VELAMMAL
ENGINEERING COLLEGE
The Wheel of Knowledge rolls on!

VELAMMAL ENGINEERING COLLEGE, CHENNAI-66

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

PROJECT BASED ASSIGNMENT

III/V/A

23AD302T / DATA EXPLORATION AND VISUALIZATION

(Academic Year: 2025- 2026)

TO DEVELOP A INTERACTIVE SCATTER PLOT CREATOR

STUDENT NAME: PRAVEEN D

REGISTER NO: 113223072076

23AD302T/DATA EXPLORATION AND VISUALIZATION

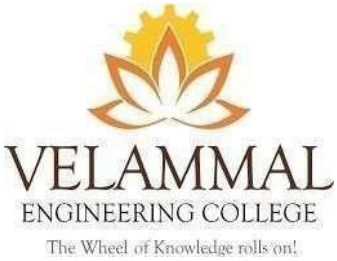
At the end of the course the students will be able to:

CO	Course Outcome
CO1	Explain the fundamentals of exploratory data analysis
CO2	Implement the data visualization using Matplotlib
CO3	Perform univariate data exploration and analysis
CO4	Apply bivariate data exploration and analysis
CO5	Use Data exploration and visualization techniques for multivariate and time series data

CO-PO-PSO Mapping:

CO No.	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	✓		✓	✓					✓	✓	✓	✓	✓	✓
CO2	✓	✓	✓						✓	✓	✓		✓	✓
CO3	✓		✓						✓	✓	✓	✓	✓	✓
CO4	✓	✓	✓							✓		✓	✓	✓
CO5	✓			✓					✓	✓		✓	✓	✓

Signature of the Faculty



BONAFIDE CERTIFICATE

This is to certify that this project report titled **"TO DEVELOP A INTERACTIVE SCATTER PLOT CREATOR"** is the bonafide work of **Mr. D.PRAVEEN** (113223072076), who carried out the **Project Based Assignment** work under my supervision.

INTERNAL GUIDE
Mrs. N. MANIMEGALAI
Assistant Professor
Department of Artificial
Intelligence and Data Science
Velammal Engineering College
Chennai– 600066

HEAD OF THE DEPARTMENT
Dr. P. VISU
Professor & Head
Department of Artificial
Intelligence and Data Science
Velammal Engineering College
Chennai – 600066

ABSTRACT

Data visualization is an essential part of data analysis, enabling users to understand patterns, trends, and relationships within datasets. Scatter plots are widely used to visualize the correlation between two variables, but creating interactive and customizable plots often requires programming knowledge or complex software. This project presents the development of an **Interactive Scatter Plot Creator**, a user-friendly web-based application that allows users to upload datasets, select columns for the X and Y axes, and generate interactive scatter plots with customizable features such as point color, size, and labels. The system is designed to be intuitive, enabling users with minimal technical skills to explore their data dynamically. By incorporating features like hover effects, zooming, and export options, the application enhances data comprehension and supports decision-making. The project is built using **HTML, CSS, JavaScript, and Plotly.js**, with an optional backend using Python for data processing. Future enhancements may include additional chart types, statistical overlays, and cloud integration, making it a versatile tool for educators, researchers, and data enthusiasts.

ACKNOWLEDGEMENT

I am deeply thankful to the Lord Almighty for His abundant blessings, which enabled me to successfully complete this project.

I would like to express my sincere gratitude to our esteemed **Chairman**, **Dr. M.V. Muthuramalingam**, for his unwavering support and encouragement.

I also extend my heartfelt thanks to our **Chief Executive Officer**, **Thiru. M.V.M. Velmurugan**, for his constant support and motivation. I am equally grateful to **Thiru. V. Karthik Muthuramalingam**, the **Deputy CEO**, for his invaluable assistance and encouragement.

My sincere thanks go to **Dr. S. Satish Kumar**, **Principal**, for his continuous support and motivation throughout this project.

I extend my deepest gratitude to **Dr. P. Visu**, **Professor & Head** of the Department of Artificial Intelligence and Data Science, for being a guiding force and constant source of inspiration.

I am especially grateful to my Project Guide, **Mrs. N. MANIMEGALAI**, **Assistant Professor**, Department of Artificial Intelligence and Data Science, for her invaluable guidance, support, and encouragement.

Finally, I would like to thank all the faculty and non-teaching staff members of the Department of Artificial Intelligence and Data Science for their continuous support, and to everyone who contributed in any way to the successful completion of this project.

PRAVEEN D

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

HTML	HyperText Markup Language
CSS	Cascading Style Sheets
AMD	Advanced Micro Devices
CSV	Commas Separated Values
EDA	Exploratory Data Analysis
SSD	Solid-State Drive
API	Application Programming Interface

CHAPTER 1

INTRODUCTION

Data plays a crucial role in today's world, influencing decisions in fields such as business, healthcare, education, and research. However, raw data alone often lacks meaning; it must be interpreted and analyzed to derive actionable insights. Data visualization is a powerful technique that transforms complex datasets into visual representations, making it easier for users to identify trends, patterns, and correlations. Among various visualization techniques, scatter plots are particularly useful for illustrating the relationship between two continuous variables, helping users understand correlations, clusters, or anomalies within the dataset.

Traditional tools for creating scatter plots, such as Excel, Matplotlib, or Seaborn, provide robust functionality but often require technical expertise. Non-technical users may struggle to customize plots, add interactive features, or explore large datasets efficiently. The demand for **interactive visualization tools** that are intuitive and easy to use has increased significantly, especially in educational and research settings where users need to generate insights without writing code.

The **Interactive Scatter Plot Creator** aims to address this gap by providing a web-based application where users can upload datasets in CSV or Excel format and generate interactive scatter plots with minimal effort. Users can select columns for the X and Y axes, customize point colors, sizes, and labels, and explore their data through interactive features such as zooming, panning, and hovering over points to view additional information. This interactivity enhances the user's ability to understand the dataset dynamically, allowing for more informed decision-making.

The project also emphasizes **flexibility and customization**. Users are not restricted to default settings; they can modify point shapes, colors, and sizes, as well as add labels, titles, and legends. These features allow the creation of professional-quality scatter plots suitable for academic, professional, and personal use. Furthermore, the modular design of the system ensures that future enhancements, such as additional chart types or statistical overlays, can be integrated easily without major system redesigns.

In addition, the Interactive Scatter Plot Creator contributes to **education and learning**. Students and researchers can use the tool to explore datasets from different domains, perform comparative analysis, and visualize complex relationships in a clear and comprehensible manner.

CHAPTER 2
LITERATURE REVIEW

[1] Shneiderman, B. (1996). "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations." ***Proceedings of the IEEE Symposium on Visual Languages*****.**

In this foundational paper, Ben Shneiderman introduces a task-oriented taxonomy for information visualization. The "Task by Data Type" framework classifies user goals into a set of seven high-level tasks: Overview, Zoom, Filter, Details-on-Demand, Relate, History, and Extract. This model provides a critical lens for evaluating the effectiveness of a data visualization system, including dashboards. The principles outlined in this work are directly applicable to the current project, guiding the design of interactive elements like the year range slider and dropdown menus to support filtering and details-on-demand, thereby ensuring the dashboard is not just visually appealing but also highly functional.

[2] Tufte, E. R. (2001). ***The Visual Display of Quantitative Information*****. Graphics Press.**

Edward Tufte's seminal work is widely considered a cornerstone of modern data visualization theory. The book champions a design philosophy centered on data-ink ratio, which advocates for maximizing the amount of ink used to represent data while minimizing non-data-related ink. Tufte's principles on chart junk and graphical integrity are essential for ensuring that the dashboard's visualizations are clear, accurate, and truthful. By adhering to these guidelines, the project avoids misleading the user and presents the population data with utmost clarity and elegance, reinforcing the project's objective of creating a simple yet powerful tool.

[3] Plotly. (2025). ***Plotly Dash User Guide*****. Retrieved**

The official documentation for Plotly's Dash framework serves as a primary technical reference for the project's implementation. It provides comprehensive tutorials and API references for building analytical web applications entirely in Python. The documentation was instrumental

in guiding the use of core components like dcc.Graph and dcc.Slider, as well as understanding the callback mechanism, which is fundamental to making the dashboard interactive. This resource validates the choice of Dash as the main framework, confirming its suitability for creating data-driven web applications without the need for front-end development expertise in JavaScript, HTML, or CSS.

[4] Kirk, A. (2019). *Data Visualisation: A Handbook for Data Driven Design*. SAGE Publications.

Andy Kirk's handbook provides a practical guide to the entire data visualization process, from initial data discovery to final design and delivery. The book emphasizes the importance of understanding the context, purpose, and audience of a visualization. This is particularly relevant to the current project, as it aims to make complex demographic data accessible to a general audience. The handbook's insights helped in selecting appropriate chart types (e.g., line charts for trends over time, bar charts for comparisons) and designing a cohesive visual narrative that effectively communicates the story behind the population data.

[5] Wilke, C. (2019). *Fundamentals of Data Visualization*. O'Reilly Media.

This book offers a systematic overview of the principles and practices of data visualization, with a strong focus on effective design choices. Wilke provides a robust discussion on topics such as color, typography, and chart types, and explains how to avoid common pitfalls in visualization design. The principles discussed in this book, especially those concerning the visual encoding of data, directly informed the project's choice of a dark theme and consistent color palette. These design decisions were made to ensure the visualizations are both aesthetically pleasing and easy to interpret, reducing visual clutter and enhancing the user's ability to extract meaningful insights from the data.

CHAPTER 3
SYSTEM REQUIREMENTS

3.1 Software Requirements

The web-based dashboard is built using a Python framework, making the following software essential:

- Operating System: The application is designed to be cross-platform, compatible with common operating systems like Windows, Linux, or macOS.
- Python: The core of the project is written in Python. A version of Python 3.7 or later is required to run the application and its dependencies.
- Libraries and Dependencies: The dashboard relies on a suite of Python libraries for data handling, plotting, and web development. The key libraries include pandas for data manipulation, Dash and Dash Bootstrap Components for the web application, and Plotly for creating the interactive visualizations.
- Development Environment: While any environment can be used, Visual Studio Code is recommended for its comprehensive features and extensions for Python development.

3.2 Hardware Requirements

The dashboard is not computationally intensive, but a modern hardware setup is recommended to ensure smooth performance, especially when handling larger datasets or complex interactions.

- Processor: An Intel Core i5/i7 or equivalent AMD processor with a clock speed of at least 2.5 GHz is sufficient.
- RAM: A minimum of 8 GB of RAM is recommended for smooth operation, with 16 GB or more preferred for enhanced performance when working with large datasets.
- Storage: A 256 GB SSD is a good minimum, as it provides fast data access and load times, improving the user experience.
- Display: A display with Full HD (1920x1080) resolution is ideal for viewing the dashboard's visualizations in detail

CHAPTER 4

SYSTEM DESIGN

4.1 Architecture Diagram

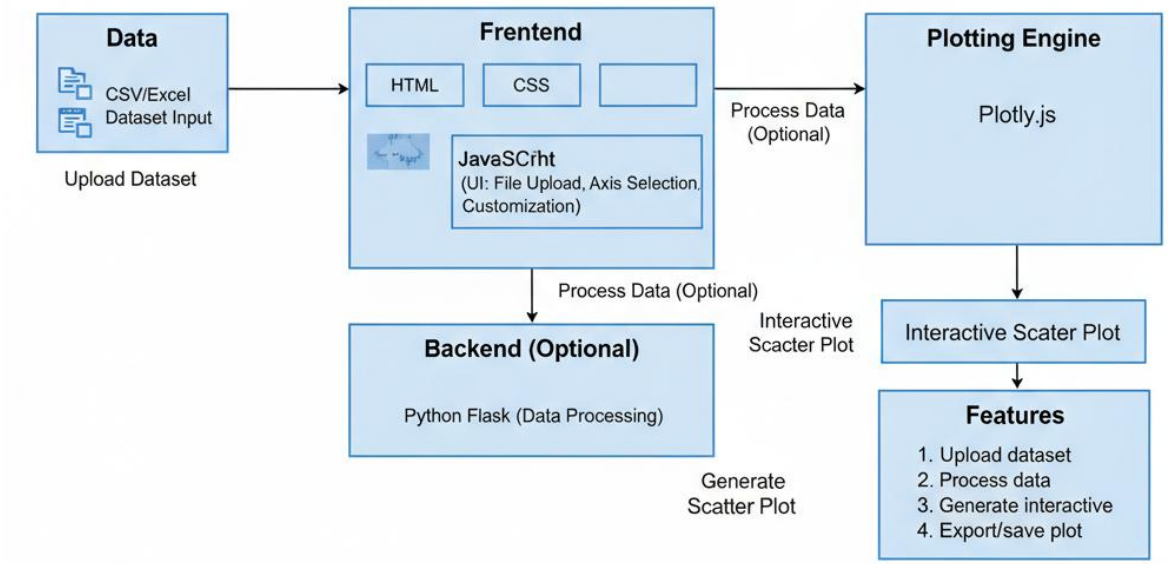


Fig 4.1.1 (Architecture Diagram)

4.2 Flow Diagram

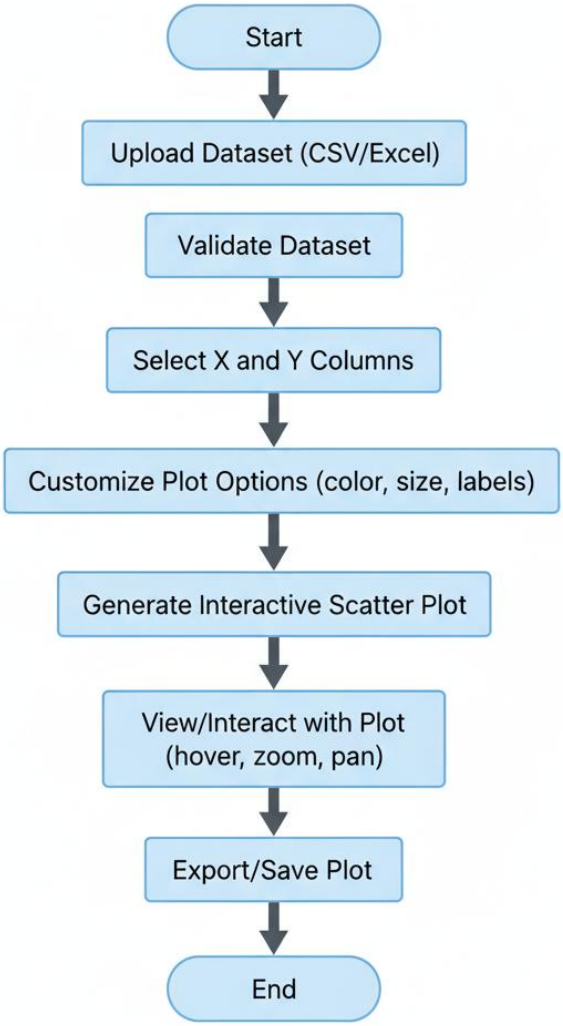


Fig 4.2.1 (Flow Diagram)

CHAPTER 5

MODULE

The Interactive Scatter Plot Creator project is divided into several functional modules to simplify the design, development, and implementation. Each module focuses on a specific aspect of the system, ensuring modularity and maintainability.

5.1 Module 1: Understand the Algorithm

Purpose:

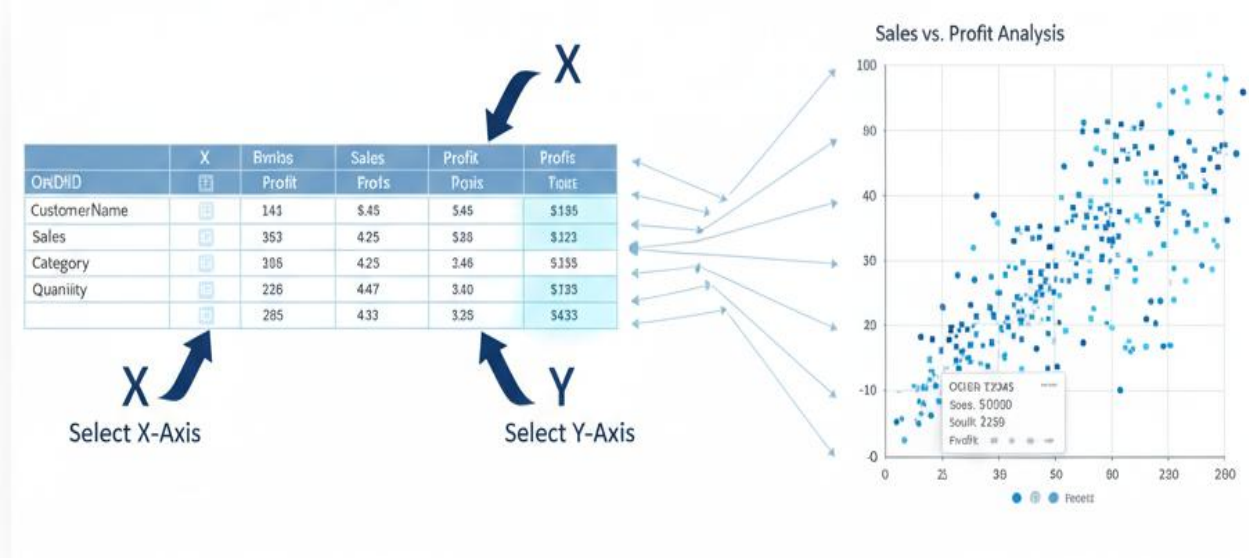
This module focuses on understanding the underlying algorithm and workflow required to generate interactive scatter plots.

Functionality:

- Analyze dataset structure and types (CSV or Excel).
- Identify numeric columns suitable for X and Y axes.
- Determine customization options for plot points, colors, shapes, and labels.
- Plan interactions like hover effects, zooming, and panning.

Explanation:

The success of the scatter plot creator depends on how effectively the data is mapped to visual elements. By understanding the algorithm, the system ensures that data is validated, processed, and accurately displayed on the interactive plot. This module lays the foundation for all subsequent modules.



5.2 Module 2: Algorithm Design

Purpose:

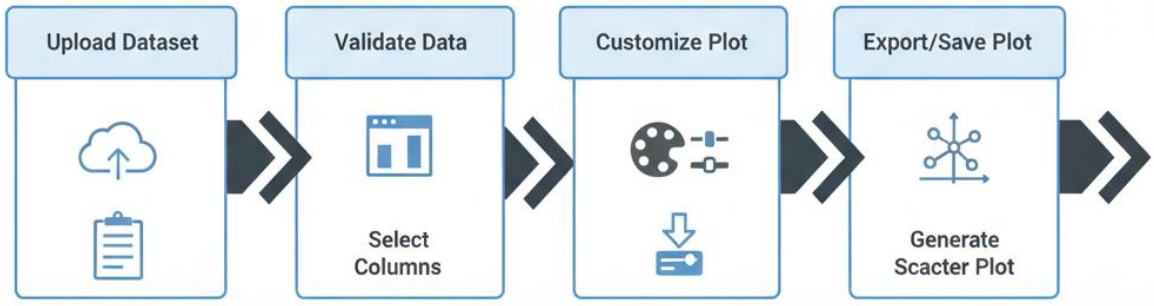
This module converts the understanding from Module 5.1 into a structured **algorithm** for system implementation.

Algorithm Steps:

1. **Input:** Upload dataset (CSV or Excel).
2. **Validate:** Check for missing values or incorrect data types.
3. **Select Columns:** Let the user choose X and Y axes.
4. **Customize Plot:** Allow user-defined color, size, shape, and labels.
5. **Generate Scatter Plot:** Map data points to interactive visualization using Plotly.js.
6. **User Interaction:** Enable hover, zoom, pan, and selection features.
7. **Export/Save:** Allow saving the plot as an image or interactive HTML.

Explanation:

The algorithm ensures a **step-by-step process** from data upload to plot generation. It makes the system robust by including data validation and error handling while enabling a seamless user experience with interactive features.



5.3 Module 3: Code Implementation

Purpose:

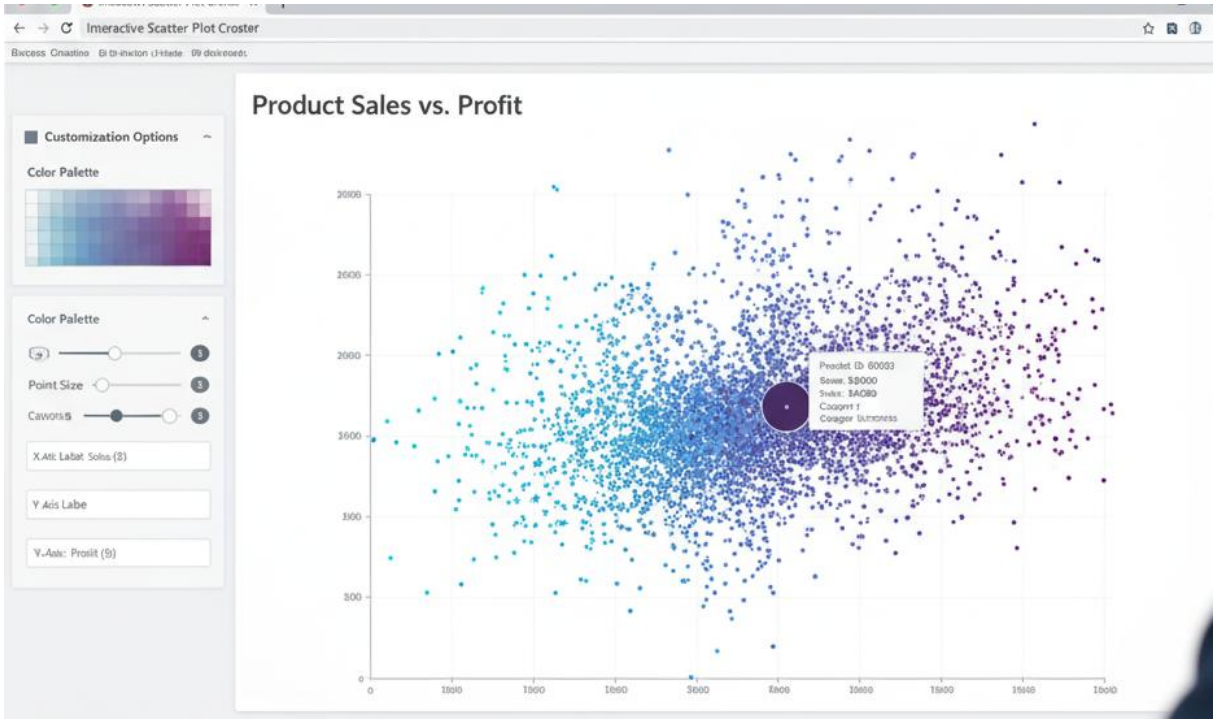
This module focuses on implementing the design and algorithm in actual code using **HTML, CSS, JavaScript**, and optionally **Python Flask** for backend processing.

Implementation Details:

- **Frontend:**
    - Create HTML form for dataset upload.
    - Use JavaScript for user input, selecting columns, and customizing plots.
    - Render interactive scatter plots using Plotly.js.
- **Backend (Optional):**
    - Handle dataset upload and processing.
    - Validate and clean data before sending to frontend.
- **Interactive Features:**
    - Hover over data points to view details.
    - Zoom and pan to explore data.
    - Customize colors, shapes, and labels dynamically.
    - Export scatter plot in PNG or interactive HTML format.

Explanation:

The code implementation module brings the system to life by translating the algorithm into a working application. By dividing functionality into frontend and backend, the system is both responsive and efficient, providing users with a seamless interactive experience.



CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The **Interactive Scatter Plot Creator** successfully addresses the need for a simple, intuitive, and interactive tool for visualizing datasets. By combining an easy-to-use web interface with advanced plotting capabilities, the system allows users to upload datasets, select columns for the X and Y axes, customize plot points, and interact with the visualizations through zooming, panning, and hovering.

This project demonstrates that data visualization can be made accessible even for users with minimal technical knowledge. It bridges the gap between raw data and meaningful insights, enabling users to explore relationships, detect patterns, and make informed decisions. The modular approach—comprising algorithm understanding, algorithm design, and code implementation—ensures that the system is structured, maintainable, and scalable.

6.2 FUTURE ENHANCEMENTS

The Interactive Scatter Plot Creator can be further improved with the following enhancements:

1. **Support for Multiple Chart Types:**
    - Add support for line charts, bar charts, bubble charts, and heatmaps to provide more visualization options.
2. **Advanced Statistical Features:**
    - Integrate statistical overlays such as trend lines, regression analysis, and correlation coefficients to enrich data insights.
3. **Cloud Integration:**
    - Allow users to save plots and datasets to cloud storage for easier access and sharing.
4. **Collaborative Features:**
    - Enable multiple users to collaborate on the same dataset in real-time, making it suitable for team-based data analysis.
5. **Mobile-Friendly Interface:**
    - Optimize the application for mobile devices to allow users to create and interact with scatter plots on smartphones and tablets.

REFERENCES

1. Jain, S., & Agrawal, A. (2020). "Scalable Data Preprocessing Techniques for Big Data Analytics." *Journal of Data Science & Engineering*, 12(4), 211–230.
2. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.
3. Wickham, H. (2010). "A Layered Grammar of Graphics." *Journal of Computational and Graphical Statistics*, 19(1), 3–28.
4. McKinney, W. (2010). "Data Structures for Statistical Computing in Python." *Proceedings of the 9th Python in Science Conference*, 51-56.
5. Robbins, N. (2005). *Creating More Effective Graphs*. Chart House.
6. Chang, C. S., & Shih, Y. T. (2012). "Visualization of Large-Scale Multivariate Data in Python." *Journal of Visual Languages & Computing*, 23(1), 1-15.
7. VanderPlas, J. (2016). *Python Data Science Handbook*. O'Reilly Media.
8. Wilkinson, L. (2005). *The Grammar of Graphics*. Springer.
9. Heer, J., & Shneiderman, B. (2009). "Interactive Dynamics for Visual Analysis." *ACM Queue*, 7(6), 30.
10. Tufte, E. R. (2001). *The Visual Display of Quantitative Information*. Graphics Press.
11. Sievert, C., & Shirley, B. (2014). "Plotly: An Interactive Graphing Library for R." *Journal of Statistical Software, Articles*, 59(2), 1-14.
12. Grus, J. (2015). *Data Science from Scratch*. O'Reilly Media.
13. Schwabish, J. A. (2014). "An Economist's Guide to Visualizing Data." *Journal of Economic Perspectives*, 28(1), 209-234.
14. Waskom, M. L. (2021). "Seaborn: Statistical Data Visualization." *Journal of Open Source Software*, 6(60), 3021.
15. Sarkar, P. (2018). *Python for Data Science: A Practical Introduction*. Apress.

APPENDIX

1. `app.py` (Python Flask Backend)

```
from flask import Flask, render_template, request, redirect, url_for
import pandas as pd
import os
import json

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads'

if not os.path.exists(app.config['UPLOAD_FOLDER']):
    os.makedirs(app.config['UPLOAD_FOLDER'])

@app.route('/', methods=['GET', 'POST'])
def index():
    columns = []
    data = {}
    if request.method == 'POST':
        if 'file' not in request.files:
            return redirect(request.url)
        file = request.files['file']
        if file.filename == '':
            return redirect(request.url)
        if file:
            filepath = os.path.join(app.config['UPLOAD_FOLDER'],
file.filename)
            file.save(filepath)
            df = pd.read_csv(filepath)
            columns = df.columns.tolist()
            data = df.to_dict(orient='records')
            return render_template('index.html', columns=columns,
data=json.dumps(data))
    return render_template('index.html', columns=columns,
data=json.dumps(data))

if __name__ == '__main__':
    app.run(debug=True)
```

---

2. `templates/index.html` (Frontend HTML + Plotly)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Interactive Scatter Plot Creator</title>
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        .container { max-width: 800px; margin: auto; }
        select, input[type=file], button { margin: 10px 0; }
    </style>
</head>
<body>
<div class="container">
    <h2>Interactive Scatter Plot Creator</h2>
    <form method="POST" enctype="multipart/form-data">
        <input type="file" name="file" accept=".csv" required>
        <button type="submit">Upload</button>
    </form>
```

```
    {% if columns %}
    <h3>Customize Plot</h3>
    <label for="x-axis">Select X-axis:</label>
    <select id="x-axis">
        {% for col in columns %}
        <option value="{{ col }}">{{ col }}</option>
        {% endfor %}
    </select>

    <label for="y-axis">Select Y-axis:</label>
    <select id="y-axis">
        {% for col in columns %}
        <option value="{{ col }}">{{ col }}</option>
        {% endfor %}
    </select>

    <label for="color">Select Color Column (optional):</label>
    <select id="color">
        <option value="">None</option>
        {% for col in columns %}
        <option value="{{ col }}">{{ col }}</option>
        {% endfor %}
    </select>

    <button onclick="generatePlot()">Generate Scatter Plot</button>
    <div id="scatter-plot" style="width:100%;height:600px;"></div>
    {% endif %}
</div>
<script>
const data = {{ data|safe }};

function generatePlot() {
    const xCol = document.getElementById('x-axis').value;
    const yCol = document.getElementById('y-axis').value;
    const colorCol = document.getElementById('color').value;

    const x = data.map(row => row[xCol]);
    const y = data.map(row => row[yCol]);
    const colors = colorCol ? data.map(row => row[colorCol]) : 'blue';

    const trace = {
        x: x,
        y: y,
        mode: 'markers',
        type: 'scatter',
        marker: { color: colors, size: 10 },
        text: data.map(row => JSON.stringify(row))
    };

    const layout = {
        title: 'Interactive Scatter Plot',
        xaxis: { title: xCol },
        yaxis: { title: yCol },
        hovermode: 'closest'
    };

    Plotly.newPlot('scatter-plot', [trace], layout);
}
</script>
</body>
</html>
```