KLA India Software Workshop 2022

3/8/2022

Agenda

Problem Overview

Terminology

Milestones

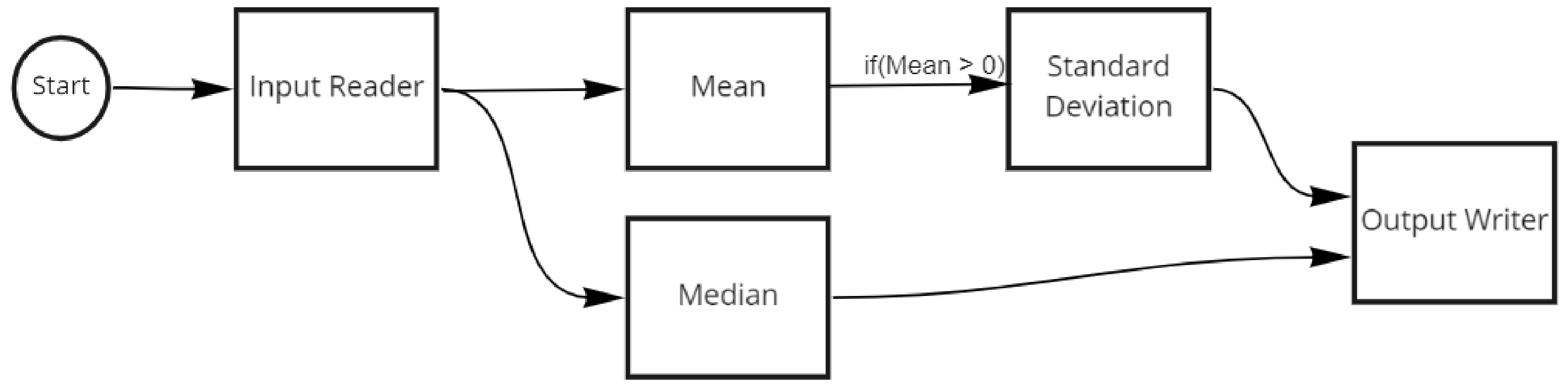
Results Validator

Workflow Framework

Definition

A **Workflow** is a sequence of tasks that processes a set of data or performs a set of connected activities

A **Workflow Framework** is a standalone application or an embedded framework responsible for reliably executing workflows



Simple Workflow Example

The steps are defined in a YAML file called a workflow configuration

Problem Statement

Design and Implement a generic Workflow Framework

A. To perform predefined tasks in a specified order

   I. Sequential

   II. Concurrent

   III. Nested

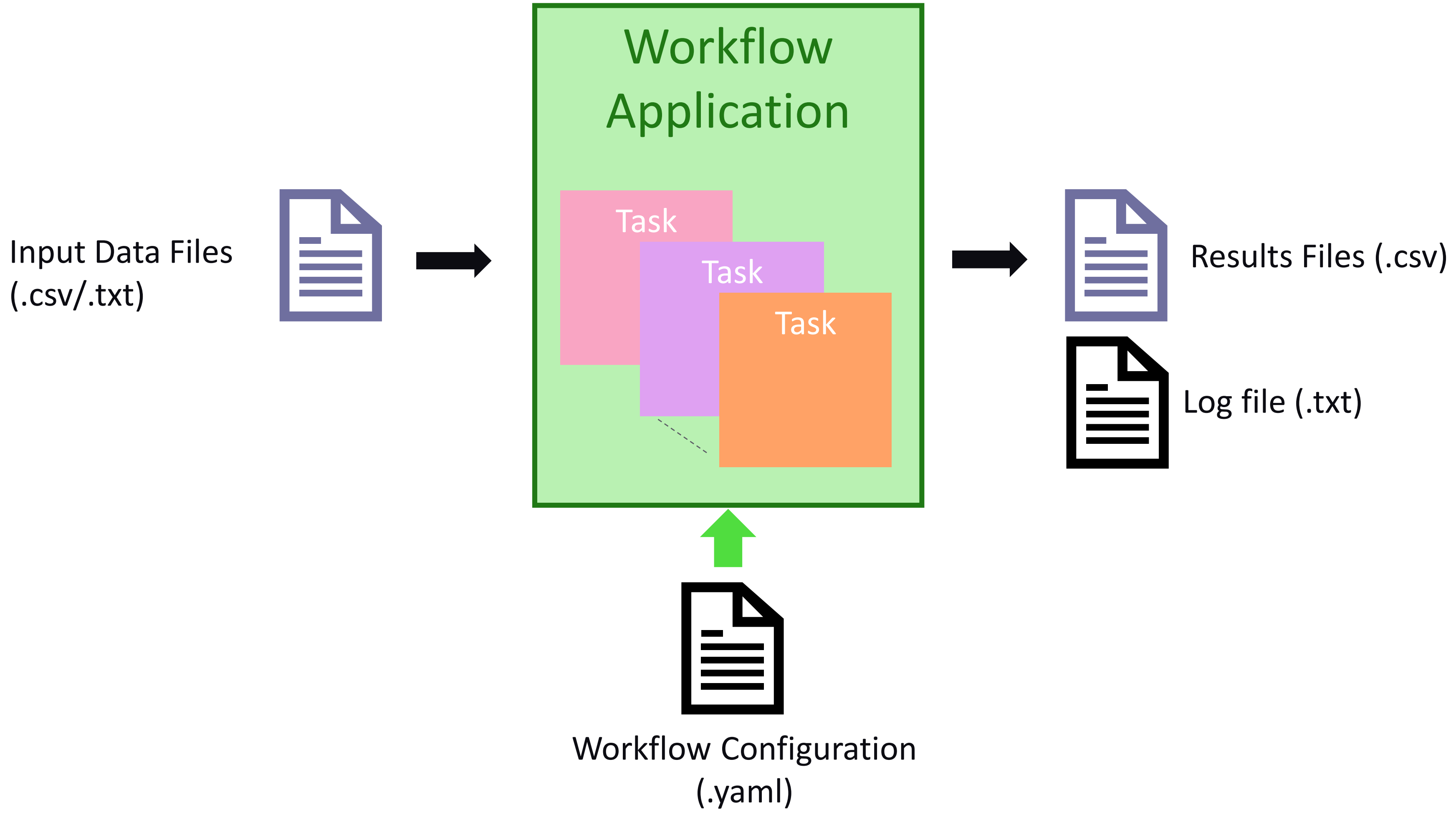
   IV. Conditional

B. To perform defect processing using the framework

   I. Defect Binning

Overview

Solution Environment with Input/Output



Agenda

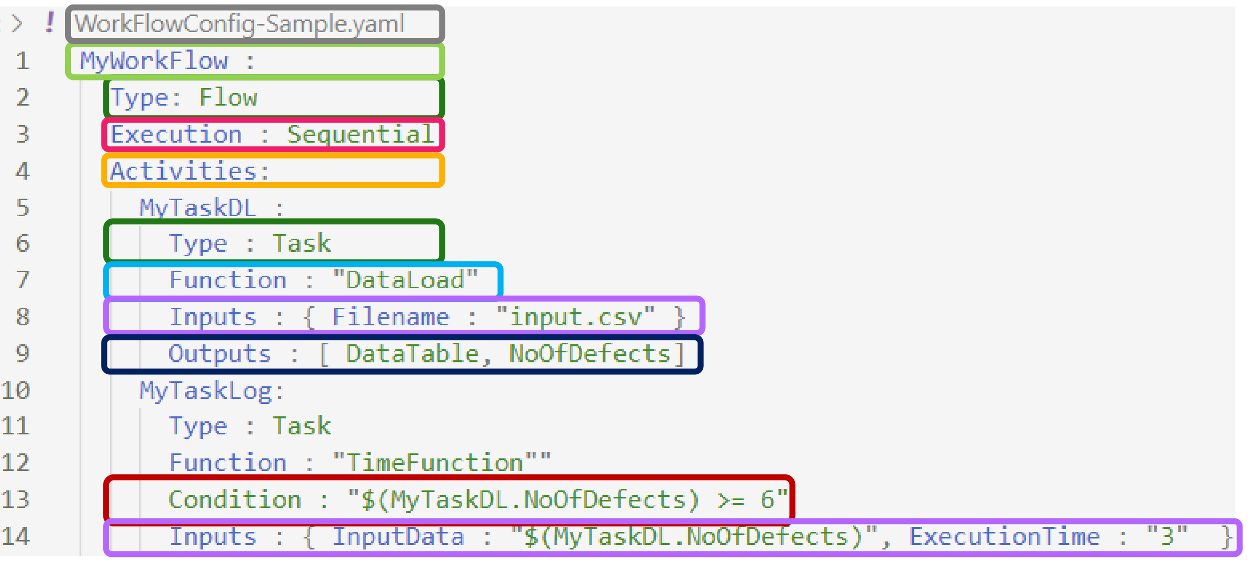
Problem Overview

Terminology

Milestones

Results Validator

Understanding Workflow Configuration File

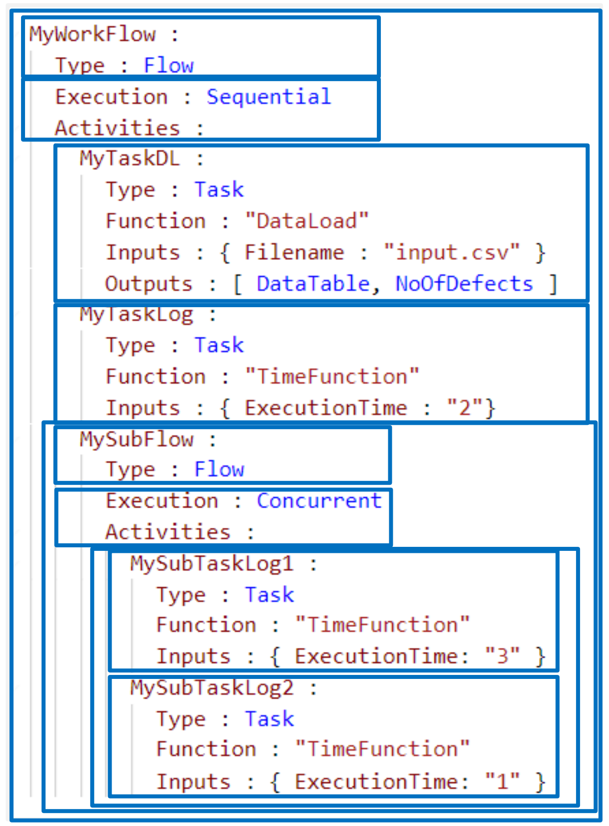


```yaml
WorkFlowConfig-Sample.yaml
MyWorkFlow :
  Type: Flow
  Execution : Sequential
  Activities:
    MyTaskDL :
      Type : Task
      Function : "DataLoad"
      Inputs : { Filename : "input.csv" }
      Outputs : [ DataTable, NoOfDefects]
    MyTaskLog:
      Type : Task
      Function : "TimeFunction""
      Condition : "$(MyTaskDL.NoOfDefects) >= 6"
      Inputs : { InputData : "$(MyTaskDL.NoOfDefects)", ExecutionTime : "3"  }
```

Keyword	Definition
Type	Represents the type of workflow unit {Flow, Task}
Flow	A collection of Activities with Execution approach
Execution	Sequential –Tasks/Flows must be executed sequentially Concurrent – Tasks/Flows can be executed concurrently
Activities	Contains multiple Flows and/or Tasks
Task	Smallest piece of work that will be performed by calling a function/method with inputs, outputs and conditions
Function	Name of the function/method to be called by the task. Here, MyTaskDL executes the function DataLoad
Inputs	Input arguments as key/value pairs Example for simple argument – {Filename : “input.csv”} Example for reference argument – { InputData : "$(MyTaskDL.NoOfDefects)" } Reference Argument type is used to pass data between tasks In this ex, “NoOfDefects” is output from DataLoad function which is fed as input to TimeFunction ExecutionTime – Wait/Sleep time (sec.) to be introduced by the Task between Start & End logs
Outputs	Return parameters such as In-memory data structure or an integer
Condition	Optional. Constrains the task to be executed only if the declared condition is met. The dependent task should wait till the completion of the task generating the reference argument

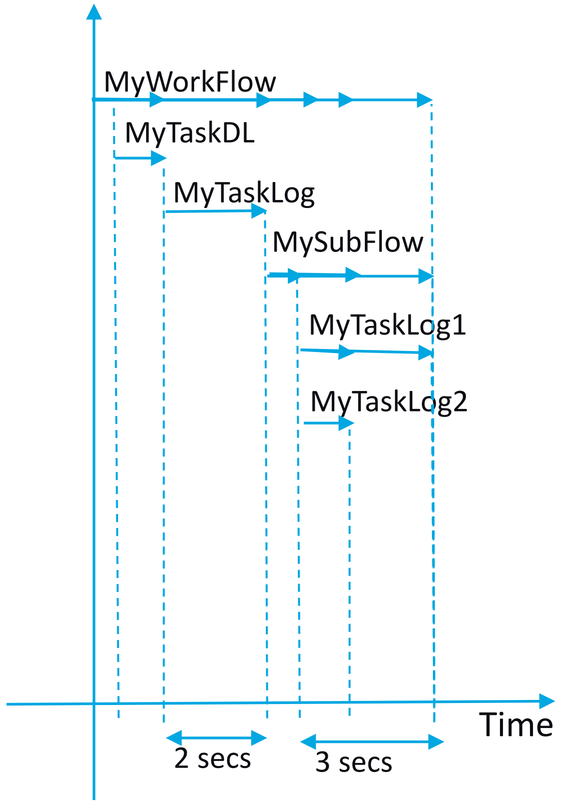
Understanding Execution Sequence & Output Log

Sample Workflow Configuration



```
MyWorkFlow :
  Type : Flow
  Execution : Sequential
  Activities :
    MyTaskDL :
      Type : Task
      Function : "DataLoad"
      Inputs : { Filename : "input.csv" }
      Outputs : [ DataTable, NoOfDefects ]
    MyTaskLog :
      Type : Task
      Function : "TimeFunction"
      Inputs : { ExecutionTime : "2"}
    MySubFlow :
      Type : Flow
      Execution : Concurrent
      Activities :
        MySubTaskLog1 :
          Type : Task
          Function : "TimeFunction"
          Inputs : { ExecutionTime: "3" }
        MySubTaskLog2 :
          Type : Task
          Function : "TimeFunction"
          Inputs : { ExecutionTime: "1" }
```

Execution Timing Sequence



Expected output in log file

```
2022-02-09 11:30:00.000000;MyWorkFlow Entry
2022-02-09 11:30:01.000000;MyWorkFlow.MyTaskDL Entry
2022-02-09 11:30:01.000000;MyWorkFlow.MyTaskDL Executing DataLoad(input.csv)
2022-02-09 11:30:01.000000;MyWorkFlow.MyTaskDL Exit
2022-02-09 11:30:02.000000;MyWorkFlow.MyTaskLog Entry
2022-02-09 11:30:02.000000;MyWorkFlow.MyTaskLog Executing TimeFunction(2)
2022-02-09 11:30:04.000000;MyWorkFlow.MyTaskLog Exit
2022-02-09 11:30:04.000000;MyWorkFlow.MySubFlow Entry
2022-02-09 11:30:04.000000;MyWorkFlow. MySubFlow.MySubTaskLog1 Entry
2022-02-09 11:30:04.000000; MyWorkFlow.MySubFlow.MySubTaskLog2 Entry
2022-02-09 11:30:04.000000;MyWorkFlow. MySubFlow.MySubTaskLog1 Executing TimeFunction(3)
2022-02-09 11:30:04.000000;MyWorkFlow.MySubFlow.MySubTaskLog2 Executing TimeFunction(1)
2022-02-09 11:30:05.000000;MyWorkFlow.MySubFlow.MySubTaskLog2 Exit
2022-02-09 11:30:07.000000;MyWorkFlow.MySubFlow.MySubTaskLog1 Exit
2022-02-09 11:30:07.000000;MyWorkFlow.MySubFlow Exit
2022-02-09 11:30:07.000000;MyWorkFlow Exit
```

The log trace for Executing statement must contain function name and argument values printed. Notice the highlighted traces

MyWorkFlow exits as all its activities are finished

Task Functions

Below listed task functions required to be implemented

- TimeFunction
- DataLoad
- Binning
- MergeResults
- ExportResults

Task Functions

TimeFunction

Declaration of TimeFunction task instance in configuration yaml file

```
MyTask2 :
  Type : Task
  Function : “TimeFunction"
  Inputs : { FunctionInput : "Task2_InputValue", ExecutionTime : "2"  }
```

Description: *TimeFunction* is a function implementing Wait/Sleep time specified by *ExecutionTime*

Inputs:

FunctionInput: An input argument
ExecutionTime: Wait/Sleep time (sec.) to be introduced by the Task between Entry & Exit log traces

Outputs: None

Task Functions

DataLoad

Declaration of DataLoad task instance in configuration yaml file

```
MyTaskDL :
  Type : Task
  Function : "DataLoad"
  Inputs : { Filename : "input.csv" }
  Outputs : [ DataTable, NoOfDefects]
```

***Description**: DataLoad* function reads input CSV file into an in-memory data structure

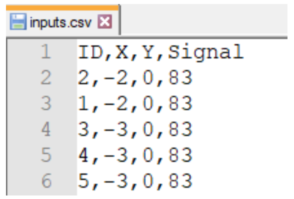
Inputs:

Filename: Specifies the input CSV file for the function

Outputs:

DataTable: In-memory data structure holding the data loaded from input CSV file
NoOfDefects: Count of defects (rows) read from the input CSV file

How to pass data between tasks?



```
ID,X,Y,Signal
2,-2,0,83
1,-2,0,83
3,-3,0,83
4,-3,0,83
5,-3,0,83
```

Declaration of data/variable chaining between functions in configuration file

```
MyTask1 :
  Type : Task
  Function : "DataLoad"
  Inputs : { Filename : "input.csv" }
  Outputs : [ DataTable, NoOfDefects ]
MyTask2 :
  Type : Task
  Function : “TimeFunction"
  Inputs : { InputData :  "$(MyTask1.NoOfDefects)", ExecutionTime : "2"   }
```

Expected output Log File

```
2022-02-12 10:00:00.000000;MyTask1 Entry
2022-02-12 10:00:00.000000; MyTask1 Executing DataLoad (input.csv)
2022-02-12 10:00:01.000000; MyTask1 Exit
2022-02-12 10:00:02.000000; MyTask2 Entry
2022-02-12 10:00:02.000000; MyTask2 Executing TimeFunction(5,2)
2022-02-12 10:00:04.000000; MyTask2 Exit
```

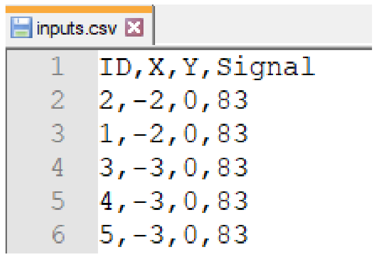
In this example,

- output of MyTask1 is fed as input to MyTask2

  $ symbol – used to reference a variable/output of another task in the Workflow configuration

Note: MyTask2 has dependency on output variable *NoOfDefects* from MyTask1.
Hence, MyTask2 must wait for completion of MyTask1. This dependency should be enforced even if Execution is *Concurrent*

Conditional task execution



inputs.csv

```
ID,X,Y,Signal
2,-2,0,83
1,-2,0,83
3,-3,0,83
4,-3,0,83
5,-3,0,83
```

Declaration of task execution based on conditional expression using variables

```
MyTask1 :
  Type : Task
  Function : "DataLoad"
  Inputs : { Filename : "input.csv" }
  Outputs : [ DataTable, NoOfDefects ]
MyTask2 :
  Type : Task
  Function : "TimeFunction"
  Condition : "$(MyTask1.NoOfDefects) >= 6"
  Inputs : { InputData :  "$(MyTask1.NoOfDefects)", ExecutionTime : "2" }
```

Expected entries in the Log File (if conditional expression pass)

```
2022-02-12 10:00:00.000000;MyTask1 Entry
2022-02-12 10:00:00.000000; MyTask1 Executing DataLoad (input.csv)
2022-02-12 10:00:01.000000; MyTask1 Exit
2022-02-12 10:00:02.000000; MyTask2 Entry
2022-02-12 10:00:02.000000; MyTask2 Executing TimeFunction (5, 2)
2022-02-12 10:00:04.000000; MyTask2 Exit
```

Expected entries in the Log File (if conditional expression fails)

```
2022-02-12 10:00:00.000000; MyTask1 Entry
2022-02-12 10:00:00.000000; MyTask1 Executing DataLoad (input.csv)
2022-02-12 10:00:01.000000; MyTask1 Exit
2022-02-12 10:00:02.000000; MyTask2 Entry
2022-02-12 10:00:02.000000; MyTask2 Skipped
2022-02-12 10:00:02.000000; MyTask2 Exit
```

In this example, MyTask2 execution depends on the value of *NoOfDefects* produced by MyTask1.

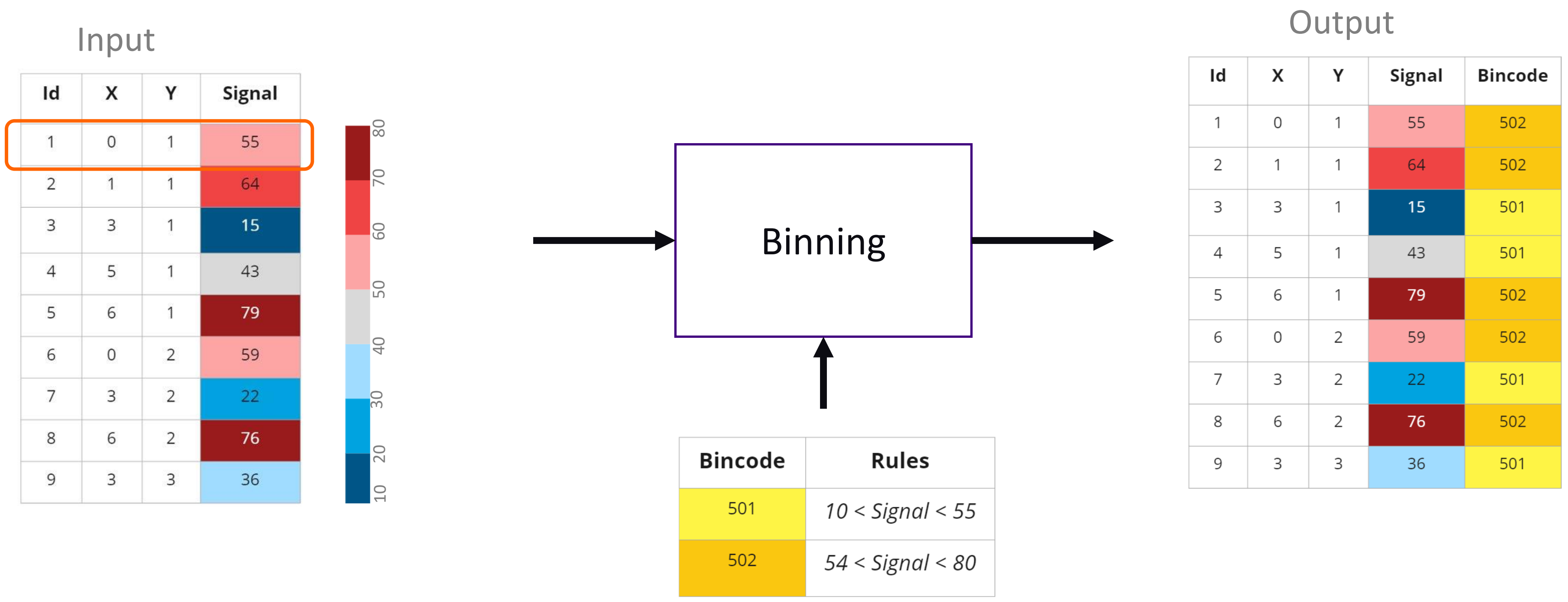
If condition succeeds → MyTask2 executed
If condition fails → MyTask2 skipped

Here, Given input.csv has only 5 defects (rows), the condition fails and hence MyTask2 skipped

Note: MyTask2 has dependency on output variable *NoOfDefects* from MyTask1.
Hence, MyTask2 must wait for completion of MyTask1. This dependency should be enforced even if Execution is *Concurrent*

Binning Explained

Binning is a way to group a number of more or less continuous values into a smaller number of "bins"



Input

Id	X	Y	Signal
1	0	1	55
2	1	1	64
3	3	1	15
4	5	1	43
5	6	1	79
6	0	2	59
7	3	2	22
8	6	2	76
9	3	3	36

Bincode	Rules
501	*10 < Signal < 55*
502	*54 < Signal < 80*

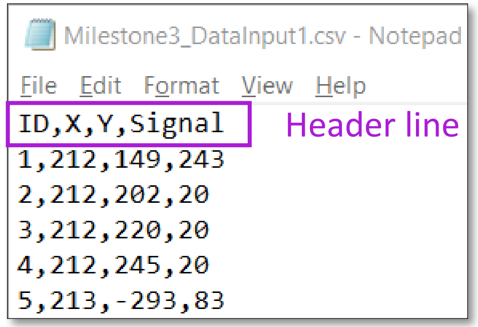
Output

Id	X	Y	Signal	Bincode
1	0	1	55	502
2	1	1	64	502
3	3	1	15	501
4	5	1	43	501
5	6	1	79	502
6	0	2	59	502
7	3	2	22	501
8	6	2	76	502
9	3	3	36	501

- Each row in the input represents one **defect** with unique ID, X, Y, and Signal
- Signal is a measure of size of the defect

Input Files

Data (.csv)

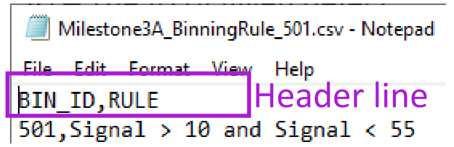

Milestone3_DataInput1.csv - Notepad
File Edit Format View Help
Header line

```
ID,X,Y,Signal
1,212,149,243
2,212,202,20
3,212,220,20
4,212,245,20
5,213,-293,83
```

Each row in DataInput.csv represents a defect with below information in 4 columns:

- ID – The unique ID of defect
- X – X coordinate of defect
- Y – Y coordinate of defect
- Signal – Size of the defect at given XY location

Binning Rules (.csv)


Milestone3A_BinningRule_501.csv - Notepad
File Edit Format View Help
Header line

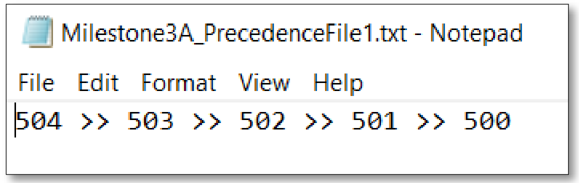
```
BIN_ID,RULE
501,Signal > 10 and Signal < 55
```

A row in the BinningRule.csv contains the following information about a rule:

- BIN_ID – The Bin ID to be assigned to a defect if the rule criteria met
- RULE – Rule criteria to be used for binning the defect

In this example, defects with 10 < *Signal* < 55 will be assigned Bin ID of 501

Precedence (.txt)


Milestone3A_PrecedenceFile1.txt - Notepad
File Edit Format View Help

```
504 >> 503 >> 502 >> 501 >> 500
```

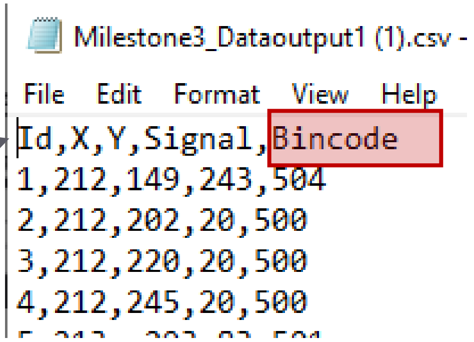
Specifies the order of precedence of Bin ID to be considered during results merging in MergeResults function.

In this example, 504 takes the highest precedence while 500 the lowest.

Output Files

Binning task will always generate CSV results output file in this format

Header line
Strict adherence



```
Id,X,Y,Signal,Bincode
1,212,149,243,504
2,212,202,20,500
3,212,220,20,500
4,212,245,20,500
```

- **ID** – The ID of given defect (same as DataInput file content)
- **X** – X coordinate of a given defect (same as DataInput file content)
- **Y** – Y coordinate of a given defect (same as DataInput file content)
- **Signal** – Size of the defect at given XY location (same as DataInput file content)
- **Bincode** – Bin ID result assigned by Binning algo routine

Task Functions

Binning

Declaration of Binning task instance in configuration yaml file

```
MyLoadData :
  Type : Task
  Function : "DataLoad"
  Inputs : { Filename : "input.csv" }
  Outputs : [ DataTable, NoOfDefects  ]
BinningFor500Task :
  Type : Task
  Function : "Binning"
  Inputs : { RuleFileName : "rule.csv", DataSet1 : "$(MyLoadData.DataTable)" }
  Outputs : [ BinningResultsTable, NoOfDefects ]
```

Description*: Binning* function implements algorithm to process the input defect list and assign Bin ID based on rules described in input file

Inputs:

RuleFilename: Specifies the Binning rule file in CSV format
DataSet1: In memory data structure with defect list to be processed and binned

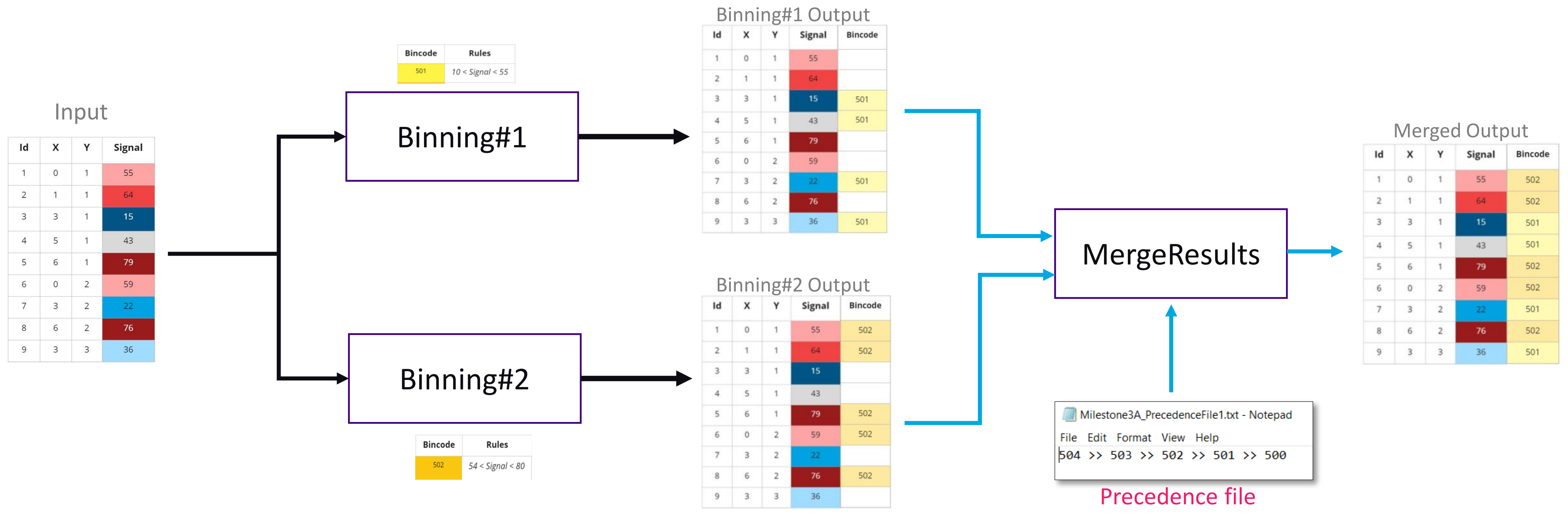
Outputs:

BinningResultsTable: In-memory data structure with binning results appended to the input *DataSet*
NoOfDefects: Count of defects in the BinningResultsTable

Processing multiple Bin Rules

What if there are multiple binning rule files to be processed?

Workflow configuration can have multiple tasks calling Binning function, once for each rule file



Input

Id	X	Y	Signal
1	0	1	55
2	1	1	64
3	3	1	15
4	5	1	43
5	6	1	79
6	0	2	59
7	3	2	22
8	6	2	76
9	3	3	36

Bincode	Rules
501	*10 < Signal < 55*

Bincode	Rules
502	*54 < Signal < 80*

Binning#1 Output

Id	X	Y	Signal	Bincode
1	0	1	55	
2	1	1	64	
3	3	1	15	501
4	5	1	43	501
5	6	1	79	
6	0	2	59	
7	3	2	22	501
8	6	2	76	
9	3	3	36	501

Binning#2 Output

Id	X	Y	Signal	Bincode
1	0	1	55	502
2	1	1	64	502
3	3	1	15	
4	5	1	43	
5	6	1	79	502
6	0	2	59	502
7	3	2	22	
8	6	2	76	502
9	3	3	36	

Merged Output

Id	X	Y	Signal	Bincode
1	0	1	55	502
2	1	1	64	502
3	3	1	15	501
4	5	1	43	501
5	6	1	79	502
6	0	2	59	502
7	3	2	22	501
8	6	2	76	502
9	3	3	36	501

Milestone3A_PrecedenceFile1.txt - Notepad

File Edit Format View Help

```
504 >> 503 >> 502 >> 501 >> 500
```

Precedence file

Task Functions

MergeResults

Declaration of MergeResults task instance in configuration yaml file

```
MergeBinningResult :
  Type : Task
  Function : "MergeResults"
  Inputs : { PrecedenceFile: "Precedence.txt",  DataSet1 : "$(BinningFor500Task.BinningResultsTable)", DataSet2: "$(BinningFor501Task.BinningResultsTable)" }
  Outputs : [ MergedResults, NoOfDefects ]
```

Description: *MergeResults* function implements algorithm to merge results from one or more *Binning* tasks and produce combined/merged results

Inputs:

PrecedenceFile: Specifies order of precedence to be considered in assigning final Bin ID for each defect
DataSet1: In memory data structure of defects with partial BIN results
DataSet2…DataSetn: Optional, *a*dditional Defect lists with partial BIN results

Outputs:

MergedResults: In-memory data structure with combined results with final Bin ID
NoOfDefects: Count of defects in the MergedResults

Task Functions

ExportResults

Declaration of ExportResults task instance in configuration yaml file

```
ExportResult :
 Type : Task
 Function : "ExportResults"
 Inputs : { FileName : "ExportResult.csv", DefectTable : "$(MergeBinningResults.MergedResults)” }
```

Description*:* *ExportResults* implements the task of exporting results to a file in csv format

Inputs:

FileName*:* Output CSV file name
DefectTable: In-memory data structure containing results

Outputs:

None

Agenda

Problem Overview

Terminology

Milestones

Results Validator

Milestones and ground rules

- Milestones
  - All different data sets should be solved in sequence
    - These data sets are sequenced in the order of increasing feature set and complexity
  - Clean coding practices with modularity and readability adopted
  - Code shall get updated to the GitHub repository (created by students) on a per hour basis
    - The code from GitHub shall be the reference for validation
- Rules
  - Choose any programming language of your choice to solve the problem at hand
  - The log file and/or output csv files from each milestone shall get validated using validator program with the help of KLA (link will be given during the workshop)

Milestone#1: Sequential, Concurrent tasks

- Goals
  - Create base framework to parse workflow configuration file and execute tasks in sequential and/or concurrent fashion as defined
  - Generate execution log file as per format defined in Terminology slides
- Milestone1A
  - Input: Milestone1A.yaml
  - Output: Milestone1A.txt having log traces of execution flow
- Milestone1B
  - Input: Milestone1B.yaml
  - Output: Milestone1B.txt having log traces of execution flow

Milestone#2: Conditions and Passing data between tasks

- Goals
  - Extend the framework to parse and execute tasks with Conditions and passing data between tasks
  - Generate execution log file as per format defined in Terminology slides
- Milestone2A
  - Input: Milestone2A.yaml
  - Output: Milestone2A.txt having log traces of execution flow
- Milestone2B
  - Input: Milestone2B.yaml
  - Output: Milestone2B.txt having log traces of execution flow

Milestone#3: Binning and Merge

- Goals
  - Implement generic Binning algorithm to process input defect list based on the binning rules
  - Execute binning flow and merge results as specified in workflow configuration file

- Milestone3A
  - Input: Milestone3A.yaml,
    - Binning rule file, Precedence file, Input data file
  - Output: Milestone3A.csv having defect list with binning results

Agenda

Problem Overview

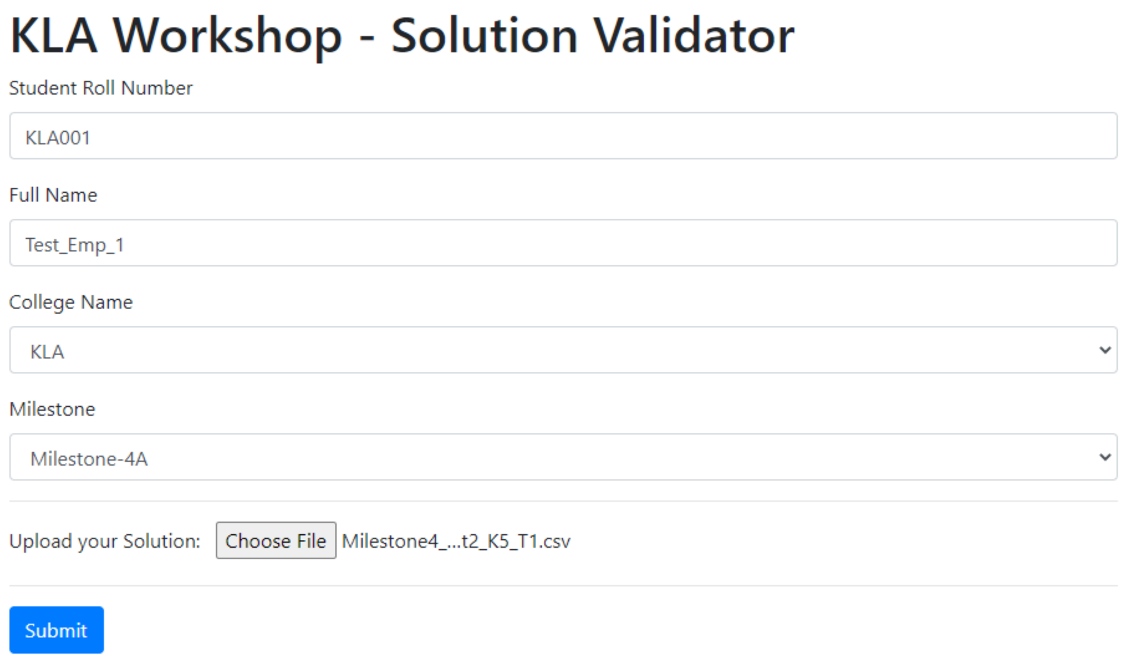
Terminology

Milestones

Results Validator

Validator

Web UI to submit output file for each level



KLA Workshop - Solution Validator

Student Roll Number: KLA001

Full Name: Test_Emp_1

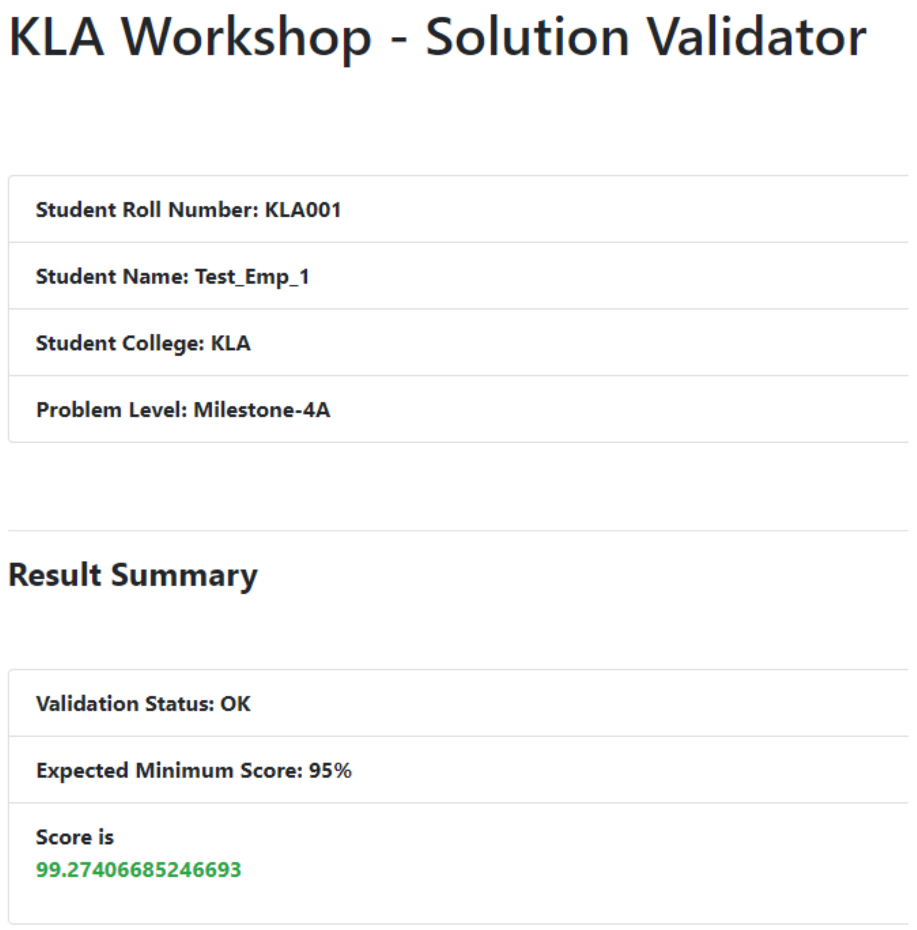
College Name: KLA

Milestone: Milestone-4A

Upload your Solution: Milestone4_...t2_K5_T1.csv

https://kla-hackathon2022.herokuapp.com/

Results summary Page



KLA Workshop - Solution Validator

Student Roll Number: KLA001

Student Name: Test_Emp_1

Student College: KLA

Problem Level: Milestone-4A

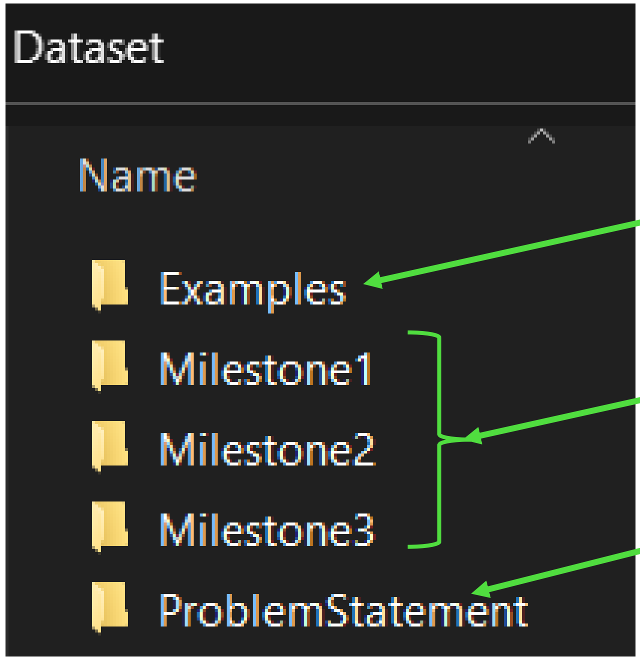
Result Summary

Validation Status: OK

Expected Minimum Score: 95%

Score is 99.27406685246693

Dataset



Dataset

Name

- Examples
- Milestone1
- Milestone2
- Milestone3
- ProblemStatement

Sample Input/Output for each milestone (Only for checking solution on student PC, do not upload to validation server)

Input data (YAML, csv, txt)

This PowerPoint, Explanation video

Thank You! Good Luck!!