Software Requirement Specifications

Visualizing Student Engagement Data Using ggplot2 and R Markdown



Submitted by

Abuhuraira Saddique (S22BINFT1M01202)

Submitted to

Dr. Mustafa Hameed

Department of Information Technology

Faculty of Computing

The Islamia University of Bahawalpur

Meeting Details

|  |  |  |  |
| --- | --- | --- | --- |
| Sr No | Details | Date | Supervisor  Signature |
| 1 | Initial project discussion and topic finalization | 15-10-2025 |  |
| 2 | SRS requirements discussion and feedback | 5-11-2025 |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Summary

The project **“Visualizing Student Engagement Data Using Python”** focuses on transforming raw educational data into meaningful insights through data analysis and visualization. Using Python libraries such as **Pandas, NumPy, Matplotlib, and Seaborn**, the system processes student engagement information—such as attendance, participation, assignments, and quiz performance—and converts it into clear, easy-to-understand visual representations.

The system provides a structured workflow where users import datasets, the system cleans and prepares the data, and visualizations are generated to highlight trends, patterns, and areas of concern. Teachers, administrators, and researchers can interact with the data, apply filters, and gain valuable insights to improve teaching strategies and student outcomes. Additionally, the project supports report generation through Jupyter Notebook or Markdown, allowing users to export charts and analysis into PDF or HTML formats.

Overall, this Python-based application is an accessible, flexible, and user-friendly solution that supports data-driven decision-making in educational environments. It enhances understanding of student engagement and contributes to better academic planning and intervention strategies.

Table of Contents

[1. Introduction 6](#_Toc33642)

[1.1. Purpose 6](#_Toc33643)

[1.2. Scope 7](#_Toc33644)

[1.3. Product Perspective 7](#_Toc33645)

[1.4. User Characteristics 8](#_Toc33646)

[1.5. Similar apps and systems/Literature Review 9](#_Toc33647)

[1.6. Proposed Technologies 11](#_Toc33648)

[2. Requirements 13](#_Toc33649)

[2.1. Functional Requirements 13](#_Toc33650)

[2.1.1. Sign Up 13](#_Toc33651)

[2.1.2. Login 13](#_Toc33652)

[2.1.3. Role-Based Dashboard Access 14](#_Toc33653)

[2.1.4. Group Creation 14](#_Toc33654)

[2.1.5. Group Joining 14](#_Toc33655)

[2.1.6. Anonymous Messaging 14](#_Toc33656)

[2.1.7. Message Viewing 15](#_Toc33657)

[2.1.8. Abuse Reporting 15](#_Toc33658)

[2.1.9. Identity Revelation by CR/GR 15](#_Toc33659)

[2.1.10. Logout 15](#_Toc33660)

[2.1.11. Password Encryption 15](#_Toc33661)

[2.1.12. OTP Verification for Email 16](#_Toc33662)

[2.1.13. Forgot Password 16](#_Toc33663)

[2.2. Non-Functional Requirements 16](#_Toc33664)

[2.2.1. Performance Requirements 16](#_Toc33665)

[2.2.2. Availability 16](#_Toc33666)

[2.2.3. Scalability 17](#_Toc33667)

[2.2.4. Security 17](#_Toc33668)

[2.2.5. Data Integrity 17](#_Toc33669)

[2.2.6. Maintainability 17](#_Toc33670)

[2.2.7. Portability 17](#_Toc33671)

[2.2.8. Cost Efficiency 18](#_Toc33672)

[3. Use Cases and Flow of Processes 19](#_Toc33673)

[3.1. Use Case 1 19](#_Toc33674)

[3.2. Use Case 2 20](#_Toc33675)

[3.3. Use Case 3 21](#_Toc33676)

[3.4. Use Case 4 22](#_Toc33677)

[3.5. Use Case 5 22](#_Toc33678)

[3.6. Use Case 6 23](#_Toc33679)

[3.7. Use Case 7 24](#_Toc33680)

[3.8. Use Case 8 24](#_Toc33681)

[3.9. Use Case 9 25](#_Toc33682)

[3.10. Use Case 10 26](#_Toc33683)

[4. References 37](#_Toc33684)

# Introduction

Student engagement plays a vital role in understanding learning behavior, academic performance, and classroom participation. Educational institutions increasingly rely on data analytics to track and improve these engagement levels. The purpose of this project, “Visualizing Student Engagement Data Using Python,” is to analyze and present student engagement patterns through clear, meaningful, and interactive visualizations.

Instead of using ggplot2 and R Markdown, this project utilizes Python, along with powerful visualization and reporting libraries such as Matplotlib, Seaborn, Pandas, and Jupyter Notebook/Markdown. The goal is to transform raw student engagement data into easy-to-understand charts, graphs, and summaries that help educators identify trends, spot low-engagement areas, and support data-driven decision-making.

This introduction highlights the overall purpose of the system, sets the foundation of the SRS, and explains how Python-based visualization tools will support effective student engagement analysis.

## Purpose

The purpose of this Software Requirements Specification (SRS) document is to clearly define and describe all functional and non-functional requirements for the project “Visualizing Student Engagement Data Using ggplot2 and R Markdown.”

This document provides a complete overview of how the system will collect, process, analyze, and visualize student engagement data to help educators gain meaningful insights.

It will serve as a guideline for developers, stakeholders, and evaluators throughout the system development life cycle.

## Scope

The scope of the project “Visualizing Student Engagement Data Using Python” includes the development of a data visualization system that helps educators analyze and understand student engagement patterns. The system will use Python-based tools and libraries to process data and generate meaningful visual insights.

The major activities covered in this project include:

• Collecting and importing student engagement datasets (attendance, participation, quiz scores, assignment submissions, etc.).

• .Cleaning, preprocessing, and organizing the data using Python (Pandas, NumPy).

•. Creating clear and informative visualizations using Matplotlib and Seaborn.

• .Generating summary reports using Jupyter Notebook, Markdown, or Python-based reporting tools.

•. Highlighting trends, patterns, and areas of low engagement to support decision-making.

• .Providing educators with an easy-to-understand dashboard or report without requiring technical expertise.

The project does not include:

• Real-time data collection or live dashboards

• Machine learning prediction or AI-based recommendation

• Integration with external LMS platforms

• Mobile or web application development (unless added later)

This scope defines the boundaries of the project and ensures that the focus remains on data visualization and analysis using Python.

## Product Perspective

The project “Visualizing Student Engagement Data Using Python” is an independent analytical tool designed to help educational stakeholders better understand student engagement patterns. This system functions as a standalone data visualization and reporting application built using Python and its data analysis libraries.

The product will operate by importing student engagement datasets from external sources such as CSV, Excel, or databases. Using Python libraries like Pandas, NumPy, Matplotlib, and Seaborn, the system processes, analyzes, and visualizes the data in the form of charts, graphs, and summary reports.

This tool does not replace existing learning management systems (LMS) or student information systems (SIS). Instead, it complements them by offering an easy-to-understand visual layer that can be used by teachers, administrators, and academic coordinators to make informed decisions based on trends and insights.

The product will be developed in a modular manner, where data preprocessing, visualization, and reporting components work together but can also function independently. Users will interact with the tool through Jupyter Notebook, Python scripts, or a simple reporting interface, depending on the final design.

## User Characteristics

The users of the “Visualizing Student Engagement Data Using Python” system come from educational and administrative backgrounds. Most users will not have advanced technical or programming knowledge. Therefore, the system must present data and visualizations in an easy-to-understand format. The key user groups are:

1. **Teachers / Lecturers**

• Basic computer skills

• Limited knowledge of programming or data analysis

• Need simple, clear visualizations to understand student participation and performance

2. Academic Coordinators / Department Heads

• Intermediate understanding of educational data

• Able to interpret charts and graphs

• Interested in class-level and department-level engagement patterns

# Institution Administrators

• High-level decision-makers

• Require summarized reports rather than detailed technical analysis

• Prefer clean dashboards and reports created via Python tools

4. Education Researchers / Analysts

• Familiar with data analysis concepts

• Will use Python-based visualizations to study engagement trends

## Similar apps and systems/Literature Review

Several tools and systems exist for analyzing and visualizing student engagement data. Most of these applications aim to help educators understand learning patterns and make data-driven decisions. However, many require either advanced technical skills or rely on proprietary software. This project uses Python to provide a flexible, open-source, and user-friendly alternative.

1. Existing Systems

1. Learning Management Systems (LMS) Analytics Modules

• Examples: Moodle Analytics, Blackboard Analytics

• Features: Track attendance, quiz scores, participation, and engagement metrics

• Limitations: Limited customization of visualizations; some require technical expertise or subscriptions

1. Commercial Educational Analytics Tools

• Examples: Tableau, Power BI for Education

• Features: Powerful visualization dashboards, interactive charts, and reporting

• Limitations: High cost; may require advanced training; less flexibility in handling raw datasets

1. Python-based Academic Visualization Projects

• Features: Use Python libraries such as Pandas, Matplotlib, Seaborn, Plotly to create custom visualizations

• Benefits: Fully customizable, open-source, and suitable for producing detailed insights

• Limitations: Often lack user-friendly interfaces for non-technical educators

## Proposed Technologies

1.6 Proposed Technologies

The “Visualizing Student Engagement Data Using Python” project will utilize modern, open-source technologies that are widely used for data analysis, visualization, and reporting. The selected technologies are chosen to ensure flexibility, ease of use, and reproducibility of results.

1. Programming Language

Python: The primary programming language for data processing, visualization, and reporting. Python is widely used in data analytics due to its simplicity, readability, and extensive library support.

2. Data Handling and Processing

• Pandas: For importing, cleaning, and manipulating student engagement datasets.

• NumPy: For numerical computations and efficient data operations.

3. Data Visualization

• Matplotlib: To create static, publication-quality charts and graphs.

• Seaborn: To generate advanced, statistical visualizations with a clean design.

Plotly (optional): For interactive and web-based visualizations, allowing users to explore data dynamically.

4. Reporting and Documentation

• Jupyter Notebook: To create interactive reports combining code, visualizations, and explanatory text.

• Markdown: To structure textual descriptions and comments alongside visualizations in a readable format.

### **5. Frontend Presentation Technologies**

If creating a simple interface or dashboard:

* **HTML** – Used to structure the webpage or dashboard layout.
* **CSS** – Used to style the interface and adjust visual elements.
* **Tailwind CSS** – A utility-first CSS framework for building modern, responsive, and clean UI designs quickly.

6. Data Sources

CSV, Excel, or Database files: Standard formats for student engagement data, enabling easy import and preprocessing in Python.

7. Optional Technologies

Streamlit or Dash: For building lightweight dashboards or web interfaces to present visualizations to non-technical users.

# Requirements

The “Visualizing Student Engagement Data Using Python” system requires both functional and non-functional specifications to ensure smooth operation, usability, and performance. Functional requirements describe the features and behavior of the system, while non-functional requirements define system qualities, performance standards, and constraints.

2.1 Functional Requirements

Functional requirements specify the operations the system must perform to meet the objectives. For this project:

1. Data Import and Processing

• The system must be able to import student engagement data from CSV, Excel, or database sources.

• It must handle missing values, incorrect formats, and clean the data for analysis using Python libraries like Pandas.

1. Data Visualization

• The system must generate charts and graphs to show trends in student engagement.

• Visualizations will include bar charts, line charts, heatmaps, and scatter plots using Matplotlib and Seaborn.

1. Report Generation

• The system must produce dynamic, readable reports combining visualizations and text explanations using Jupyter Notebook/Markdown.

• Reports should be exportable to PDF or HTML formats for sharing with educators.

1. User Interaction

• Users should be able to select datasets, filter data by date, class, or student group, and view updated visualizations.

• Optional: Interactive dashboards using Plotly or Streamlit.

1. Insight Extraction

• The system should highlight low engagement patterns and trends.

• Provide summary statistics (e.g., average participation, attendance percentage) automatically.

2.2 Non-Functional Requirements

Non-functional requirements describe the quality attributes and constraints of the system:

1. Performance

• The system should process datasets up to 10,000 records without noticeable delay.

• Visualizations should render in under 5 seconds for medium-sized datasets.

1. Usability

• The interface must be simple and intuitive for users without technical knowledge.

• Reports and visualizations must be easily interpretable by teachers and administrators.

1. Reliability

• The system should handle errors gracefully (e.g., missing or corrupted data).

• Must ensure data integrity during processing and reporting.

1. Portability

• The system should run on major operating systems: Windows, macOS, and Linux.

• Compatible with standard Python environments and IDEs like Jupyter Notebook, VS Code, or PyCharm.

1. Maintainability

• The Python code should follow modular design principles to allow easy updates and addition of new features.

• Documentation and code comments must be provided for maintainers.

# Use Cases and Flow of Processes

## 

Figure 1: System Level Use Case Diagram

# **Use Case for Student Engagement Visualization System**

### **Use Case 1: Import Student Engagement Data**

**Actor:** Teacher / Researcher  
**Description:** User uploads a CSV or Excel file containing engagement data.  
**Precondition:** The dataset must be available on the user’s device.  
**Main Flow:**

1. User selects the dataset.
2. System reads the file using Pandas.
3. System validates file format and checks for missing values.
4. System loads and displays a preview of the data.  
   **Postcondition:** Data is successfully imported into the system.

## **1. User Login**

### **Flow**

1. User opens the web application (HTML + Tailwind UI).
2. Login page appears with fields: **Email / Username** and **Password**.
3. User enters credentials.
4. Backend (Python + Django/Flask) verifies credentials from the database.
5. If correct → user is redirected to **Dashboard**.
6. If incorrect → error message appears (“Invalid Credentials”).

### **Use Case 2: Clean & Preprocess Data**

**Actor:** System (Automated)  
**Description:** The system cleans and prepares the dataset.  
**Main Flow:**

1. Detect missing values and incorrect data types.
2. Apply cleaning operations using Pandas.
3. Standardize columns (names, formats).
4. Store cleaned data for visualization.  
   **Postcondition:** A clean dataset is ready for analysis.

## **2. Upload Student Engagement Dataset**

### **Flow**

1. User navigates to **Upload Data Page** from the dashboard.
2. User selects a file (CSV/Excel).
3. Frontend sends the file to the backend through POST request.
4. Backend validates:
   * File format
   * Missing values
   * Column names
5. If valid → data stored temporarily or in database.
6. System shows **“Upload Successful”** message.

### **Use Case 3: Generate Visualizations**

**Actor:** Teacher / Administrator  
**Description:** User views visual graphs of engagement.  
**Main Flow:**

1. User selects type of visualization (bar chart, line chart, heatmap, etc.).
2. System processes data and creates plots using Matplotlib/Seaborn.
3. System displays graphs in Jupyter Notebook.  
   **Postcondition:** Visual graphs are presented to the user.

## **3. Data Processing & Cleaning**

### **Flow**

1. After upload, backend (Python) starts preprocessing:
   * Removing missing/invalid rows
   * Converting data types
   * Normalizing values
2. System logs and shows the summary of cleaned data (e.g., total rows, cleaned rows).
3. Cleaned dataset is saved for visualization.

### **Use Case 4: Filter & Analyze Data**

**Actor:** User  
**Description:** User applies filters to explore specific engagement patterns.  
**Main Flow:**

1. User selects filters (date range, class, section, activity).
2. System updates dataset based on filters.
3. System regenerates visualizations.  
   **Postcondition:** User sees filtered and updated insights.

## **4. Generate Visualizations**

### **Flow**

1. User selects a visualization type from dashboard:
   * Bar chart
   * Line chart
   * Pie chart
   * Heatmap
2. System fetches cleaned dataset.
3. Python library (Matplotlib / Seaborn / Plotly) generates the graph.
4. The graph is displayed on the website using HTML + Tailwind.
5. User can download or export the graph if needed.
   1. Activity Diagram 1

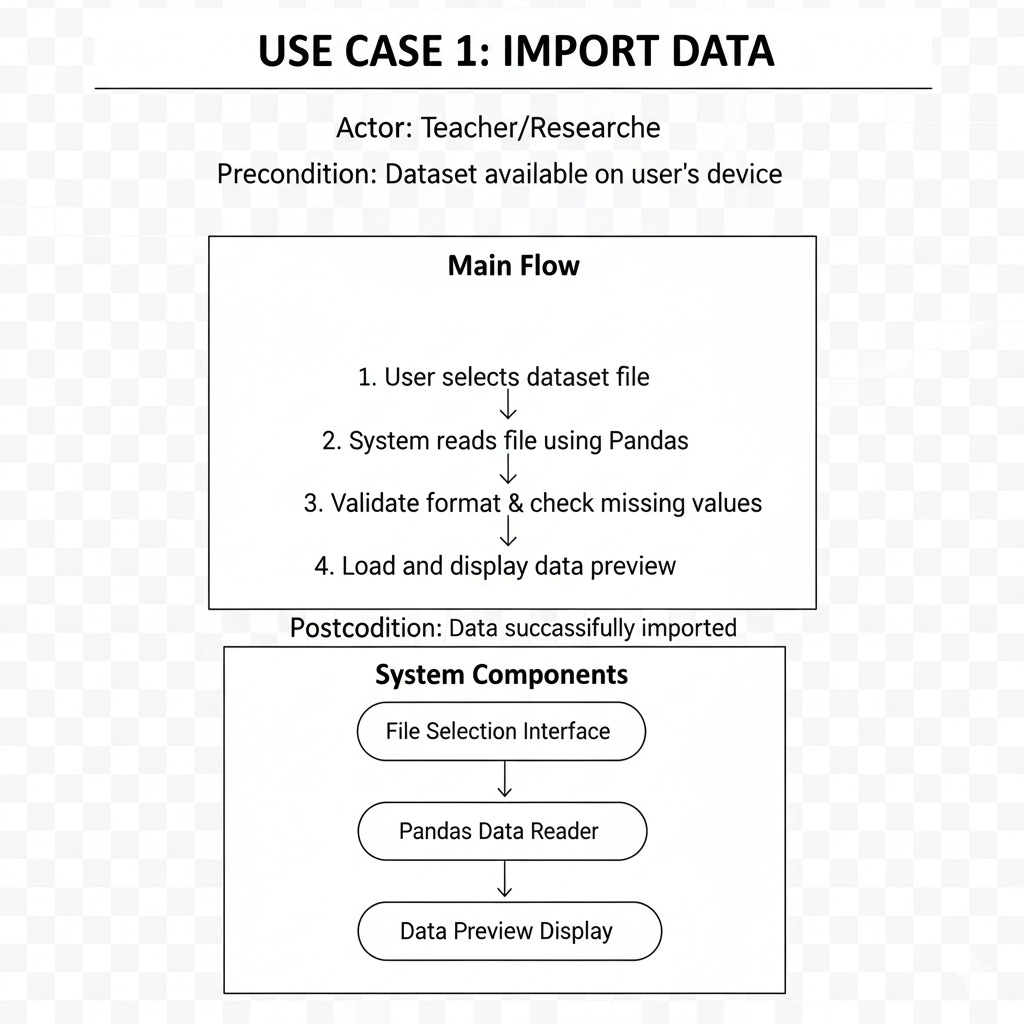
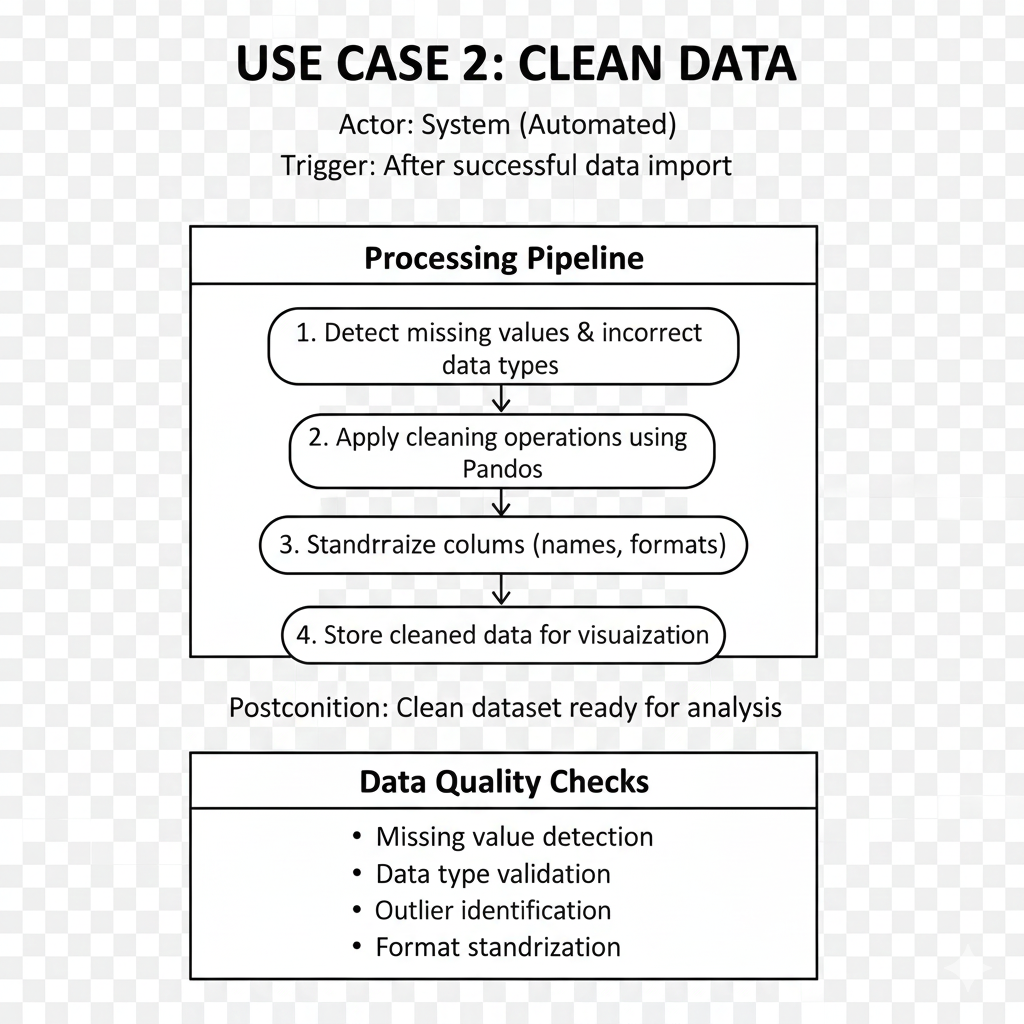
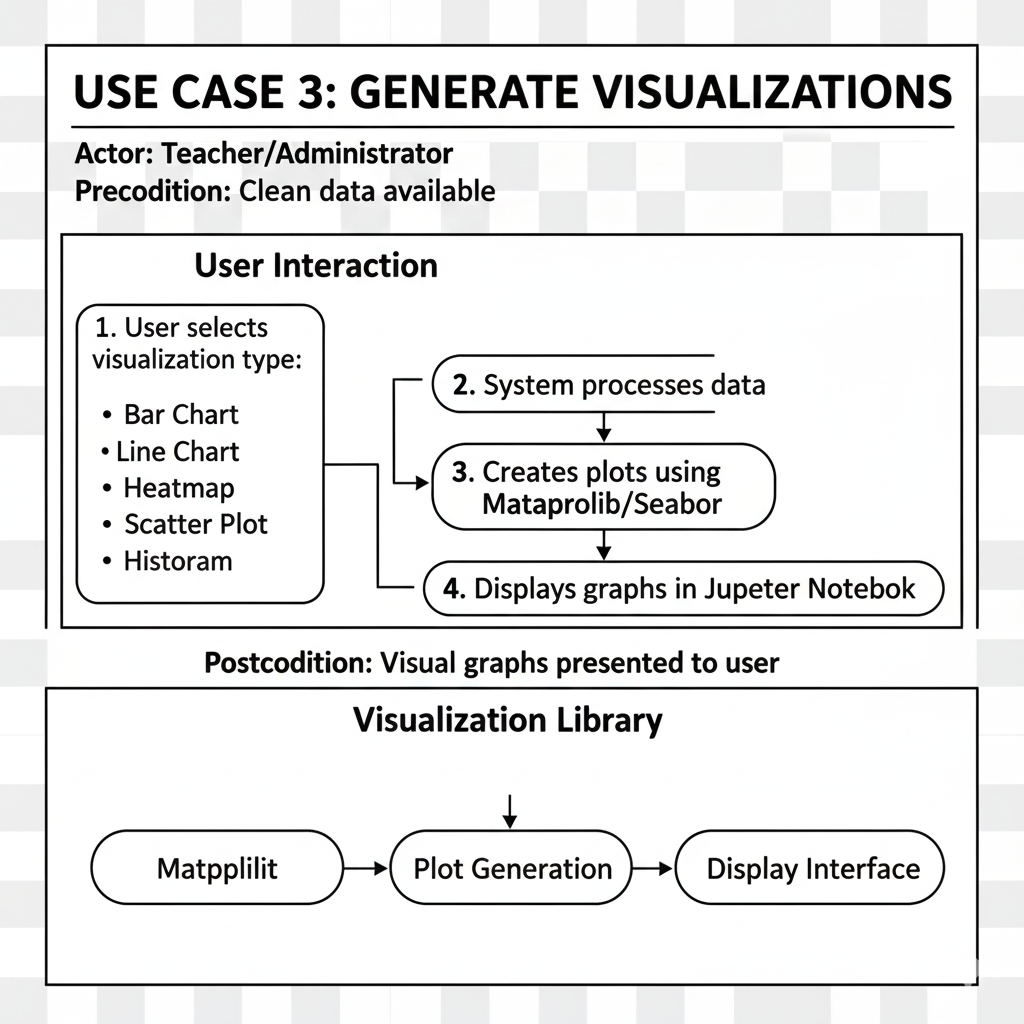


Figure1.Import Data

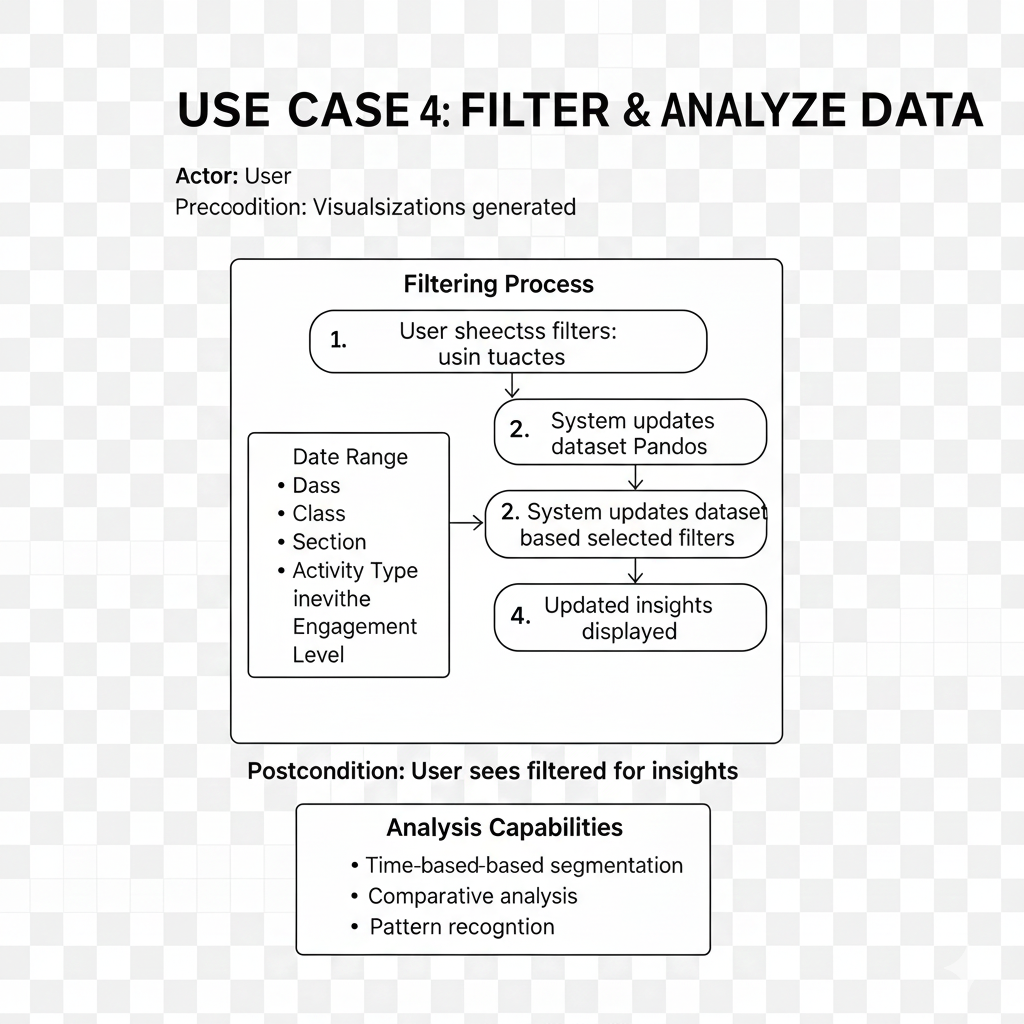
2. Activity Diagram 2

Figure2. Clean data

1. Activity Diagram 3

Fiqure3. Generate Visualizations

1. Activity Diagram 3

Fiqure4. Filter and Analyze Data