Internship Report On
Automatic Data Entry In DMIS For Metrology Equipment At Semi-Conductor Laboratory

Utkarsh Jain
b17029@students.iitmandi.ac.in

Computer Science and Engineering, IIT Mandi

December 2018-Feburary 2019

IIT Mandi, Himachal Pradesh-175005

Certificate of Originality

This is to certify that Mr. Utkarsh Jain, S/O Shri Paritosh Jain, of B.tech Computer Science and Engineering under Enrollment No. B17029 of IIT Mandi, Himachal Pradesh has undergone eleven weeks of Industrial Training from 03 December, 2018 to 15 Feburary, 2019 at Semi-Conductor Laboratory(Dept of Space, ISRO). He has worked on the project titled **Automatic Data Entry In DMIS For Metrology Equipment** during the training under the guidance of Mrs. Anjali Negi.

During his tenure at Semi-Conductor Laboratory, he has been found hardworking, cooperative and has shown keen interest in the training. He was friendly towards his team and other staff members and has never been caught in any illicit activities.

Project Guide:

Mrs. Anjali Negi
Scientist / Engr. SD
IT&ND, SCL

Acknowledgement

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to Mr. Sanjay Bhatnagar for giving me the chance to do an internship at SCL. I would also like to thank the authority in my college who allowed me to do the internship and sent me the necessary documents on time.

I am highly indebted to my father and members of SCL for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing it. Furthermore, I would also like to acknowledge with much appreciation the crucial role of Mrs. Anjali Negi, who gave me the necessary guidance and permission to use all required equipment and the necessary material in the fab to complete the auto-backup and file parsing. I have to appreciate the support given by very friendly and cooperative colleague Mr. Vikas Kumar who motivated me and helped in developing the project. I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals who hepled me from time to time. I would like to extend my sincere thanks to all of them.

Utkarsh Jain
B17029
IIT Mandi

Contents

List of Figures

1 Introduction

1.1 The organization SCL

Semiconductor Laboratory
Department of Space

The Semi-Conductor Laboratory, Mohali (SCL) is a research institute of the Department of Space, Government of India. Its aims include research and development in the field of semiconductor technology.

SCL had its origin as the Semiconductor Complex Limited, a public sector undertaking of the Government of India. It came under the administrative control of Department of Space in March 2005 and has since undergone organizational restructuring to become focused on research and development. The society was registered in November 2005.

SCL is a society under the Department of Space with the main objective to undertake, aid, promote, guide and coordinate the R&D in the field of semiconductor technology, Micro-Electro-Mechanical Systems (MEMS) and process technologies relating to semiconductor processing in the existing 6" wafer fab. SCL has over the years developed and supplied a number of key VLSIs, majority of which have been Application Specific Integrated Circuits (ASICs) for high reliability applications in industrial and space sectors. Steps have been initiated to upgrade the facilities to fabricate devices in 0.25 micron or better technology.

1.2 Internship actvities

I started my internship at SCL on December 3, 2018 under the Information Technology and Networking Division(IT&ND) which takes care of DMIS, a form-submission webpage used in SCL to upload data manually and keep track of the wafer lots.

My responsibility at IT&ND was to completely automate the process of data submission on DMIS to enhance the efficiency of the fab. This involved fetching the data spit by the metrology equipment and then parsing it to extract necessary information which later is uploaded on the main database. This data holds importance in later stages of fabrication where it is used for process control and tool health monitoring.

SCL would benefit from the achieved automation. Previously, all the data was manually fed into the system which involved laborious human effort and accompanying errors thus reducing the efficiency of the fab. Now with the proposed automation, the software itself will feed the information coming directly from the equipment without any human involvement. This speeds up the overall operation of wafer processing and inspection.

During the internship I learned working in .NET framework and usage of databases in automation. I learned developing client-side interface and scripting each component in server-side. Client-side scripting was done in ASP.NET and the server-side scripts were written in VB.NET. ADO.NET was used to access data and data services from the database. I also learned about how to interact with others in the community who are very different from me.

1.2.1 Contribution

There are **twelve** measurement tools which are used extensively to generate bulk measurement data and out of these I have successfully automated **two**.

The output files spit by the tools were manually parsed by the engineers and the data was manually fed into the DMIS. This process was time inefficient and also involved the risk of human-error. I contributed towards speeding up this process by modifying the DMIS server-side script so that it automatically parses the uploaded files and pushes all the necessary data into the DBMS, hence achieving complete automation.

1.2.2 Outlining procedure

There are two networks isolated from each other which are SCL network and the Tool network. The following steps are implemented on each tool to automate them.

1. Installation of IIS on the tool to make it a FTP server.
2. FTP Script:

Listing 1: File.txt

```
        Open 10.10.98.1
        user: admin
        pwd : admin
        lcd G:/backup
        mget *.001
```

3. Create a .BAT

Listing 2: Tool.BAT

```
        FTP: -seq File.txt
```

4. Using Task Scheduler, schedule Tool.BAT file to 24 hours. This will automatically run the .BAT file in every 24 hours.

When the above steps are implemented, the client computer will automatically run the .BAT file which fetches all the data from the tool into the computer. This data is later uploaded on DMIS to be parsed.

2 Metrology Equipment

2.1 YEDI1: KLA 2139 Inspection Tool

2.1.1 Operating procedure

This inspection tool uses optical microscope for automatic wafer inspection system by comparing three dies. Its working is broadly divided into four categories.

1. **Image Acquisition System**

   - In this stage the wafer is moved under the objective in a continuous swath.
   - Thus, image formed by optics moves continuously across stationary TDI sensor.

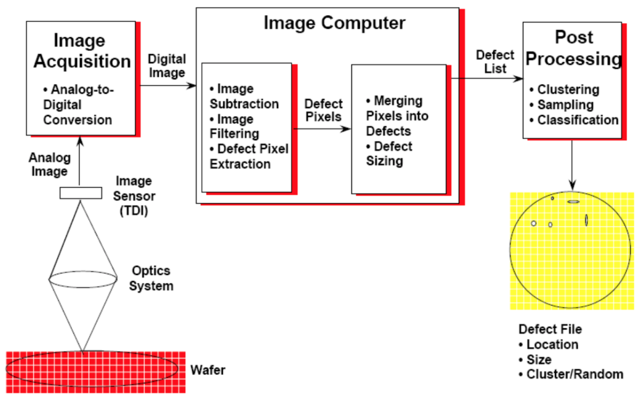


Fig. 1. Image acquisition processing

2. **Defect Detection**

   - In this stage every die(candidate die) is compared to the die on its left and on its right(reference dies). So every row needs at least three dies.
   - Since a perfect die cannot be fabricated and the tool cannot be hard-coded with all the defects, the tool has to use a comparison strategy. It compares the candidate die to the reference die to spot any differences between them. Since all the dies are meant to be same any feature which is not present in candidate die, but in reference die is marked as a defect.

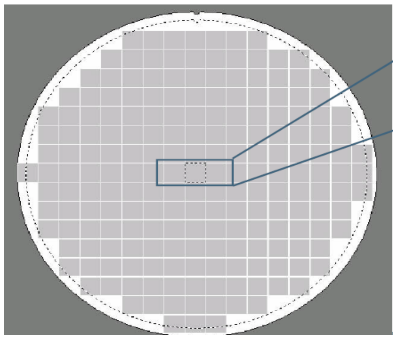


Fig. 2. Candidate and reference die

3. Difference Image Histogram

- A defect is only considered a defect when the absolute value of the gray-level difference is larger than the threshold

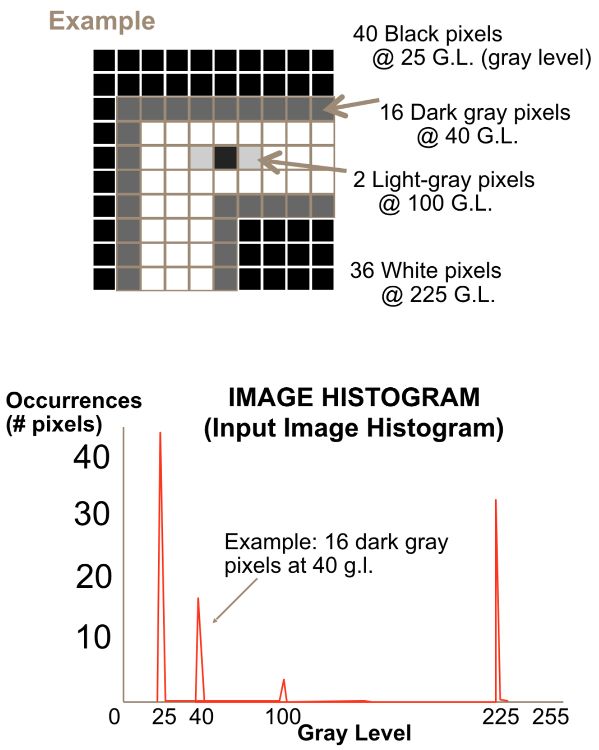
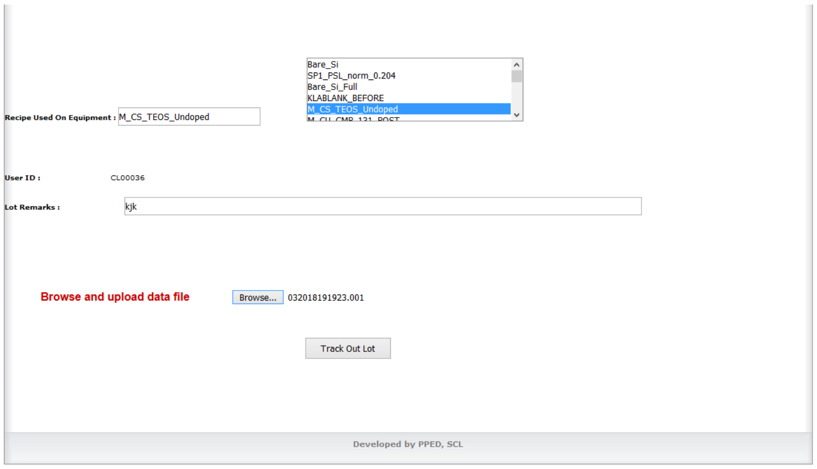


Fig. 3. Difference Image Histogram

4. **Data fetching through FTP**

    - Data from the tool is fetched using FTP. The whole procedure is explained in **procedure** section.

2.1.2 Understanding the code [Listing 3, Annexure]



Recipe Used On Equipment : M_CS_TEOS_Undoped

User ID : CL00036

Lot Remarks : kjk

Browse and upload data file 032018191923.001

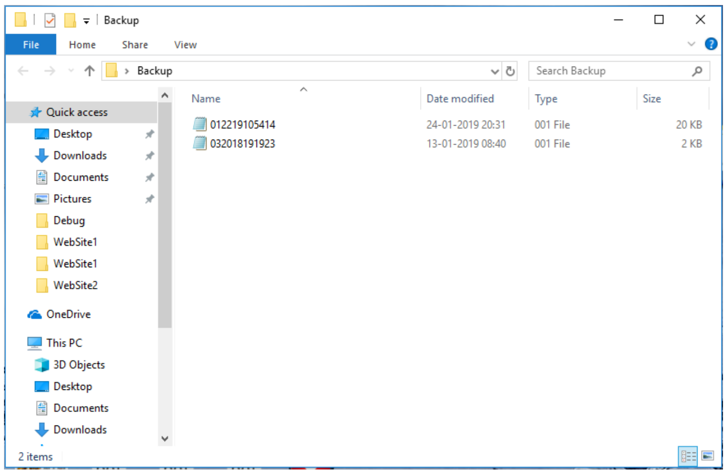
Fig. 4. Track Out Page

1. At the track-out page, the user browses the files he/she wants to upload. Each time the *upload* button is clicked, the selected file is saved in a specified folder.

    - **Lines 240-252** This part of the code handle the *upload* button event.In the variable named as *filepath1* the path of the specified folder is passed and the files gets saved there [Fig. 5].

2. After uploading all the required files, the user clicks the *Track Out Lot*. *Track Out Lot* button click fires the main script which verifies whether correct files are uploaded and then reads them to extract the information.

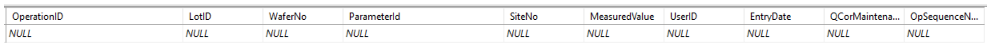
    - **Line 6-9** In this the connection string is declared which sets a connection between the web-page and the microsoft SQL server. The configuration of the connection string is done in the web config. file. The connection is then set open.
    - **Line 25-62** It iterates through all the files in filepath1 to check the LotID in the file matches to the value of the parameter *strLotID* in function *UploadYEDI1_Production()*. In case the LotIDs don't match, value of variable *incorrectfile* is set to 1.



Name	Date modified	Type	Size
012219105414	24-01-2019 20:31	001 File	20 KB
032018191923	13-01-2019 08:40	001 File	2 KB

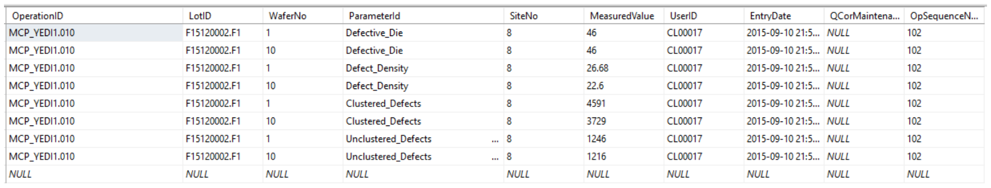
Fig. 5. Files saved in dest.

- **Line 64-71** It takes care that the uploaded files have different slot i.e duplicated files are not uploaded. If in any case same file is uploaded twice, the value of *incorrectfile* is set to 1. After this iteration the value of *incorrectfile* decides if the rollback function is to called or not. Once rollback function is called, error message is displayed and the web-page is reset and reloaded.
- **Line 105-174** After the verification code, if it passes successfully, each file from the *filepath1* is looped through and read. Necessary data is extracted, typecast-ed and trimmed to remove unwanted characters, for example ";", that remain clung to the actual data as debris. Data that is parsed from every file:
  (a) Wafer number
  (b) Parameter ID
  (c) Defective die
  (d) Defect density
  (e) Unclustered defect
  (f) Clustered defect
- **Line 181-234** In this a parameterized SQL query is constructed and values are added to their corresponding parameters. The query is then executed and the parmeters are removed. Processed files are deleted afterwards.



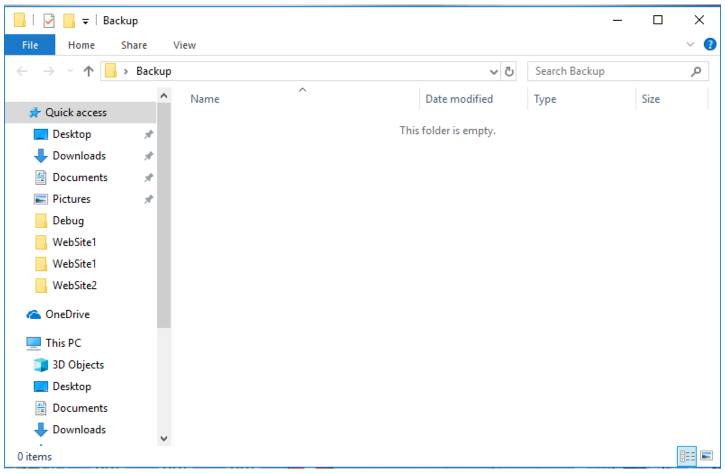
OperationID	LotID	WaferNo	ParameterId	SiteNo	MeasuredValue	UserID	EntryDate	QCorMaintena...	OpSequenceN...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig. 6. Before tracking out



OperationID	LotID	WaferNo	ParameterId	SiteNo	MeasuredValue	UserID	EntryDate	QCorMaintena...	OpSequenceN...
MCP_YEDI1.010	F15120002.F1	1	Defective_Die	8	46	CL00017	2015-09-10 21:5...	NULL	102
MCP_YEDI1.010	F15120002.F1	10	Defective_Die	8	46	CL00017	2015-09-10 21:5...	NULL	102
MCP_YEDI1.010	F15120002.F1	1	Defect_Density	8	26.68	CL00017	2015-09-10 21:5...	NULL	102
MCP_YEDI1.010	F15120002.F1	10	Defect_Density	8	22.6	CL00017	2015-09-10 21:5...	NULL	102
MCP_YEDI1.010	F15120002.F1	1	Clustered_Defects	8	4591	CL00017	2015-09-10 21:5...	NULL	102
MCP_YEDI1.010	F15120002.F1	10	Clustered_Defects	8	3729	CL00017	2015-09-10 21:5...	NULL	102
MCP_YEDI1.010	F15120002.F1	1	Unclustered_Defects ...	8	1246	CL00017	2015-09-10 21:5...	NULL	102
MCP_YEDI1.010	F15120002.F1	10	Unclustered_Defects ...	8	1216	CL00017	2015-09-10 21:5...	NULL	102
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig. 7. After tracking out



This folder is empty.

Fig. 8. Files deleted after processing

2.2 YEDR1: CP Measurement Tool

Note: There are two codes for this tool and both serve different purposes. One is for automating the tool[Listing 4, Annexure] and other one is to classify the clustered data in each file into two bin types.[Listing 5, Annexure].

Role of YEDR1 in defect analysis:

- Providing information about particles and surface defects on unpatterned substrates such as
  - Number of defects
  - Location of defects
  - Size of defects
- Providing a measure of surface quality such as haze, pits, scratches, mounds etc.

2.2.1 Operational procedure

1. Uniform, axi-symmetric collection optics ensure exceptional measurement repeatability

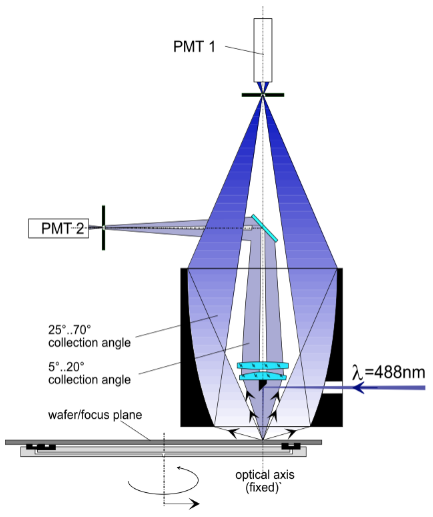


Fig. 9. Mechanism

2. Fixed illumination path increases measurement sensitivity
3. Rotating wafer design provides < 1mm edge exclusion (no edge artifacts)
4. Multiple dark-field/bright- field collection channels capture all defect types

Dark field detection: the collection and registration of scattered radiation. **Bright field detection:** operations performed on the reflected light (specular beam or retrobeam).

2.2.2 Understanding the code [Listing 4, Annexure]

There are two types of output files. One contains pre-processing data and the other contains post-processing data. Each file carries a measured value which corresponds to the type of process it belongs to. Every step function can have multiple parameter Ids. There are three possible parameters for every step function.

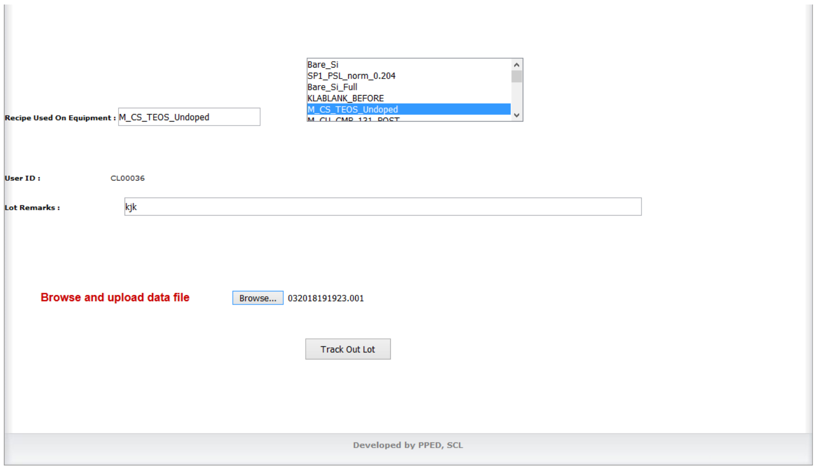
1. Pre
2. Post
3. Delta(= Post - Pre)

If the list of parameter Ids contains multiple parameters against a common Lot Id, then we need two files to calculate the parameters. Same goes if the list contains *delta* parameter. One file upload button is always visible, but not the second one.

1. The script given below runs at the page load event of the track-out page. In this script a connection is made with the database. The value of the operation Id is fetched from the web-page and attached to the parameterized SQL query. The query is then run to fetch all the parameters needed to calculated against that particular Id. If multiple parameters or *delta* parameter is/are listed, the second file upload button is also made visible.

Listing 3: Page load script

```

    If GvLots.SelectedRow.Cells(6).Text.Contains("YEDR1") Then
                conn = New SqlConnection(conString)
                conn.Open()
                cmd = New SqlCommand
                cmd.Connection = conn
                cmd.CommandType = CommandType.Text

                cmd.CommandText = "select Operationid from
                    LLotLocationStatus where lotid=@lotcpx"
                cmd.Parameters.AddWithValue("@lotcpx",
                    Trim(GvLots.SelectedRow.Cells(1).Text))
                Dim opidcpx As String
                opidcpx = cmd.ExecuteScalar
                cmd.Parameters.RemoveAt("@lotcpx")
                cmd.CommandText = "Select * From LOperationLimits
                    WHERE OperationID = '" & Trim(opidcpx) & "'"
                Dim ds As SqlDataReader
                ds = cmd.ExecuteReader
                Dim x As String

                Dim rowcount As Integer = 0

                If ds.HasRows Then
                    While ds.Read
                        rowcount = rowcount + 1
                        x = ds.Item("OpParameterID")

                        If (x.IndexOf("delta", 0,
                            StringComparison.CurrentCultureIgnoreCase)
                            > -1) Then
                            FileUpload3.Visible = True
                            lblFile1.Visible = True
                            lblFile2.Visible = True
                        End If
                    End While
                End If
                ds.Close()
                If (rowcount > 1) Then
                    FileUpload3.Visible = True
                    lblFile1.Text = "Upload Pre and Post files."
                    lblFile1.Visible = True
                    lblFile2.Visible = True
                End If
            End If
```



Bare_Si
SP1_PSL_norm_0.204
Bare_Si_Full
KLABLANK_BEFORE
M_CS_TEOS_Undoped

Recipe Used On Equipment : M_CS_TEOS_Undoped

User ID : CL00036

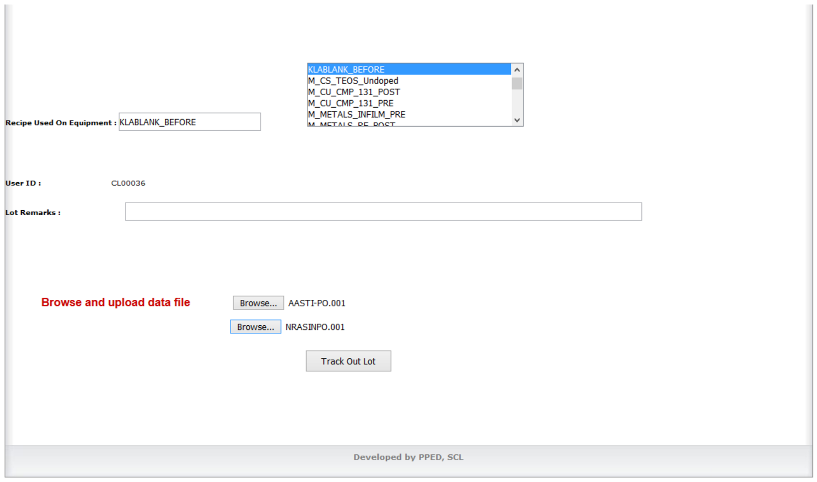
Lot Remarks : kjk

Browse and upload data file Browse... 032018191923.001

Track Out Lot

Developed by PPED, SCL

Fig. 10. Single parameter



KLABLANK_BEFORE
M_CS_TEOS_Undoped
M_CU_CMP_131_POST
M_CU_CMP_131_PRE
M_METALS_INFILM_PRE

Recipe Used On Equipment : KLABLANK_BEFORE

User ID : CL00036

Lot Remarks :

Browse and upload data file Browse... AASTI-PO.001

Browse... NRASINPO.001

Track Out Lot

Developed by PPED, SCL

Fig. 11. Multiple or *delta* parameter

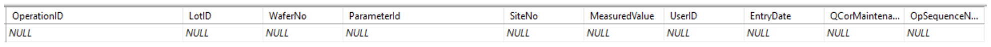
2. After uploading the required files, the user click the *Track Out Lot*. *Track Out Lot* button click fires the main code in which the uploaded files are parsed and the necessary data is inserted into the DBMS.

    - **Line 96-166** In this the first file is opened and checked whether it's

Lot Id matches with the Lot Id given on the web-page(parameter *strLotID*). If they doesn't match, a error message is shown and the roll back function is called which resets and reloads the page. It checks the type of data(pre/post) the file contains and extracts the below given data.

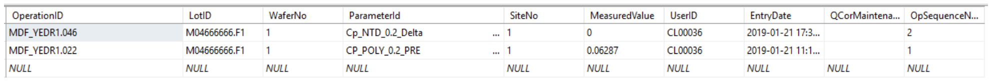
(a) Wafer number
(b) Parameter ID
(c) Measured value of pre/post/delta

- **Line 170-216** This code only runs if the second button is visible. The explanation of this code is same to the one for the first file.
- **Line 218-244** Multiple parameters or *delta* parameter requires two different files to calculate the values. It checks if the files contains data corresponding to both the processes(pre/post). If the parameters for the operation Id contains *delta*, it's value is calculated. In case of any mismatch, the rollback function is called and an error message is displayed.
- **Line 246-282** In this the parameterized SQL query is constructed and values are added to their corresponding parameters. The query is then executed and parameters are removed.



OperationID	LotID	WaferNo	ParameterId	SiteNo	MeasuredValue	UserID	EntryDate	QCorMaintena...	OpSequenceN...
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Fig. 12. Before tracking out



OperationID	LotID	WaferNo	ParameterId	SiteNo	MeasuredValue	UserID	EntryDate	QCorMaintena...	OpSequenceN...
MDF_YEDR1.046	M04666666.F1	1	Cp_NTD_0.2_Delta ...	1	0	CL00036	2019-01-21 17:3...		2
MDF_YEDR1.022	M04666666.F1	1	CP_POLY_0.2_PRE ...	1	0.06287	CL00036	2019-01-21 11:1...		1
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Fig. 13. After tracking out

2.2.3 Understanding the code [Listing 5, Annexure]

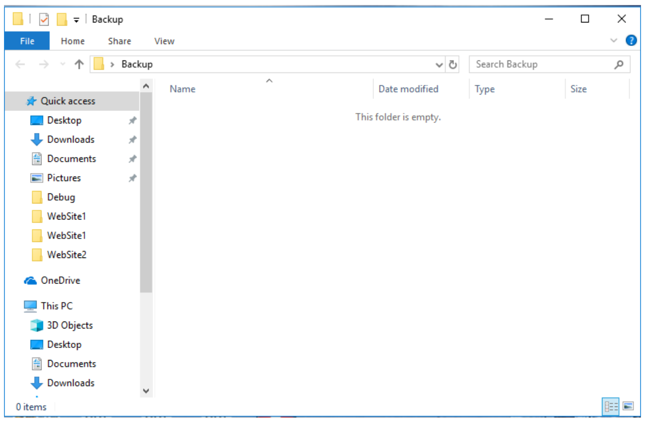
To keep a check on the fluctuations and tool health, the data coming from it has to monitored manually. For this data present in each file is classified in two bins depending upon it's particle size value.

1. BIN1 for particle size lesser than 0.5 microns.
2. BIN2 for particle size greater than 0.5 microns.

After the data from these files has been uploaded on the database, it is then converted in .CSV form for manual checking.

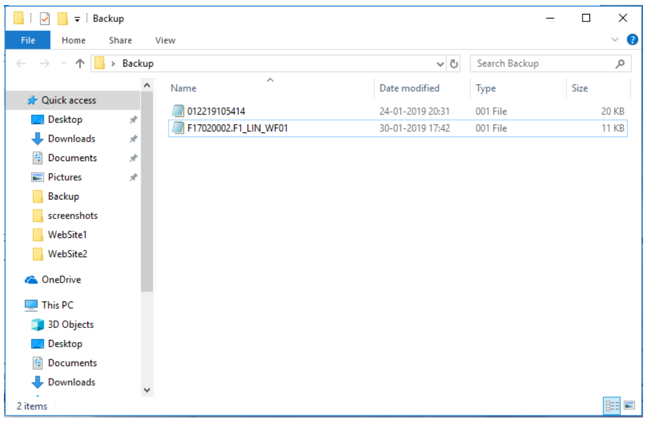
1. The tool spits all of its data on the tool server. These files are copied to a backup folder to be worked upon. Each *klarf* file from the folder is read and classified as BIN1 or BIN2. The data from these files are then uploaded on the database and the files are deleted afterwards.

   - **Line 16-20** In this the connection string is declared which sets a connection between the web-page and the microsoft SQL server. The configuration of the connection string is done in the web config. file. The connection is then set open.
   - **Line 31** This is the server-side script which fires whenever the *upload* button in clicked on the web-page. The function *Directory.GetFiles()* returns the names of files (including their paths) that match the specified search pattern in the specified directory.



This folder is empty.

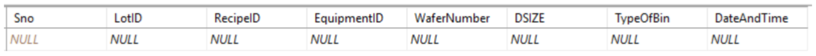
Fig. 14. Before code run



Name	Date modified	Type	Size
012219105414	24-01-2019 20:31	001 File	20 KB
F17020002.F1_LIN_WF01	30-01-2019 17:42	001 File	11 KB

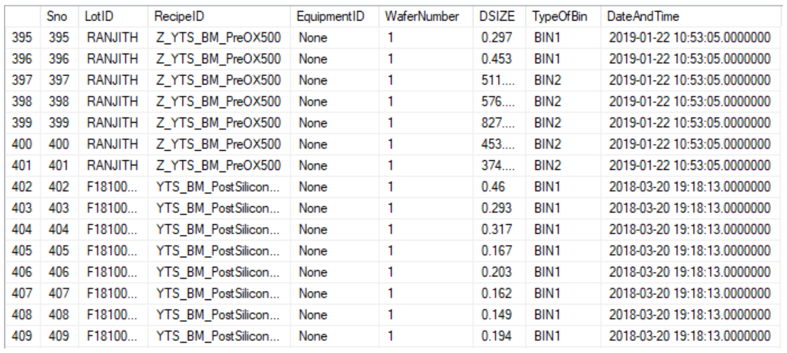
Fig. 15. During code run

- **Line 35-80** Each file is looped through and read. Necessary data is collected and trimmed to remove unwanted characters for example ";" that remain clung to the actual data as debris. Depending upon the particle size, the data in the files is classified as bin1 or bin2.



Sno	LotID	RecipeID	EquipmentID	WaferNumber	DSIZE	TypeOfBin	DateAndTime
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig. 16. Before code run



	Sno	LotID	RecipeID	EquipmentID	WaferNumber	DSIZE	TypeOfBin	DateAndTime
395	395	RANJITH	Z_YTS_BM_PreOX500	None	1	0.297	BIN1	2019-01-22 10:53:05.0000000
396	396	RANJITH	Z_YTS_BM_PreOX500	None	1	0.453	BIN1	2019-01-22 10:53:05.0000000
397	397	RANJITH	Z_YTS_BM_PreOX500	None	1	511....	BIN2	2019-01-22 10:53:05.0000000
398	398	RANJITH	Z_YTS_BM_PreOX500	None	1	576....	BIN2	2019-01-22 10:53:05.0000000
399	399	RANJITH	Z_YTS_BM_PreOX500	None	1	827....	BIN2	2019-01-22 10:53:05.0000000
400	400	RANJITH	Z_YTS_BM_PreOX500	None	1	453....	BIN2	2019-01-22 10:53:05.0000000
401	401	RANJITH	Z_YTS_BM_PreOX500	None	1	374....	BIN2	2019-01-22 10:53:05.0000000
402	402	F18100...	YTS_BM_PostSilicon...	None	1	0.46	BIN1	2018-03-20 19:18:13.0000000
403	403	F18100...	YTS_BM_PostSilicon...	None	1	0.293	BIN1	2018-03-20 19:18:13.0000000
404	404	F18100...	YTS_BM_PostSilicon...	None	1	0.317	BIN1	2018-03-20 19:18:13.0000000
405	405	F18100...	YTS_BM_PostSilicon...	None	1	0.167	BIN1	2018-03-20 19:18:13.0000000
406	406	F18100...	YTS_BM_PostSilicon...	None	1	0.203	BIN1	2018-03-20 19:18:13.0000000
407	407	F18100...	YTS_BM_PostSilicon...	None	1	0.162	BIN1	2018-03-20 19:18:13.0000000
408	408	F18100...	YTS_BM_PostSilicon...	None	1	0.149	BIN1	2018-03-20 19:18:13.0000000
409	409	F18100...	YTS_BM_PostSilicon...	None	1	0.194	BIN1	2018-03-20 19:18:13.0000000

Fig. 17. After code run

- **Line 85-129** In this the database connection is made. Parameterized SQL query is declared and all the data is fed into the parameters and the Query is then executed. Files are only deleted when all the rows in the file have been uploaded on the datbase.

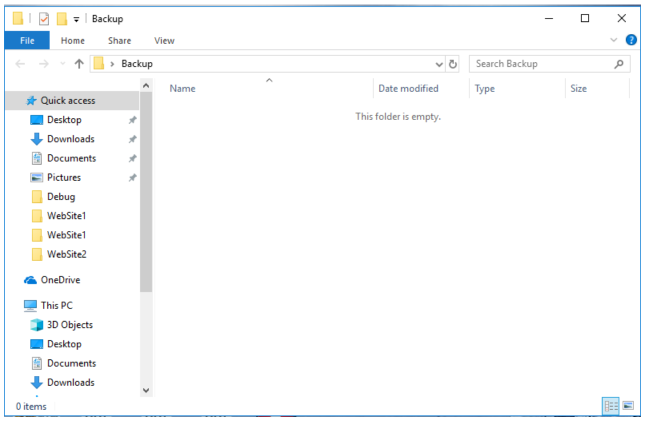


Fig. 18. After code run

Annexure

1. YEDI1: KLA 2139 Inspection Tool

Objective

To automate the YEDI1 Inspection Tool. This involves making a FTP server to fetch all the output files. When uploaded on DMIS, these files must be automatically parsed to extract important information which is inserted into DBMS.

Server-side Script

Listing 4: Code for YEDI1

```

    Private Sub UploadYEDI1_Production(ByVal strLotID As String, ByVal
        strOpid As String, ByVal strOpSno As String)
       Try

           Dim Paraid() As String = {"NDEFDIE", "defdensity",
               "clusture", "unclusture"}
           Dim sConnectionString As String =
               ConfigurationManager.ConnectionStrings("DMIS.MDFConnectionString").ConnectionString
           Dim objConn As New SqlConnection(sConnectionString)
           objConn.Open()
           Dim filepath1 As String = Server.MapPath("MTOP1\")
           Dim fileEntries As String() = Directory.GetFiles(filepath1,
               "*.001")
           Dim fileName As String
           Dim LotID, Slot As String
           Dim words As String()
           Dim cluster As Integer = 0
           Dim uncluster As Integer = 0
           Dim reqvalue_index As Integer = 0
           Dim ndefdie, defdensity As Double
           Dim slotid As Integer
           Dim line As String

           Dim incorrectfile As Boolean = False
           Dim filenumber As Integer = 0
           Dim fileSlotArray As ArrayList = New ArrayList

           'CHECKING IF THE FILES UPLOADED ARE CORRECT OR NOT
           For Each fileName In fileEntries
               If (System.IO.File.Exists(fileName)) Then
```

```
                Using sr As StreamReader = New StreamReader(fileName)
                    filenumber = filenumber + 1

                    line = sr.ReadLine()

                    While (line <> Nothing)

                        If (line.IndexOf("Slot", 0,
                            StringComparison.CurrentCultureIgnoreCase)
                            > -1) Then
                            words = line.Split(" ")
                            Slot = words(1)
                            If Slot.Contains(";") Then
                                Slot = Slot.TrimEnd(";")
                                Integer.TryParse(Slot, slotid)
                                fileSlotArray.Add(slotid)
                            End If
                        End If

                        If (line.IndexOf("LotID", 0,
                            StringComparison.CurrentCultureIgnoreCase)
                            > -1) Then
                            words = line.Split(""""c)
                            MsgBox(words(0))
                            MsgBox(words(1))
                            MsgBox(words(2))

                            'CHECKING IF LotID ARE MATCHING OR NOT
                            If (words(1).IndexOf(strLotID, 0,
                                StringComparison.CurrentCultureIgnoreCase)
                                = -1) Then
                                incorrectfile = True
                            End If
                        End If

                        line = sr.ReadLine()
                    End While

                End Using
            End If
        Next

        'CHECKING IF DUPLICATE SLOTS ARE PRESENT.
        fileSlotArray.Sort()

        For Index As Integer = 0 To (fileSlotArray.Count - 2)
            If (fileSlotArray(Index) = fileSlotArray(Index + 1)) Then
                incorrectfile = True
            End If
        Next
        If (fileSlotArray.Count <> CInt(txttrackoutqty.Text)) Then
            'CHECKING IF CORRECT NUMBER OF FILES ARE UPLOADED
            incorrectfile = True
        End If

        'IF ANYTHING IS NOT RIGHT, IT DELETES ALL THE FILES AND EXITS
            THE SUB
        If (incorrectfile = True) Then
            General1.myShowPopup(Me.Page, "Incorrect files uploaded.
                Error may be caused if 1>LotIDs dont't match.
                2>Required numbers of files are not uploaded. 3>Same
                slot is uploaded twice")
            For Each fileName In fileEntries
                File.Delete(fileName)
            Next
            Exit Sub
        End If

        'IF EVERYTHING IS RIGHT THEN IT RUNS THE FURTHER

        'CHECKING OF PARAMETERS IN LOPERATIONLIMITS

        Dim parameters As ArrayList = New ArrayList
        Dim sq2 As String = "Select * From LOperationLimits WHERE
            OperationID = '" & strOpid & "'"
        Dim cmd2 As SqlCommand
        cmd2 = New SqlCommand(sq2, objConn)
        cmd2.CommandType = CommandType.Text
        Dim ds As SqlDataReader
        ds = cmd2.ExecuteReader

        If ds.HasRows Then
            While ds.Read
                parameters.Add(item("OpParameterID"))
            End While
        End If

        'DATA PARSING STARTS HERE
        For Each fileName In fileEntries
            If (System.IO.File.Exists(fileName)) Then

                Dim msrvalue(4) As Double
                cluster = 0
                uncluster = 0
                reqvalue_index = 0

                Using sr As StreamReader = New StreamReader(fileName)
                    line = sr.ReadLine()
                    While (line <> Nothing)
```

```

                        If (line.IndexOf("Slot", 0,
                            StringComparison.CurrentCultureIgnoreCase)
                            > -1) Then
                            words = line.Split(" ")
                            Slot = words(1)
                            If Slot.Contains(";") Then
                                Slot = Slot.TrimEnd(";")
                                Integer.TryParse(Slot, slotid)
                            End If
                        End If

                        If (line.IndexOf("DefectList", 0,
                            StringComparison.CurrentCultureIgnoreCase)
                            > -1) Then
                            line = sr.ReadLine()
                            While (line.IndexOf("SummarySpec", 0,
                                StringComparison.CurrentCultureIgnoreCase)
                                = -1)

                                words = line.Split(" ")
                                If (words(14) = 0) Then
                                    uncluster = uncluster + 1
                                Else
                                    cluster = cluster + 1
                                End If

                                line = sr.ReadLine()
                            End While
                        End If

                        If (line.IndexOf("SummaryList", 0,
                            StringComparison.CurrentCultureIgnoreCase)
                            > -1) Then
                            line = sr.ReadLine()
                            words = line.Split(" ")

                            For Each word As String In words

                                If (word <> Nothing) Then
                                    reqvalue_index = reqvalue_index + 1

                                    If (reqvalue_index = 3) Then
                                        Double.TryParse(word, defdensity)

                                    End If

                                    If (reqvalue_index = 5) Then
                                        Double.TryParse(word, ndefdie)

```

```
                    End If

                End If
            Next
        End If

        line = sr.ReadLine()
    End While
End Using

msrvalue(0) = ndefdie
msrvalue(1) = defdensity
msrvalue(2) = cluster
msrvalue(3) = uncluster

'DECLARING THE SQL QUERY
Dim sSQL As String = "Insert into ParseALG
    values(@opid,@lotid,@waferno,@paraid,@site,@msrval,@userid,@entrydate,@qcor,@ops
Dim objCmd As New SqlCommand(sSQL, objConn)

'FEEDING THE PARAMETERS WITH VALUES
objCmd.Parameters.AddWithValue("@opid", strOpid)
objCmd.Parameters.AddWithValue("@lotid", strLotID)
objCmd.Parameters.AddWithValue("@waferno", slotid)
objCmd.Parameters.AddWithValue("@site", CInt("1"))
objCmd.Parameters.AddWithValue("@userid",
    Session("username"))
objCmd.Parameters.AddWithValue("@entrydate", Date.Now)
objCmd.Parameters.AddWithValue("@qcor", "")
objCmd.Parameters.AddWithValue("@opseq",
    CInt(strOpSno))

For i As Integer = 0 To 3

 objCmd.Parameters.AddWithValue("@paraid",
     parameters(i))

    If (parameters(i).IndexOf("cluster", 0,
        StringComparison.CurrentCultureIgnoreCase) >
        -1) Then
        objCmd.Parameters.AddWithValue("@msrval",
            msrvalue(2))

    ElseIf (parameters(i).IndexOf("uncluster", 0,
        StringComparison.CurrentCultureIgnoreCase) >
        -1) Then
        objCmd.Parameters.AddWithValue("@msrval",
            msrvalue(3))

```

```
                        ElseIf (parameters(i).IndexOf("die", 0,
                            StringComparison.CurrentCultureIgnoreCase) >
                            -1) Then
                            objCmd.Parameters.AddWithValue("@msrval",
                                msrvalue(0))

                        ElseIf (parameters(i).IndexOf("density", 0,
                            StringComparison.CurrentCultureIgnoreCase) >
                            -1) Then
                            objCmd.Parameters.AddWithValue("@msrval",
                                msrvalue(1))

                        End If

                        objCmd.ExecuteNonQuery()

                        objCmd.Parameters.RemoveAt("@paraid")
                        objCmd.Parameters.RemoveAt("@msrval")

                    Next

                    objCmd.Parameters.RemoveAt("@opid")
                    objCmd.Parameters.RemoveAt("@lotid")
                    objCmd.Parameters.RemoveAt("@waferno")
                    objCmd.Parameters.RemoveAt("@site")
                    objCmd.Parameters.RemoveAt("@userid")
                    objCmd.Parameters.RemoveAt("@entrydate")
                    objCmd.Parameters.RemoveAt("@qcor")
                    objCmd.Parameters.RemoveAt("@opseq")

                    Erase msrvalue
                End If

                'DELETING FILES AFTER PROCESSING
                File.Delete(fileName)
            Next
        Catch ex As Exception
            General1.myShowPopup(Me.Page, "Invalid File: " & ex.Message &
                " ")
        End Try
    End Sub

    'UPLOAD BUTTON CLICK EVENT
    Protected Sub Button3_Click(ByVal sender As Object, ByVal e As
        EventArgs) Handles Button3.Click
        Try
            Dim filename1 As String =
                Path.GetFileName(FileUpload2.FileName)
            Dim extension1 As String = Path.GetExtension(filename1)
            Dim filepath1 As String = Server.MapPath("MTOP1\" & filename1)
            FileUpload2.SaveAs(filepath1)

        Catch ex As Exception
            General1.myShowPopup(Me.Page, "Invalid File: " & ex.Message &
                " ")
        End Try
    End Sub
End Class
```

2. YEDR1: CP Measurement tool

2.1 Objective

To automate the YEDR1. This involves making a FTP server to fetch all the output files. When uploaded on DMIS, these files must be automatically parsed to extract important information which is inserted into DBMS.

Server-side Script

Listing 5: Code for CP Msr. Tool

```

''Structure to save the data from the files
    Public Structure filevar
        Public word As String
        Public Slot As String
        Public x As String
        Public index As Integer
    End Structure

    Private Sub UploadYEDR1_Production(ByVal strLotID As String, ByVal
         strOpid As String, ByVal strOpSno As String)
        Try
            Dim file(5) As filevar
            Dim line As String
            Dim words As String()
            Dim visibility As Integer = 0
            Dim rowcount As Integer = 0
            Dim conn3 As SqlConnection

            'ESTABLISHING CONNECTION WITH DATABASE
            Dim constring As String =
            ConfigurationManager.ConnectionStrings("DMIS.MDFConnectionString").ConnectionString
            conn3 = New SqlConnection(constring)
            conn3.Open()

            FileUpload3.Visible = False

            For ii As Integer = 0 To 4
                file(ii).index = 0
            Next

            'CHECKING PARAMTERS AGAINST THE Operation ID
            Dim sq2 As String = "Select * From LOperationLimits WHERE
                OperationID = '" & strOpid & "'"
            Dim cmd2 As SqlCommand
            cmd2 = New SqlCommand(sq2, conn3)
            cmd2.CommandType = CommandType.Text
            Dim ds As SqlDataReader
            ds = cmd2.ExecuteReader

            Dim index As Integer
            Dim isdelta As Integer = 0          ''check if delta parameter is present
            Dim isdeltaindex As Integer = -1
            Dim ispre As Integer = 0            ''check if pre parameter is present
            Dim ispreindex As Integer = -1
            Dim ispost As Integer = 0           ''check if post parameter is present
            Dim ispostindex As Integer = -1

            If ds.HasRows Then
                index = index + 1

                While ds.Read
                   rowcount = rowcount + 1
                   file(index - 1).x = ds.Item("OpParameterID")

                   If (file(index - 1).x.IndexOf("pre", 0, StringComparison.CurrentCultureIgnoreCase) > -1) Then
                       ispre = 1
                       ispreindex = rowcount
                   End If

                   If (file(index - 1).x.IndexOf("post", 0, StringComparison.CurrentCultureIgnoreCase) > -1) Then
                       ispost = 1
                       ispostindex = rowcount
                   End If

                   If (file(index - 1).x.IndexOf("delta", 0, StringComparison.CurrentCultureIgnoreCase) > -1) Then
                       FileUpload3.Visible = True
                       visibility = 1
                       isdelta = 1
                       isdeltaindex = rowcount
                       rowcount = rowcount + 1
                   End If
                End While

            End If

            If (rowcount > 1) Then
                FileUpload3.Visible = True
                visibility = 1
```

```
                General1.myShowPopup(Me.Page, "Insert pre file in first
                    slot and post file in second slot.")
            End If

            ds.Close()

            Dim file1index As Integer = 0
            Dim file2index As Integer = 0

            If (rowcount > 1) Then
                file1index = 3
                file2index = 4
            End If

            Dim isprefile As Integer = 0
            Dim ispostfile As Integer = 0
            Dim postflag As Integer = 0
            Dim preflag As Integer = 0

            ''FILE 1 PARSING STARTS HERE
            Dim filename1 As String =
                Path.GetFileName(FileUpload2.PostedFile.FileName)
            Dim extension1 As String = Path.GetExtension(filename1)
            Dim filepath1 As String = Server.MapPath("LotDataFiles\" &
                filename1)
            FileUpload2.SaveAs(filepath1)

            Using sr As StreamReader = New StreamReader(filepath1)

                line = sr.ReadLine()

                Dim reqvalue_index As Integer = 0
                While (line <> Nothing)

                    If (line.Contains("Slot")) Then
                        words = line.Split(" ")
                        file(file1index).Slot = words(1)
                        If (file(file1index).Slot.Contains(";")) Then
                            file(file1index).Slot =
                                file(file1index).Slot.TrimEnd(";")
                        End If
                    End If

                    If (line.Contains("LotID")) Then
                        words = line.Split(""""c)
                        If (strLotID <> words(1)) Then
                            General1.myShowPopup(Me.Page, "LotID don't
                                match")
                            Exit Sub
                        End If
                    End If


                    If (line.Contains("Post")) Then
                        ispostfile = 1
                        postflag = 1
                    End If
                    If (line.Contains("Pre")) Then
                        isprefile = 1
                        preflag = 1
                    End If

                    If (line = "SummaryList ") Then
                        line = sr.ReadLine()
                        words = line.Split(" ")
                        For i As Integer = 1 To 30
                            If (words(i - 1) > "0.00") Then
                                reqvalue_index = reqvalue_index + 1
                            End If
                            If (reqvalue_index = 3) Then
                                file(file1index).word = words(i - 1)
                                Exit While
                            End If
                        Next
                    End If
                    line = sr.ReadLine()

                End While
                sr.Close()

                If (rowcount > 1) Then
                    If (ispre And preflag) Then
                        file(ispreindex - 1).index = 10
                        file(ispreindex - 1).Slot = file(file1index).Slot
                        file(ispreindex - 1).word = file(file1index).word
                    End If

                    If (ispost And postflag) Then
                        file(ispostindex - 1).index = 10
                        file(ispostindex - 1).Slot = file(file2index).Slot
                        file(ispostindex - 1).word = file(file2index).word
                    End If
                End If
            End Using
            postflag = 0
            preflag = 0

            'FILE 2 PARSING STARTS HERE
            If (visibility = 1) Then

```

```
            Dim filename2 As String =
                Path.GetFileName(FileUpload3.PostedFile.FileName)
            Dim extension2 As String = Path.GetExtension(filename2)
            Dim filepath2 As String = Server.MapPath("LotDataFiles\"
                & filename2)
            FileUpload3.SaveAs(filepath2)

            Using sr As StreamReader = New StreamReader(filepath2)
                line = sr.ReadLine()
                While (line <> Nothing)

                    If (line.Contains("Slot")) Then
                        words = line.Split(" ")
                        file(file2index).Slot = words(1)
                        If (file(file2index).Slot.Contains(";")) Then
                            file(file2index).Slot =
                                file(file2index).Slot.TrimEnd(";")
                        End If
                    End If

                    If (line.Contains("LotID")) Then
                        words = line.Split(""""c)
                        If (strLotID <> words(1)) Then
                            General1.myShowPopup(Me.Page, "LotID don't
                                match")
                            'Exit Sub
                        End If
                    End If


                    If (line.Contains("Post")) Then
                        ispostfile = 1
                        postflag = 1
                    End If
                    If (line.Contains("Pre")) Then
                        isprefile = 1
                        preflag = 1
                    End If

                    If (line = "SummaryList ") Then
                        line = sr.ReadLine()
                        words = line.Split(" ")
                        file(file2index).word = words(10)
                    End If
                    line = sr.ReadLine()
                End While
                sr.Close()
            End Using

            If (rowcount > 1) Then
```

```
                If (ispre And preflag) Then
                    file(ispreindex - 1).index = 10
                    file(ispreindex - 1).Slot = file(file1index).Slot
                    file(ispreindex - 1).word = file(file1index).word
                End If

                If (ispost And postflag) Then
                    file(ispostindex - 1).index = 10
                    file(ispostindex - 1).Slot = file(file2index).Slot
                    file(ispostindex - 1).word = file(file2index).word
                End If

                If (isdelta) Then
                    file(isdeltaindex - 1).index = 10
                    file(isdeltaindex - 1).Slot = file(file2index).Slot
                    file(isdeltaindex - 1).word =
                        file(file2index).word - file(file1index).word
                End If
            End If
        End If

        If (rowcount > 1 And (isprefile + ispostfile < 2)) Then
            General1.myShowPopup(Me.Page, "Upload correct files!")
            FileUpload2.Focus()

            Exit Sub
        End If

        'INSERTION INTO DATABSE STARTS HERE
        For i As Integer = 0 To 2
            If (file(i).index = 10) Then

            'DECLARING THE SQL QUERY
                sq2 = "Insert into LOperationMeasuredValueRaw
                    values(@opid,@lotid,@waferno,@paraid,@site,@msrval,@userid,@entrydate,@qcor,@ops
                cmd2 = New SqlCommand(sq2, conn3)

                'FEEDING THE PARAMETERS WITH VALUES
                cmd2.Parameters.AddWithValue("@opid", strOpid)
                cmd2.Parameters.AddWithValue("@lotid", strLotID)
                cmd2.Parameters.AddWithValue("@waferno", file(i).Slot)
                cmd2.Parameters.AddWithValue("@paraid", file(i).x)
                cmd2.Parameters.AddWithValue("@site", CInt("1"))
                cmd2.Parameters.AddWithValue("@msrval",
                    Convert.ToDouble(file(i).word))
                cmd2.Parameters.AddWithValue("@userid",
                    Session("username"))
                cmd2.Parameters.AddWithValue("@entrydate", Date.Now)
                cmd2.Parameters.AddWithValue("@qcor", "")
                cmd2.Parameters.AddWithValue("@opseq", CInt(strOpSno))
```

```

                    If cmd2.ExecuteNonQuery() > -1 Then
                        General1.myShowPopup(Me.Page, "datasaved!")
                    End If

                    cmd2.Parameters.RemoveAt("@opid")
                    cmd2.Parameters.RemoveAt("@lotid")
                    cmd2.Parameters.RemoveAt("@waferno")
                    cmd2.Parameters.RemoveAt("@paraid")
                    cmd2.Parameters.RemoveAt("@site")
                    cmd2.Parameters.RemoveAt("@msrval")
                    cmd2.Parameters.RemoveAt("@userid")
                    cmd2.Parameters.RemoveAt("@entrydate")
                    cmd2.Parameters.RemoveAt("@qcor")
                    cmd2.Parameters.RemoveAt("@opseq")
                End If

            Next

            conn3.Close()

        Catch ex As Exception
            General1.myShowPopup(Me.Page, "Invalid File: " & ex.Message &
                " ")
        End Try
    End Sub
```

2.2 Objective

Recipe wise bin classification from YEDR tool data. Classification is based on particle size.

Code

```
''Importing necessary libraries.
Imports System.Data.SqlClient
Imports System.Data
Imports System.IO
Imports System
Imports System.Runtime.InteropServices
Imports System.Collections
Imports Microsoft.VisualBasic.Strings
Imports System.Globalization
Imports System.Configuration

Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        'ESTABLISHING CONNECTION WITH THE DATABASE
        Dim sConnectionString As String = ConfigurationManager.ConnectionStrings("DMIS.CPX").ConnectionString
        Dim objConn As New SqlConnection(sConnectionString)
        objConn.Open()                  ''setting the connection open

        Dim provider As CultureInfo = CultureInfo.InvariantCulture

        Dim words As String()
        Dim LotID, RecipeID, format, EquipmentID, wafernumberstring As String
        Dim DateAndTime As DateTime
        Dim DSIZE As New ArrayList
        Dim WaferNumber As Integer
        Dim dsizedouble As Double
        format = " MM-dd-yy HH:mm:ss;"

        Dim fileEntries As String() = Directory.GetFiles("E:\cpx_Backup\", "*.001") ''fetching files and their path from the folder
        Dim fileName As String

        'LOOPING THROUGH EACH FILE IN THE FOLDER
        For Each fileName In fileEntries
            If (System.IO.File.Exists(fileName)) Then

               Console.WriteLine(fileName)
               Using sr As StreamReader = New StreamReader(fileName)
                   Dim line As String
                   line = sr.ReadLine()
                   While (line <> Nothing)

                       If (line.IndexOf("LotID", 0,
                            StringComparison.CurrentCultureIgnoreCase)) >
                            -1 Then           ''Extrating the Lot Id
                          words = line.Split(""""c)
                          LotID = words(1)
                       End If

                       If (line.IndexOf("SetupID", 0,
                            StringComparison.CurrentCultureIgnoreCase) >
                            -1) Then           ''Extrating the Setup Id
                          words = line.Split(""""c)
                          RecipeID = words(1)
                          DateAndTime = DateTime.ParseExact(words(2),
                              format, provider)
                       End If

                       If (line.IndexOf("DeviceID", 0,
                            StringComparison.CurrentCultureIgnoreCase) >
                            -1) Then           ''Extrating the Device Id
                          words = line.Split(""""c)
                          EquipmentID = words(1)
                       End If

                       If (line.IndexOf("Slot", 0,
                            StringComparison.CurrentCultureIgnoreCase) >
                            -1) Then              ''Extrating the wafer
                            number
                          words = line.Split(" ")
                          wafernumberstring = words(1).TrimEnd(";")
                          Integer.TryParse(wafernumberstring,
                              WaferNumber)
                       End If

                       If (line.IndexOf("DSIZE", 0,
                            StringComparison.CurrentCultureIgnoreCase) >
                            -1) Then
                          line = sr.ReadLine()
                          line = sr.ReadLine()
                          While (line.Contains("SummarySpec") <> True)

                              words = line.Split(" ")
                              Double.TryParse(words(9), dsizedouble)
                              DSIZE.Add(dsizedouble)
                              line = sr.ReadLine()
                   End While
               End If

               line = sr.ReadLine()
           End While
       End Using

       Dim numberofrows As Integer = 0                ''Count of
           rows of data in file
       Dim rowaffectedcount As Integer = 0            ''Count of
           rows affected in the database

       'INSERTION INTO DATABASE STARTS HERE
       Dim sSQL As String = "INSERT INTO BinClassification
           (LotID,RecipeID,DeviceID,WaferNumber,DSIZE,TypeOfBin,DateAndTime)
           VALUES
           (@LotID,@RecipeID,@EquipmentID,@WaferNumber,@dsize,@bin,@datetime)"
       Dim objCmd As New SqlCommand(sSQL, objConn)

       objCmd.Parameters.AddWithValue("@LotID", LotID)
       objCmd.Parameters.AddWithValue("@RecipeID", RecipeID)
       objCmd.Parameters.AddWithValue("@EquipmentID",
           EquipmentID)
       objCmd.Parameters.AddWithValue("@WaferNumber",
           WaferNumber)
       objCmd.Parameters.AddWithValue("@datetime", DateAndTime)

       For Each word As Double In DSIZE

           numberofrows = numberofrows + 1
           objCmd.Parameters.AddWithValue("@dsize", word)
           If (word < 0.5) Then
               objCmd.Parameters.AddWithValue("@bin", "BIN1")

           ElseIf (word >= 0.5) Then
               objCmd.Parameters.AddWithValue("@bin", "BIN2")

           End If

           If (objCmd.ExecuteNonQuery() > -1) Then
               rowaffectedcount = rowaffectedcount + 1
           End If


           objCmd.Parameters.RemoveAt("@dsize")
           objCmd.Parameters.RemoveAt("@bin")

       Next
       DSIZE.Clear()
       objCmd.Parameters.RemoveAt("@LotID")
```

```
                objCmd.Parameters.RemoveAt("@RecipeID")
                objCmd.Parameters.RemoveAt("@EquipmentID")
                objCmd.Parameters.RemoveAt("@WaferNumber")
                objCmd.Parameters.RemoveAt("@datetime")

                'If both are same then the files are deleted.
                If (rowaffectedcount = numberofrows) Then
                    File.Delete(fileName)

                End If
            End If
        Next

    End Sub
End Class
```