VISUAL QUALITY CONTROL TOOL

A Mini Project Report Submitted
In partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering (Cyber Security)

by

JANGAM SWATHI
(Regd No: 21N31A6228)
BHRAMANAPALLY SAI LIKHITH
(Regd No: 21N31A6215)
ERUKALA VISHAL
(Regd No: 21N31A6223)

Under the Guidance of

Mrs. R Annapurna
Assistant Professor
Department of Emerging Technologies
MRCET (Autonomous Institution, UGC Govt. of India)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-CYBER SECURITY
(SCHOOL OF EMERGING TECHNOLOGIES)
MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY
(Autonomous Institution – UGC, Govt. of India)
(Affiliated to JNTU, Hyderabad, Approved by AICTE, Accredited by NBA & NAAC – 'A' Grade, ISO 9001:2015 Certified)
Maisammaguda (v), Near Dullapally, Via: Kompally, Hyderabad – 500 100, Telangana State, India

2024-2025

DECLARATION

We hereby declare that the project entitled "**Visual Quality Control Tool**" submitted to **Malla Reddy College of Engineering and Technology,** affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) as part of IV Year B.Tech – I Semester and for the partial fulfillment of the requirement for the award of **Bachelor of Technology** in **Computer Science and Engineering (Cyber Security)** is a result of original research work done by us.

It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree.

Jangam Swathi (21N31A6228)
Bhramanapally Sai Likhith (21N31A6215)
Erukala Vishal (21N31A6223)

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous Institution – UGC, Govt. of India)
(Sponsored by CMR Educational Society)
Recognized under 2(f) and 12 (B) of UGC ACT 1956
Estd : 2004
(Affiliated to JNTUH, Hyderabad, Approved by AICTE- Accredited by NBA & NAAC– 'A' Grade - ISO 9001:2015 Certified)

CERTIFICATE

This is to certify that this is the Bonafide record of the project titled "**Visual Quality Control Tool** submitted by **Swathi**, **Sai Likhith, Vishal** bearing **21N31A6228, 21N31A6215, 21N31A6223** of **B.Tech IV YEAR – I Semester** in the partial fulfillment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering (Cyber Security)**, Dept. of CSE (Emerging Technologies) during the year 2024-2025. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree.

Project Guide
Department of CSE (ET)

Project Coordinator
Department of CSE (ET)

HEAD
OF THE DEPARTMENT

EXTERNAL EXAMINER

Date of Viva-Voce Examination held on:_____

ACKNOWLEDGEMENTS

We feel ourself honored and privileged to place our warm salutation to our college "Malla Reddy College of Engineering and Technology (Autonomous Institution – UGC Govt. of India)" and our Principal **Dr. S Srinivasa Rao,** Professor who gave us the opportunity to do the Project during our IV Year B.Tech I Semester and profound the technical skills.

We express our heartiest thanks to our Director **Dr. V S K Reddy,** Professor for encouraging us in every aspect of our project and helping us realize our full potential.

We are also thankful to our Head of the Department **Dr. M V Kamal,** Professor for providing training and guidance, excellent infrastructure and a nice atmosphere for completing this project successfully.

We would like to express our sincere gratitude and indebtedness to our project supervisor **Mrs. R Annapurna,** Assistant Professor for her valuable suggestions and interest throughout the course of this project.

We convey our heartfelt thanks to our Project Coordinator **Dr P Dileep,** Professor and our Class Teacher and Project Incharge **Mrs. P Satyavathi, Assistant Professor** for allowing for their regular guidance and constant encouragement during our dissertation work.

We would like to thank all our supporting **staff** of the Department of CSE(Emerging Technologies) and even all other department who have been helpful directly and in-directly in making our project a success.

Finally, we would like to take this opportunity to thank our **families** for their support and blessings for completion of our project that gave us the strength to do our project.

Jangam Swathi (21N31A6228)
Bhramanapally Sai Likhith (21N31A6215)
Erukala Vishal (21N31A6223)

ABSTRACT

The project aim is to visual the data that is collected from different sources and check for the spikes and outliers of the data and clean the data and used later for different modelling and other usage of the data.

Each and every minute nearly some millions of data has been generated and it is very difficult to check each record separately manually whether it is correct or not. So, to solve this problem VQC tool is designed, and it helps to visualize the data graphically and can easily identify the spikes and duplicates of the data and can easily be removed and clean the data and use for future purposes.

The increasing reliance on environmental data for scientific research and decision-making necessitates robust tools for data quality control. This project introduces a Visual Quality Control (VQC) Tool, a Python-based graphical user interface (GUI) application designed to enhance data integrity through interactive visualization and manual flagging of anomalies. The VQC Tool integrates with datasets containing key environmental parameters such as temperature, pressure, and humidity, providing users with an interface to select date ranges, explore time-series plots, and engage with geospatial data via a world map visualization.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The VQC tool is an innovative solution designed to visualize data, enabling users to swiftly and effectively identify and rectify anomalies. By graphically representing the data, users can easily spot spikes—sudden and unexpected jumps in data values—and duplicates, ensuring that the dataset is clean and reliable for subsequent analysis and modeling.

In oceanographic data management, maintaining data quality is crucial for ensuring reliable and accurate analysis. Quality control (QC) tools are essential for identifying and addressing anomalies or errors in datasets. This project presents the development of a Visual Quality Control (VQC) Tool, a user-friendly GUI application built using Python that integrates multiple data sources and visualization techniques to aid in data quality assessment.

The VQC Tool is designed for oceanographers and data analysts working with environmental sensor data. It provides an intuitive interface to interact with datasets, allowing users to visualize key parameters such as temperature, pressure, and humidity over time. Additionally, it includes a world map to identify specific geospatial data points of interest. Users can flag data points during the visualization process for further evaluation, ensuring that anomalies are appropriately documented and addressed.

Built with a combination of popular data science libraries such as Pandas, Matplotlib, and Tkinter, the VQC Tool supports interactive engagement with data. Its features include selectable date ranges, customizable plots, and map-based visualization, which enable users to scrutinize data comprehensively. By facilitating the manual identification and flagging of suspicious data, the tool enhances data reliability and aids in maintaining high standards of data integrity within oceanographic research and similar fields.

1.2 MOTIVATION

In today's rapidly evolving digital landscape, organizations across various sectors are witnessing an unprecedented explosion in data generation. With every operational transaction, social media interaction, or sensor measurement, vast quantities of data are produced, creating an invaluable repository of information that, if harnessed correctly, can drive innovation and strategic advantage.

However, the sheer volume of this data brings with it the challenge of maintaining data quality. Reliable and accurate data is the cornerstone of intelligent decision-making, providing insights and forecasts that guide everything from daily operations to long-term strategic planning. High-quality data ensures that business models and predictions reflect reality, rather than skewing the truth with erroneous assumptions.

The traditional approach of manual data inspection is increasingly untenable in this fast-paced environment, as it is not only time-consuming and labor-intensive but also prone to human error. This is where a dedicated Visual Quality Control Tool becomes invaluable. By automating the identification and correction of data anomalies, such a tool empowers data analysts to maintain the integrity and accuracy of data without the bottleneck of manual processing.

1.3 PROBLEM DEFINITION

The main challenge addresses the difficulty in managing the quality of massive datasets being continuously generated and aggregated from multiple sources. Within these vast seas of data, hidden beneath the surface, lie spikes and outliers that can distort analyses and lead to misguided conclusions if not accurately identified and managed.

These anomalies may arise from various sources, including data entry errors, sensor malfunctions, or communication glitches, often manifesting as erroneous or extreme values that do not align with expected data patterns.

Furthermore, the issue of duplicate records introduces complications, such as inaccurate averages and misleading trends, which can significantly affect decision-making and operational strategies if left unaddressed. Traditional visualization techniques and data inspection methods are inadequate for the scale and complexity of modern datasets, making automation essential.

The challenge, therefore, lies in developing a system capable of efficiently and accurately detecting these irregularities and facilitating their rectification, hence ensuring the usability and reliability of data for subsequent analysis and application.

The Visual Quality Control Tool is designed to tackle these specific challenges, providing a comprehensive solution that simplifies data quality management and enhances analytical precision.

1.4 OBJECTIVE OF THE PROJECT

The primary objective of the Visual Quality Control (VQC) Tool is to provide a powerful and intuitive platform for data analysts to effectively manage and enhance data quality. In a landscape where data is generated at an unprecedented pace, ensuring this data is accurate, consistent, and free from anomalies is essential for generating reliable insights and making informed decisions. The VQC Tool is designed to address this need by offering a suite of features that streamline the quality control process through automation and visualization.

At its core, the VQC Tool aims to automate the detection of data anomalies such as spikes, outliers, and duplicates, which are often indicators of errors or inconsistencies. By leveraging advanced algorithms and visual representations, the tool allows users to quickly identify these issues without the need for time-consuming manual inspection.

This automation not only improves efficiency but also increases the accuracy of data quality assessments, reducing the risk of human error.

Furthermore, the tool is designed with user-friendliness in mind, providing an intuitive graphical interface that simplifies the complex task of data visualization. This enables analysts to gain a clear understanding of data patterns and quality issues at a glance, facilitating more rapid decision-making. The VQC Tool also integrates data cleaning functionalities, allowing users to correct identified anomalies directly within the platform, thereby streamlining the end-to-end data preparation process.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM AND PROPOSED SYSTEM

EXISTING SYSTEM

1. Excel:

Excel is used as a quality check tool for data, in excel it uses the "Conditional Formatting" feature to highlight duplicate values in a column. This can help identify and address any redundancy or inconsistency in the data. Set up data validation rules to control the type and range of data entered in specific cells. This helps ensure data accuracy and consistency. Utilize Excel's formula auditing tools to trace precedents and dependents. This can help identify errors or inconsistencies in formulas. Compare data across different columns to ensure consistency.

For example, if you have a "Country" column, use sorting or filtering to identify any inconsistencies in country names. If your data contains information in a single column that should be separated into multiple columns (e.g., splitting a full name into first and last names), use the "Text to Columns" feature. Use pivot tables and charts to profile your data. This can help you understand data distributions, identify outliers, and spot potential issues. Apply conditional formatting to highlight cells that meet specific criteria. For instance, you can use it to highlight values that are above or below a certain threshold. Use the spell check feature to identify and correct typos or misspellings in your data. Utilize Excel's filtering capabilities to focus on specific subsets of data.

This can be helpful for identifying outliers or values that don't conform to expected patterns. If you have two sets of data to compare (e.g., two versions of a dataset), use Excel's functions like VLOOKUP or IF statements to highlight differences.

Disadvantages:

Performance: Struggles with large datasets and slower processing than specialized tools.

Automation: Limited in automating complex quality checks compared to dedicated tools.

Version Control: Difficult to manage in collaborative environments.

Advanced Analytics: Not ideal for complex data analysis tasks.

2. Talend Data Quality is a module within Talend's Data Fabric platform that ensures data integrity and governance, with capabilities for real-time data profiling, cleansing, and masking.

Who it's for: Best suited for organizations seeking a comprehensive, real-time, and user-friendly data quality management solution.

Benefits: Talend's user-friendly interface and built-in machine learning components help address data quality issues.

- Automatically profiles and cleans data in real-time, ensuring high data quality and trust.
- Integrated machine learning provides actionable recommendations for data quality improvements.
- In-built features for masking sensitive data help meet data privacy and protection regulations.
- Talend'sTrust Score offers an immediate, understandable, and actionable assessment of data confidence.
- Talend was acquired by Qlik in May 2023.

PROPOSED SYSTEM

The proposed system is an advanced data quality management tool designed to address the limitations of using Excel for data checks. It offers seamless integration with existing databases and applications, capable of handling various data formats and sources. This system efficiently processes large datasets in real-time, providing immediate feedback and supporting robust automation of repetitive tasks through customizable workflows.

Enhanced data validation and cleaning features include comprehensive error detection and standardization. The tool facilitates collaboration with strong version control and audit trails while offering advanced analytics tools for deep insights and interactive visualizations.

Advantages:

- The VQC Tool Checks and controls the data in the plotting, it helps the analyst to understand the data easily whether it is good data or bad data.
- In the VQC Tool the Time delay is less. Comparing to the existing system.
- Automation of data checking is done.

2.2 FUNCTIONAL REQUIREMENTS

Requirements analysis in systems engineering and software engineering encompasses those tasks that go into determining the needsor conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, suchas beneficiaries or users.

A software requirements specification (SRS) is a document that is created when a detailed description of all aspectsof the software to be built must be specified before the project is to commence. It is important to note that a formal SRS is not always written.

In fact, there are many instances in which effort expended on anSRS might be better spent in other software engineering activities. Requirements analysis is critical to the success of a developmentproject.

Requirements must be actionable, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

SOFTWARE REQUIREMENTS

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints, and user

documentation. The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

1. **Operating system:** Windows 7 or newer. (32-bit or 64-bit).
2. **Python:** Python is a popular, high-level programming language known for its readability and versatility, often used for web development, data science, and scripting. Python IDLE 3.7 version.
3. Code editor like **VSCODE.**
4. **Tkinter:** Tkinter is Python's built-in library for creating graphical user interfaces (GUIs) with widgets like buttons, labels, and text boxes. It offers a simple way to build cross-platform desktop applications.
5. **DATA BASE:** SQL server (STRUCTURED QUERY LANGUAGE).
6. **Matplotlib:** Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Create publication quality plots. Make interactive figures that can zoom, pan, update.
7. **Pandas:** Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. Pandas is mainly used for data analysis and associated manipulation of tabular data in Data Frames.
8. **Basemap:** Basemap is a great tool for creating maps using python in a simple way. It's a matplotlib extension, so it has got all its features to create data visualizations, and adds the geographical projections and some datasets to be able to plot coast lines, countries, and so on directly from the library.
9. **Numpy:** NumPy, which stands for Numerical Python, is a powerful open-source library in Python primarily used for numerical and scientific computing. It provides support for multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

HARDWARE REQUIREMENTS

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python /Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

1. Ram: 4GB or more.
2. Processor: i3 7th gen or more.
3. Processor Speed: 250MHz to 833MHz.
4. Hard disk space: 10GB or more.

CHAPTER 3

SOFTWARE REQUIREMENTS

3.1 SOFTWARE

1. **Python**:

Python is a powerful yet beginner-friendly programming language that has become a favorite for its versatility. From web development and data analysis to automation and scientific computing, Python tackles a wide range of tasks. Its clean, readable syntax, thanks to its reliance on indentation, makes code easy to understand and maintain. Backed by a large and active community and a vast array of libraries, Python empowers both beginners and seasoned programmers to bring their ideas to life.

We are using Python IDLE 3.7 version for this project.

2. Tkinter:

Tkinter provides a variety of widgets like buttons, labels, and text boxes, letting you design user-friendly applications. With Python's clear syntax and Tkinter's intuitive tools, you can build cross-platform desktop applications without needing to learn a separate GUI framework.

Buttons: The code uses tk.Button to create four buttons labeled "1", "2", "3", and "4". Each button is associated with a function (red, blue, green, yellow) that changes the color of the selected data points on the plots and saves the updated data to a CSV file. These buttons provide the user interface for classifying the data points.

"Delete" Button: A "Delete" button is also created using tk.Button, linked to a delete() function that removes data points with a specific flag from the CSV file.

Layout Management (Implicit): The .place() method is used to position the buttons on the GUI window. While not explicitly shown, there's likely a tk.Tk() window and potentially other widgets (like frames or labels) to organize the layout, though this is not directly visible in the snippet.

3. **Matplotlib:**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Create publication quality plots. Make interactive figures that can zoom, pan, update.

4. **Pandas:**

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. Pandas is mainly used for data analysis and associated page-09 manipulation of tabular data in Data Frames. Pandas allows importing data from various file formats such as comma-separated values, JSON, Parquet, SQL database tables or queries, and Microsoft Excel.

Pandas is the core data manipulation library in this project. It's used for:

- **Data Loading:** The code uses pd.read_csv (inferred from the to_csv function) to load data from a CSV file ("maindata.csv") into a Pandas DataFrame (df1). This DataFrame likely stores the time series data (dates, temperature, pressure, humidity). Another DataFrame, mapd, likely holds the geographical data.
- **Data filtering:** The onselect, onselect2, and onselect3 functions use boolean indexing with Pandas to filter df1 based on the user's selection from the plots. The condition

  (df1['date'] >= xmind) & (df1['date'] <= xmaxd) & (df1['meantemp'] >= ymin) & (df1['meantemp'] <= ymax) (and similar for pressure and humidity) select rows matching the selected region.
- **Data Modification:** The red, blue, green, and yellow functions modify the df1 DataFrame by adding a 'flag' column (or updating existing values) to indicate the classification of selected data points. These changes are then saved back to the CSV file.
- **Data Access:** Pandas provides efficient ways to access data columns (df1['date'], df1['meantemp'], etc.) for plotting and analysis.

5. Numpy:

While NumPy isn't explicitly used in many direct functions calls in this snippet, it's very likely implicitly used *behind the scenes* by Pandas. Pandas is built on top of NumPy, and NumPy arrays are the underlying data structure for Pandas Data Frames. Therefore, NumPy's efficient array operations are implicitly used for data manipulation within the Pandas operations (filtering, indexing, etc.). The code might benefit from more explicit use of NumPy if performance becomes a bottleneck.

3.2 MODULES

Making of this project consisted of different modules such as,

- User Interface Module
- Data Processing Module
- Data Visualization Module
- System Integration

1. User Interface (UI) Module:

Functionality: This module is responsible for the visual elements and user interaction of the application. It's what the user sees and interacts with.

Components:

- **Platform Selection Dropdown:** Allows the user to choose a location (in this case, "mumbai"). This likely affects the data loaded and displayed.
- **Date Range Selection:** Enables the user to specify the time period for the data to be visualized.

- **"Plot" Button:** Triggers the data processing and visualization based on the user's selections.
- **Map Display:** Shows a geographical representation of data points (likely the locations where measurements were taken).
- **Graph Displays:** Presents the time-series data in multiple graphs (Temperature, Pressure, Humidity).
- **Navigation Controls:** Buttons for navigation within the application (home, back, forward, zoom).
- **Delete Button:** Presumably allows the user to clear selections or reset the view.
- **Technology:** Likely developed using a GUI framework (e.g., Java Swing, PyQt, a web framework like React or Angular).

2. Data Processing Module:

Functionality: This is the "behind-the-scenes" component that handles the data. It takes the user's input (platform and date range) and prepares the data for visualization.

Tasks

- **Data Loading:** Reads the relevant data from a data source (database, CSV file, etc.) based on the selected platform and date range.
- **Data Filtering/Cleaning:** This may involve removing outliers, handling missing values, or transforming the data into a suitable format.
- **Data Aggregation:** May perform calculations or aggregations on the data (e.g., calculating averages, calculating daily totals).
- **Data Formatting:** Prepares the data into the correct structure for the visualization module (e.g., creating arrays or data structures that the plotting library can understand).
- **Technology:** Likely uses programming languages like Python (with libraries like Pandas, NumPy), R, or Java.

3. Data Visualization Module:

Functionality: This module takes the processed data from the Data

Processing Module and creates the visual representations (graphs and maps).

Components:

- Time-Series Graphs: Generates line graphs for temperature, pressure, and humidity over time.
- Geographical Map: Creates a map showing the spatial distribution of data points.
- Technology: Likely uses a plotting library (e.g., Matplotlib, Seaborn in Python; ggplot2 in R; D3.js, Chart.js in JavaScript). The map visualization might utilize a mapping library (e.g., Leaflet, Google Maps API).

4. System Integration:

Functionality: This module handles the communication and data flow between the different parts of the system.

Tasks:

- Connecting UI and Data Processing: Manages the transfer of user input (platform, date range) to the data processing module.
- Connecting Data Processing and Visualization: Passes the processed data from the data processing module to the visualization module.
- Data Source Management: Handles the connection to and retrieval of data from the underlying data source (database, files).
- Technology: This involves the overall architecture of the system, including how the modules communicate (e.g., through function calls, message queues, or APIs). It might also involve database interaction libraries

CHAPTER 4

SYSTEM DESIGN AND UML DIAGRAM

4.1 DATAFLOW DIAGRAM

The data flow diagram depicts a Visual Quality Control (VQC) tool's workflow. A user initiates the tool, selecting a data range and location to filter data. The system then visualizes the data, allowing for spike detection and manual data flagging. Data cleaning and saving options are also provided, enabling users to refine and store processed data. The diagram shows a linear flow, but likely involves iterative steps for data refinement and visualization. The absence of error handling and data source specifics suggests a simplified representation of the tool's functionality.

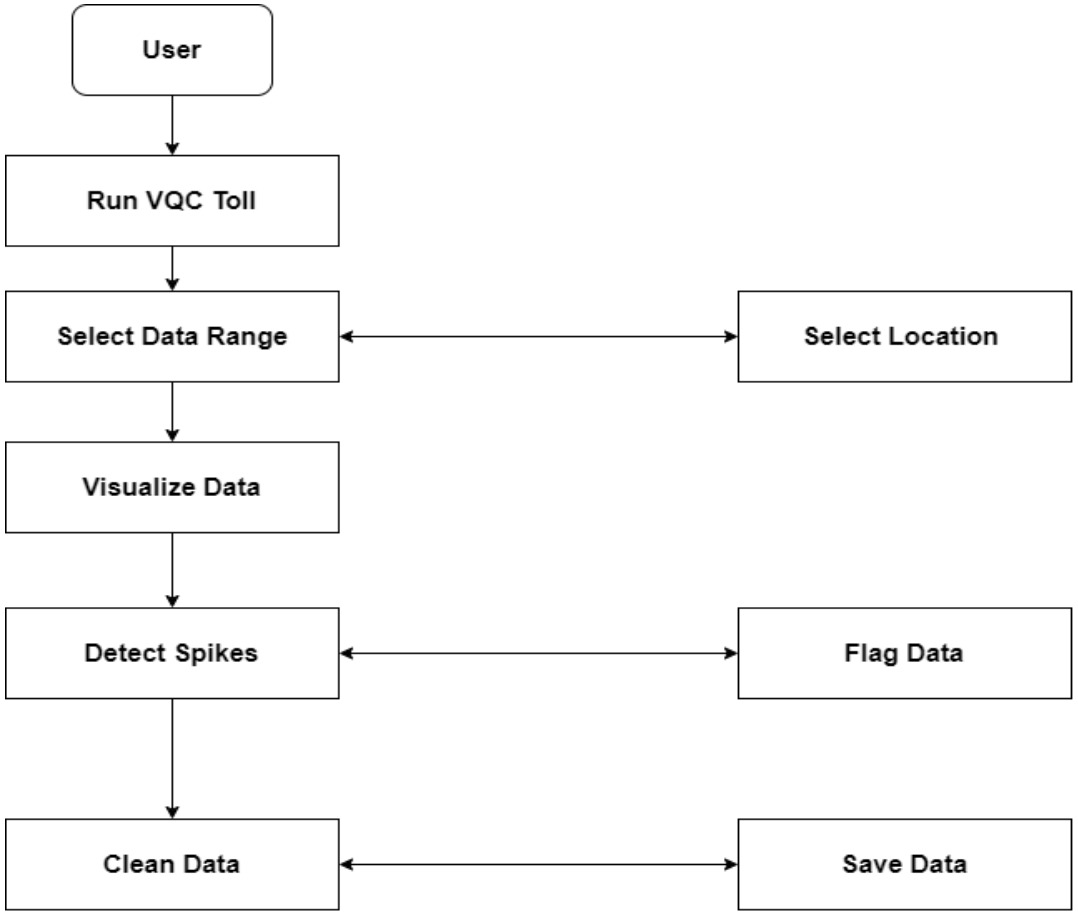


Fig 4.1 DATAFLOW DIAGRAM

4.2 ARCHITECTURE DIAGRAM

An Architectural diagram is a visual representation of the structure, components, relationships, and interactions within a system or application. It shows cases how different elements works either a simple element or diagram or even complex ones.

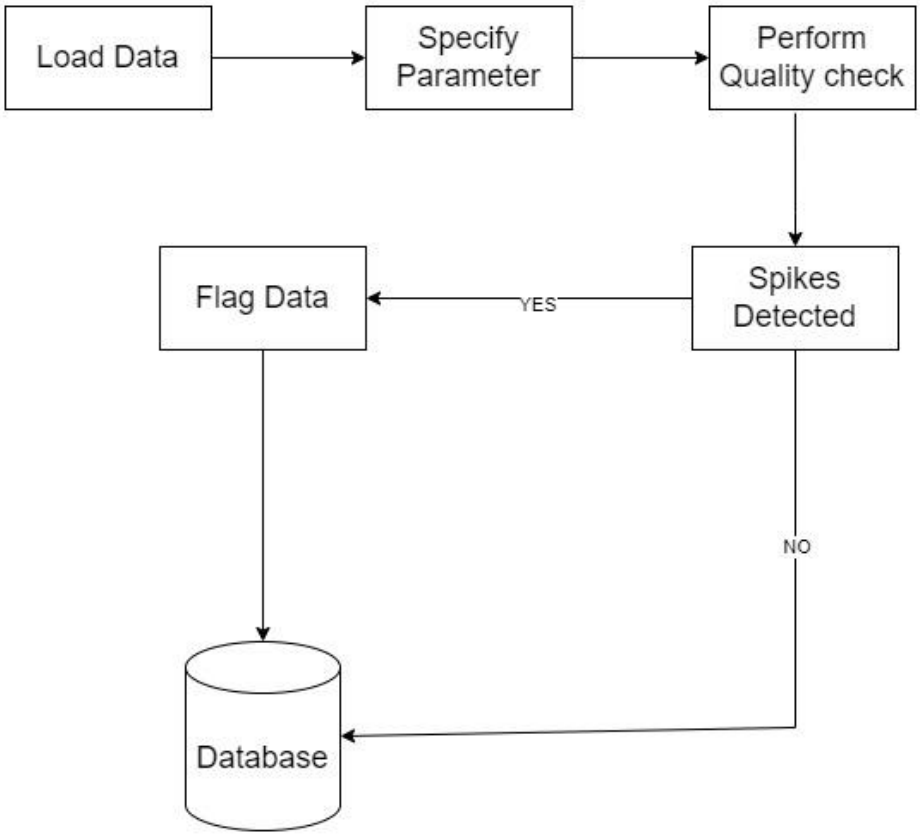


Fig 4.2 ARCHITECTURE DIAGRAM

4.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML 2.0 helped extend the original.UML specification to cover a wider portion of software development efforts including agile practices. Improved integration between structural models like class diagrams and behavior models like activity diagrams. Added the ability to define a hierarchy and decompose a software system into structure diagram, interaction overview diagram, and timing diagram. It also renamed statechart diagrams to state machine diagrams, also known as state diagrams.

USECASE DIAGRAM

A use case describes a specific interaction between a user (or actor) and a system to achieve a particular goal. It outlines the sequence of steps involved, focusing on what the system does from the user's perspective. It's *not* a detailed technical specification but a high-level description of functionality.

Run VQC Tool: This is the overarching use case. The user initiates the tool. All other use cases are sub-functions within this.
Select Data Range: The user selects a time period for analysis.
Select Location: The user chooses a geographic location for the data.
Visualize Data: The system displays the selected data graphically.
Detect Spikes: The system automatically identifies unusual data points.
Flag Data: The user manually marks data points for review.
Clean Data: The user or system modifies or removes flagged data points.
Save Data: The user saves the processed data.

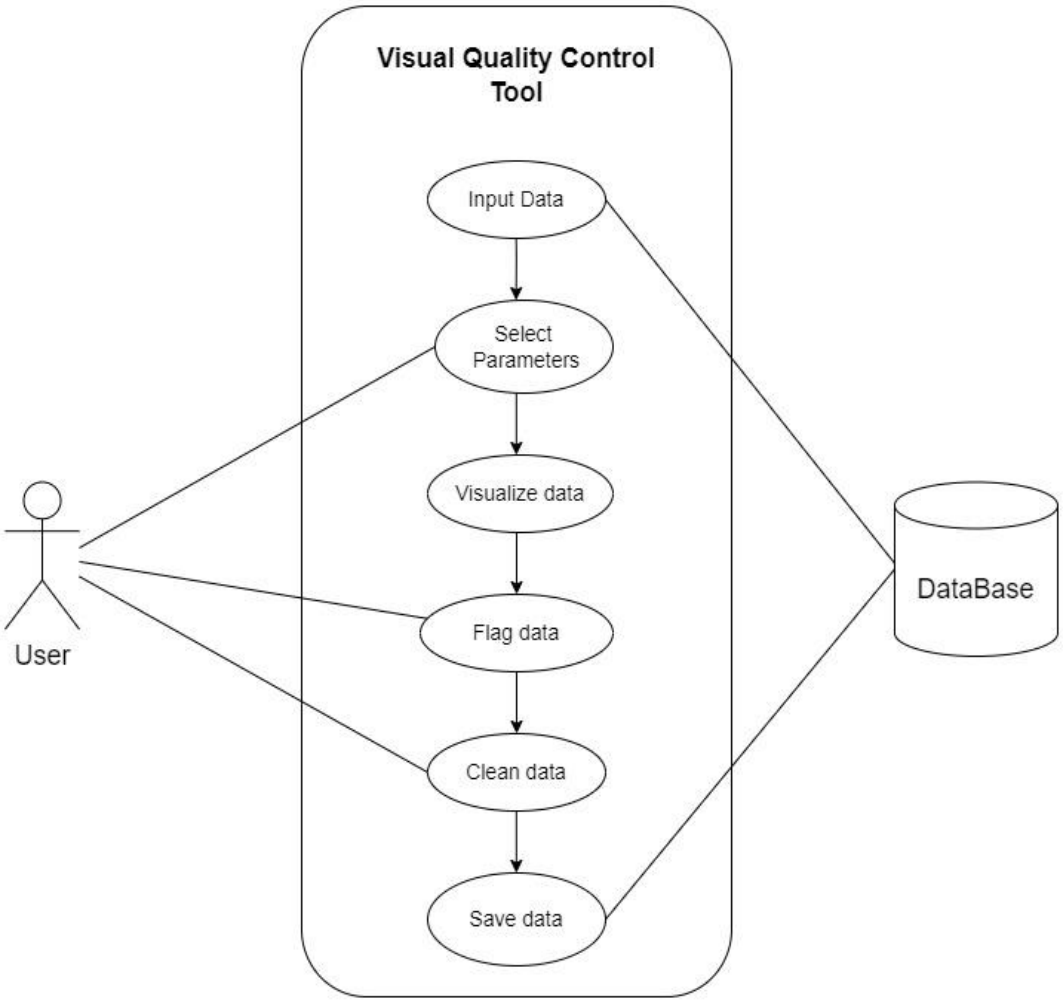


Fig 4.3.1 USECASE DIAGRAM

CLASS DIAGRAM

A class diagram is a static model in the Unified Modeling Language (UML) that describes the structure of a system by showing the classes, their attributes (data), methods (functions), and the relationships between them. It's a blueprint of the system's objects.

Relationships:

a. User interacts with GUI**:** The user interacts with the system through the GUI.

b. GUI uses DataLoader: The GUI uses the DataLoader to get data.

c. DataLoader passes data to DataVisualizer and SpikesDetector: The loaded data is passed to these classes for processing.

d. DataVisualizer shows data on GUI: Visualizations are displayed on the GUI.

e. SpikesDetector flags data for Cleaner: The Cleaner receives flagged data points from the SpikesDetector.

f. Cleaner sends cleaned data to DataSaver: The cleaned data is passed to the DataSaver for storage.

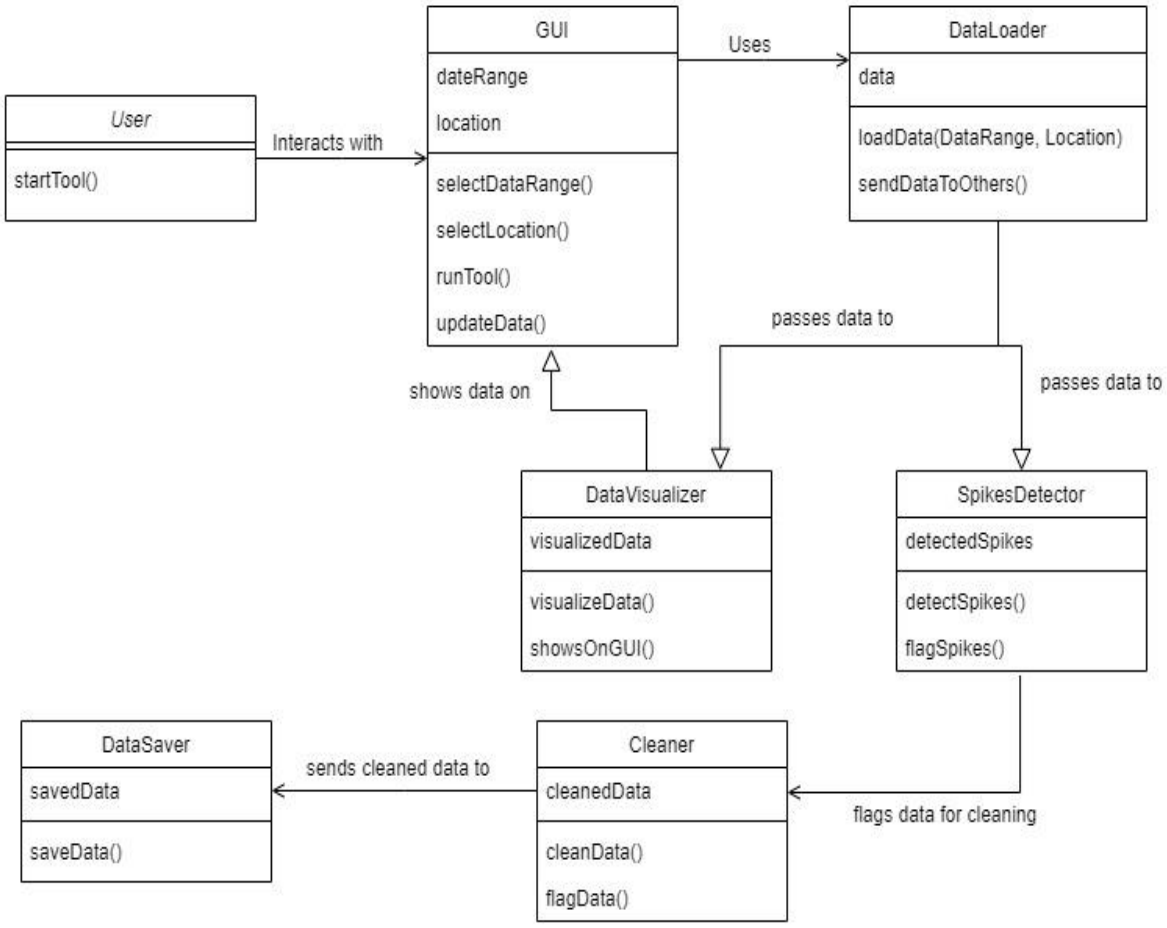


Fig 4.3.2 CLASS DIAGRAM

SEQUENCE DIAGRAM

A sequence diagram in UML is an interaction diagram that details how operations are carried out—showing the objects and classes involved and the order of messages exchanged between them. It focuses on the timing and sequence of messages, illustrating how different parts of the system collaborate to perform a task. It's often used to model the interactions within a single use case.

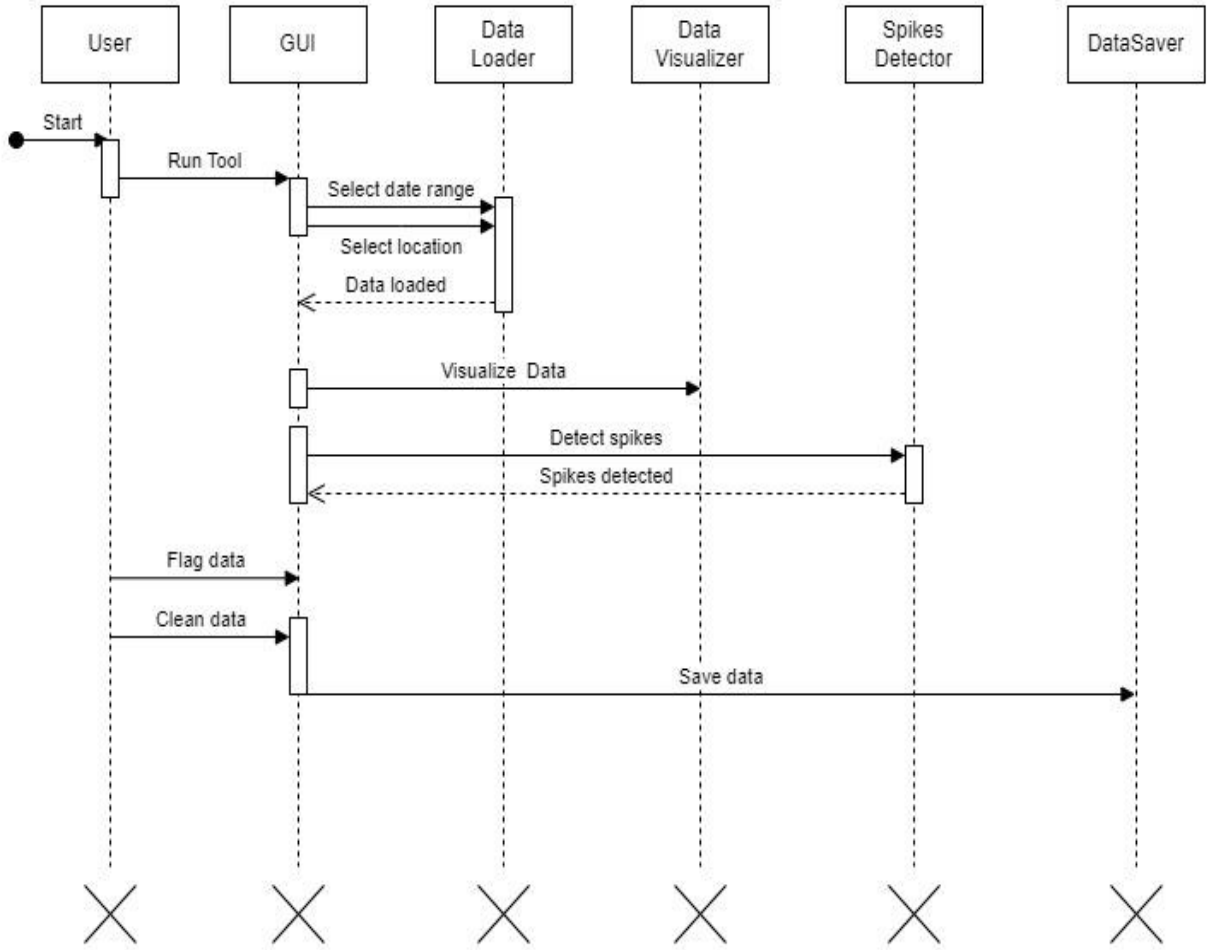


Fig 4.3.3 SEQUENCE DIAGRAM

CHAPTER 5

SOFTWARE DEVELOPMENT LIFE CYCLE

5.1 PHASES OF SDLC

The Software Development Life Cycle (SDLC) is a structured process that enables the production of high-quality, low-cost software, in the shortest possible production time. The goal of the SDLC is to produce superior software that meets and exceeds all customer expectations and demands.

In other words, a life cycle model maps the various activities performed on a software product from its inception to retirement. Different life cycle models may plan the necessary development activities to phases in different ways. Thus, no element which life cycle model is followed, the essential activities are contained in all life cycle models though the action may be carried out in distinct orders in different life cycle models. During any life cycle stage, more than one activity may also be carried out.

Need of SDLC

The development team must determine a suitable life cycle model for a particular plan and then observe to it. Without using an exact life cycle model, the development of a software product would not be in a systematic and disciplined manner. When a team is developing a software product, there must be a clear understanding among team representative about when and what to do. Otherwise, it would point to chaos and project failure.

This problem can be defined by using an example. Suppose a software development issue is divided into various parts and the parts are assigned to the team members. From then on, suppose the team representative is allowed the freedom to develop the roles assigned to them in whatever way they like.

It is possible that one representative might start writing the code for his part, another might choose to prepare the test documents first, and some other engineer might begin with the design phase of the roles assigned to him. This would be one of the perfect methods for project failure.

The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. In detail, the SDLC methodology focuses on the following phases of software development:

- Requirement analysis
- Planning Stage
- Analysis Stage
- Design Stage
- Testing
- Implementation
- Deployment

Planning Stage

Before we even begin with the planning stage, the best tip we can give you is to take time and acquire proper understanding of app development life cycle. The planning stage (also called the feasibility stage) is exactly what it sounds like: the phase in which developers will plan for the upcoming project. It helps to define the problem and scope of any existing systems, as well as determine the objectives for their new systems. By developing an effective outline for the upcoming development cycle, they'll theoretically catch problems before they affect development and help to secure the funding and resources, they need to make their plan happen.

Analysis Stage

The analysis stage includes gathering all the specific details required for a new system as well as determining the first ideas for prototypes.

Developers may:

- Define any prototype system requirements
- Evaluate alternatives to existing prototypes
- Perform research and analysis to determine the needs of end-users

Furthermore, developers will often create a software requirement specification or SRS document. This includes all the specifications for software, hardware, and network requirements for the system they plan to build. This will prevent

them from overdrawing funding or resources when working at the same place as other development teams.

Design Stage

The design stage is a necessary precursor to the main developer stage. Developers will first outline the details for the overall application, alongside specific aspects , such as its:

- User interfaces
- System interfaces
- Network and network requirements
- Databases

They'll typically turn the SRS document they created into a more logical structure that can later be implemented in a programming language.

Operation, training, and maintenance plans will all be drawn up so that developers know what they need to do throughout every stage of the cycle moving forward.

Once complete, development managers will prepare a design document to be referenced throughout the next phases of the SDLC.

Development Stage

The development stage is the part where developers actually write code and build the application according to the earlier design documents and outlined specifications.

This is where Static Application Security Testing or SAST tools come into play. Product program code is built per the design document specifications.

In theory, all of the prior planning and outlined should make the actual development phase relatively straight forward. Developers will follow any coding guidelines as defined by the organization and utilize different tools such as compilers, debuggers, and interpreters. Programming languages can include staples such as C++, PHP, and more. Developers will choose the right programming code to use based on the project specifications and requirements.

Testing Stage

Building software is not the end. Now it must be tested to make sure that there aren't any bugs and that the end-user experience will not negatively be affected at any point. During the testing stage, developers will go over their software with a fine-tooth comb, noting any bugs or defects that need to be tracked, fixed, and later retested. It's important that the software overall ends up meeting the quality standards that were previously defined in the SRS document.

Depending on the skill of the developers, the complexity of the software, and the requirement for the end-user, testing can either be an extremely short phase or take a very long time. Take a look at our top 10 best practices for software testing projects for more information.

Implementation and Integration Stage

After testing, the overall design for the software will come together. Different modules Or designs will be integrated into the primary source code through developer efforts, usually by leveraging training environments to detect further errors or defects.

The information system will be integrated into its environment and eventually installed. After passing this stage, the software is theoretically ready for market and may be provided to any end-users.

Maintenance Stage

The SDLC doesn't end when software reaches the market. Developers must now move to a maintenance mode and begin practicing any activities required to handle issues reported by end-users. Furthermore, developers are responsible for implementing any changes that the software might need after deployment.

This can include handling residual bugs that were not able to be patched before launch or resolving new issues that crop up due to user reports. Larger systems may require longer maintenance stage compared to smaller systems.

CHAPTER-6

IMPLEMENTATION

6.1 SAMPLE CODE

```
import tkinter as tk  #To create GUI application
from tkinter import ttk
from tkinter import *
from tkcalendar import DateEntry
import pandas as pd
from datetime import datetime       #To convert into time series
import matplotlib.pyplot as plt     #To create plot graphs
from matplotlib.backends.backend_tkagg
import FigureCanvasTkAgg,NavigationToolbar2Tk
import matplotlib.dates as mdates
from mpl_toolkits.basemap import Basemap     #To create the world map
from matplotlib.figure import Figure
from matplotlib.widgets import RectangleSelector


top=tk.Tk()     # Create a Tkinter application window
top.title("INCOIS")
top.geometry('1536x864')
Label(top,text='VQC Tool',font=("Arial", 30)).place(x=30,y=10)
frame1=LabelFrame(top,width=500,height=804,bg='#999999' )
frame1.place(x=0,y=60)
canvas = tk.Canvas(top,width=1036,height=804)
canvas.place(x=500,y=60)
frame2=Frame(canvas)
frame2.place(x=500,y=60)
Label(frame1,text='Platform :',font=30,bg='#999999').place(x=20,y=20) options
= ["mumbai"]
combobox = ttk.Combobox(frame1, values=options, state='readonly',
width=30,justify='center',font=30)
combobox.place(x=130,y=20)
```

```
Label(frame1,text='Date Range ',font=30,bg='#999999').place(x=20,y=60)
sd=DateEntry(frame1,selectmode='day',font=45,justify='center',width=12)
sd.place(x=20,y=105)
Label(frame1,text='To',font=45,bg='#999999').place(x=200,y=105)
ed=DateEntry(frame1,selectmode='day',font=45,justify='center',width=12)
ed.place(x=250,y=105)
data_loaded=True
    mapd=pd.read_excel(r"C:\Users\swathi\Downloads\incoisdata.xlsx")
    mapd = pd.DataFrame(mapd)
except Exception as e:
    data_loaded = False
    error_label = tk.Label(top, text="Input the data file!", font=("Arial", 16),
fg="red")
    error_label.place(x=500, y=30)
c=Canvas(frame1)
c.place(x=20,y=250)
fig = plt.Figure(figsize=(4.4, 3.4))
basemap = Basemap(projection='cyl', resolution='l',
ax=fig.add_subplot(111))
basemap.ax.set_position([0, 0.08, 0.995, 0.85])
basemap.drawcoastlines()
basemap.drawcountries()
basemap.drawstates()
canvasm = FigureCanvasTkAgg(fig, master=c)
canvasm.draw()
canvasm.get_tk_widget().pack()
toolbar1 = NavigationToolbar2Tk(canvasm, c)
toolbar1.update()
try:
    df1=pd.read_csv(r'C:\Users\swathi\Downloads\maindata.csv')
    df1 = pd.DataFrame(df1)
    df1['date'] = pd.to_datetime(df1['date'], format='%Y-%m-%d')
except Exception as e:
    data_loaded = False
```

```
    error_label = tk.Label(top, text="Input the data file!", font=("Arial", 16),
fg="red")
    error_label.place(x=500, y=30)

# create a Matplotlib figure
figure = plt.Figure(figsize=(10.6, 7.4), dpi=100)
ax1,ax2,ax3= figure.subplots(3, sharex=True)
a,=ax1.plot(df1.date,df1.meantemp, marker='o',alpha=0.5)
b,=ax2.plot(df1.date,df1.meanpressure,marker='o',alpha=0.5)
d,=ax3.plot(df1.date,df1.humidity,marker='o',alpha=0.5)
line = FigureCanvasTkAgg(figure, master=frame2)
def onselect(eclick, erelease):     #creation of function with name onselect
    rs2.set_visible(False)
    rs3.set_visible(False)
    rs1.set_visible(False)
    global xmin, xmax, ymin, ymax
    xmin, xmax = sorted([eclick.xdata, erelease.xdata])
    ymin, ymax = sorted([eclick.ydata, erelease.ydata])
    xmind= mdates.num2date(xmin)
    xmaxd=mdates.num2date(xmax)
    date_string = xmind.strftime('%Y-%m-%d %H:%M:%S')
    date_string1 = xmaxd.strftime('%Y-%m-%d %H:%M:%S')
    xmind = datetime.strptime(date_string, '%Y-%m-%d %H:%M:%S')
    xmaxd = datetime.strptime(date_string1, '%Y-%m-%d %H:%M:%S')
    df1['date'] = df1['date'].dt.to_pydatetime()
    selected_data = df1[(df1['date'] >= xmind) & (df1['date'] <= xmaxd) &
(df1['meantemp']>= ymin) & (df1['meantemp'] <= ymax)]
    a.set_data(df1['date'], df1['meantemp'])
    ax1.draw_artist(a)
    def red():
        ax1.scatter(selected_data['date'], selected_data['meantemp'], color='red',
s=30)
        df1.loc[selected_data.index,'flag']=1
        df1.to_csv(r'C:\Users\swathi\Downloads\maindata.csv' , index=False)
```

```
            ax1.relim()
            ax1.autoscale_view()
            line.draw()
        def blue():
            ax1.scatter(selected_data['date'], selected_data['meantemp'], color
='blue', s=30)
            df1.loc[selected_data.index,'flag']=2
            df1.to_csv(r'C:\Users\swathi\Downloads\maindata.csv' , index=False)
            ax1.relim()
            ax1.autoscale_view()
            line.draw()
        def green():
            ax1.scatter(selected_data['date'], selected_data['meantemp'],
color='green', s=30)
            df1.loc[selected_data.index,'flag']=3
            df1.to_csv(r'C:\Users\swathi\Downloads\maindata.csv' , index=False)
            ax1.relim()
            ax1.autoscale_view ()
            line.draw ()
        def yellow ():
            ax1.scatter(selected_data['date'], selected_data['meantemp'],
color='yellow', s=30)
            df1.loc[selected_data.index,'flag']=4
            df1.to_csv (r'C:\Users\swathi\Downloads\maindata.csv' , index=False)
            ax1.relim()
            ax1.autoscale_view()
            line.draw()
        button = tk.Button(frame1, text="1",command=red).place(x=40,y=670)
        button = tk.Button(frame1,text="2",command=blue).place(x=120,y=670)
        button = tk.Button(frame1, text="3",command=green).place(x=200,y=670)
        button = tk.Button(frame1, text="4",command=yellow).place(x=280,y=670)

    #create of box selection for the graphs
    rect = plt.Rectangle((0,0), 0, 0, alpha=0.2, facecolor='blue', edgecolor='red',
```

```
visible=False)
ax1.add_patch(rect)
rs = RectangleSelector(ax1, onselect, button=[1],minspanx=5, minspany=5,
spancoords='pixels',interactive=True)

def delete():
    csv_file_path= r'C:\Users\swathi\Downloads\maindata.csv'
    df = pd.read_csv(csv_file_path)
    df = df[df['flag'] != 4]
    df.to_csv(csv_file_path, index=False)
button =tk.Button(frame1,text="Delete",command=delete).place(x=130,y=720)

def onselect2(eclick, erelease):
    rs.set_visible(False)
    rs3.set_visible(False)
    rs1.set_visible(False)
    global xmin, xmax, ymin, ymax
    xmin, xmax = sorted([eclick.xdata, erelease.xdata])
    ymin, ymax = sorted([eclick.ydata, erelease.ydata])
    xmind= mdates.num2date(xmin)
    xmaxd=mdates.num2date(xmax)
    date_string = xmind.strftime('%Y-%m-%d %H:%M:%S')
    date_string1 = xmaxd.strftime('%Y-%m-%d %H:%M:%S')
    xmind = datetime.strptime(date_string, '%Y-%m-%d %H:%M:%S')
    xmaxd = datetime.strptime(date_string1, '%Y-%m-%d %H:%M:%S')
    df1['date'] = df1['date'].dt.to_pydatetime()
    selected_data = df1[(df1['date'] >= xmind) & (df1['date'] <= xmaxd)
&(df1['meanpressure']>= ymin) & (df1['meanpressure'] <= ymax)]
    b.set_data(df1['date'], df1['meanpressure'])
    ax2.draw_artist(b)
    def red():
        ax2.scatter(selected_data['date'], selected_data['meanpressure'],
color='red')
        df1.loc[selected_data.index,'flag']=1
```

```
        df1.to_csv(r'C:\Users\swathi\Downloads\maindata.csv' , index=False)
        ax2.relim()
        ax2.autoscale_view()
        line.draw()
    def blue():
        ax2.scatter(selected_data['date'], selected_data['meanpressure'],
color='blue')
        df1.loc[selected_data.index,'flag']=2
        df1.to_csv(r'C:\Users\swathi\Downloads\maindata.csv' , index=False)
        ax2.relim()
        ax2.autoscale_view()
        line.draw()
    def green():
        ax2.scatter(selected_data['date'], selected_data['meanpressure'],
color='green')
        df1.loc[selected_data.index,'flag']=3
        df1.to_csv(r'C:\Users\swathi\Downloads\maindata.csv' , index=False)
        ax2.relim()
        ax2.autoscale_view()
        line.draw()
    def yellow():
        ax2.scatter(selected_data['date'], selected_data['meanpressure'],
color='yellow')
        df1.loc[selected_data.index,'flag']=4
        df1.to_csv(r'C:\Users\swathi\Downloads\maindata.csv' , index=False)
        ax2.relim()
        ax2.autoscale_view()
        line.draw()
    button = tk.Button(frame1, text="1",command=red).place(x=40,y=670)
    button = tk.Button(frame1, text="2",command=blue).place(x=120,y=670)
    button = tk.Button(frame1, text="3",command=green).place(x=200,y=670)
    button = tk.Button(frame1, text="4",command=yellow).place(x=280,y=670)
rect2 = plt.Rectangle((0,0), 0, 0, alpha=0.2, facecolor='blue', edgecolor='red',
visible=False)
```

```
ax2.add_patch(rect2)
rs2 = RectangleSelector(ax2, onselect2, button=[1],minspanx=5,
minspany=5, spancoords='pixels',interactive=True)

def onselect3(eclick, erelease):
    rs.set_visible(False)
    rs2.set_visible(False)
    rs1.set_visible(False)
    global xmin, xmax, ymin, ymax
    xmin, xmax = sorted([eclick.xdata, erelease.xdata])
    ymin, ymax = sorted([eclick.ydata, erelease.ydata])
    xmind= mdates.num2date(xmin)
    xmaxd=mdates.num2date(xmax)
    date_string = xmind.strftime('%Y-%m-%d %H:%M:%S')
    date_string1 = xmaxd.strftime('%Y-%m-%d %H:%M:%S')
    xmind = datetime.strptime(date_string, '%Y-%m-%d %H:%M:%S')
    xmaxd = datetime.strptime(date_string1, '%Y-%m-%d %H:%M:%S')
    df1['date'] = df1['date'].dt.to_pydatetime()
    selected_data = df1[(df1['date'] >= xmind) & (df1['date'] <= xmaxd) &
(df1['humidity'] >=ymin) & (df1['humidity'] <= ymax)]
    d.set_data(df1['date'], df1['humidity'])
    ax3.draw_artist(d)
    def red():
        ax3.scatter(selected_data['date'], selected_data['humidity'], color='red')
        df1.loc[selected_data.index,'flag']=1
        df1.to_csv(r'C:\Users\swathi\Downloads\maindata.csv' , index=False)
        ax3.relim()
        ax3.autoscale_view()
        line.draw()
    def blue():
        ax3.scatter(selected_data['date'], selected_data['humidity'], color='blue')
        df1.loc[selected_data.index,'flag']=2
        df1.to_csv(r'C:\Users\swathi\Downloads\maindata.csv' , index=False)
        ax3.relim()
```

```
            ax3.autoscale_view()
            line.draw()
        def green():
            ax3.scatter(selected_data['date'], selected_data['humidity'],
color='green')
            df1.loc[selected_data.index,'flag']=3
            df1.to_csv(r'C:\Users\swathi\Downloads\maindata.csv' , index=False)
            ax3.relim()
            ax3.autoscale_view()
            line.draw()
        def yellow():
            ax3.scatter(selected_data['date'], selected_data['humidity'],
color='yellow')
            df1.loc[selected_data.index,'flag']=4
            df1.to_csv(r'C:\Users\swathi\Downloads\maindata.csv' , index=False)
            ax3.relim()
            ax3.autoscale_view()
            line.draw()
        button = tk.Button(frame1, text="1",command=red).place(x=40,y=670)
        button = tk.Button(frame1, text="2",command=blue).place(x=120,y=670)
        button = tk.Button(frame1, text="3",command=green).place(x=200,y=670)
        button = tk.Button(frame1, text="4",command=yellow).place(x=280,y=670)

    rect3 = plt.Rectangle((0,0), 0, 0, alpha=0.2, facecolor='blue', edgecolor='red',
    visible=False)
    ax3.add_patch(rect3)
    rs3 = RectangleSelector(ax3, onselect3, button=[1],minspanx=5,
    minspany=5, spancoords='pixels',interactive=True)

    def onselect1(eclick, erelease):
        rs.set_visible(False)
        rs3.set_visible(False)
        rs2.set_visible(False)
        global xmmin, xmmax, ymmin, ymmax
```

```
    xmmin, xmmax = sorted([eclick.xdata, erelease.xdata])
    ymmin, ymmax = sorted([eclick.ydata, erelease.ydata])
    indices = (mapd['LONGITUDE'] >= xmmin) & (mapd['LONGITUDE'] <=
xmmax) &(mapd['LATITUDE'] >= ymmin) & (mapd['LATITUDE'] <= ymmax)
    def red():
        basemap.scatter(mapd.loc[indices, 'LONGITUDE'], mapd.loc[indices,
'LATITUDE'],c='red', edgecolors='red')
        fig.canvas.draw_idle()
    def blue():
        basemap.scatter(mapd.loc[indices, 'LONGITUDE'], mapd.loc[indices,
'LATITUDE'],c='blue', edgecolors='blue')
        fig.canvas.draw_idle()
    def green():
        basemap.scatter(mapd.loc[indices, 'LONGITUDE'], mapd.loc[indices,
'LATITUDE'],c='green', edgecolors='green')
        fig.canvas.draw_idle()
    def yellow():
        basemap.scatter(mapd.loc[indices, 'LONGITUDE'], mapd.loc[indices,
'LATITUDE'],c='yellow', edgecolors='yellow')
        fig.canvas.draw_idle()
    button = tk.Button(frame1, text="1",command=red).place(x=40,y=670)
    button = tk.Button(frame1, text="2",command=blue).place(x=120,y=670)
    button = tk.Button(frame1, text=”3",command=green).place(x=200,y=670)
    button = tk.Button(frame1, text="4",command=yellow).place(x=280,y=670)

rect1 = plt.Rectangle((0,0), 0, 0, alpha=0.2, facecolor='blue', edgecolor='red',
visible=False)
basemap.ax.add_patch(rect1)
rs1 = RectangleSelector(basemap.ax, onselect1, button=[1],minspanx=5,
minspany=5, spancoords='pixels',interactive=True)

def main():
    sdate=sd.get()      #getting dates from the user
    edate=ed.get()
```

```
    ssdate=datetime.strptime(sdate,'%m/%d/%y')
    eedate=datetime.strptime(edate,'%m/%d/%y')
    ax1.set_xlim(ssdate, eedate)
    ax2.set_xlim(ssdate, eedate)
    ax3.set_xlim(ssdate, eedate)
    ax1.set_ylabel("Temperature(celsius)”)
    ax2.set_ylabel("Pressure(pascal)")
    ax3.set_ylabel("Humidity(grams/m3)")
    ax3.set_xlabel("Time-series")
    basemap.scatter(mapd['longitude'], mapd['latitude'])
    fig.canvas.draw_idle()
    canvasm.get_tk_widget().pack()
    line.draw()
    line.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
    canvas.create_window((0, 0), window=frame2, anchor='nw')
    toolbar.update()
    toolbar.pack( side=tk.TOP,fill=tk.X)

toolbar = NavigationToolbar2Tk(line, frame2)
toolbar.pack_forget()

button = tk.Button(frame1,
text="Plot",font=30,command=main).place(x=20,y=155)
top.mainloop()
```

CHAPTER 7

TESTING

7.1 INTRODUCTION

SOFTWARE TESTING

Software Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction. Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, and history is full of such examples.

- Meets the business and technical requirements that guided its design and Development.
- Works as expected and can be implemented with the same characteristics.

White Box Testing

It is a testing technique in which software's internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

Examples of White Box Test Cases

- **Data Validation Module:**

  Scenario: The system needs to validate email addresses. A white box test would examine the specific regular expression (regex) used for validation.

  White Box Test Case: The tester would:

  - Review the regex code within the validation module.
  - Design test cases to cover various scenarios: valid email addresses (various formats), invalid email addresses (missing "@" symbol, incorrect domain extensions, etc.), edge cases (extremely long email addresses).
  - Execute the tests and verify that the validation function correctly identifies valid and invalid email addresses according to the implemented regex. This goes beyond simple input/output; it examines *how* the validation is done.

- **Data Cleaning Module:**

  Scenario: The system deduplicates data based on a combination of fields (e.g., Temperature, Humidity, Pressure).

  White Box Test Case: The tester would:

  - Examine the deduplication algorithm (e.g., fuzzy matching, exact matching).
  - Design test cases with various levels of similarity in the data to cover different matching scenarios.
  - Use debugging tools to step through the algorithm and verify that it correctly identifies and handles duplicate records.
  - Test scenarios involving partial matches, considering how the algorithm handles different levels of similarity.

- **Data Integration Module:**

  Scenario: The module handles connections to different databases.

  White Box Test Case: The tester would:

  - Inspect the code that handles database connections.

- Design tests to check that the correct database drivers are loaded and used.
- Test the error handling when database connections fail (e.g., network issues, incorrect credentials).
- Verify that the code correctly handles different database types.

Black Box Testing

It is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

Examples of Black Box Test Cases:

- **Data Integration:** Test importing data from various sources (databases, files) and formats. Verify that the data is correctly imported and handled.
- **Data Validation:** Test the system's ability to detect different types of data errors (e.g., missing values, invalid formats, outliers). Verify that appropriate error messages are generated.
- **Data Cleaning:** Test the system's data cleaning capabilities (e.g., deduplication, standardization). Verify the accuracy and completeness of the cleaning process.
- **Reporting and Analytics:** Test the generation of reports and dashboards. Verify that the reports are accurate, complete, and easy to understand.
- **Security:** Test access controls and data encryption to ensure that sensitive data is protected.

FIG 7.1.1 BLACK BOX TESTING

TESTING METHODS

Unit Testing

A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system. In most programming languages, that is a function, a subroutine, a method or property.

The isolated part of the definition is important. In his book "Working Effectively with Legacy Code", author Michael Feathers states that such tests are not unit tests when they rely on external systems: “If it talks to the database, it talks across Network, it touches the file system, it requires system configuration, or it can't be run at the same time as any other test.

A unit can be almost anything you want it to be -- a line of code, a method, or a class. Generally, though, smaller is better. Smaller tests give you a much more Granular view of how your code is performing. There is also the practical aspect that when you test very small units, your tests can be run fast; like a thousand tests in a second fast.

Functional Testing

Functional Testing Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

- Functions: Identified functions must be exercised.
- Output: Identified classes of software outputs must be exercised.
- Systems/Procedures: system should work properly

Integration Testing

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group.

Test Case for Excel Sheet Verification: Here in machine learning we are

dealing with dataset which is in excel sheet format so if any test case is needed, we need to check excel file. Later on, classification will work on the respective columns of dataset.

The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units. Once all the components or modules are working independently, then we need to check the data flow between the dependent modules is known as integration testing.

End-to-End Testing

End-to-end (E2E) testing for a Visual Quality Control (VQC) project verifies the complete application flow, from data loading and processing (using Pandas/NumPy) to data visualization (Matplotlib) and user interaction, ensuring accurate data display and functionality.

Performance Testing

Performance testing, on the other hand, assesses the application's responsiveness, stability, and scalability under various conditions. This includes load testing to determine performance under multiple users, stress testing to find breaking points, response time testing to measure speed, resource usage monitoring (CPU, memory, disk I/O), and scalability testing to evaluate the application's ability to handle increasing data and user loads.

PARAMETERS

Maximum and Minimum Ranges for the Parameters.

Sl No	Parameter	Range	
		Lower Limit	Upper Limit
1	Humidity	0	100
2	Air pressure	985	1025
3	Air Temperature	10	40
4	Wind direction	0	360
5	Wind Speed	0	35
6	Wind Gust	0	45
7	Current Speed	0	300
8	Current Direction	0	360
9	SST Skin	15	35
10	SST3m	15	35
11	Conductivity	20	65
12	Salinity	15	40
13	Wave height	0	15
14	Wave Direction	0	360

TABLE 1. MINIMUM AND MAXIMUM RANGES FOR THE PARAMETERS

Sensor Specifications

Sensor	Range	Accuracy	Resolution	Duration / Freq.
Air pressure	800 – 1100 hPa	± 0.1 hPa	0.01 hPa	5 sec, 1 Hz
Air temperature	10 – 50oC	± 0.1oC	0.01oC	10 min, 1 Hz
Wind* (spd, dir)	0 – 60 ms-1, 0 – 360o	± 1.5% FS, ± 3.6o	0.07 ms-1, 0.1o	10 min, 1 Hz
Water Temp**	-5 – 45oC	± 0.1oC	0.01oC	10 min, 1 Hz
Conductivity**	2 – 77 m mho cm-1	± 0.06 mmho cm-1	0.01 m mhocm-1	10 min, 1 Hz
Surf. Cur.* *(spd, dir)	0 – 6 ms-1, 0 – 360o	± 3% FS, ± 2o	0.005 ms-1, 0.36o	10 min, 1 Hz
Wave** (ful lspectrum)	± 20m, 0 –360c	± 10 cm, ± 5o	1 cm, < 0.1o	17 min, 1 Hz
Humidity & Air Temp.*	Humidity 0:100% Ai r Temp:-40 to + 60c	Humidity±1%R Air Temp… ±3 oC	-	10 min, 1 Hz

TABLE 2. SENSOR SPECIFICATIONS

Threshold Values referred from TAO Quality Control Documents

Parameter	**Daily parameters that will generate error alerts**
Barometric pressure	BP changes > 5 mb from previous day
Air temperature	Daily AT changes > 5°C from previous day
Relative humidity(RH)	Daily RH changes > 20% from previous day
Wind direction	Direction varies more than 90° from previous day
Wind Speed	Speed changes more than 5 m/s from previous day
SST	SST changes > 5 °C from previous day
Current Direction	Direction varies more than 90° from previous day
Current Speed	Speed changes greater than 50 cm/s from previous day
Salinity	Salinity > 0.5 psu from previous day

TABLE 3. THRESHOLD VALUES

DATA TESTS

(i) Impossible Date Test:

The test requires that the observation date and time from the float be sensible.

- ❖ Year greater than 1997
- ❖ Month in range 1 to 12
- ❖ Day in range expected for month
- ❖ Hour in range 0 to 23
- ❖ Minute in range 0 to 59

Action: If any one of the conditions is failed, the date should be flagged as bad data.

(ii) Stuck Value Test

This test looks for consecutive identical measurements (O1, O2) of each parameter of an individual buoy.

Action: If O2 = O1 then O2 should be flagged as wrong. Where O2 is the measurement being tested and O1 is the previous value.

(iii) Impossible Location Test

The test requires that the observation latitude and longitude from the float be sensible.

- ❖ Latitude in range 0 to 30.
- ❖ Longitude in range 70 to 100.
- ❖ The change in Buoy position from the deployment position should not greater than 6 Nano Meters.

Action: If either latitude or longitude fails, the position should be flagged as bad data.

(iv) Spike Test

The difference between sequential measurements, where one measurement is quite different than adjacent ones, is a spike. The difference between 2 consecutive measurements is to be compared with the threshold values given by TAO data quality control program. The threshold values for some of the parameters are shown in Table 2.

Test Value = | O2 - O1| (“| | “ represents MOD and gives the absolute value). Where O2 is the measurement being tested as a spike, and O1 is the previous value.

Action: If the Test Value crosses the threshold value of the corresponding parameter then O2 should be flagged as wrong.

(vi) Range Test

This test checks range limits for each parameter. The Lower and Upper Limits for the ranges are shown in Table 3.

Action: if the value fails this test, it should be flagged as bad data.

Flagging of Data

We can flag the data into four types

1.Good data

2.Probably good data

3.Stuck Values

4.Worst data

Graph Representation of Data

- **Data Before Quality Check and After Quality Check**

  Spike Check (Before)

  This section shows the results of Real Time Quality Control applied to data obtained from 2 buoys (CVAL and OB12), currently transmitting data for the period October 2008.

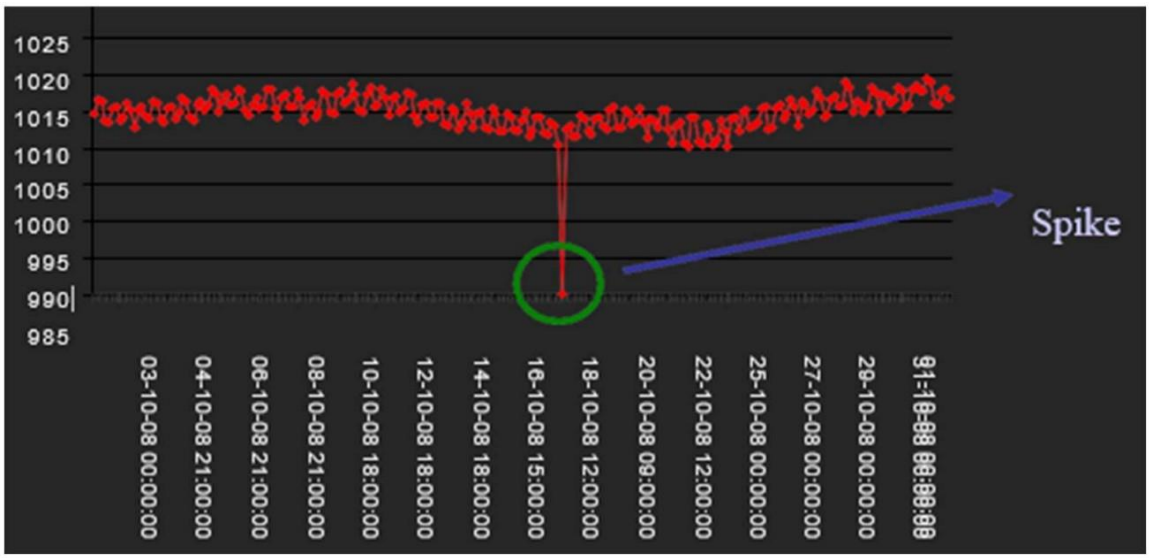


FIG 7.1.2 SPIKE CHECK(BEFORE)

Spike Check (After)

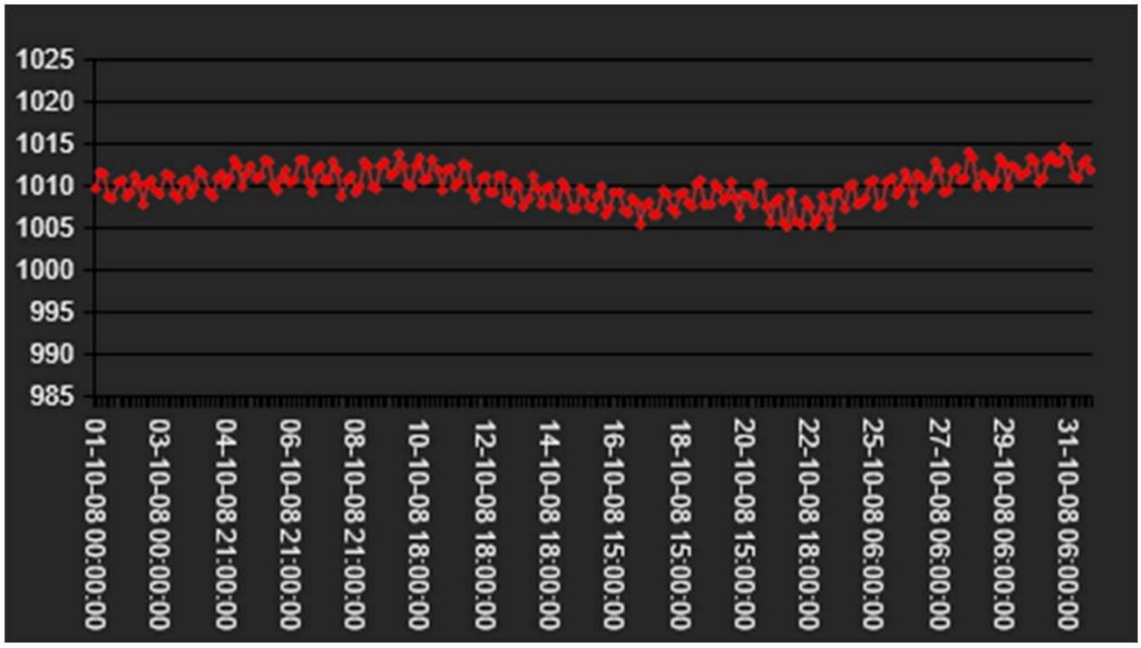


FIG 7.1.3 SPIKE CHECK(AFTER)

Observe the spike in 17/10/2008. It is because of the observation that crossed the range limit. So it is flagged as ‘ 4 ‘ during quality control procedure.

7.2 SAMPLE TEST CASES

V	Description	Input (steps)	Expected result	Obtained result	Result	Remarks (If fails)
TC_1.	Date Selection	Attempt to select dates outside the defined range of the data.	The graph is plotted successfully.	Error message indicating the date range is invalid.	Fail	
TC_2.	Data File Input	No data file is selected or uploaded for processing.	The graph is plotted successfully, displaying the correct data based on the content of .csv file.	An error message is displayed, clearly stating that a data file is required.	Fail	
TC_3.	Data File Input	(maindata .csv) is uploaded.	The graph is plotted successfully, displaying the correct data based on the content of .csv file.	The graph is plotted successfully, displaying the correct data based on the content of maindata. csv.	Pass	

TABLE 4. SAMPLE TEST CASES

CHAPTER 8

OUTPUT SCREENS

8.1 SCREENSHOTS

Application Window

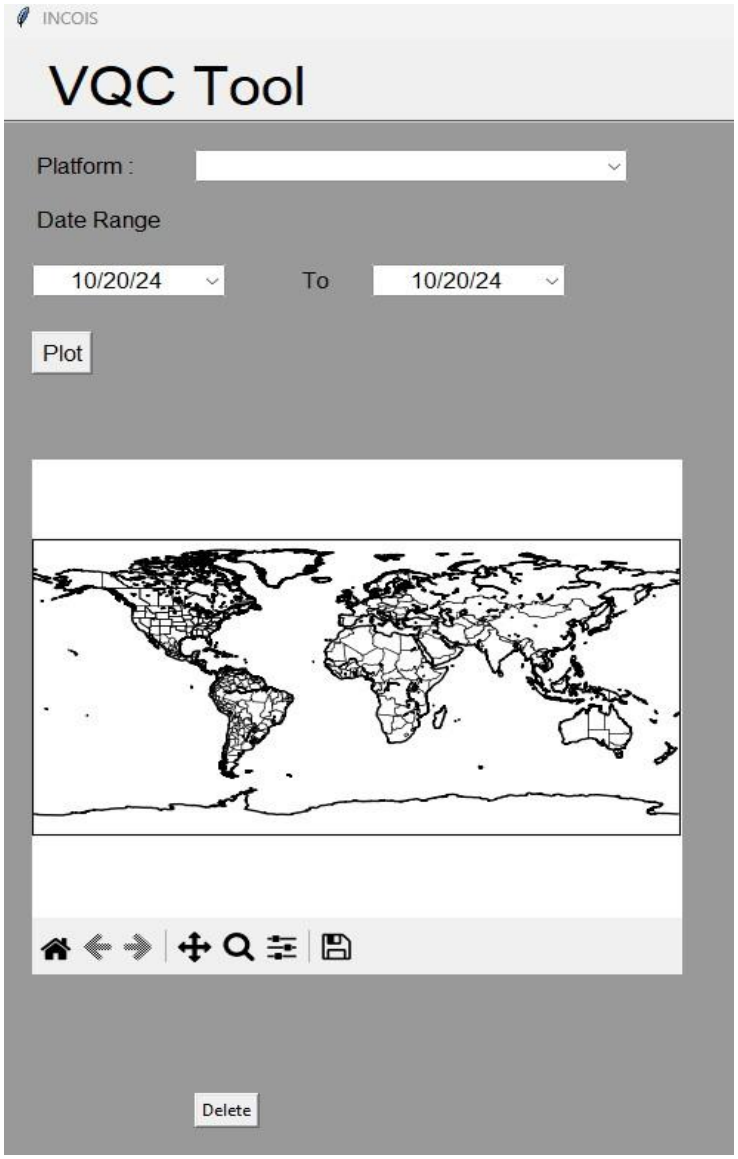


FIG 8.1.1 APPLICATION WINDOW PAGE

VQC Data Visualization

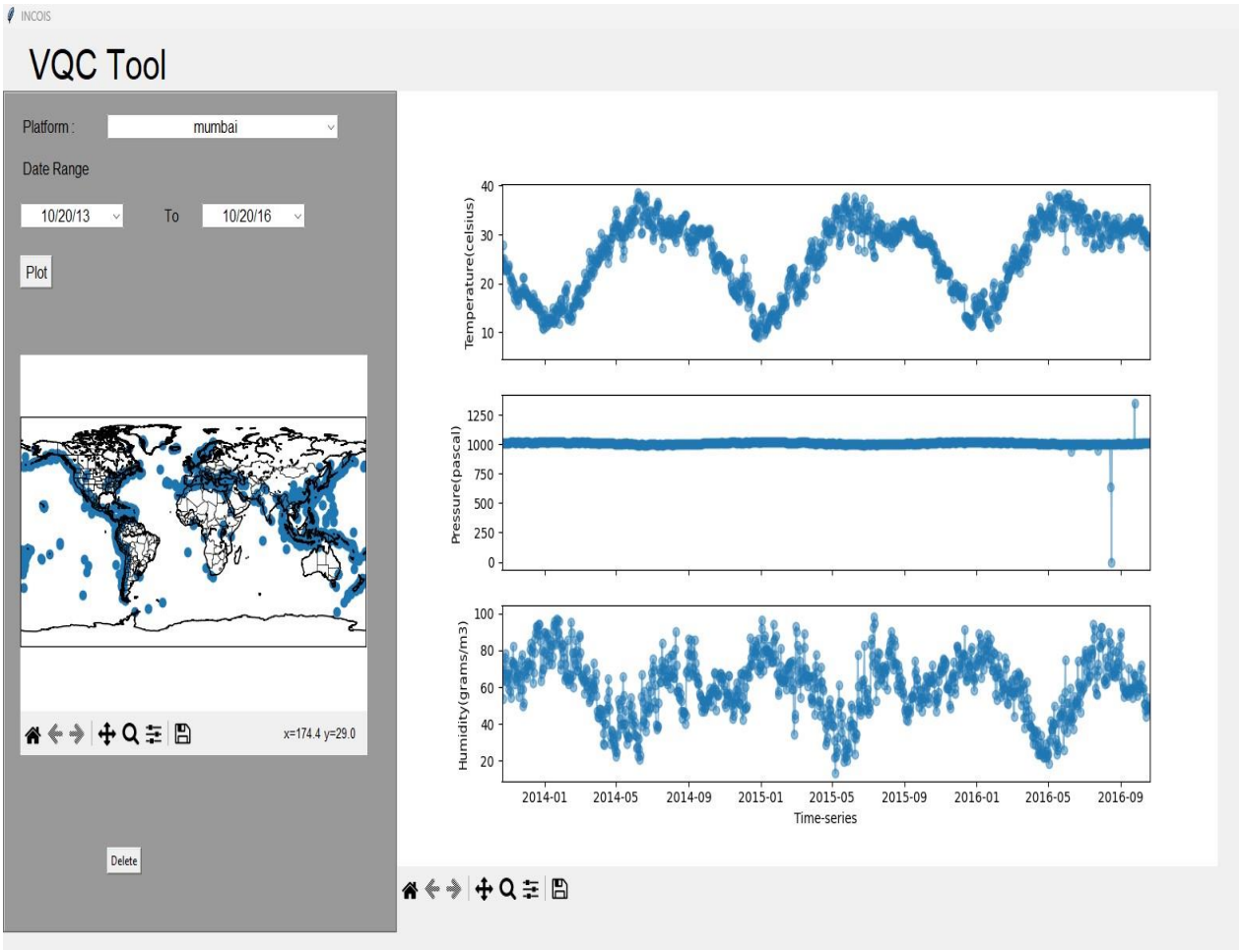


FIG 8.1.2 VQC DATA VISUALIZATION

Spikes Detection

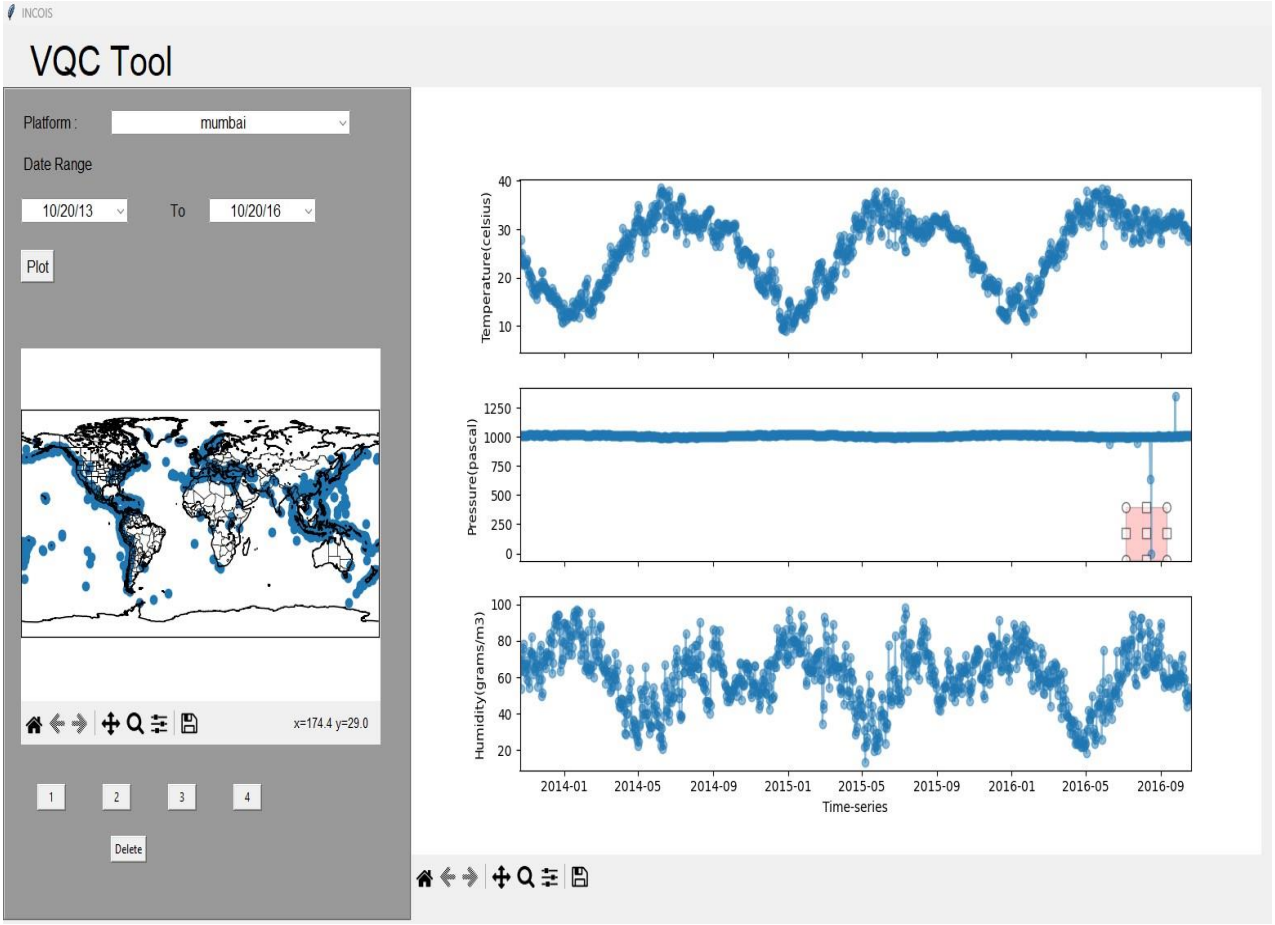


FIG 8.1.3 SPIKES DETECTION

Flagging data points with spikes

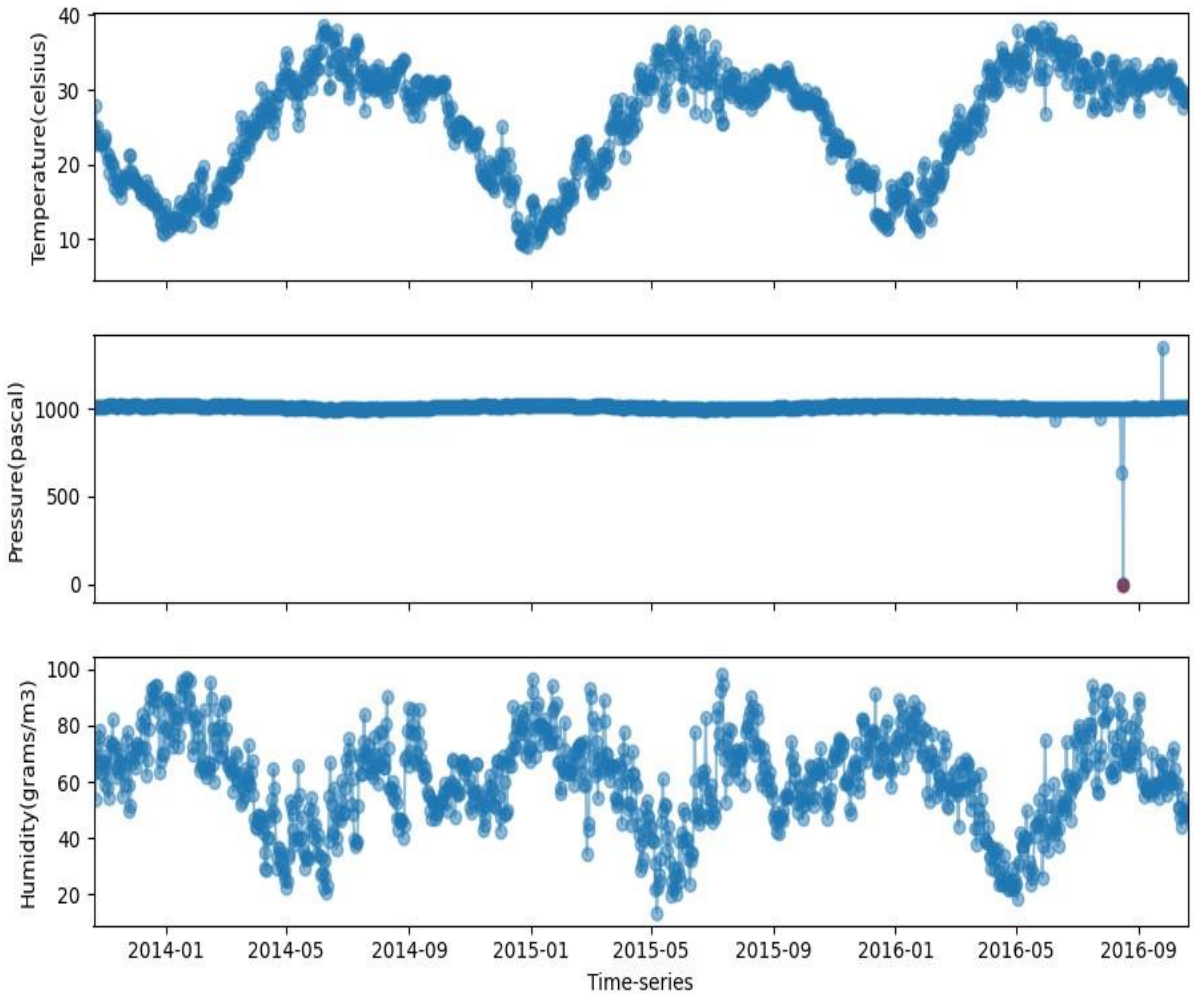


FIG 8.1.4 FLAGGING DATA

Spike Deletion

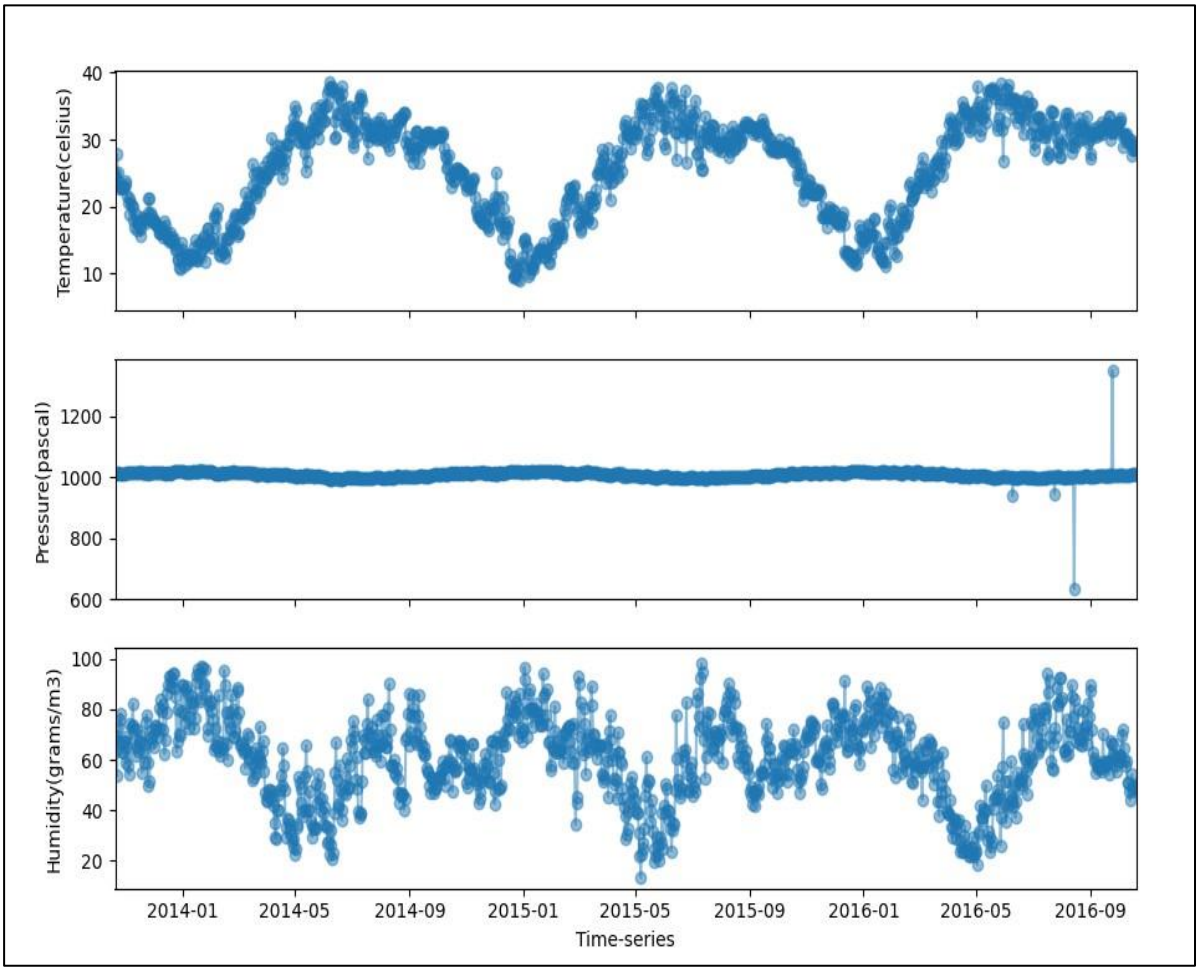
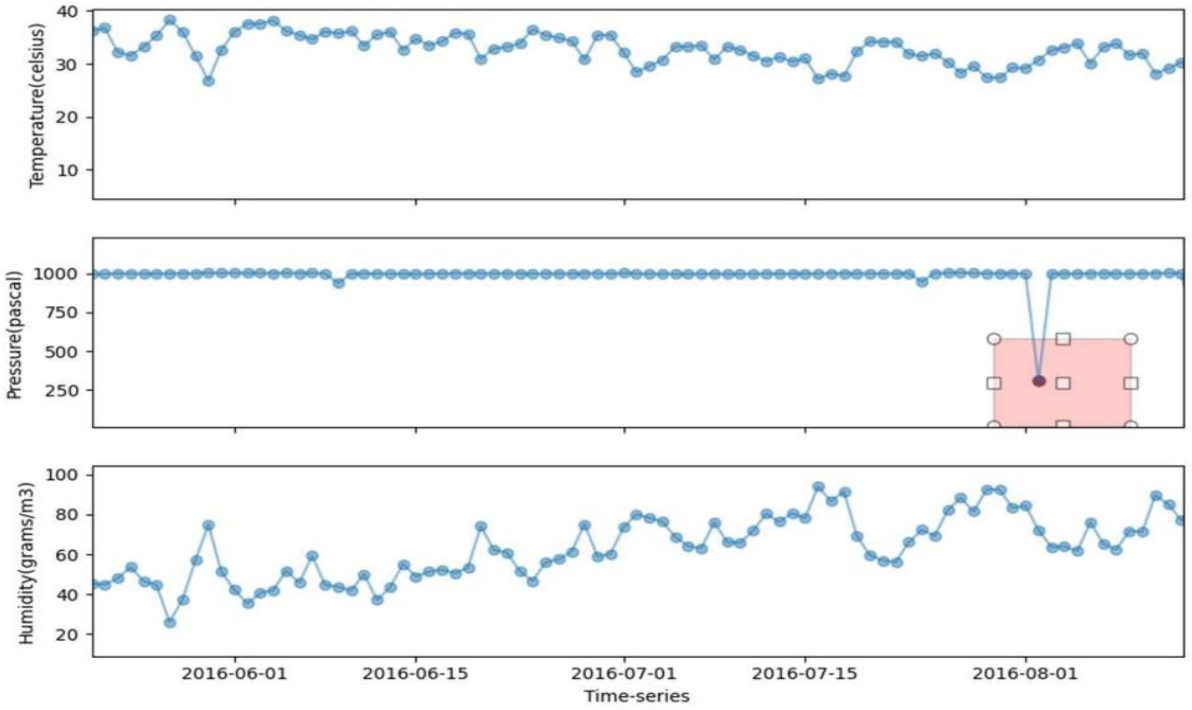
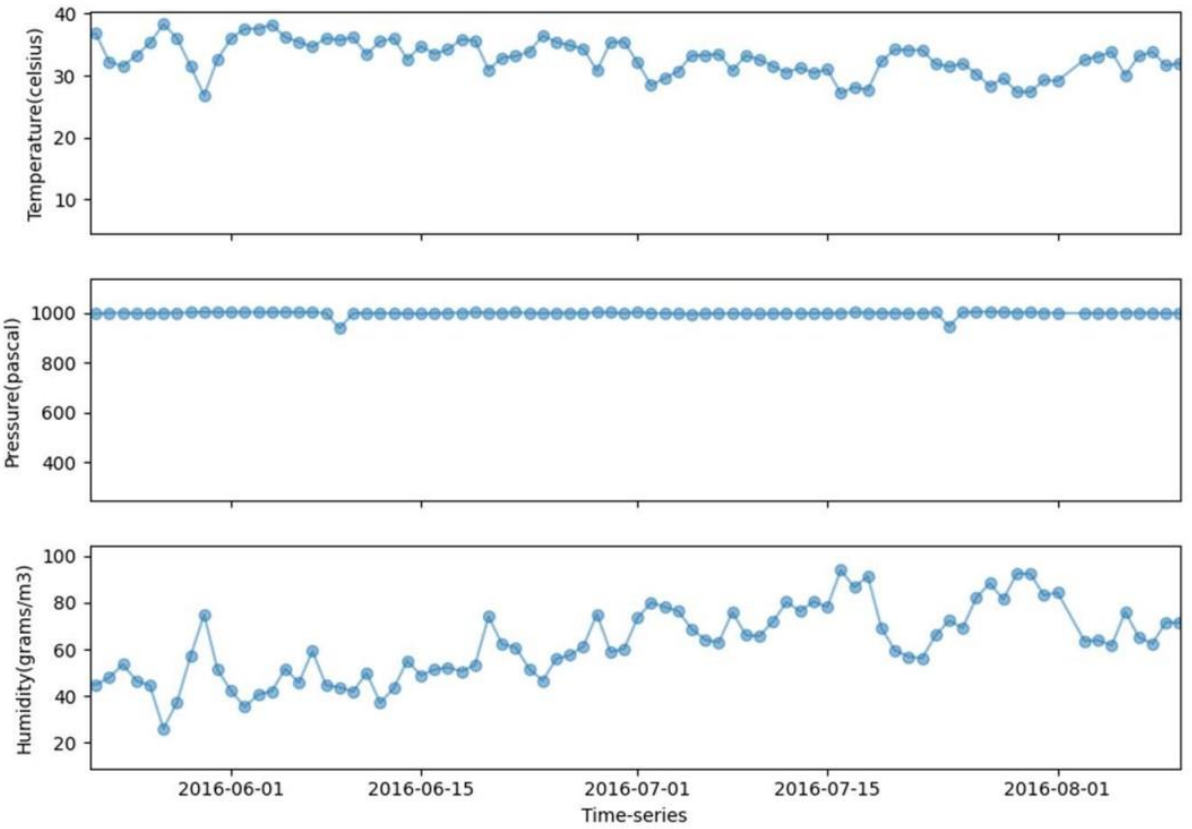


FIG 8.1.5 SPIKE DELETION

Before



After



CHAPTER 9

CONCLUSION AND FUTURE SCOPE

9.1 CONCLUSION

In conclusion, the development of a Visual Quality Control Tool represents a significant advancement in the field of data management. The increasing volume and complexity of data necessitate automated solutions that go beyond traditional manual inspection methods. This tool addresses the critical need for efficient and accurate identification and remediation of data quality issues, such as anomalies and duplicates, within large datasets. By combining automated anomaly detection with intuitive data visualization, the tool empowers data analysts to maintain data integrity, improve the reliability of analyses, and ultimately support more informed and effective decision-making.

The future scope of the project, encompassing machine learning integration, real-time processing, and enhanced collaborative features, further underlines its potential to become an indispensable component of modern data management strategies across diverse industries. The successful implementation of this tool will contribute significantly to improving data quality and the trustworthiness of data-driven insights, thereby enhancing operational efficiency and strategic planning.

9.2 FUTURE SCOPE

The future scope of this Visual Quality Control (VQC) tool project is significant, encompassing several areas of improvement and expansion:

1. **Enhanced Anomaly Detection:**
   - More sophisticated algorithms: Move beyond simple spike detection to incorporate machine learning (ML) techniques like anomaly detection using algorithms. This would allow for the identification of more subtle and complex anomalies that might go unnoticed with simpler methods.
   - Adaptive thresholds: Implement dynamic thresholding that adjusts based on the historical data and learned patterns, rather than relying

on static thresholds set manually.

- Anomaly classification: Develop the capacity to classify detected anomalies into different types (e.g., sensor malfunction, sudden environmental change, etc.), providing more context and actionable insights.
- Root cause analysis: Integrate features that help pinpoint the root cause of detected anomalies, perhaps by correlating data from multiple sensors or other data sources.

2. Improved Visualization and User Experience:

- Interactive dashboards: Develop interactive dashboards allowing users to explore data in different ways, zoom in on specific time periods or locations, and customize visualizations.
- Advanced charting options: Add more advanced chart types (e.g., heatmaps, scatter plots, geographical maps) to better represent the data and highlight trends.
- User-friendly interface: Improve the overall user experience with a more intuitive and user-friendly interface, potentially using modern UI/UX design principles.
- Real-time data updates: Enable real-time data streaming and visualization for immediate feedback and monitoring.

3. Expanded Functionality:

- Predictive modeling: Incorporate predictive modeling techniques to forecast future data points and potential anomalies, allowing for proactive interventions.
- Automated data cleaning: Develop more sophisticated automated data cleaning algorithms to reduce the need for manual intervention.
- Multi-platform support: Expand support to different operating systems and devices (web,mobile).
- Support for different data types: Expand support beyond the current data types to include other relevant sensor data or external data sources.

- Alerting and notification system: Implement a robust alerting system to notify users of critical events or anomalies, potentially with customizable severity levels and notification methods (email, SMS, etc.).

4. Scalability and Performance:

- Database optimization: Optimize the database for efficient storage and retrieval of large datasets.
- Parallel processing: Implement parallel processing techniques to improve the speed of data analysis and visualization, especially for large datasets.
- Cloud deployment: Consider cloud deployment for scalability and accessibility.

5. Advanced Reporting and Analytics:

- Customizable reports: Allow users to generate custom reports based on specific parameters and timeframes.
- Data export options: Provide a variety of data export options (CSV, JSON, XML, etc.) for integration with other systems.
- Trend analysis: Add features for long-term trend analysis to identify patterns and potential issues over time.

CHAPTER 10

REFERENCES

10.1 WEBSITES

- **Kaggle (www.kaggle.com):** Kaggle is a platform for data science and machine learning. It provides numerous datasets, kernels (code examples), and discussions that can be valuable for understanding data visualization and anomaly detection techniques.
- **Towards Data Science (towardsdatascience.com):** This is a medium publication that covers a wide range of data science topics, including data visualization, anomaly detection, and data cleaning with practical examples and case studies.
- **Data Science Central (www.datasciencecentral.com):** This site offers articles, webinars, and resources related to data science, including best practices in data quality management and visualization techniques.
- **Tableau Public (public.tableau.com):** Tableau's community platform offers numerous examples and discussions around data visualization techniques that can be implemented in similar tools.
- https://www.ndbc.noaa.govrealtime.html.accessed on 28th Nov 2008.
- https://www.pmel.noaa.govtao proj over/qc.html.accessed on 28th November 2008.

10.2 BOOKS

- Handbook of Automated Data quality control and procedures of the National Data Buoy center by National Data Buoy center, stennis space center, Febuary 2003.
- Handbook of Quality control procedures and methods for surface meteorology Data by Shawn R.smith j.parks camp and David M.legler, August 1996.
- “Data Visualization: A Practical Introduction" by Kieran Healy: This book provides a comprehensive foundation in data visualization using R's ggplot2, which can be adapted to other tools and context.

10.2 RESEARCH PAPERS

- Argo Science Team, 2001. The Global Array of Profiling Floats, in observing Oceans in 21st century. C.Z. Koblinsky \and N.R. Smith, eds., Godae Proj. Off., Bur. Meteorol., Melbourne, Australia, (2001) 248 – 258.
- “Automated Data Quality Control and Error Correction: A Review" by Jacek M. Zurada: This paper reviews methodologies for automated data quality control systems, which can inform the design and functionality of the VQC Tool.
- "Survey of Outlier Detection Methodologies" by Varun Chandola, Arindam Banerjee, and Vipin Kumar: A comprehensive review of techniques available for identifying anomalies in datasets, which is foundational for designing an anomaly detection system.