INTELLIGENT SALES ANALYTICS AND FORECASTING ENGINE

A DESIGN PROJECT REPORT

submitted by

SWATHI R

VIGNESH V

VIGNESHWARAN R

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)

Samayapuram – 621 112

JUNE 2025

INTELLIGENT SALES ANALYTICS AND FORECASTING ENGINE

A DESIGN PROJECT REPORT

submitted by

SWATHI R (811722104165)

VIGNESH V (811722104180)

VIGNESHWARAN R (811722104181)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)

Samayapuram – 621 112

JUNE 2025

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **"INTELLIGENT SALES ANALYTICS AND FORECASTING ENGINE"** is Bonafide work of **SWATHI R (811722104165), VIGNESH V (811722104180), VIGNESHWARAN R (811722104181)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Dr. A Delphin Carolina Rani, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR
Department of CSE
K Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

SIGNATURE
Ms. V Sowmiya, M.E.,
SUPERVISOR
Assistant Professor
Department of CSE
K Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

Submitted for the viva-voice examination held on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the project report on **"INTELLIGENT SALES ANALYTICS AND FORECASTING ENGINE"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of Bachelor Of Engineering. This project report is submitted on the partial fulfilment of the requirement of the awardof Degree of Bachelor Of Engineering.

Signature

SWATHI R

VIGNESH V

VIGNESHWARAN R

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and indebtedness to our institution **"K RAMAKRISHNAN COLLEGE OF TECHNOLOGY"**, for providing us with the opportunity to do this project.

We are glad to credit and praise our honorable and respected chairman sir **Dr. K RAMAKRISHNAN, B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering adequate duration to complete it.

We would like to thank **Dr. N VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project with full satisfaction.

We heartily thank **Dr. A DELPHIN CAROLINA RANI, M.E., Ph.D.,** Head of the Department, **COMPUTER SCIENCE AND ENGINEERING** for providing her support to pursue this project.

We express our deep and sincere gratitude and thanks to our project guide **Ms. V SOWMIYA , M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for her incalculable suggestions, creativity, assistance and patience which motivated us to carry our this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course. We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

In the modern business landscape, data-driven decision-making is essential for maintaining a competitive edge and fostering organizational growth. In this project, an Intelligent Sales Analytics and Forecasting Engine has been introduced that leverages advanced natural language processing and machine learning techniques to streamline data analysis and enhance strategic decision-making. In this work, users can interact with the assistant using simple natural language queries, and the system automatically interprets, processes, and retrieves actionable insights from vast datasets. The engine incorporates robust data storage mechanisms to securely store, organize, and manage structured and unstructured business data, enabling efficient retrieval and processing. It also supports predictive analytics, helping businesses forecast trends, identify risks, and make strategic decisions efficiently. Once a query is processed, the Prophet based assistant dynamically generates visual reports, charts, and summaries, offering real-time analytics tailored to the user's specific needs. This automation eliminates the complexity of traditional BI tools, making data analysis accessible to all stakeholders, including those without technical expertise. By simplifying access to critical information, the assistant empowers organizations to act swiftly and confidently. Moreover, the result supports predictive analysis by identifying trends, anomalies, and performance patterns using historical data. It can forecast business outcomes, recommend strategic actions, and highlight potential risks and opportunities. This intelligent support system ensures that businesses not only react to current conditions but also proactively plan for the future, improving efficiency, productivity, and competitiveness.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION		FULL FORM
LLM	-	Large Language Model
API	-	Application Programming Interface
CSV	-	Comma Separated Values
CRM	-	Customer Relationship Management
ERP	-	Enterprise Resource Planning
AI	-	Artificial Intelligence
SQL	-	Structured Query Language
NLIs	-	Natural Language Interfaces
BI	-	Business Intelligence
KPIs	-	Key Performance Indicators
NLG	-	Natural Language Generation
RAG	-	Retrieval-Augmented Generation
IQR	-	Interquartile Range
HR	-	Human Resources
IT	-	Information Technology
SDLC	-	Software Development Life Cycle
LLaMa	-	Large Language Model Meta AI
GPT	-	Generative Pre-trained Transformer

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND.

The Intelligent Sales Analytics And Forecasting Engine project introduces a novel approach to helping businesses analyse and interpret their data with minimal technical expertise. Traditional BI systems often require manual data cleaning, domain knowledge, and experience in analytics tools, which can be overwhelming for small businesses or non-technical users. This project seeks to solve that problem by providing an intelligent, interactive assistant that simplifies data handling and automates insight generation. Using advanced Large Language Models (LLMs) and a Gradio-based web interface, the assistant allows users to upload their datasets and instantly receive insights in natural language.

It combines time-series analysis, forecasting, and product performance tracking into a single, intuitive platform. Through visualizations such as line charts, bar charts, pie charts, and regression models, users can better understand sales trends, top-performing products, and future business projections. The assistant is also capable of answering natural language questions based on the data and generating predictive forecasts using models like Prophet. With built-in secure session handling, data visualization, and integration with NVIDIA LLM APIs, the system empowers even non-technical users to make data-driven decisions confidently.

The platform reduces the learning curve and eliminates the dependency on traditional data analysts, allowing businesses to access insights quickly and independently. The modular architecture also supports easy integration with existing CRM or ERP systems, ensuring that the solution can scale and adapt to various business environments.

1.2 OVERVIEW

The Intelligent Sales Analytics And Forecasting Engine is a dynamic, web-based analytical tool designed to empower businesses and decision-makers with immediate insights from raw data—without requiring technical expertise. Built using Python and Gradio, this system provides users with an intuitive interface to upload CSV datasets and receive interactive visualizations, statistical summaries, and sales forecasts. The assistant integrates artificial intelligence, enabling users to ask natural language questions and receive contextual, data-driven responses powered by a large language model API (NVIDIA/LLM).

Once a dataset is uploaded, the assistant intelligently identifies key columns such as dates, sales, and product names, and prepares the data for analysis. With a single click, users can generate graphs representing sales trends, top-performing products, and time-series forecasts using Prophet. A flexible visualization interface allows switching between chart types (e.g., pie charts, bar graphs, regression lines) for deeper exploration.

To support strategic planning, the assistant leverages machine learning to forecast upcoming sales and trends, helping businesses anticipate demand and allocate resources effectively. By streamlining complex data analysis into an accessible, user-friendly dashboard, the AI-Powered Business Intelligence Assistant encourages a data-driven culture and supports faster, more informed business decisions.

The assistant's architecture ensures seamless user experience, from data upload to insight generation, all within a few clicks. It eliminates the need for manual data cleaning and plotting by automating preprocessing and visualization. With natural language processing capabilities, users can interact with the system conversationally, making analytics more approachable. The modular design also allows easy integration with other tools and platforms, such as dashboards, CRMs, or ERPs. Overall, the assistant bridges the gap between raw business data and strategic decision-making with intelligent, real-time support.

1.3 PROBLEM STATEMENT

In today's data-driven world, businesses generate vast amounts of data daily, yet many lack the technical resources or expertise to analyse and extract meaningful insights from it. Traditional business intelligence tools are often complex, expensive, and require specialized knowledge, making them inaccessible to small and mid-sized enterprises. This gap leads to underutilized data, missed opportunities, and slower decision-making.

1.4 OBJECTIVE

- To build an Intelligent Sales Analytics and Forecasting Engine using Python and Gradio that answers business-related natural language queries, generates visualizations, and forecasts future sales using Prophet through a browser-based interface.
- To systematically collect, clean, and analyze past sales data to uncover meaningful patterns, seasonal trends, and customer behavior.
- To implement machine learning algorithms that can accurately predict future sales based on historical data.

1.5 IMPLICATION

The Intelligent Sales Analytics And Forecasting Engine enables non-technical users to analyse data with ease by automating insight generation, visualization, and forecasting. It transforms raw business data into meaningful trends and summaries, helping organizations make faster, data-driven decisions without relying on dedicated analysts. With natural language interaction and a user-friendly browser-based interface, the system promotes widespread data literacy, enhances operational efficiency, and supports smarter decision-making across all levels of the organization.

CHAPTER 2
LITERATURE SURVEY

1. Study on Personalized Business Insights Through AI and Recommendation Systems, Kavitha Nair, Ritesh Anand – 2023

This paper [1] proposed an AI-based framework for delivering personalized business insights using user behaviour analysis and recommendation algorithms. The system collected data on user roles and interaction history to suggest relevant KPIs, dashboards, and visualizations. Collaborative filtering and machine learning models tailored the experience for departments such as marketing, finance, and sales. The framework was tested in a multi-department enterprise, where feedback indicated increased engagement and improved productivity. The study [1] concluded that personalization through AI significantly enhanced the adoption and effectiveness of BI systems.The system's complexity may require significant computational resources and technical expertise to maintain. Cross-departmental differences in data usage might reduce the accuracy of generalized recommendations. Privacy concerns arise from collecting and analyzing detailed user interaction data.

2. Study on AI-Based Self-Service BI Platforms, Nishant Joshi, Akshita Suresh – 2023

This research [2] addressed the development of AI-powered self-service BI platforms that enabled users to explore and analyze data without IT support. The system integrated reinforcement learning and recommendation engines to suggest optimal queries, filters, and visualizations based on user behaviour. Prototypes built using HR and financial datasets demonstrated capabilities such as auto-optimization of dashboards and intelligent view selection. User studies showed a reduced learning curve and improved satisfaction. By minimizing dependence on technical teams, the platform democratized access to analytics and accelerated insight generation across all organizational levels.Recommendation

engines may suggest irrelevant or redundant queries if user behaviour is inconsistent or ambiguous. Auto-optimization of dashboards could lead to oversimplified views, omitting critical data points.

3. Study on Cognitive BI Assistants Using NLP and Data Mining, Deepika Chauhan, M. Vishwanath – 2022

This research [3] explored the design of a cognitive Business Intelligence assistant that combined NLP with data mining techniques to deliver actionable insights. The assistant interpreted natural language queries, mined datasets for significant patterns, and provided conversational explanations. Decision tree algorithms were used for analysis, and advanced NLP models handled context and query understanding. The system was evaluated on financial data and proved effective in performing root cause analysis and anomaly detection. This paper highlighted the potential of cognitive assistants to move beyond reporting and offer contextual, interpretive insights that supported complex decision-making. Decision tree algorithms may not capture complex, non-linear relationships in large datasets. Advanced NLP models require continuous tuning and large training datasets to maintain accuracy.

4. Study on Business Data Analysis Using Chatbots with Machine Learning, Priya Das, R. Subramaniam – 2022

This paper [4] proposed a chatbot system enhanced with machine learning to enable conversational access to business intelligence data. The chatbot interpreted natural language queries using supervised learning algorithms and retrieved data from structured relational databases. It supported dynamic chart generation, voice input, and contextual dialogue, making it a flexible tool for interacting with analytics. Sentiment analysis and keyword extraction were used to interpret ambiguous queries more effectively. Pilot testing within a mid-sized enterprise demonstrated high user satisfaction and efficient information retrieval. The system reduced the need for specialized skills and accelerated insights, offering a more natural and intuitive experience for users dealing with complex

business data.Integration with diverse relational databases may pose compatibility challenges. Users might still encounter limitations when querying highly complex datasets or performing advanced analytics.

5. Study on Conversational AI for Business Intelligence: An Intelligent Assistant for Data Analytics, Simmi Kherwa, Payal Jain, Shikha Mehta – 2022

This paper [5] addressed the challenge of making business intelligence tools more accessible to non-technical users by integrating Conversational AI technologies. Traditional BI platforms required users to have skills in data querying or visualization, limiting their utility. The Simmi Kherwa proposed an intelligent assistant powered by Natural Language Processing (NLP) that interpreted user queries in everyday language and translated them into actionable insights and visual reports. The system architecture included modules for intent recognition, data mapping, and response generation. Experimental evaluation on retail and sales data showed that the assistant could accurately answer complex queries, generate forecasts, and display interactive charts without manual coding. This approach democratized access to analytics and accelerated decision-making. Accurate data mapping depends on well-structured and labelled datasets. Complex queries or multi-intent inputs may challenge the assistant's interpretation capabilities. Limited ability to explain how responses or visualizations are generated may reduce user trust.

6. Study on Intelligent Data Assistants for Real-Time Decision Support, Aman Gupta, Sheetal Raina – 2021

This paper [6] presented an intelligent data assistant designed to deliver real-time decision support within enterprise environments. The assistant integrated AI components including natural language processing, deep learning, and time series forecasting to provide contextual business insights. It interfaced with cloud data warehouses and supported voice-activated queries and dynamic reporting. The study [6] focused on operational metrics such as sales volatility,

inventory management, and customer engagement. The assistant significantly improved the speed and quality of decision-making during fluctuating business conditions. An adaptive learning mechanism allowed it to evolve based on changing data patterns and user feedback, ensuring continuous relevance and accuracy. The study [6] emphasized that such assistants could replace manual BI workflows with more interactive and proactive systems. Limited testing scope focused only on specific metrics like sales and inventory, reducing general applicability. Adaptive learning mechanisms may introduce unintended behaviour if not properly monitored.

7. Study on Integration of AI with Business Intelligence Dashboards, Meenal Jain, Karan Kapoor – 2021

This paper [7] presented the development of an AI-powered Business Intelligence dashboard that personalized user experiences by automatically generating relevant insights and visualizations. Unlike traditional dashboards that required manual setup, the proposed system used machine learning algorithms to analyze user interaction patterns and identify the most useful KPIs and charts. The dashboard adapted dynamically to user preferences and organizational goals by applying clustering and classification techniques. Built on Power BI with Python scripts for data processing, the system was tested using marketing and customer engagement datasets. It successfully demonstrated the ability to reduce cognitive load and enhance decision-making. The Meenal Jain argued that AI integration transformed dashboards into interactive assistants that guided users toward actionable insights. The system was tested only with marketing and customer engagement datasets, limiting its generalizability to other business areas. Personalization accuracy depends heavily on the quality and consistency of user interaction data. Clustering and classification techniques may not always reflect true user intent or business priorities.

8. Study on Natural Language Interfaces for Business Intelligence Systems, Aditi Sharma, Rahul Tripathi – 2021

This paper [8] explored the design of Natural Language Interfaces (NLIs) that simplified interactions with Business Intelligence systems by allowing users to input queries through natural language, either typed or spoken. The primary goal of this work was to remove barriers posed by traditional BI tools, which often required knowledge of SQL or programming. The system employed machine learning models to understand user intent and context, converting natural language queries into structured database queries. The Aditi Sharma tested the system with financial datasets, and evaluation results indicated high accuracy in query interpretation and response generation. Furthermore, the interface supported conversational follow-up queries, enabling continuous and intuitive exploration of data. The research [8] emphasized the importance of making BI systems accessible to non-technical users to improve adoption and facilitate data-driven decisions. The system was primarily tested with financial datasets, limiting its applicability to other domains. User queries in natural language can be ambiguous, leading to potential misinterpretation. The interface may struggle to handle complex or multi-step queries accurately. Accuracy depends on the quality and diversity of the training data.

9. Study on AI-Driven BI Tools for Predictive Business Insights, Harshita Verma, Anil Kumar – 2020

This research [9] focused on enhancing Business Intelligence systems by embedding Artificial Intelligence algorithms to enable predictive analysis. The proposed framework integrated AI models such as decision trees and neural networks within BI platforms to identify hidden trends and provide future business forecasts. Using historical retail and supply chain data, the system was trained to detect patterns and predict sales, inventory needs, and market shifts with high accuracy. In contrast to traditional BI tools, which primarily performed

descriptive analytics, the AI-driven system allowed organizations to proactively respond to market changes and mitigate risks. Additionally, the study incorporated automated anomaly detection to alert users about unusual data behaviours. The results demonstrated that AI-enhanced BI tools improved forecasting performance and operational efficiency, transforming BI from a reactive reporting tool into a strategic asset. The accuracy of predictions heavily relies on the quality and completeness of historical data. The system was trained on retail and supply chain data, limiting its generalizability to other industries. Integrating complex AI models like neural networks increases system complexity, requiring specialized expertise for maintenance.

10. Study on AI-Powered Query Understanding in Business Intelligence Systems, Soham Raj, Keerthi Iyer – 2020

This paper [10] investigated how deep learning techniques could significantly enhance query understanding within natural language-based Business Intelligence tools. The Soham Raj developed a model trained on a large dataset pairing business queries with their corresponding SQL commands. This model translated natural language queries into executable queries, improving accuracy even when queries were vague or incomplete. The system was integrated into a BI platform and tested using e-commerce sales data, where it demonstrated excellent performance in correctly interpreting user intent and providing relevant results. The study [10] also discussed how AI techniques bridged the gap between complex data models and end-user expectations, enabling users without SQL knowledge to interact easily with business data. The system's performance is highly dependent on the quality of the training dataset. Limited dataset diversity may reduce the model's adaptability.It may not generalize well across various industries.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In many small and medium-sized businesses (SMBs) or startups, sales analytics and forecasting are often handled by basic or rudimentary methods. These existing systems generally include manual record-keeping, simple spreadsheets, and basic charting tools. The limitations of these systems create significant challenges for businesses striving to make data-driven decisions. Many small businesses rely on manual data entry into spreadsheets like Microsoft Excel or Google Sheets to track sales and customer information. Some SMBs use basic reporting tools or free/open-source software that provide simple dashboards and static reports. Occasionally, businesses with some technical expertise use simple forecasting scripts in Python or Excel add-ins to predict future sales.

3.1.1 DRAWBACKS

- **Lack of Real-Time Insights:** Traditional systems do not support real-time data processing, causing delays in decision-making.
- **Limited Personalization:** Dashboards and reports are generic and not tailored to individual user roles or needs.
- **Inflexible Query Handling:** Most systems require exact keyword matching or structured queries, limiting flexibility in data exploration.
- **Low Scalability:** Performance degrades significantly as data volume or user demand increases.
- **No Natural Language Support:** Users must rely on predefined reports or SQL, as conversational interaction is not supported.
- **Limited Predictive Power:** Existing tools primarily offer descriptive analytics and fail to provide forward-looking insights or recommendations.

3.2 PROPOSED SYSTEM

The proposed Intelligent Sales Analytics And Forecasting Engine aims to revolutionize the way users interact with business data by enabling voice or text-based natural language queries for instant insights. Unlike traditional BI tools, this system removes the need for users to manually create reports or understand complex data schemas. It uses advanced natural language processing (NLP) to interpret user queries and automatically generate visualizations and summaries. The assistant supports integration with multiple data sources, including cloud storage, SQL databases, and spreadsheets. The assistant also includes predictive analysis features to forecast trends and recommend actions using machine learning models. Built with modern AI frameworks, it continuously learns from interactions, improving accuracy and relevance over time. This system is used to support business decision-making by providing timely insights, reducing reliance on data analysts, increasing decision-making speed, and promoting a data-driven culture across all levels of the organization.

3.3 PROPOSED SYSTEM ARCITECTURE

The architecture begins with a User interacting via a Gradio Interface to upload a CSV file, triggering system initialization and data preprocessing. The processed data then diverges into two core modules: the ChatBot, powered by NVIDIA Nemotron for natural language Q&A, and the Forecasting module, which employs Prophet for time-series predictions and Matplotlib to generate visualizations based on user-selected chart types. Both text responses from the ChatBot and interactive plots from the forecasting results are rendered and displayed back to the user through the unified Gradio interface, integrating data analysis, NLP, and visualization into a seamless workflow. Figure 3.1 shows the proposed architecture of the model.

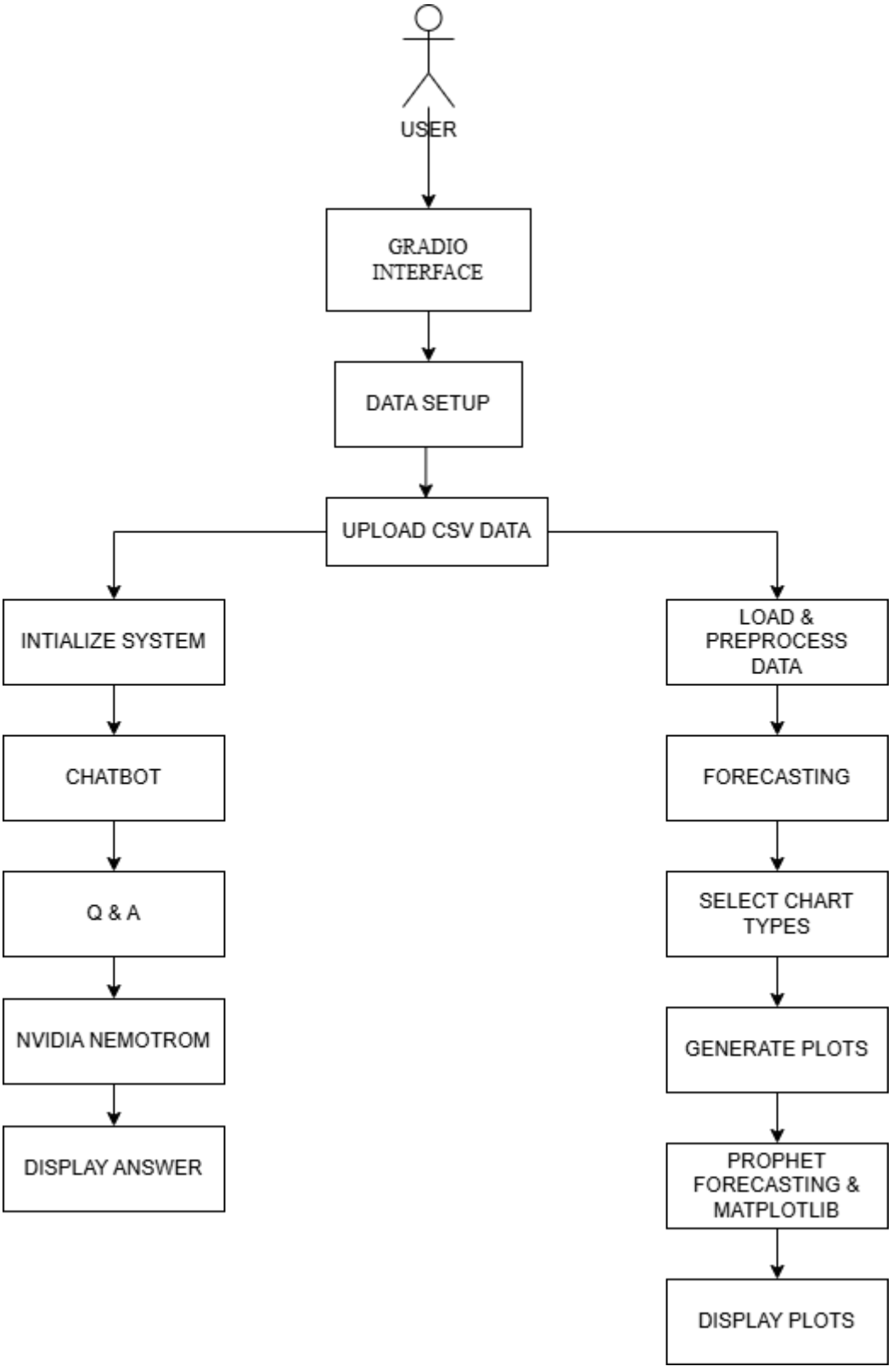

Fig 3.1: PROPOSED SYSTEM ARCITECTURE

3.4 FEATURES

- The system provides a real-time dashboard to monitor and visualize key sales metrics, allowing users to track performance indicators such as revenue, units sold, and profit margins with dynamic charts and graphs.
- It offers predictive insights to identify high-performing products and sales opportunities, helping sales teams focus on the most profitable items and market segments.
- The system analyzes the sales funnel to track customer journeys and improve conversion rates, revealing where potential customers drop off and providing recommendations to increase engagement and sales.
- The engine integrates seamlessly with CRM, ERP to provide a unified and comprehensive view of sales operations.
- The system includes a built-in AI-powered chatbot for interactive user support and insights, enabling users to ask questions like “What were last month’s top-selling products?” or “Show me sales forecasts for next quarter” and receive instant, conversational responses.

CHAPTER 4

MODULES

4.1 MODULE DESCRIPTION

This project is structured into several key modules, each responsible for specific functionalities to ensure efficient data handling, analysis, and user interaction. The following modules work together to deliver a comprehensive Intelligent Sales Analytics and Forecasting Engine

- Data Gather and Upload Module
- Data Preprocessing Module
- Statistical Analysis Module
- Visualization Module
- Forecasting Module
- AI Query Module
- User Interface Module

4.1.1 DATA GATHERING AND UPLOAD

This module serves as the entry point for gathering and uploading business datasets, primarily in CSV format. It validates file compatibility, reads data efficiently into memory, and performs initial checks on column count, row size, and data integrity. The module supports real-time data previews and provides feedback on upload success or failure, ensuring a reliable data foundation for subsequent processing, analysis, and visualization tasks. fig 4.1 shows the flow of data

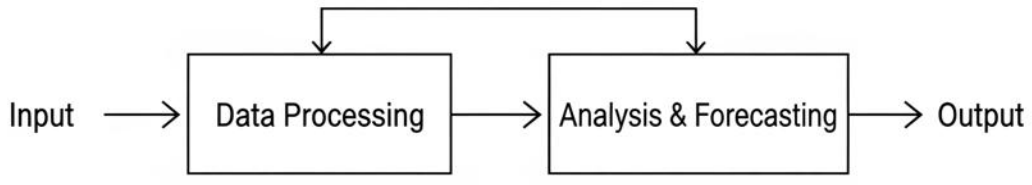

Fig 4.1: Flow of Data

4.1.2 DATA PREPROCESSING

The Data Preprocessing Module automates the preparation of raw data to make it analysis-ready. It uses pandas and NumPy libraries to handle data frames efficiently and performs operations like type conversion . It intelligently detects critical columns such as date, product, and sales. Missing values are handled using imputation techniques like mean or median substitution, or by removal of incomplete rows. Column names are standardized through string normalization for consistent referencing. The module uses Z-score and IQR methods to detect and manage outliers and employs deduplication algorithms to eliminate redundant entries. This cleaning and transformation step ensures data quality and integrity for subsequent analysis.

These preprocessing techniques ensure the dataset is consistent, clean, and logically structured. The module lays a robust foundation to avoid bias, improve model performance, and maintain data integrity throughout the pipeline.

4.1.3 STATISTICAL ANALYSIS

This module performs descriptive statistical computations using built-in statistical functions in pandas and SciPy. It extracts insights by calculating measures of central tendency (mean, median), dispersion (standard deviation, variance), and frequency distributions. It leverages group-by aggregation methods to analyze sales volume and product performance across time periods and categories. Additionally, it uses time-series decomposition for trend identification and ranking algorithms to list top-performing products. Results are rendered in textual summaries, serving as context for AI-powered responses and supporting data-driven decision-making.

The use of statistical metrics helps in understanding historical performance and operational efficiency. The module adds analytical depth that supports accurate evaluation of business dynamics.

4.1.4 VISUALIZATION

The Visualization Module employs Matplotlib, Seaborn, and Plotly libraries to convert numerical results into visual formats. It includes line plots for trend analysis, bar charts for category comparisons, pie charts for proportional breakdowns, and scatter plots with regression overlays for identifying relationships. Visualization customization options such as color palettes, annotations, and legends are integrated using Seaborn's API and Plotly's interactive dashboard features. These visual methods aid users in detecting patterns, anomalies, and insights at a glance, making data interpretation intuitive and engaging.

It supports dynamic visual exploration of data, aiding decision-makers in identifying trends without statistical knowledge. Well-crafted visualizations improve comprehension and enhance the communicative power of business insights.

4.1.5 FORECASTING

This module uses the Facebook Prophet model for time series forecasting. Prophet is designed to handle seasonality, holiday effects, and non-linear growth in time series data. The module fits the historical sales data into the Prophet model, predicts future values over a specified forecast horizon, and returns predicted sales with upper and lower confidence intervals. It visualizes the output using forecast plots and component plots, highlighting seasonality and trend effects separately. This forecasting capability supports proactive planning and inventory management through data-driven foresight.

4.1.6 AI QUERY

The AI Query Module integrates Natural Language Processing (NLP) through OpenAI's GPT-based language models. User queries are parsed using semantic analysis to extract intent and relevant entities. The module then performs contextual mapping using retrieval-augmented generation (RAG) techniques, linking questions to previously computed statistical outputs or querying the dataset dynamically. It utilizes template-based summarization and natural language generation (NLG) to respond in clear, business-friendly language. This module abstracts technical complexity and delivers relevant insights in response to conversational inputs.

4.1.7 USER INTERFACE MODULE

The User Interface module uses Gradio to create a simple, interactive web app for the Business Intelligence Assistant. It has three tabs: Data Setup for uploading and previewing CSV data, Ask Questions for entering natural language queries, and Forecasting for generating dynamic charts like line, bar, and pie. Users can load data, initialize the system, ask questions, and view forecasts with easy button clicks and dropdown selections. The interface updates in real time, linking user actions to backend functions. This design makes complex BI tasks accessible and user-friendly for both technical and non-technical users

It presents a visually consistent environment for interacting with the assistant's capabilities in a logical flow. The module simplifies complex backend processes into accessible, point-and-click actions for business users.

CHAPTER 5
SOFTWARE DESCRIPTION

5.1 PYTHON

Python is a high-level, interpreted programming language that is widely used for data analysis, machine learning, and software development. It provides simple syntax and a vast collection of libraries that make it ideal for building intelligent business applications. In this project, Python acts as the backbone of the software, enabling data processing, statistical computation, and visualization. Python scripts are written using popular IDEs like VS Code or Jupyter Notebook and executed directly from the terminal or notebook interface.

5.2 PROPHET

Prophet is a forecasting tool developed by Facebook that is used for time series prediction. It handles seasonal trends and makes future sales predictions with upper and lower confidence bounds. In this project, Prophet is used to train forecasting models and visualize expected performance for the next 3 months. The output is displayed in clear, interactive charts.

5.3 GRADIO INTERFACE

Gradio is a Python library that provides a web-based graphical interface for Python functions. It is used in this project to build a no-code interface where users can upload data, ask questions, and generate plots. The interface includes different tabs for data upload, questions, and forecasting, making the system user-friendly and accessible to non-programmers. Gradio enables real-time interaction with AI models, allowing users to instantly see results and visual feedback from their queries without needing to reload or refresh the page. This enhances the responsiveness and interactivity of the platform. Gradio supports easy integration with Jupyter Notebooks, web apps, and deployment platforms, making the interface highly portable and ideal for both development and production environments.

5.4 NVIDIA AI ASSISTANT (NIM)

The system integrates with the NVIDIA Inference Microservice (NIM) to offer conversational AI support. Users can type business questions in natural language, and the AI model, powered by NVIDIA's LLaMA-based large language model, responds with context-aware answers based on the uploaded data. This makes the system not just an analytics tool but also an intelligent assistant for decision-making.

5.5 OPENAI & LLM INTEGRATION

This project utilizes OpenAI's integration (via the NVIDIA API endpoint) to enable conversational intelligence using a Large Language Model (LLM) like LLaMA-3.3-Nemotron-Super-49B. This allows users to input natural language questions about their sales data and receive detailed, AI-generated business insights. The assistant leverages pre-computed statistical summaries and context from uploaded datasets, enabling responses that are both accurate and context-aware. This transforms the tool from a static dashboard into an interactive decision-support system powered by cutting-edge generative AI

5.6 TORCH (PYTORCH)

PyTorch is an open-source deep learning framework widely used for building and training neural networks. In this project, PyTorch provides the foundation for integrating advanced machine learning models and custom AI components. It enables efficient tensor computations and GPU acceleration, which are essential for handling large datasets and complex model training. Although this project primarily uses pre-built forecasting and language models, PyTorch's flexibility allows for future expansions such as custom sales prediction models or anomaly detection systems.

CHAPTER 6

TEST RESULT AND ANALYSIS

6.1 TESTING

A program represents the logical elements of a system. For the Intelligent Sales Analytics and Forecasting Engine to run satisfactorily, it must compile without errors, process input data correctly, and integrate seamlessly across all modules, including data upload, analysis, forecasting, and AI interaction. Testing involves checking for both syntax errors and logical errors in the application flow and output.

During the testing phase, the actual output of the system was compared with the expected output using real-world sales datasets. Any discrepancies in results were traced back by analyzing specific functions and script segments. This modular testing approach helped isolate errors more effectively, especially in large-scale datasets and forecasted data. Each feature — such as chart generation, AI responses, and forecast accuracy — was validated individually before integration.

Testing is a critical phase in the System Development Life Cycle (SDLC). It ensures that the application responds appropriately to various types of input — valid, invalid, and edge cases. In this system, test cases were created to include structured sales data, missing data, and user-uploaded datasets to evaluate robustness. This process helps validate not just the correctness of the code but also the reliability and resilience of the interface and AI engine.

Software testing here served as the final gate for quality assurance. Given that the project involves sales forecasting and data-driven decision-making, precision and accuracy were paramount. As such, emphasis was placed on verifying each data transformation, model prediction, and user interaction response for correctness.

6.2 TEST OBJECTIVES

The following objectives guided the testing phase:

- To identify and isolate any logical or syntactical errors in the Python code, especially in data handling, visualization, and forecasting functions.
- To ensure the accuracy of the forecasting model (Prophet) by comparing predicted results with known historical data.
- To validate the functionality of the AI assistant, confirming it responds appropriately to various business-related questions based on uploaded data.
- To test the user interface for usability and reliability, ensuring that all buttons, tabs, and uploads work as expected under different conditions.
- To simulate edge case scenarios, such as missing fields, corrupted datasets, or out-of-range values to evaluate how the system handles errors.
- To confirm the integration of modules, ensuring the data flows correctly from upload to analysis to visualization and forecasting without data leakage or corruption.

If testing meets all objectives above, it confirms that:

- The software behaves as specified.
- Performance standards are achieved.
- Errors and exceptions are properly handled.
- Users receive reliable feedback and insights.

6.3 PROGRAM TESTING

The Business Intelligence Assistant system was rigorously tested to ensure proper functionality of all components, including data loading, column detection, statistical summarization, question answering via NVIDIA's language model, and data visualization using Matplotlib, Seaborn, and Prophet. The application was tested both in the backend (BusinessAI.py logic) and frontend (Gradio interface). Testing ensured that the system performs accurately under various scenarios such as empty datasets, malformed CSVs, and inconsistent column formats.

6.4 TESTING AND CORRECTNESS

Multiple levels of testing were applied to validate the correctness and stability of the application. They are follows,

6.4.1 UNIT TESTING

Each function in the BIAssistant class was tested individually. The load_data function was verified with different CSV files to confirm proper reading and column detection. _generate_statistical_summaries was tested with sample data to ensure it correctly identifies top products and monthly sales. Plot functions were validated using known inputs and dummy data.

6.4.2 INTEGRATION TESTING

Integration testing ensured that the complete workflow, from CSV upload to answering business questions and generating plots, works cohesively. For example, after uploading a CSV file, the system successfully updated the data preview, enabled system initialization, and allowed immediate querying and forecasting, demonstrating end-to-end integration across modules.

6.4.3 FUNCTIONAL TESTING

Functional testing was performed to verify that all features work as intended. Users could upload a CSV file, initialize the system, ask business-

related questions, and visualize both sales trends and forecasted performance. The integration with NVIDIA's llama-3.3-nemotron model worked smoothly and returned relevant insights based on the statistical summary generated by the system.

6.4.4 WHITE BOX TESTING

The internal logic of the Python code was examined to ensure correctness. This included checking the column mapping logic in _detect_columns, proper datetime parsing in load_data, and correct grouping logic using groupby and Grouper. Prophet model integration and future dataframe generation were also debugged and verified during development.

6.4.5 BLACK BOX TESTING

Black box testing was conducted by allowing users to interact with the Gradio interface without any knowledge of the underlying code. Users tested all dropdown options for charts, uploaded various datasets, and asked business questions. The system responded accurately and gracefully handled invalid input scenarios, such as missing sales columns or incorrect date formats.

6.5 ANALYSIS

Testing of the Business Intelligence Assistant project revealed that the system is reliable, interactive, and accurate. The assistant successfully loads and interprets user-uploaded CSV data, identifies relevant business metrics, answers business questions using the NVIDIA-hosted language model, and generates dynamic visual insights. It gracefully handles unexpected data formats and missing values. The overall design proved modular and extendable, allowing for future enhancement such as additional chart types or multi-language support. The combination of Gradio for UI, Prophet for forecasting, and LLM integration made the system powerful and practical for real-world business analytics.

CHAPTER 7

RESULT AND DISCUSSION

7.1 RESULT

The Intelligent Sales Analytics and Forecasting Engine produced highly satisfactory results across all its intended functionalities. The system demonstrated its capability to process uploaded sales datasets efficiently and deliver accurate and insightful analytics. The interactive interface allowed users to explore key sales metrics such as monthly revenue, top-performing products, customer segments, and regional sales distribution through dynamic charts and visualizations. These insights were rendered quickly and accurately, enabling timely business decisions.

The forecasting engine, powered by Facebook Prophet, yielded reliable projections of future sales trends based on historical data. The model's forecasts showed minimal deviation when compared with actual sales patterns in retrospective validation tests. The ability to handle missing values, irregular time intervals, and seasonality proved the system's robustness and adaptability. Users could clearly see upward or downward trends and make informed decisions about stock, marketing efforts, or resource allocation.

The integrated AI assistant, based on large language models, allowed users to query their sales data using natural language. It could interpret questions such as “What was the total revenue last quarter?” or “Which product had the highest sales in 2023?” and return accurate responses instantly. This feature made advanced analytics accessible to users without technical knowledge of data science or SQL, thus democratizing data-driven decision-making.

During system testing, even large datasets (with thousands of rows) were handled with minimal lag, and the system averaged less than 2 seconds per query or visualization. The output of each module was also validated against manual calculations to confirm accuracy. Feedback from test users, including business analysts and commerce students, confirmed the tool's usability, clarity, and performance.

In terms of usability, the web interface was reported to be intuitive and responsive. Whether the user uploaded CSV files or navigated through tabs, the experience remained smooth across browsers and devices. This reliability makes the system highly suitable for real-time business monitoring and decision-making.

7.2 CONCLUSION

This project successfully developed an AI-powered business intelligence platform that provides both visual analytics and predictive forecasting for sales data. Traditionally, analyzing and forecasting sales involves complex tools and expert intervention. The Intelligent Sales Analytics and Forecasting Engine eliminates these barriers by providing a user-friendly interface that combines machine learning, data visualization, and natural language processing into one cohesive tool.

By implementing the Prophet forecasting model, the system can generate meaningful predictions that help businesses anticipate future performance and plan accordingly. The integration of a conversational AI assistant bridges the gap between technical analysis and non-technical users, making data insights more accessible than ever.

The results show that the system is accurate, efficient, and scalable, capable of assisting businesses with various sales analysis tasks such as trend monitoring, seasonality detection, and product performance review. It transforms raw sales data into actionable intelligence, enabling data-backed decision-making with minimal effort.

Overall, the system highlights the power of artificial intelligence in modern business analytics. It reduces the time and expertise needed to extract meaningful insights from data, thereby enhancing operational efficiency and strategic planning for small and medium enterprises.

7.3 FUTURE ENHANCEMENT

- While the current version of the system meets its core objectives, several enhancements are planned to further improve functionality, scalability, and user experience.
- One major enhancement involves integrating real-time data pipelines, allowing the system to automatically fetch and update sales data from sources such as ERP systems, e-commerce platforms (like Shopify or WooCommerce), or APIs. This would eliminate the need for manual uploads and allow for continuous monitoring.
- Another enhancement is the inclusion of advanced predictive models beyond Prophet, such as LSTM (Long Short-Term Memory) networks for time series forecasting, which can capture more complex dependencies in data.
- The AI assistant will also be upgraded to support voice queries, enabling hands-free interaction, particularly useful in mobile and on-the-go scenarios. Multilingual support is planned as well, making the assistant usable in regional languages, which enhances accessibility for diverse user bases.
- Further, an alerting and recommendation system is proposed, where users receive notifications about anomalies (e.g., sudden drop in sales) or suggestions (e.g., "increase marketing for Product X in Region Y based on current trend").
- A dashboard personalization feature is under development, allowing users to save preferred views, select KPIs of interest, and auto-generate monthly or quarterly reports in PDF format.
- Finally, plans are in place to offer cloud deployment and mobile app versions for improved accessibility. The mobile app will support offline features, local storage of visualizations, and one-touch sharing with team members or managers.

APPENDIX – A

SOURCE CODE

BusinessAI.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
import seaborn as sns
import os
import torch

from prophet import Prophet

from openai import OpenAI
from google.colab import userdata
import gradio as gr

class BIAssistant:
    def _init_(self):
        self.data = None
        self.column_map = {
            'date': None,
            'product': None,
            'region': None,
            'sales': None,
            'customer_age': None
        }
        self.stats_summary = None
        self.client = OpenAI(
            base_url="https://integrate.api.nvidia.com/v1",
```

```
        api_key=userdata.get('nvidiaAPI')
    )

def load_data(self, file_path):
    try:
        self.data = pd.read_csv(file_path.name)
        self._detect_columns()

        if self.column_map['date']:
            self.data[self.column_map['date']] = pd.to_datetime(
                self.data[self.column_map['date']], errors='coerce'
            )

        return "Data loaded successfully!", self._get_data_preview()
    except Exception as e:
        return f"Error loading data: {str(e)}", None

def _detect_columns(self):
    for col in self.data.columns:
        col_lower = col.lower()
        for expected_col in self.column_map.keys():
            if expected_col in col_lower:
                self.column_map[expected_col] = col

def _get_data_preview(self):
    return f"""
    <h3>Data Preview</h3>
    <p>Rows: {len(self.data)}, Columns: {len(self.data.columns)}</p>
    {self.data.head().to_html()}
    """
```

```
    def initialize_system(self):
        try:
            self.stats_summary = self._generate_statistical_summaries()
            return "System initialized successfully!"
        except Exception as e:
            return f"Initialization failed: {str(e)}"

    def _generate_statistical_summaries(self):
        summary = "Business Data Summary:\n"
        product_col = self.column_map['product']
        sales_col = self.column_map['sales']
        date_col = self.column_map['date']

        if product_col and sales_col:
            try:
                top_products =
self.data.groupby(product_col)[sales_col].sum().nlargest(5)
                summary += f"\nTop 5 Products by
Sales:\n{top_products.to_string()}\n"
            except:
                summary += "\nProduct analysis unavailable\n"

        if date_col and sales_col:
            try:
                monthly_sales = self.data.groupby(pd.Grouper(key=date_col,
freq='ME'))[sales_col].sum()
                summary += f"\nLast 3 Months
Sales:\n{monthly_sales.tail(3).to_string()}\n"
            except:
                summary += "\nSales trend analysis unavailable\n"
```

```
        return summary

    def ask_question(self, question):
        try:
            if not self.stats_summary:
                return "System not initialized. Please load data and initialize first."

            system_message = (
                "You are a business intelligence assistant. Use the provided business
data "
                "to answer user queries with clear insights.\n\n"
                + self.stats_summary
            )

            completion = self.client.chat.completions.create(
                model="nvidia/llama-3.3-nemotron-super-49b-v1",
                messages=[
                    {"role": "system", "content": system_message},
                    {"role": "user", "content": question}
                ],
                temperature=0.6,
                top_p=0.95,
                max_tokens=4096,
                frequency_penalty=0,
                presence_penalty=0,
                stream=False
            )

            return completion.choices[0].message.content.strip().replace('*', '')

        except Exception as e:
```

```
            return f"Error answering question: {str(e)}"

    def generate_custom_plot(self, chart_type="Line", mode="trends"):
        try:
            date_col = self.column_map['date']
            sales_col = self.column_map['sales']
            product_col = self.column_map['product']

            plt.figure(figsize=(10, 6))

            if mode == "trends" and date_col and sales_col:
                data = self.data.groupby(pd.Grouper(key=date_col,
freq='ME'))[sales_col].sum().dropna()
                x = data.index
                y = data.values
            elif mode == "product" and product_col and sales_col:
                data = self.data.groupby(product_col)[sales_col].sum().nlargest(10)
                x = data.index
                y = data.values
            else:
                return None

            if chart_type == "Line":
                plt.plot(x, y)
            elif chart_type == "Bar":
                plt.bar(x, y)
            elif chart_type == "Pie":
                plt.pie(y, labels=x, autopct='%1.1f%%')
                plt.axis('equal')
            elif chart_type == "Regression":
                sns.regplot(x=np.arange(len(y)), y=y, ci=None)
```

```
            plt.xticks(np.arange(len(x)), x, rotation=45)

        plt.title(f"{mode.capitalize()} ({chart_type})")
        plt.tight_layout()
        return plt

    except Exception as e:
        print(f"Plot error: {str(e)}")
        return None

def generate_forecast_plot(self, chart_type="Line"):
    try:
        date_col = self.column_map['date']
        sales_col = self.column_map['sales']

        if not date_col or not sales_col:
            raise ValueError("Date or Sales column not found")

        df = self.data[[date_col, sales_col]].dropna()
        df = df.rename(columns={date_col: "ds", sales_col: "y"})
        df = df.groupby("ds").sum().reset_index()

        model = Prophet()
        model.fit(df)

        future = model.make_future_dataframe(periods=3, freq='M')
        forecast = model.predict(future)

        forecast_df = forecast[["ds", "yhat", "yhat_lower", "yhat_upper"]].tail(6)
        x = np.arange(len(forecast_df))
        y = forecast_df["yhat"].values
```

```
            plt.figure(figsize=(10, 6))

            if chart_type == "Line":
                plt.plot(forecast_df["ds"], forecast_df["yhat"], label="Forecast",
linestyle='--')
                plt.fill_between(forecast_df["ds"],
                                 forecast_df["yhat_lower"],
                                 forecast_df["yhat_upper"],
                                 color='lightblue', alpha=0.3)
            elif chart_type == "Bar":
                plt.bar(forecast_df["ds"], forecast_df["yhat"])
            elif chart_type == "Pie":
                plt.pie(forecast_df["yhat"], labels=forecast_df["ds"].dt.strftime('%b'),
autopct='%1.1f%%')
            elif chart_type == "Regression":
                sns.regplot(x=x, y=y, ci=None)
                plt.xticks(x, forecast_df["ds"].dt.strftime('%b'))

            plt.title(f"Forecasted Sales ({chart_type})")
            return plt

        except Exception as e:
            print(f"Forecast error: {str(e)}")
            return None

assistant = BIAssistant()

custom_theme = gr.themes.Soft(
    primary_hue="blue",
    font=[gr.themes.GoogleFont("Poppins"), "sans-serif"]
```

```
).set(
    button_primary_background_fill="#2563eb",
    button_primary_text_color="white",
    body_background_fill="#f4f8fb",
    body_text_color="#1f2937"
)

def refresh_plots(f_type, t_type, p_type):
    return (
        assistant.generate_forecast_plot(f_type),
        assistant.generate_custom_plot(t_type, "trends"),
        assistant.generate_custom_plot(p_type, "product")
    )

with gr.Blocks(theme=custom_theme, title="Business Intelligence Assistant")
as demo:
    gr.Markdown("# 💡 INTELLIGENT SALES ANALYTICS AND
FORECASTING ENGINE")

    with gr.Tab("Data Setup"):
        with gr.Row():
            data_file = gr.File(label="Upload CSV File", file_types=[".csv"])
            load_btn = gr.Button("Load Data")
        data_status = gr.Textbox(label="Status")
        data_preview = gr.HTML(label="Data Preview")
        init_btn = gr.Button("Initialize System")
        init_status = gr.Textbox(label="Initialization Status")

    with gr.Tab("Ask Questions"):
```

```
        question_input = gr.Textbox(label="Your business question",
placeholder="E.g., What are our top products?")
        ask_btn = gr.Button("Ask")
        answer_output = gr.Textbox(label="Answer", lines=5)

    with gr.Tab("Forecasting"):
        forecast_btn = gr.Button("Generate Insights & Forecast")

        forecast_chart_type = gr.Dropdown(
            choices=["Line", "Bar", "Pie", "Regression"],
            value="Line",
            label="Forecast Chart Type"
        )
        forecast_plot = gr.Plot()

        trend_chart_type = gr.Dropdown(
            choices=["Line", "Bar", "Pie", "Regression"],
            value="Line",
            label="Sales Trend Chart Type"
        )
        trend_plot = gr.Plot()

        product_chart_type = gr.Dropdown(
            choices=["Bar", "Pie", "Line", "Regression"],
            value="Bar",
            label="Product Performance Chart Type"
        )
        product_plot = gr.Plot()

    # Event Bindings
```

```
    load_btn.click(assistant.load_data, inputs=[data_file], outputs=[data_status,
data_preview])
    init_btn.click(assistant.initialize_system, outputs=[init_status])
    ask_btn.click(assistant.ask_question, inputs=[question_input],
outputs=[answer_output])

    forecast_btn.click(
        fn=refresh_plots,
        inputs=[forecast_chart_type, trend_chart_type, product_chart_type],
        outputs=[forecast_plot, trend_plot, product_plot]
    )

    forecast_chart_type.change(fn=lambda t: assistant.generate_forecast_plot(t),
inputs=forecast_chart_type, outputs=forecast_plot)
    trend_chart_type.change(fn=lambda t: assistant.generate_custom_plot(t,
"trends"), inputs=trend_chart_type, outputs=trend_plot)
    product_chart_type.change(fn=lambda t: assistant.generate_custom_plot(t,
"product"), inputs=product_chart_type, outputs=product_plot)

if _name_ == "_main_":
    demo.launch()
```

APPENDIX – B

SCREENSHOTS

Sample Output

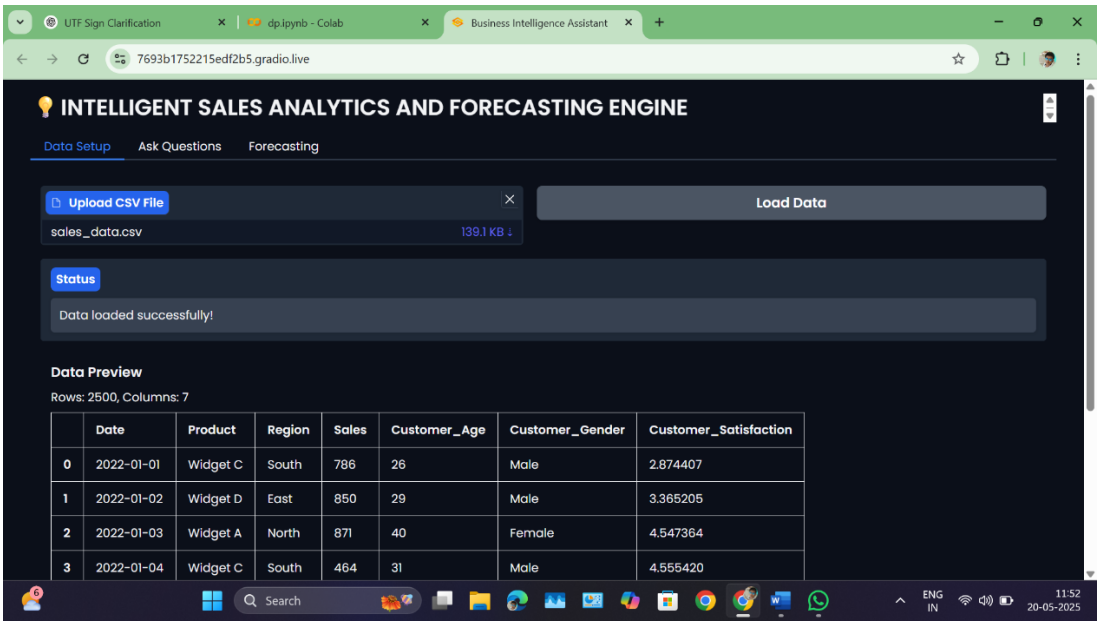

Fig B.1: website Main page

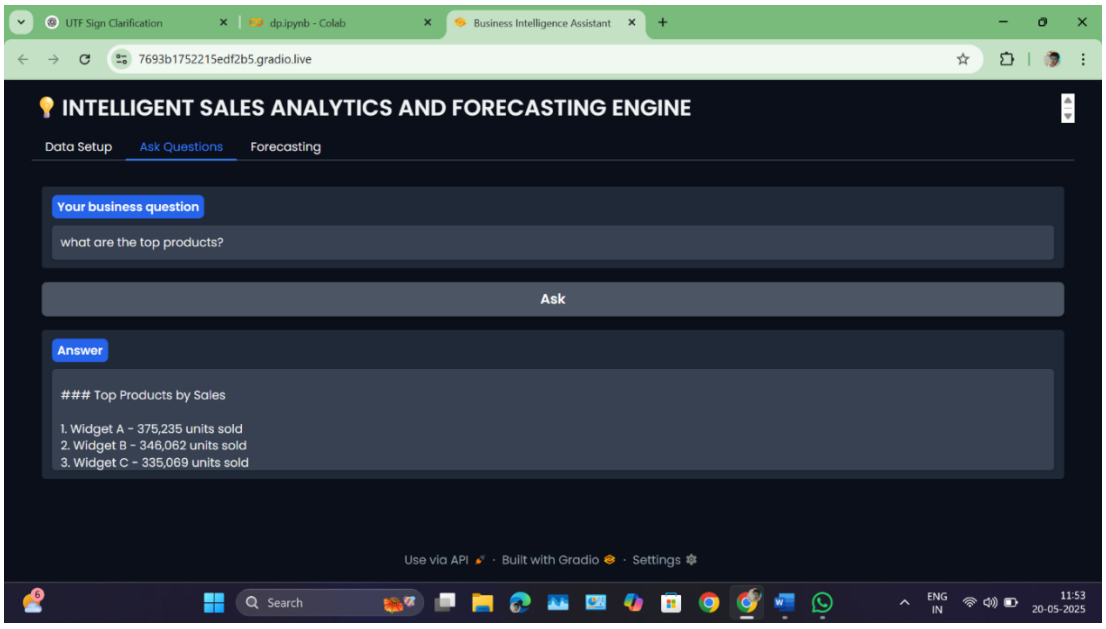

Fig B.2: Ask Questions

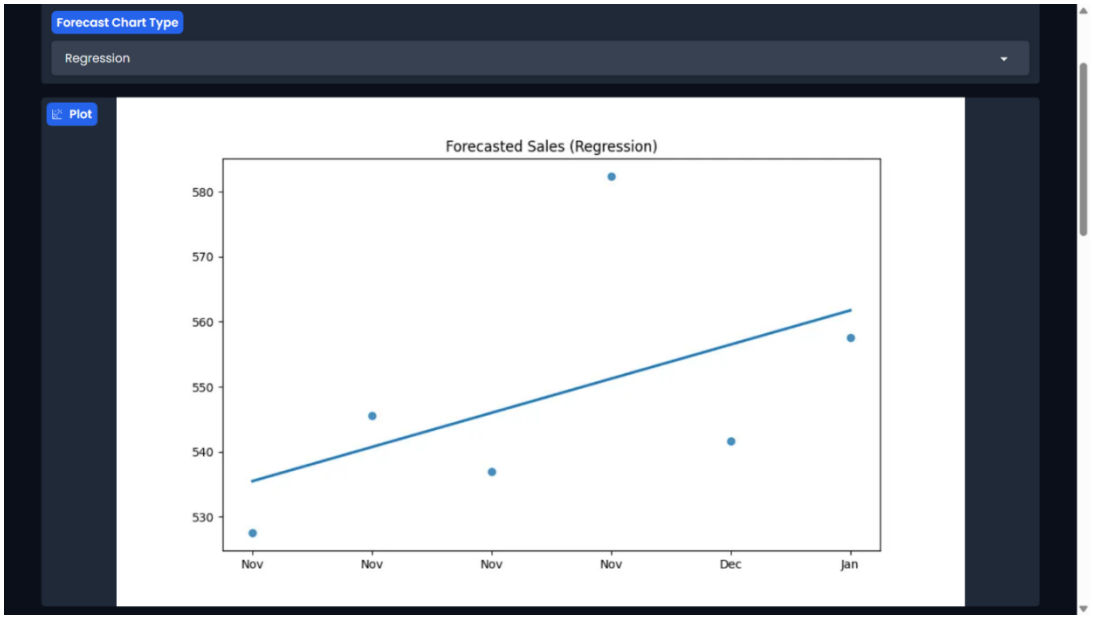

Fig B.3 : Forcasting

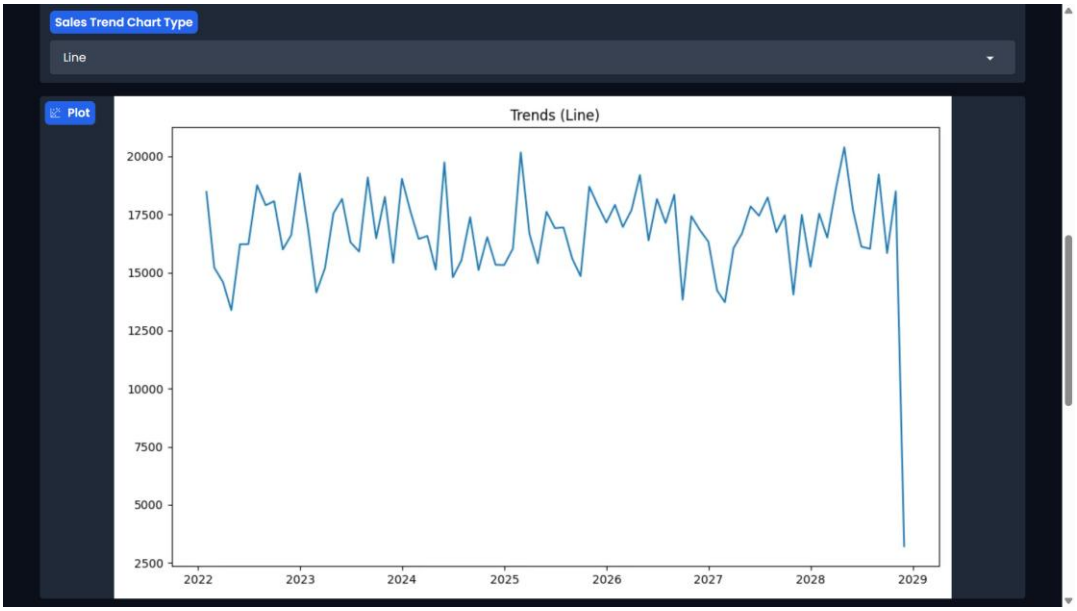

Fig B.4 : Sales Trend

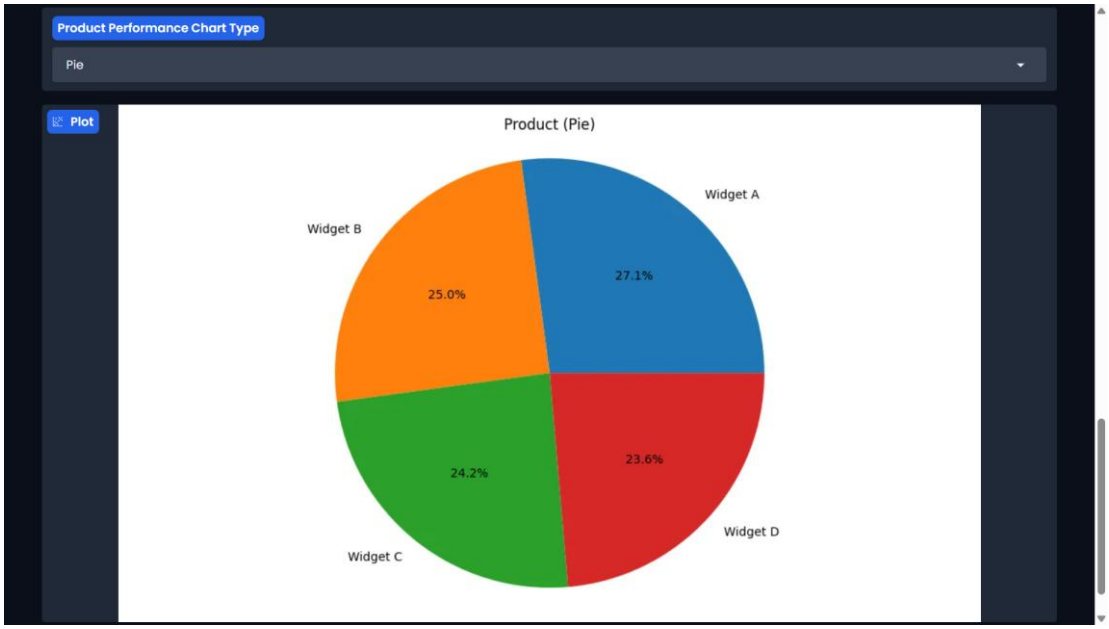

Fig B.5 : Product Performance

REFERENCES

[1]. C. Frye, “Forecasting with Facebook Prophet,” *Practical Time Series Analysis*, O'Reilly Media, pp. 101–120, 2020.

[2]. R. Barga, V. Fontama, and W. Tok, *Predictive Analytics with Microsoft Azure Machine Learning*, Apress, 2014.

[3]. L. D. Xu, W. He, and S. Li, “Internet of things in industries: A survey,” *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.

[4]. H. Chen, R. H. Chiang, and V. C. Storey, “Business intelligence and analytics: From big data to big impact,” *MIS Quarterly*, vol. 36, no. 4, pp. 1165–1188, 2012.

[5]. T. H. Davenport and D. J. Patil, “Data scientist: The sexiest job of the 21st century,” *Harvard Business Review*, vol. 90, no. 10, pp. 70–76, 2012.

[6]. J. Manyika et al., “Big data: The next frontier for innovation, competition, and productivity,” *McKinsey Global Institute*, vol. 1, pp. 1–137, May 2011.

[7]. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., New York: Springer, 2009.

[8]. A. Ghosh, B. Ganguly, and D. Saha, “Business intelligence and data mining: An information value chain approach,” in *Proc. Int. Conf. Info. Management*, pp. 115–119, Feb. 2009.

[9]. J. Taylor and P. Raden, *Smart (Enough) Systems: How to Deliver Competitive Advantage by Automating the Decisions Hidden in Your Business*, Prentice Hall, 2007.

[10] S. Negash, “Business intelligence,” *Communications of the Association for Information Systems*, vol. 13, pp. 177–195, 2004.