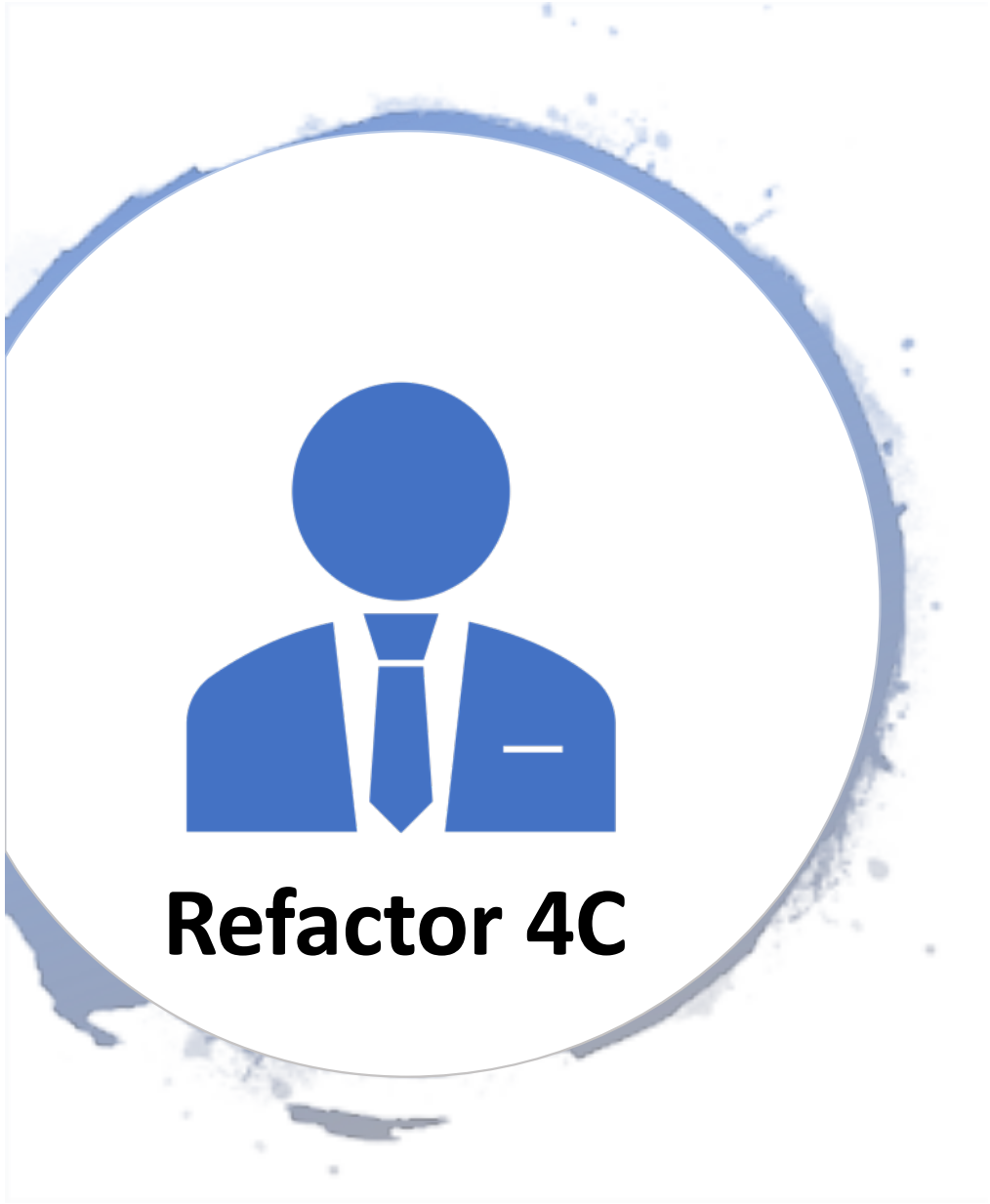

# Refactored the code to Introduce CSV Exception to ensure CSV Builder encapsulate all changes

- **Note:**
- Ensure all Test Cases are Refactored and working
- Note typically for refactored code no new test cases are needed




# Refactored the code to create a separate jar and include the jar in the Census Analyser Project

- **Note:**
- Ensure all Test Cases are Refactored and working
- Note typically for refactored code no new test cases are needed

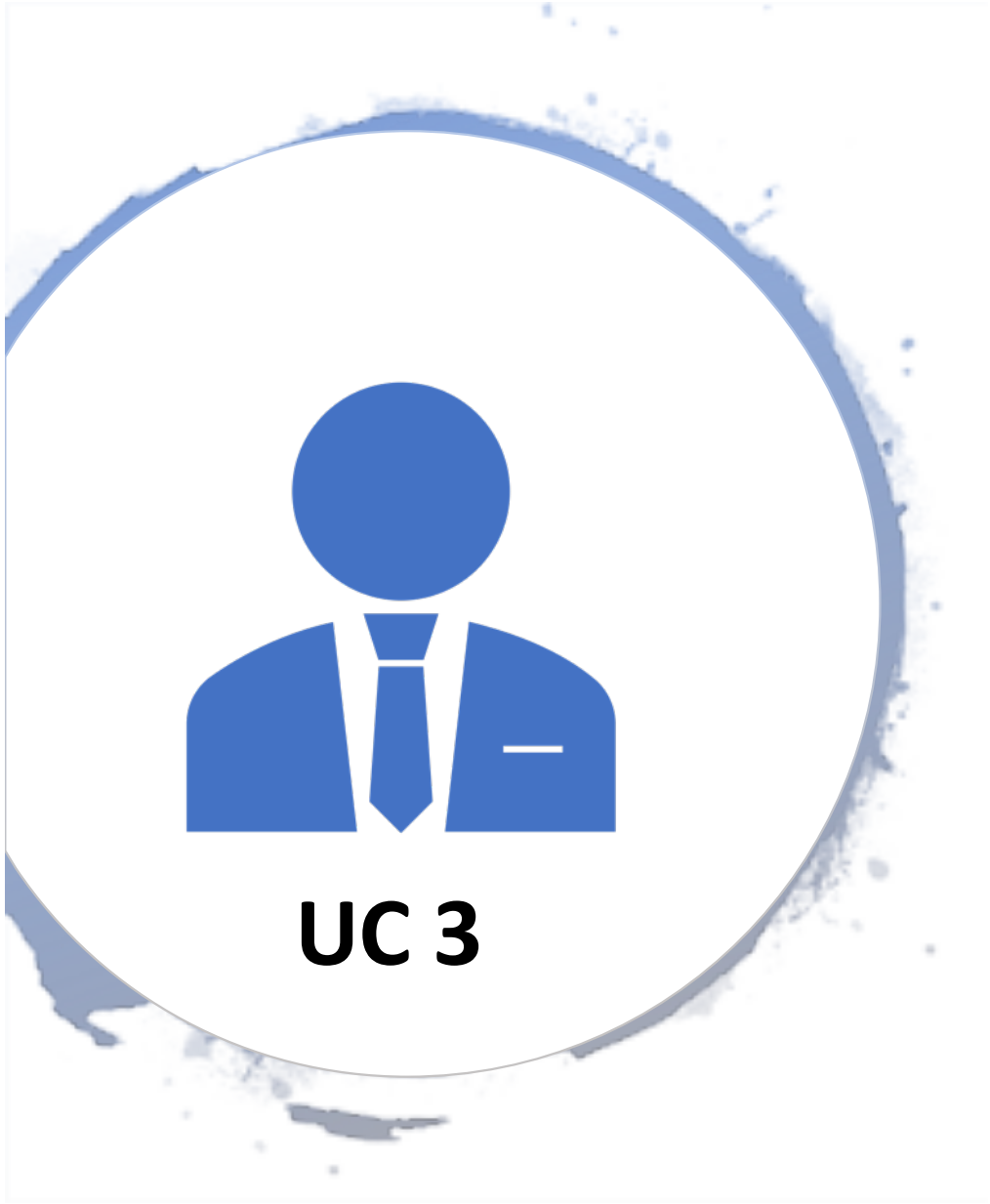



As we have used OpenCSV Library there is also Commons CSV Library. Also implement Commons CSV

- **Note:**
- Commons CSV Download

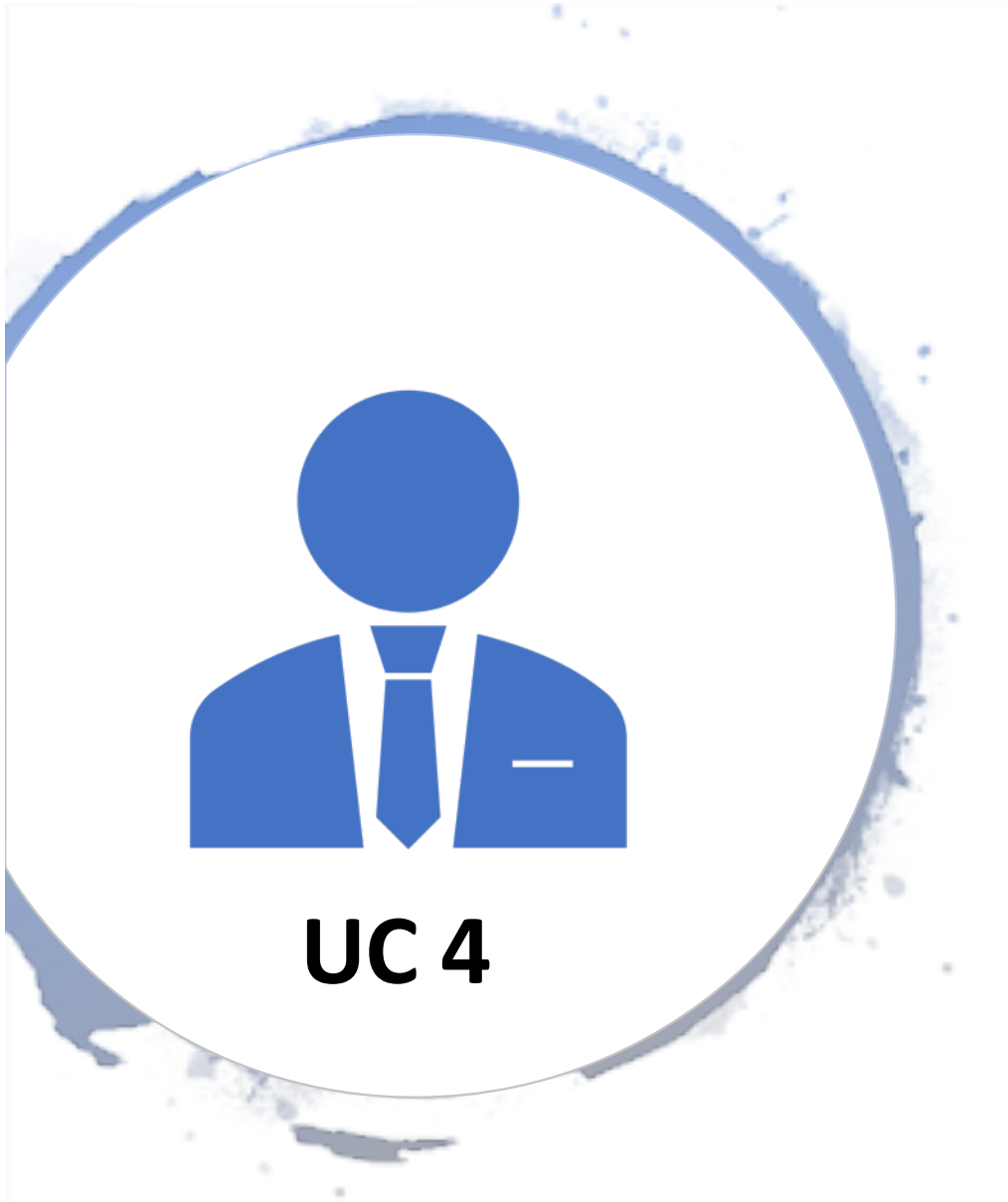

# Refactored Code to take in List of Census Data instead of Iterating through the File

- **Note:**
- Ensure all Test Cases are Refactored and working
- Note typically for refactored code no new test cases are needed




# Ability for Analyser to report the State Census Data in a Json Format according to State alphabetical order

- Create a Sorting function in the Analyser to Sort the States in the alphabetical order
- Return the Sorted Output in a JSON Format
- For Test Case to pass check the start and end states
- Write Test Cases for all the Error cases to ensure 100% code coverage

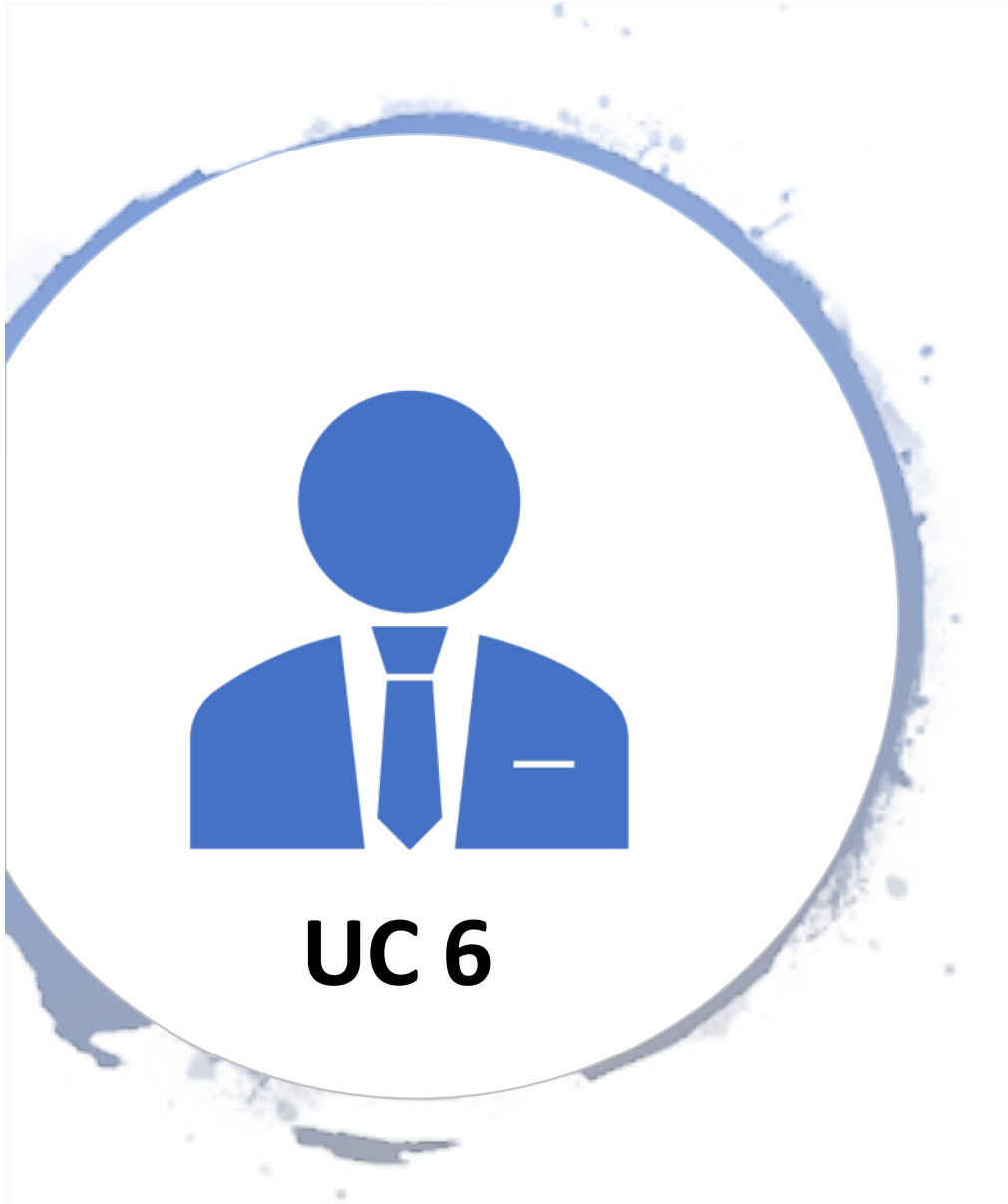



# Ability for Analyser to report the State Census Data in a Json Format as per State Code in an alphabetical order

- Create a Sorting function in the Analyser to Sort as per state code in an alphabetical order
- Return the Sorted Output in a JSON Format
- For Test Case to pass check the start and end states
- Write Test Cases for all the Error cases to ensure 100% code coverage


# Ability to report the State Census Data in a Json Format from most populous state to the least one

- Create a Sorting function in the Analyser to Sort by State population
- Write the Sorted Output into a JSON File
- For Test Case to pass report number of states sorted
- Write Test Cases for all the Error cases



**UC 6**

# Ability to report the State Census Data in a Json File from most population density state to the least one

- Return JSON Output sorted based on Population Density


# Ability to report the State Census Data in a Json format from Largest State by Area to the smallest state

- Write the Sorted Output into a JSON File

BridgeLabz

Employability Delivered

# Thank You