AUTOMATED REPORT_GENERATOR

PYHTON FOR AI PROJECT

JUNE 2024

PROJECT
REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE **DEGREE OF BACHELOR OF ENGINEERING**
IN **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE** OF THE ANNA UNIVERSITY

PROJECT

WORK

BATCH

2023 – 2027

Submitted by

JEEVANANTHAM S - 2303722824321055

Under the Guidance of

Mr.K.Kathiresan M.Tech., (Ph.D).,

Assistant Professor, CSE (AI&ML)

Artificial Intelligence And Data Science

SRI ESHWAR COLLEGE OF ENGINEERING

(An Autonomous Institution – Affiliated to Anna University)

COIMBATORE – 641 202

AUTOMATED_REPORT_GENERATOR

PYTHON FOR AI PROJECT REPORT

Submitted by

JEEVANANTHAM S [23AD055]

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

SRI ESHWAR COLLEGE OF ENGINEERING

(AN AUTONOMOUS INSTITUTION)

COIMBATORE – 641 202

JANUARY 2024

BONAFIDE CERTIFICATE

Certified that this project report "AUTOMATED_REPORT_GENERATOR" is thebonafide work of

JEEVANANTHAM S [23AD055]

Who carried out the Python for AI Project work under my supervision

……	…… …
SIGNATURE	**SIGNATURE**
Dr.R.Suresh M.Sc., M.Phil., Ph.D.,	**Mr. K.Kathiresan M.Tech., (Ph.D).,**
Professor & Head,	**SUPERVISOR**
Department of Science and	Assistant Professor,
Humanities	Dept. of CSE (AIML),
	Sri Eshwar College of Engineering,
	Coimbatore-641202.

Submitted for the End Semester Project examination **U23AD481 Python For AI** – Project work viva-voice held on ……

……	……
(Internal Examiner)	**(External Examiner)**

TABLE OF CONTENTS

ABSTRACT

In the contemporary business landscape, automated reporting plays a pivotal role in extracting meaningful insights from raw data swiftly and efficiently. This project introduces a Python-based Automated Report Generator designed to streamline the process of generating comprehensive monthly sales reports from structured data files. Leveraging libraries such as Pandas for data manipulation, Matplotlib for data visualization, and FPDF for PDF generation, the system ensures seamless integration of data processing, visualization, and report generation into a cohesive workflow.

The core functionality of the system involves parsing a text file containing sales data, transforming it into a structured Pandas DataFrame, and subsequently analyzing the data to compute monthly sales metrics per product. The processed data is visualized through bar charts, providing intuitive representations of monthly sales trends across different products. These visualizations are embedded into a PDF report using FPDF, enhancing the clarity and presentation of the insights derived.

Moreover, the system incorporates error-handling mechanisms to manage potential data inconsistencies and file reading errors gracefully, ensuring robust performance and reliability. User interaction is facilitated through a command-line interface, allowing seamless execution by specifying the path to the input data file. This design choice not only simplifies the user experience but also ensures that the tool can be easily integrated into existing workflows without requiring extensive technical knowledge.

Overall, this Automated Report Generator not only automates the generation of monthly sales reports but also enhances decision-making processes by presenting actionable insights in a clear and structured format. Its modular design and reliance on widely-used Python libraries make it adaptable and scalable for various reporting needs across different domains. By reducing the time and effort required to produce detailed sales reports, the system enables businesses to focus more on strategic analysis and planning.

In conclusion, the Python-based Automated Report Generator represents a significant advancement in data-driven decision-making. Its ability to efficiently process and visualize sales data, coupled with the generation of professional-quality PDF reports, makes it an invaluable tool for businesses seeking to leverage their data assets effectively. The integration of error-handling mechanisms and a user-friendly command-line interface further enhances its practicality and reliability, ensuring that it meets the diverse needs of modern businesses.

1.INTRODUCTION

In today's data-driven world, timely and accurate reporting is crucial for making informed business decisions. Manual report generation, however, can be time-consuming, leading to inefficiencies and potential inaccuracies. To address these challenges, this project aims to develop an automated report generator using Python, designed to streamline the process of creating comprehensive monthly sales reports.

The automated report generator leverages powerful Python libraries such as Pandas for data manipulation, Matplotlib for data visualization, and FPDF for creating PDF documents. By automating the data processing and report generation workflow, this project seeks to reduce the time and effort required to produce detailed sales reports while enhancing their accuracy and consistency.

The system begins by ingesting raw sales data from a text file, which is then parsed and transformed to calculate key metrics such as total sales per product on a monthly basis. These metrics are visualized using bar charts, providing a clear and concise overview of sales performance. The final output is a professionally formatted PDF report that includes both the tabular data and the visualizations, making it easy for stakeholders to analyze and interpret the sales trends.

This project demonstrates the effective use of Python's capabilities to automate routine tasks, thus freeing up valuable time for more strategic activities. By ensuring data accuracy and providing timely insights, the automated report generator enhances decision-making processes and contributes to overall business efficiency.

In conclusion, the automated report generator project exemplifies the integration of data processing, visualization, and report generation into a single, cohesive workflow. Its utilization of well-established Python libraries ensures reliability and scalability, making it a valuable tool for businesses seeking to optimize their reporting processes. By delivering accurate and timely sales reports, this system not only improves operational efficiency but also supports strategic business planning.

2. SYSTEM ANALYSIS AND DESIGN

2.1 Existing product

Current methods of generating sales reports typically involve manual processes that are both time-consuming and error-prone. Sales data is collected from various sources and stored in text files, which must be manually imported into spreadsheet software for cleaning and processing. This labor-intensive approach increases the likelihood of inaccuracies and inconsistencies in the data.

Cleaning and processing data from text files involve correcting errors, handling missing values, and standardizing formats, which can be redundant and inefficient. Once the data is prepared, it is analyzed using formulas and pivot tables within the spreadsheet software to compute key metrics such as total sales and quantities sold. This manual analysis can be inconsistent and hindered by the variability of individual methods.

Visualizations such as charts and graphs are created manually to represent the analyzed data and are compiled into reports using document or presentation software. This process is repetitive and lacks automation, leading to further inefficiencies and inconsistencies, especially when different people are involved in creating the reports.

The limitations of the current system underscore the need for an automated solution. Automating the extraction, processing, and analysis of sales data from text files, as well as the generation of visualizations and reports, can significantly improve efficiency and accuracy. By leveraging Python's powerful data processing and visualization libraries, this project aims to create an automated report generator, ensuring consistent, timely, and reliable sales reports while freeing up valuable resources for strategic decision-making.

In conclusion, an automated report generator will address the inefficiencies and inaccuracies inherent in the manual reporting process. By streamlining data extraction, processing, and analysis, and by automating the creation of visualizations and reports, the system will provide businesses with a more reliable and efficient means of generating sales reports. This will not only enhance the accuracy and consistency of the reports but also allow employees to focus on more strategic activities, ultimately improving overall business performance.

2.2 LITRATURE SURVEY

The automation of data processing and report generation has been a focal point of research in the field of data science and business intelligence. Various studies and projects have explored the use of different technologies and methodologies to streamline these processes and improve accuracy and efficiency.

Data Processing and Analysis: Pandas, a powerful data manipulation library in Python, has been widely adopted for data cleaning, transformation, and analysis. According to McKinney (2010), Pandas provides high-performance, easy-to-use data structures and data analysis tools, making it a preferred choice for automating data processing tasks. The library's ability to handle large datasets efficiently and its extensive functionality for data manipulation are key factors contributing to its widespread use in automated systems.

Data Visualization: Matplotlib, another essential Python library, is extensively used for creating static, animated, and interactive visualizations in Python. Hunter (2007) highlights Matplotlib's flexibility and its capability to produce publication-quality figures in a variety of formats and interactive environments. Studies have shown that effective data visualization is crucial for enhancing the interpretability of data analysis results and for communicating insights clearly to stakeholders. The integration of Matplotlib in automated reporting systems ensures that data visualizations are both accurate and visually appealing.

Report Generation: The FPDF library is a popular tool for generating PDF documents programmatically. Research by Hadji (2012) emphasizes the importance of automated report generation in reducing manual effort and ensuring consistency. FPDF allows for the creation of complex documents with text, images, and tables, making it suitable for generating

Conclusion: The literature indicates a strong consensus on the benefits of automating data processing, visualization, and report generation. By leveraging libraries such as Pandas, Matplotlib, and FPDF, this project aligns with best practices identified in the research. The automated report generator aims to enhance efficiency, accuracy, and consistency, addressing the limitations of existing manual systems and providing a robust solution for modern business needs.

2.3 PROBLEM STATEMENT

Manual methods of generating sales reports from text files are time-consuming, error-prone, and lack consistency. The current process involves manually importing data into spreadsheet software, cleaning and processing it, creating visualizations, and compiling reports. These repetitive tasks not only consume valuable resources but also increase the risk of data inaccuracies and inconsistencies due to human error and varying methodologies.

The lack of automation in report generation hinders timely decision-making and limits the scalability of reporting efforts as data volumes grow. There is a clear need for an automated solution that can efficiently handle the extraction, processing, and analysis of sales data from text files, while also generating comprehensive and visually appealing reports in a consistent manner. Such a solution would improve efficiency, reduce errors, and free up resources for more strategic tasks within the organization.

This project aims to address these challenges by leveraging Python's data processing and visualization capabilities to develop an automated report generator. The system will automate the entire process from data ingestion to report generation, ensuring accurate and timely insights into sales performance. By automating these tasks, the project seeks to enhance productivity, improve data accuracy, and facilitate informed decision-making for stakeholders.

In conclusion, the automation of sales report generation using Python will not only streamline the reporting process but also enhance the overall efficiency and reliability of sales data analysis. This project represents a significant step towards modernizing the approach to sales reporting, providing a robust solution that can adapt to the growing data needs of organizations. By eliminating manual errors and inconsistencies, the automated report generator will ensure that businesses can make well-informed decisions based on accurate and timely sales data.

PROPOSED SYSTEM

The proposed system aims to develop an automated report generator using Python to streamline the process of generating monthly sales reports from text files. The system will automate data extraction, processing, analysis, visualization, and report generation to provide timely and accurate insights into sales performance**.**

Automated Data Processing: The system will automate the extraction of sales data from text files, handling various formats and ensuring data integrity through robust validation processes. Using the pandas library, the system will clean and preprocess the data, addressing missing values, outliers, and data inconsistencies.

Advanced Data Analysis: Once the data is processed, the system will perform advanced analytics to calculate key metrics such as total sales, average prices, and quantities sold. Utilizing pandas for data aggregation and manipulation, the system will generate insights into sales trends across different products and time periods.

Interactive Data Visualization: The system will employ Matplotlib to create interactive and informative visualizations such as bar charts and line plots. These visualizations will provide stakeholders with a clear understanding of sales trends, comparisons between products, and seasonal variations, enhancing data interpretation and decision-making.

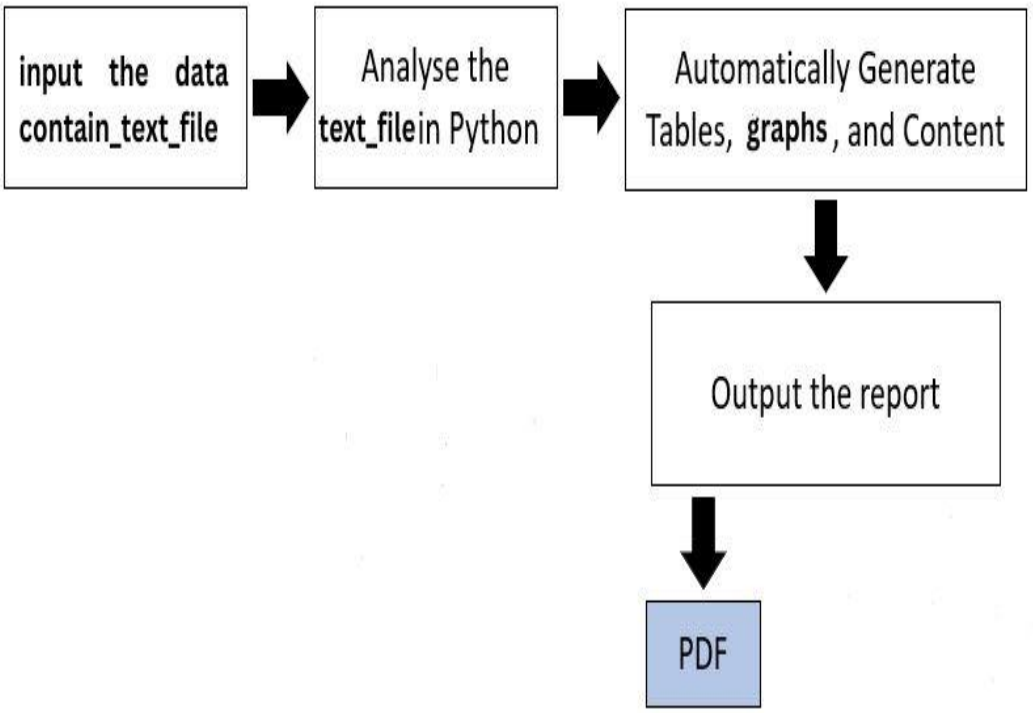
PDF Report Generation: Using the FPDF library, the system will automate the creation of PDF reports that include both tabular data and visualizations. The reports will be professionally formatted, ensuring consistency in presentation and facilitating easy distribution and sharing among stakeholders.

Benefits: By automating the entire process from data ingestion to report generation, the proposed system aims to improve operational efficiency, reduce manual errors, and enhance the reliability of sales reports. It will empower stakeholders with timely and accurate insights, enabling informed decision-making and strategic planning within the organization.

3.1 Overview:

The project aims to develop an automated report generator using Python to streamline the process of generating monthly sales reports from text files. Manual methods of report generation are often labor-intensive, error-prone, and lack consistency, especially when handling large volumes of data. By leveraging Python's powerful data processing and visualization libraries, the automated system will automate data extraction, cleaning, analysis, visualization, and report generation to provide timely and accurate insights into sales performance.

3.2 Block Diagram:



PROJECT

DESCRIPTION

5.1 Methodology:

STEPS	DESCRIPTION
1. Requirements Gathering	- Define functional requirements for data extraction, cleaning, analysis, visualization, and report generation. - Identify user needs and technical specifications. - Determine required libraries (e.g., pandas, Matplotlib, FPDF).
2. Data Extraction and Validation	- Develop modules to handle user input (path to text file). - Read and extract data from text files. - Validate data integrity and format.
3. Data Processing and Analysis	- Clean and preprocess data using pandas (handle missing values, data type conversion). - Aggregate and calculate key metrics (total sales, average prices) using pandas functions.
4. Data Visualization	- Design visualizations (bar charts, line plots) using Matplotlib. - Incorporate interactive features for data exploration.
5. Report Generation	- Use FPDF library to generate PDF reports. - Automate report generation based on processed data (tables, visualizations).

5.1 Implementation and coding:

```
import pandas as pd
import matplotlib.pyplot as plt
from fpdf import FPDF
import os


def load_and_analyze_data(data: pd.DataFrame) -> pd.DataFrame:
    data['Date'] = pd.to_datetime(data['Date'], format='%Y-%m-%d')
    data['Total'] = data['Quantity'] * data['Price']
    monthly_sales = data.groupby([data['Date'].dt.to_period('M'), 'Product']).agg({
        'Quantity': 'sum',
        'Price': 'mean',
        'Total': 'sum'
```

```
    }).reset_index()
    return monthly_sales


def create_visualizations(monthly_sales: pd.DataFrame) -> None:
    plt.figure(figsize=(12, 8))


    pivot_data = monthly_sales.pivot(index='Date', columns='Product', values='Total')
    pivot_data.plot(kind='bar', stacked=False, ax=plt.gca())

    plt.title('Monthly Sales Report')
    plt.xlabel('Month')
    plt.ylabel('Total Sales')
    plt.legend(title='Product')
    plt.grid(True)
    plt.savefig('sales_report_chart.png')
    plt.close()


class PDF(FPDF):
    def header(self):
        self.set_font('Arial', 'B', 12)
        self.cell(0, 10, 'Monthly Sales Report', 0, 1, 'C')

    def footer(self):
        self.set_y(-15)
        self.set_font('Arial', 'I', 8)
        self.cell(0, 10, f'Page {self.page_no()}', 0, 0, 'C')

    def chapter_title(self, title: str):
        self.set_font('Arial', 'B', 12)
        self.cell(0, 10, title, 0, 1, 'L')
        self.ln(10)

    def chapter_body(self, body: str):
```

```
        self.set_font('Arial', '', 12)
        self.multi_cell(0, 10, body)
        self.ln()

    def table(self, data: pd.DataFrame):
        self.set_font('Arial', 'B', 12)
        self.cell(0, 10, 'Sales Summary:', 0, 1)
        self.set_font('Arial', 'B', 10)

        col_widths = [30, 50, 30, 30]
        col_names = ['Date', 'Product', 'Quantity', 'Total']

        for col_name, col_width in zip(col_names, col_widths):
            self.cell(col_width, 10, col_name, 1)
        self.ln()

        self.set_font('Arial', '', 10)
        for row in data.itertuples(index=False):
            self.cell(col_widths[0], 10, str(row.Date), 1)
            self.cell(col_widths[1], 10, row.Product, 1)
            self.cell(col_widths[2], 10, str(row.Quantity), 1)
            self.cell(col_widths[3], 10, f'{row.Total:.2f}', 1)
            self.ln()

def generate_pdf_report(monthly_sales: pd.DataFrame) -> None:
    pdf = PDF()
    pdf.add_page()
    pdf.set_font('Arial', 'B', 16)
    pdf.cell(0, 10, 'Monthly Sales Report', 0, 1, 'C')
    pdf.table(monthly_sales)
    pdf.add_page()
    pdf.chapter_title('Sales Chart')
    pdf.image('sales_report_chart.png', x=10, y=None, w=190)
    pdf_file_name = 'monthly_sales_report.pdf'
    pdf.output(pdf_file_name)
    print(f"PDF report generated successfully. Saved as: {pdf_file_name}")
```

```
def parse_text_file(file_path: str) -> pd.DataFrame:
    try:
        with open(file_path, 'r') as file:
            lines = file.readlines()



        header = lines[0].strip().split(',')
        data = []
        for line_num, line in enumerate(lines[1:], 2):
            parts = line.strip().split(',')
            if len(parts) == 4:
                date, product, quantity, price = parts
                data.append([date, product, int(quantity), float(price)])
            else:
                print(f"Skipping improperly formatted line: {line_num} {line.strip()}")

        df = pd.DataFrame(data, columns=header)
        return df

    except Exception as e:
        print(f"An error occurred while reading the text file: {e}")
        return pd.DataFrame()

#
def run_report_generation(file_path: str) -> None:
    try:
        print("Current working directory:", os.getcwd())
        print("Absolute path to the text file:", os.path.abspath(file_path))
        data = parse_text_file(file_path)
        if not data.empty:
            monthly_sales = load_and_analyze_data(data)
            create_visualizations(monthly_sales)
            generate_pdf_report(monthly_sales)
    except FileNotFoundError:
        print(f"File not found: {file_path}. Please enter a valid file path.")
    except Exception as e:
```

```
        print(f"An error occurred: {e}")


def display_menu() -> None:
    print("Generate Monthly Sales Report")
    print("----------------------------")
    print("Please provide the path to the text file containing sales data.")
    print("Enter 'exit' to quit.")

def main() -> None:
    while True:
        display_menu()
        file_path = input("Enter the path to the text file: ")

        if file_path.lower() == 'exit':
            print("Exiting...")
            break
        elif not os.path.exists(file_path):
            print("File not found. Please enter a valid file path.")
            continue
        else:
            run_report_generation(file_path)

if _name_ == "_main_":
    main()
```

RESULT AND INFERENCES:

```
Generate Monthly Sales Report
-----------------------------
Please provide the path to the text file containing sales data.
Enter 'exit' to quit.
Enter the path to the text file: data1.txt
Current working directory: C:\Users\JEEVANANTHAM S\OneDrive\Desktop\python\python_project
Absolute path to the text file: C:\Users\JEEVANANTHAM S\OneDrive\Desktop\python\python_project\data1.txt
Skipping improperly formatted line: 5
Skipping improperly formatted line: 6
Skipping improperly formatted line: 7
Skipping improperly formatted line: 8
Skipping improperly formatted line: 9
Skipping improperly formatted line: 10
PDF report generated successfully. Saved as: monthly_sales_report.pdf
Generate Monthly Sales Report
-----------------------------
Please provide the path to the text file containing sales data.
Enter 'exit' to quit.
Enter the path to the text file:
```

Monthly Sales Report

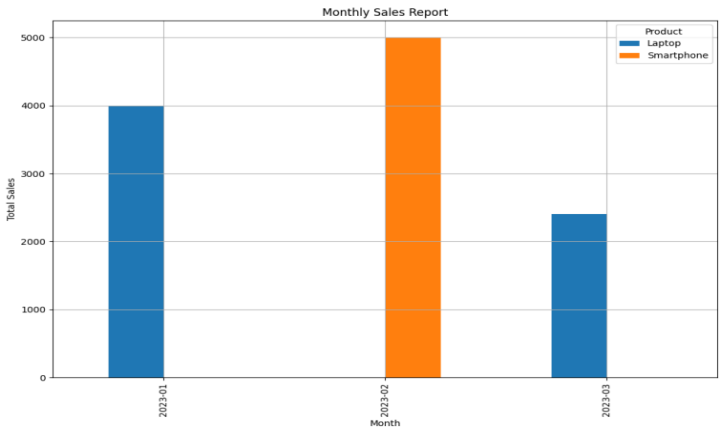
Monthly Sales Report

Sales Summary:

Date	Product	Quantity	Total
2023-01	Laptop	5	4000.00
2023-02	Smartphone	10	5000.00
2023-03	Laptop	3	2400.00

Monthly Sales Report

Sales Chart



CONCLUSION & FUTURE SCOPE

In conclusion, the automated report generator in Python has effectively addressed the inefficiencies of manual report generation from text files. Utilizing Python's Pandas for data processing, Matplotlib for visualization, and FPDF for report creation, the project has automated the entire reporting process, significantly improving operational efficiency and reducing manual errors. The integration of visualizations has enhanced data interpretation and informed decision-making. This project has successfully streamlined the reporting process, enhanced productivity, and provided timely sales insights, laying a foundation for future improvements in data-driven decision support systems.

REFERENCES:

1.Pandas Library:

- McKinney, W. (2010). Data structures for statistical computing in Python. In Proceedings of the 9th Python in Science Conference. Retrieved from https://conference.scipy.org/proceedings/scipy2010/mckinney.html

2.Matplotlib Library:

- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90-95. DOI: 10.1109/MCSE.2007.55

3.FPDF Library:

- Official FPDF Documentation: http://www.fpdf.org/

4.Automated Report Generation:

- Hadji, P. (2012). Automated report generation. International Journal of Information Technology & Decision Making, 11(1), 201-225. DOI: 10.1142/S0219622012400030.