AUTOMATION OF INTERFACE FILE ANALYSIS AND ORTHOGRAPHIC DIAGRAM TAGGING IN PIPING ENGINEERING USING VBA AND C#

Vinesh P[1], Tharun M[2], Sujan R[3], Mohammed Raashidh S[4], Dr. G. Suresh Kannan[5]

[1,2,3,4] B. E Mechanical Engineering, Coimbatore Institute of Technology, Coimbatore, India
[5] Associate Professor, Dept of Mechanical Engineering, Coimbatore Institute of Technology, Coimbatore, India

---***---

Abstract - In contemporary engineering industries, particularly in sectors such as oil and gas, petrochemicals, and energy production, the efficiency of project execution is highly dependent on the accuracy and consistency of digital engineering data. Piping engineering, being a critical component of plant design, involves extensive interaction between design tools, material databases, and interface files. One of the major challenges encountered in such environments is the inconsistency of commodity codes across different datasets, leading to discrepancies in Smart 3D (S3D) models and associated engineering workflows.
This study presents a comprehensive automation approach aimed at addressing these inconsistencies through the development of customized tools using Visual Basic for Applications (VBA) and C#. The primary tool focuses on comparing specification identification sheets to detect changes in commodity codes and extract relevant differences such as skipped, added, modified, and deleted components. A graphical user interface (GUI) developed using C# enhances usability and ensures ease of operation. Additionally, a secondary automation tool is introduced to streamline the tagging process in orthographic diagrams. This tool simplifies complex tag structures by converting them into sequential identifiers while maintaining traceability through mapping files.
The implementation of these tools demonstrates significant improvements in efficiency, accuracy, and workflow integration. The results highlight the potential of automation in reducing manual effort, minimizing errors, and enhancing coordination across engineering teams.

1.INTRODUCTION

Piping engineering plays a crucial role in the design and operation of industrial plants, particularly in sectors such as oil and gas, petrochemical, power generation, and process industries. These systems are responsible for transporting fluids such as liquids, gases, and slurries under controlled conditions. The reliability, safety, and efficiency of these systems depend on precise design, accurate data management, and seamless coordination between multiple engineering disciplines.
In recent years, the adoption of digital engineering tools has significantly transformed the way piping systems are designed and managed. Advanced software platforms such as Smart 3D and SmartPlant Materials enable engineers to create highly detailed 3D models, manage large datasets, and ensure compliance with international standards. These tools integrate geometry, material specifications, and engineering data into a unified environment.
However, despite these technological advancements, one of the major challenges in modern piping engineering is maintaining data consistency across multiple systems. Engineering data is typically distributed across various interface files such as specification sheets, catalogue workbooks, and generic data files. These files are interconnected, and any inconsistency between them can lead to significant issues in the design process.
A critical element within these datasets is the commodity code, which uniquely identifies each piping component. When updates occur in catalogue data without proper synchronization, components may lose their reference in the system. This results in missing elements in 3D models, incorrect material specifications, and increased manual verification efforts.
To overcome these challenges, automation becomes essential. By implementing automated tools, engineers can reduce manual effort, improve accuracy, and enhance overall productivity. This study focuses on developing such automation tools using VBA and C# to streamline data comparison and simplify tagging systems in piping engineering workflows.

2. INDUSTRY BACKGROUND AND DIGITAL ENGINEERING

The engineering industry is undergoing a significant transformation driven by digitalization. Traditional methods of design, which relied heavily on manual drafting and isolated data systems, are being replaced by integrated digital platforms that enable real-time collaboration and data sharing.
Digital engineering involves the use of advanced software tools, data analytics, and automation technologies to improve the efficiency and accuracy of engineering processes. In piping engineering, this includes the use of 3D modeling tools, database management systems, and automated workflows.
One of the key advantages of digital engineering is the ability to handle large volumes of data efficiently. Modern projects involve thousands of components, each with

detailed specifications. Managing this data manually is not feasible, which makes automation a necessity.
Another important aspect is interoperability between different software systems. Tools like Smart 3D and AutoCAD Plant 3D must work together seamlessly to ensure consistency across design and material data.
Despite these advancements, challenges such as data inconsistency, duplication, and synchronization errors still exist. These issues highlight the need for intelligent automation solutions that can bridge the gap between different systems and ensure reliable data flow.

3. PIPING ENGINEERING FUNDAMENTALS

Piping engineering is a multidisciplinary field that combines principles of mechanical engineering, fluid mechanics, thermodynamics, and materials science. The primary objective of piping engineering is to design systems that can safely and efficiently transport fluids under various operating conditions.

3.1 Components of a Piping System
A piping system consists of several components, each serving a specific function:

- Pipes: These are the primary elements used to transport fluids. They are selected based on material, diameter, and thickness.
- Valves: Used to control the flow of fluid within the system. Different types include gate valves, globe valves, and ball valves.
- Flanges: Provide detachable connections between pipes and equipment.
- Gaskets: Ensure leak-proof connections between flanges.
- Fittings: Used to change the direction or size of pipes (elbows, tees, reducers).
- Supports: Provide structural stability to the piping system.

3.2 Design Considerations
Designing a piping system requires careful consideration of several factors:

- Pressure and Temperature: The system must withstand operating conditions without failure.
- Fluid Properties: Corrosive or hazardous fluids require special materials.
- Thermal Expansion: Pipes expand and contract with temperature changes.
- Safety Standards: Compliance with standards ensures reliability.

3.3 Codes and Standards
Piping systems are designed according to international standards such as:

- ASME B31.3 (Process Piping)
- ASME B31.1 (Power Piping)

These standards define guidelines for design, materials, testing, and safety.

4. PROBLEM STATEMENT

One of the major challenges in piping engineering is the inconsistency of data across multiple systems. This issue arises due to frequent updates in catalogue data, which may not be properly synchronized with design models.

4.1 Commodity Code Mismatch

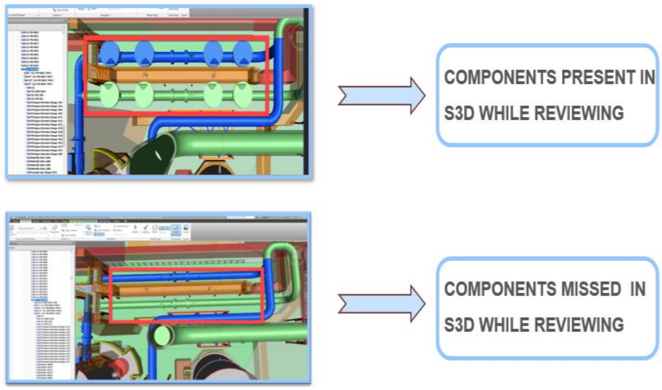


Fig -1: Components present vs. missing in S3D due to commodity code mismatch

Commodity codes serve as unique identifiers for components. When these codes are updated:

- Existing components lose their reference
- Models show missing elements
- Engineers must manually identify and correct errors

This process is time-consuming and prone to mistakes.

4.2 Manual Comparison Problem
Engineers often rely on manual methods such as Excel comparison to identify differences between datasets. This involves:

- Comparing thousands of rows
- Identifying mismatches
- Extracting relevant data

This approach is inefficient and not scalable.

4.3 Tagging Complexity
Orthographic diagrams use long tags that are difficult to interpret. This creates communication issues between teams and increases the likelihood of errors.

5. METHODOLOGY

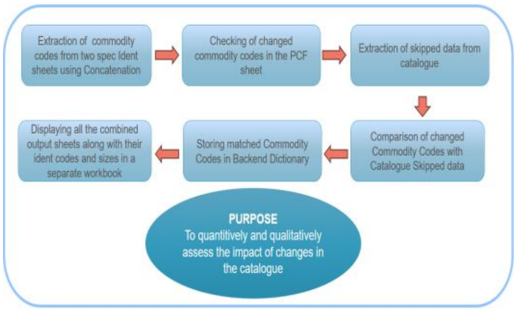


Fig -2: Proposed methodology flowchart for commodity code automation

5.1 Commodity Code Automation Tool
The automation process consists of multiple stages:

5.1.1 Data Preparation

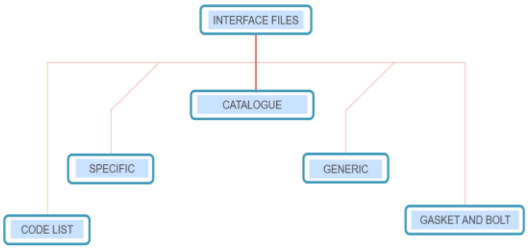


Fig -3: Hierarchy of S3D interface files

Input files include:

- Previous spec ident sheet
- Updated spec ident sheet
- Catalogue workbook

These files are loaded into the system for processing.

5.1.2 Concatenation Logic

1.CONCATENATION

B
Concatenated Data (Feb)
A100 - PIP - 1 - 18 -
A100 - PAD - 1 - 24 - 18
A100 - PAD - 1 - 24 - 14
A100 - PAD - 1 - 24 - 16
A100 - PAD - 1 - 20 - 10
A100 - PAD - 1 - 20 - 12
A100 - PAD - 1 - 20 - 14
A100 - PAD - 1 - 20 - 16
A100 - PAD - 1 - 20 - 18
A100 - PAD - 1 - 24 - 10

Fig -4: Unique key generation using concatenation of Spec Code, Short Code, Option, and Size

To uniquely identify components, multiple fields are combined:

- Short Code
- Option Code
- Size parameters

This creates a unique key for comparison.

5.1.3 Comparison Algorithm

2.DATA COMPARISON

	A	B	C	D	E	F
1	Commodity Code (Feb)	Concatenated Data (Feb)	Commodity Code (Mar)	Comparison Resi	Ident Code F	Ident Code M
90943	VBAAI0U404R0C13J073900	F10A - VB - 1 - .5 -	VBABI0U404R0C13J073900	FALSE	C9A8C60S	CAD1514M
90944	VBAAI0U404R0C13J073900	F10A - VB - 1 - .75 -	VBABI0U404R0C13J073900	FALSE	C9A8C60T	CAD1514N
90945	VBAAI0U404R0C13J073900	F10A - VB - 1 - 1 -	VBABI0U404R0C13J073900	FALSE	C9A8C60U	CAD1514P
90946	VBAAI0U404R0C13J073900	F10A - VB - 1 - 1.5 -	VBABI0U404R0C13J073900	FALSE	C9A8C60W	CAD1514R
90957	FFF120C5C406L5HS00	F10A - ORI - 1 - 10 -	FFF120C5C406L5HS00	TRUE	C9M6E3EN	C9M6E3EN
90958	FFF120C5C406L5HS00	F10A - ORI - 1 - 12 -	FFF120C5C406L5HS00	TRUE	C9M6E3GC	C9M6E3GC
90959	FFF120C5C406L5HS00	F10A - ORI - 1 - 14 -	FFF120C5C406L5HS00	TRUE	C9M6E3J5	C9M6E3J5
90960	FFF120C5C406L5HS00	F10A - ORI - 1 - 16 -	FFF120C5C406L5HS00	TRUE	C9M6E3KV	C9M6E3KV
90961	FFF120C5C406L5HS00	F10A - ORI - 1 - 18 -	FFF120C5C406L5HS00	TRUE	C9M6E3MC	C9M6E3MC
90962	FFF120C5C406L5HS00	F10A - ORI - 1 - 20 -	FFF120C5C406L5HS00	TRUE	C9M6E3RX	C9M6E3RX
90963	FFF120C5C406L5HS00	F10A - ORI - 1 - 24 -	FFF120C5C406L5HS00	TRUE	C9M6E3TK	C9M6E3TK

Fig -5: Data comparison output between February and March datasets

The algorithm compares datasets and identifies:

- Matching entries
- Mismatched entries
- Missing entries

This is done using VBA loops and dictionary-based matching.

5.1.4 Extraction Process

DATA COMPARISON

1	Commodity Code (Feb)	Concatenated Data (Feb)	Commodity Code (Mar)	Comparison Resi	Ident Code F	Ident Code M
90943	VBAAI0U404R0C13J073900	F10A - VB - 1 - .5 -	VBABI0U404R0C13J073900	FALSE	C9A8C60S	CAD1514M
90944	VBAAI0U404R0C13J073900	F10A - VB - 1 - .75 -	VBABI0U404R0C13J073900	FALSE	C9A8C60T	CAD1514N
90945	VBAAI0U404R0C13J073900	F10A - VB - 1 - 1 -	VBABI0U404R0C13J073900	FALSE	C9A8C60U	CAD1514P
90946	VBAAI0U404R0C13J073900	F10A - VB - 1 - 1.5 -	VBABI0U404R0C13J073900	FALSE	C9A8C60W	CAD1514R
90957	FFF120C5C406L5HS00	F10A - ORI - 1 - 10 -	FFF120C5C406L5HS00	TRUE	C9M6E3EN	C9M6E3EN
90958	FFF120C5C406L5HS00	F10A - ORI - 1 - 12 -	FFF120C5C406L5HS00	TRUE	C9M6E3GC	C9M6E3GC
90959	FFF120C5C406L5HS00	F10A - ORI - 1 - 14 -	FFF120C5C406L5HS00	TRUE	C9M6E3J5	C9M6E3J5
90960	FFF120C5C406L5HS00	F10A - ORI - 1 - 16 -	FFF120C5C406L5HS00	TRUE	C9M6E3KV	C9M6E3KV
90961	FFF120C5C406L5HS00	F10A - ORI - 1 - 18 -	FFF120C5C406L5HS00	TRUE	C9M6E3MC	C9M6E3MC
90962	FFF120C5C406L5HS00	F10A - ORI - 1 - 20 -	FFF120C5C406L5HS00	TRUE	C9M6E3RX	C9M6E3RX
90963	FFF120C5C406L5HS00	F10A - ORI - 1 - 24 -	FFF120C5C406L5HS00	TRUE	C9M6E3TK	C9M6E3TK

Fig -6: PCF Method Step 1 — Extraction of piping commodity filter data

The system extracts:

- Skipped components
- Modified components
- Added components
- Deleted components

Each category is stored in a separate worksheet.

5.1.5 GUI Development

Fig -7: Final output displayed in the C# graphical user interface

A user-friendly interface is developed using C#:

- File selection buttons
- Execution controls
- Output display grid

This improves usability and reduces complexity.

5.2 Tag Streamlining Tool

5.2.1 Tag Extraction

Tags are extracted using pattern matching techniques such as Regex.

6. IMPLEMENTATION

The implementation involves two main technologies:

6.1 VBA Implementation

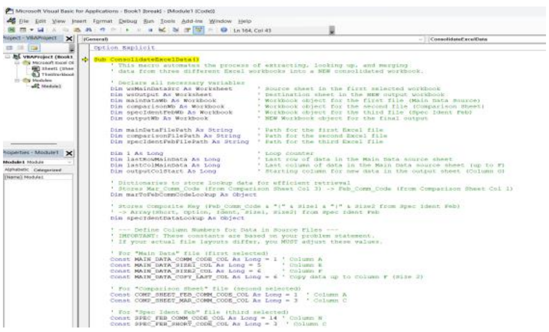


Fig -8: Corrected VBA code in Excel — Part 1

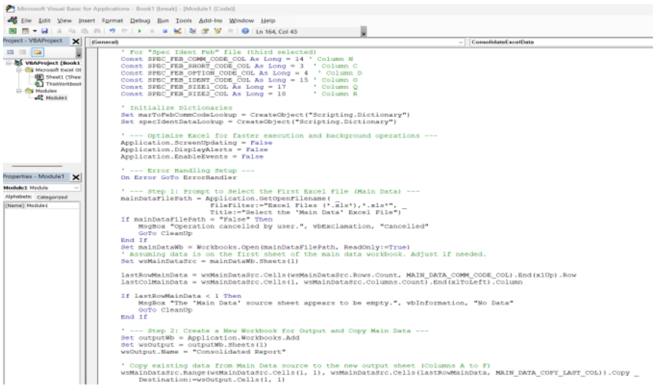
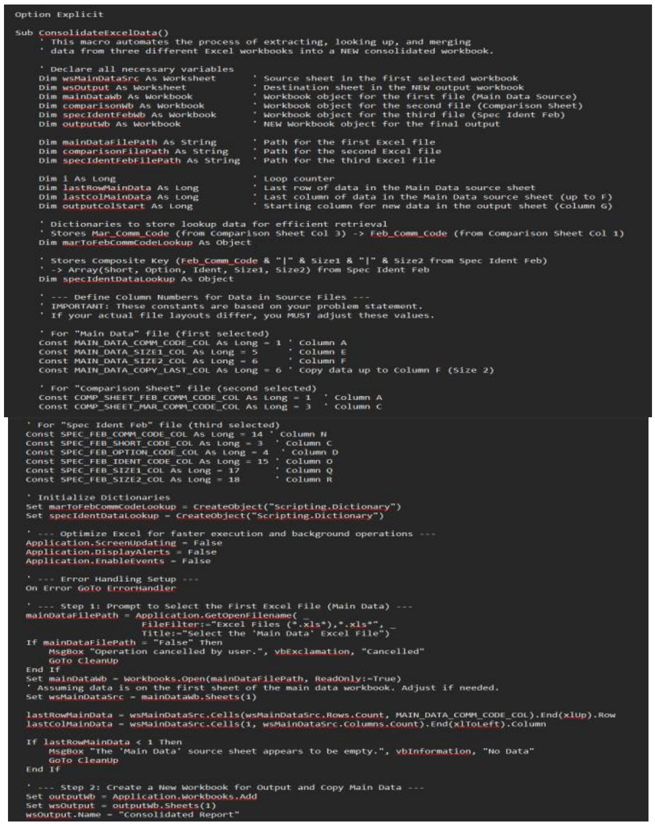


Fig -9: Corrected VBA code in Excel — Part 2

- Modules created for each task
- Functions for comparison and extraction
- Excel automation for data processing

6.2 C# Implementation



```
Option Explicit

Sub ConsolidateExcelData()
    ' This macro automates the process of extracting, looking up, and merging
    ' data from three different Excel workbooks into a NEW consolidated workbook.

    ' Declare all necessary variables
    Dim wsMainDataSrc As Worksheet        ' Source sheet in the first selected workbook
    Dim wsOutput As Worksheet             ' Destination sheet in the NEW output workbook
    Dim mainDataWb As Workbook            ' Workbook object for the first file (Main Data Source)
    Dim comparisonWb As Workbook          ' Workbook object for the second file (Comparison Sheet)
    Dim specIdentFebWb As Workbook        ' Workbook object for the third file (Spec Ident Feb)
    Dim outputWb As Workbook              ' NEW Workbook object for the final output

    Dim mainDataFilePath As String        ' Path for the first Excel file
    Dim comparisonFilePath As String      ' Path for the second Excel file
    Dim specIdentFebFilePath As String    ' Path for the third Excel file

    Dim i As Long                         ' Loop counter
    Dim lastRowMainData As Long           ' Last row of data in the Main Data source sheet
    Dim lastColMainData As Long           ' Last column of data in the Main Data source sheet (up to F)
    Dim outputColStart As Long            ' Starting column for new data in the output sheet (Column G)

    ' Dictionaries to store lookup data for efficient retrieval
    ' Stores Mar_Comm_Code (from Comparison Sheet Col 3) -> Feb_Comm_Code (from Comparison Sheet Col 1)
    Dim marToFebCommCodeLookup As Object

    ' Stores Composite Key (Feb_Comm_Code & "|" & Size1 & "|" & Size2 from Spec Ident Feb)
    ' -> Array(Short, Option, Ident, Size1, Size2) from Spec Ident Feb
    Dim specIdentDataLookup As Object

    ' --- Define Column Numbers for Data in Source Files ---
    ' IMPORTANT: These constants are based on your problem statement.
    ' If your actual file layouts differ, you MUST adjust these values.

    ' For "Main Data" file (first selected)
    Const MAIN_DATA_COMM_CODE_COL As Long = 1 ' Column A
    Const MAIN_DATA_SIZE1_COL As Long = 5     ' Column E
    Const MAIN_DATA_SIZE2_COL As Long = 6     ' Column F
    Const MAIN_DATA_COPY_LAST_COL As Long = 6 ' Copy data up to Column F (Size 2)

    ' For "Comparison Sheet" file (second selected)
    Const COMP_SHEET_FEB_COMM_CODE_COL As Long = 1  ' Column A
    Const COMP_SHEET_MAR_COMM_CODE_COL As Long = 3  ' Column C
' For "Spec Ident Feb" file (third selected)
Const SPEC_FEB_COMM_CODE_COL As Long = 14 ' Column N
Const SPEC_FEB_SHORT_CODE_COL As Long = 3  ' Column C
Const SPEC_FEB_OPTION_CODE_COL As Long = 4  ' Column D
Const SPEC_FEB_IDENT_CODE_COL As Long = 15 ' Column O
Const SPEC_FEB_SIZE1_COL As Long = 17      ' Column Q
Const SPEC_FEB_SIZE2_COL As Long = 18      ' Column R

' Initialize Dictionaries
Set marToFebCommCodeLookup = CreateObject("Scripting.Dictionary")
Set specIdentDataLookup = CreateObject("Scripting.Dictionary")

' --- Optimize Excel for faster execution and background operations ---
Application.ScreenUpdating = False
Application.DisplayAlerts = False
Application.EnableEvents = False

' --- Error Handling Setup ---
On Error GoTo ErrorHandler

' --- Step 1: Prompt to Select the First Excel File (Main Data) ---
mainDataFilePath = Application.GetOpenFilename( _
                    FileFilter:="Excel Files (*.xls*),*.xls*", _
                    Title:="Select the 'Main Data' Excel File")
If mainDataFilePath = "False" Then
    MsgBox "Operation cancelled by user.", vbExclamation, "Cancelled"
    GoTo CleanUp
End If
Set mainDataWb = Workbooks.Open(mainDataFilePath, ReadOnly:=True)
' Assuming data is on the first sheet of the main data workbook. Adjust if needed.
Set wsMainDataSrc = mainDataWb.Sheets(1)

lastRowMainData = wsMainDataSrc.Cells(wsMainDataSrc.Rows.Count, MAIN_DATA_COMM_CODE_COL).End(xlUp).Row
lastColMainData = wsMainDataSrc.Cells(1, wsMainDataSrc.Columns.Count).End(xlToLeft).Column

If lastRowMainData < 1 Then
    MsgBox "The 'Main Data' source sheet appears to be empty.", vbInformation, "No Data"
    GoTo CleanUp
End If

' --- Step 2: Create a New Workbook for Output and Copy Main Data ---
Set outputWb = Application.Workbooks.Add
Set wsOutput = outputWb.Sheets(1)
wsOutput.Name = "Consolidated Report"
```

Fig -10: Initial trial VBA code for Method 1 comparison logic

- Windows Forms used for GUI
- Event-driven programming

7. ADVANTAGES

- Reduces manual effort by automating repetitive tasks such as data comparison and extraction
- Saves significant time compared to traditional manual methods
- Improves accuracy by eliminating human errors in large dataset processing
- Provides fast and efficient comparison of specification identification sheets
- Capable of handling large volumes of engineering data (high scalability)
- Organizes data into clear categories such as added, modified, deleted, and skipped components
- Enhances readability of orthographic diagrams by simplifying complex instrument tags
- Generates structured outputs for easy analysis and decision-making
- Improves coordination between design, procurement, and engineering teams
- Reduces chances of data inconsistency and mismatch issues
- Increases overall productivity of engineers
- User-friendly interface developed using C# enables easy operation
- Requires minimal technical expertise to operate
- Reduces project delays by speeding up data processing tasks
- Supports digital transformation in engineering workflows
- Flexible and can be adapted to similar engineering applications

8. LIMITATIONS

- The system is highly dependent on the format and structure of input Excel files
- Any change in column names or file layout may cause errors in processing
- Limited compatibility with non-standard or corrupted data formats
- Requires initial setup and configuration before execution
- The tool operates mainly on Excel-based data and lacks direct integration with design software like Smart 3D
- Not fully automated in real-time; works on batch processing of data
- Requires basic knowledge of VBA or system usage for troubleshooting
- Performance may slightly decrease with extremely large datasets (very high rows)
- Error handling is limited for unexpected input conditions
- GUI functionality is basic and may not support advanced visualization features
- The tool is platform-dependent (primarily Windows-based environment)
- Security features such as data encryption and user authentication are not included
- Limited flexibility for customization without modifying the source code
- Does not include built-in backup or recovery mechanisms
- Integration with enterprise systems (ERP/PLM) is not available

9. FUTURE SCOPE

- Integration with advanced engineering software such as Smart 3D for real-time data synchronization
- Development of cloud-based platforms to enable remote access and team collaboration
- Implementation of artificial intelligence (AI) for automatic error detection and smart decision-making
- Use of machine learning algorithms to predict data inconsistencies and suggest corrections
- Enhancement of real-time data processing capabilities for instant updates and monitoring
- Expansion of the tool to support other engineering domains such as mechanical, electrical, and structural engineering
- Development of a web-based interface for improved accessibility across different devices
- Addition of advanced data visualization tools such as dashboards, graphs, and reports
- Automation of additional engineering tasks such as report generation and validation
- Integration with enterprise systems like ERP and PLM for better data management
- Improvement in user interface design for better user experience and usability
- Implementation of strong security features such as data encryption and user authentication
- Optimization of system performance to handle very large datasets efficiently
- Development of mobile application support for on-site engineering access
- Inclusion of automated backup and recovery systems for data safety
- Support for multiple file formats beyond Excel for greater flexibility

10. CONCLUSION

This study addressed critical challenges in piping engineering workflows, particularly those related to data inconsistency, manual processing, and complex tagging systems commonly encountered in large-scale industrial projects. By developing automation tools using VBA and C#, the project successfully streamlined the comparison of specification identification sheets, identified discrepancies in commodity codes, and enabled efficient extraction of skipped, modified, added, and deleted components from catalogue data. In addition, the implementation of a tag streamlining tool significantly improved the readability and usability of orthographic diagrams by converting complex instrument tags into simplified sequential identifiers while maintaining traceability through mapping files. The results demonstrated a substantial reduction in manual effort, processing time, and human errors, thereby enhancing overall workflow efficiency and accuracy. The integration of a user-friendly graphical interface further improved accessibility and usability, allowing engineers to execute complex operations with minimal technical expertise. The study highlights the growing importance of automation in modern engineering practices, emphasizing its role in improving productivity, ensuring data reliability, and supporting effective collaboration between design and procurement teams. Although certain limitations such as dependence on structured input formats and limited integration with external systems were identified, the proposed solution provides a strong foundation for future advancements. Overall, the project confirms that automation is a key enabler in digital engineering transformation and offers a practical, scalable approach to improving piping engineering workflows in industrial applications.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to the Department of Mechanical Engineering, VIT Pune, for providing the necessary facilities and academic environment to carry out this work. We extend our thanks to our project supervisor, Dr. G. Suresh Kannan, for his valuable guidance, constructive feedback, and continuous support throughout the project. We also thank the faculty members and laboratory staff for their assistance during the design, modelling, and analysis stages, and we acknowledge our classmates and friends for their cooperation and support.

REFERENCES

1. ASME B31.3, *Process Piping*, American Society of Mechanical Engineers, Latest Edition.
2. ASME B31.1, *Power Piping*, American Society of Mechanical Engineers, Latest Edition.
3. Nayyar, M. L., *Piping Handbook*, 7th Edition, McGraw-Hill Education, 2000.
4. Smith, R., *Chemical Process Design and Integration*, Wiley Publications, 2005.
5. Bentley Systems, *AutoCAD Plant 3D User Guide*, Autodesk Documentation.
6. Intergraph Corporation, *Smart 3D (S3D) Software Documentation*, Hexagon PPM.
7. Microsoft Corporation, *Visual Basic for Applications (VBA) Documentation*, Microsoft Docs.
8. Microsoft Corporation, *.NET Framework and C# Programming Guide*, Microsoft Docs.
9. Pressman, R. S., *Software Engineering: A Practitioner's Approach*, 7th Edition, McGraw-Hill, 2010.
10. Sommerville, I., *Software Engineering*, 10th Edition, Pearson Education, 2016.
11. ISO 15649, *Petroleum and Natural Gas Industries — Piping*, International Organization for Standardization.
12. Kellogg, D. W., *Design of Piping Systems*, Wiley Publications.
13. Rajput, R. K., *Mechanical Engineering Design*, Laxmi Publications.