VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590018

A Project Report on

"TEXT-TO-SQL REPORT GENERATION FOR WOOCOMMERCE"

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by

Ayman Shaikh	**2VD20CS010**
Neha Mule	**2VD20CS021**
Prajwal Naik	**2VD20CS023**
Raksha Hebbar	**2VD08CS025**

Under the Guidance of

Dr. Naveenkumar T Rudrappa

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KLS VISHWANATHRAO DESHPANDE INSTITUTE OF TECHNOLOGY
HALIYAL-581329

2023-2024

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590018

A Project Report on

"TEXT-TO-SQL REPORT GENERATION FOR WOOCOMMERCE "

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by

Ayman Shaikh	**2VD20CS010**
Neha Mule	**2VD20CS021**
Prajwal Naik	**2VD20CS023**
Raksha Hebbar	**2VD20CS025**

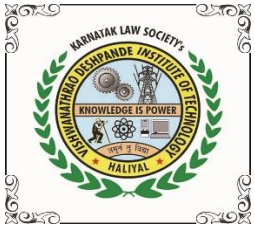
Under the Guidance of

Dr. Naveenkumar T Rudrappa

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KLS VISHWANATHRAO DESHPANDE INSTITUTE OF TECHNOLOGY
HALIYAL-581329

2023-24

KLS VISHWANATHRAO DESHPANDE
INSTITUTE OF TECHNOLOGY, HALIYAL - 581329



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate

Certified that the Project work entitled

" **TEXT-TO-SQL REPORT GENERATION FOR WOOCOMMERCE** "

is a bonafide work carried out by

Ayman Shaikh (2VD20CS010) **Prajwal Naik (2VD20CS023)**

Neha Mule (2VD20CS021) **Raksha Hebbar(2VD20CS25)**

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Computer Science and Engineering of Visvesvaraya Technological University, Belagavi,** during the year 2023-2024. It is certified that all the corrections / suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the Bachelor of Engineering degree.

Signature of the Guide	Signature of the HOD	Signature of the Principal
Dr. Naveenkumar T Rudrappa.	Dr. S. A. Sirdeshpande	Dr. V. A. Kulkarni

Name of the Examiners:	Signature with date:
1.	1.
2.	2.

ACKNOWLEDGEMENT

Task successful makes everyone happy. But the happiness will be gold without glitter if we didn't state the persons who have supported us to make it success. Success will be crowned to people who made it reality but the people whose constant guidance and encouragement made it possible will be crowned first on the eve of success.

We are grateful to our beloved principal Dr. V. A. Kulkarni for having provided us the academic environment, which nurtured our practical skills, contributing to the success of our project. We would like to thank all the teaching and non-teaching staff for their ideas and encouragement which helped us through the completion of the project.

We take this opportunity to thank Dr. S. A. Sirdeshpande, HOD of Computer Science and Engineering Department for providing the inspiration for taking the project to its completion. We wish to express our sincere gratitude towards our guide Dr. Naveenkumar T Rudrappa for their constant motivation and valuable help through the project.

We extend our sincere gratitude towards our parents who have encouraged us with their blessings to do this project successfully

ABSTRACT

Common business users within the WooCommerce ecosystem face challenges due to their limited technical expertise, particularly in Structured Query Language (SQL), which impedes their efficient interaction with databases. In response, we introduced a novel system: 'Text-to-SQL Report Generation for WooCommerce,' which leverages the power of the Gemini Pro model. This system enables users to effortlessly input queries in both Plain English and Hindi voice commands, empowering them to interact seamlessly with their WooCommerce database.By harnessing the capabilities of the Gemini Pro model, our system translates these queries into SQL commands, bridging the gap between non-technical users and database interaction.Through an intuitive graphical user interface, users receive query results in both tabular and visual graph formats, facilitating easy interpretation and analysis of data. This functionality enhances users' ability to make informed decisions and derive data-driven insights within the WooCommerce ecosystem, even without proficiency in SQL.Our project's core objective is to democratize database access and analysis for non-technical users, thereby streamlining operations and fostering data-driven decision-making within WooCommerce environments. By eliminating the need for SQL expertise and offering a user-friendly interface with multi-language support, our system represents a significant advancement towards empowering businesses to leverage their data effectively.

Table of Contents

Abstract	i
Acknowledgement	ii
List of figures	iv
Chapter 1. Introduction	1-5
1.1 Overview	1
1.2 Motivation	1
1.3 Introduction	2-5
Chapter 2. Literature Review	6-10
Chapter 3. Proposed System	11-12
3.1 Objectives	11
3.2 Expected Outcomes	12
Chapter 4. Methodology	13-15
4.1 Block Diagram	13-14
4.2 Sequence Diagram	15
Chapter 5. System Implementation	16-20
Chapter 6. Results and Discussion	21-27
Chapter 7. Conclusion and Scope of Future work	28-29
References	31-32
Appendix A: Hardware Requirements	33
Appendix B: Software Requirements	33

List of Figures

Figure No.	Figure Name	Page No.
4.1	Intelligent Bilingual Query Processing Architecture	13
4.2	Seamless Data Interaction Frame Work	15
6.1	User Interface	21
6.2	User Input	22
6.3	Query generated by Generative AI	22
6.4	Structured Output	23
6.5	Speech Recognition and Translation	24
6.6	Output for the query through mic	25
6.7	Downloaded report in the PDF Format	26
6.8	Data in the form of Line Chart	27

Chapter 1

INTRODUCTION

1.1 Overview

The idea behind "Text-to-SQL Report Generation for WooCommerce" is to bridge the gap between non-technical users and database interaction within the WooCommerce ecosystem. By leveraging advanced natural language processing technology, specifically the Gemini Pro model, the system allows users to effortlessly input queries in Plain English or Hindi voice commands. These queries are then translated into SQL commands, enabling seamless interaction with the WooCommerce database. Through an intuitive graphical user interface, users can access query results in tabular and visual graph formats, facilitating easy interpretation and analysis of data. This innovative approach democratizes database access and analysis, empowering businesses to make data-driven decisions without requiring expertise in SQL.

1.2 Motivation

The motivation for this idea stemmed from recognizing the challenges faced by common business users within the WooCommerce ecosystem due to their limited technical expertise, particularly in SQL. These users often struggle to interact efficiently with databases, hindering their ability to derive meaningful insights from their data. In response to this challenge, our team sought to develop a solution that would empower non-technical users to interact seamlessly with their WooCommerce database, without the need for SQL expertise. Drawing inspiration from advancements in natural language processing, particularly the capabilities of the Gemini Pro model, the idea emerged to enable users to input queries in Plain English or Hindi voice commands. The goal was to create a system that not only translated these queries into SQL commands but also presented query results in a user-friendly format, facilitating easy interpretation and analysis of data. By democratizing database access and analysis in this way, we aimed to streamline operations and foster data-driven decision-making within the WooCommerce ecosystem, ultimately empowering businesses to leverage their data effectively for growth and success.

1.3 Introduction

Long before we wrote things down, the primary mode of communication was through speech. Communities relied on oral traditions, passing down information, stories, and cultural knowledge from one generation to the next. This rich tradition of spoken words was central to human interaction and the transmission of collective wisdom. The shift from oral communication to written text marked a transformative leap in human communication. The transition empowered societies to preserve information beyond the confines of immediate spoken interaction. The advent of written text allowed for the recording of knowledge, enabling its longevity and wider distribution. Over the millennia, diverse scripts emerged globally, giving rise to the multitude of written languages we encounter today. This linguistic diversity fostered cultural exchange, enabling the sharing of ideas, literature, and scientific knowledge.

The transition from oral communication to written text marked a transformative leap for humanity. This shift empowered societies to capture the intangible, preserving knowledge in a tangible and lasting form. The first strokes of primitive symbols etched on cave walls and the birthing of early pictorial languages signalled the dawn of a new era. The proliferation of written texts catalysed cultural exchange, enabling the dissemination of ideas, literature, and scientific knowledge across vast distances.

Computers have their own language, a series of 0’s and 1’s that magically arrange themselves into words and pictures. The story of writing and scripts is not merely about symbols on paper or screens; it is a testament to humanity's timeless desire to keep their stories alive from ancient campfires to the pixels on our screens today. It is a journey through time, where the essence of human expression continues to evolve in the grand tapestry of our shared history. In the mid-20th century, the first computers emerged as colossal, room-filling machines with limited capabilities. The dawn of the digital era had arrived, and with it, the need for a language that could instruct these electronic marvels. Thus, the genesis of computer languages began. The earliest computer languages were low-level and closely tied to the hardware, demanding a deep understanding of the machine's architecture. Assembly languages, which used symbolic codes to represent machine instructions, were the pioneers in this realm. However, programming in assembly languages was a complex and laborious task, requiring meticulous attention to detail.

The demand for more user-friendly programming languages grew, high-level languages emerged. Fortran, developed in the 1950’s, was one of the first high-level languages designed to simplify scientific and

engineering computations. Cobol followed suit, aiming to make business data processing more accessible. The evolution continued with the introduction of programming languages like Lisp, which focused on symbolic processing and artificial intelligence. The 1960’s witnessed the birth of Beginner's All-purpose Symbolic Instruction Code (BASIC), opening programming to a wider audience. In the 1970’s, the programming landscape saw the advent of C, a powerful and versatile language that laid the groundwork for many subsequent languages. C's influence extended to the creation of the Unix operating system, shaping the foundations of modern computing.

Computers became more ubiquitous, the need for standardized, portable languages became apparent. This led to the development of languages like Pascal and Ada, designed for specific applications and industries. Amidst this dynamic landscape, a revolutionary language known as SQL (Structured Query Language) emerged in the 1970’s. SQL was specifically designed for managing and manipulating relational databases. Its syntax was geared towards expressing complex queries, making it a cornerstone in database management. SQL’s power lay in its ability to seamlessly interact with databases, allowing users to retrieve, update, and manipulate data with ease. Its standardized nature ensured compatibility across different database management systems, contributing to its widespread adoption.The journey from the invention of scripts to the evolution of computer languages, culminating in the birth of SQL, reflects the relentless pursuit of enhancing human-computer interaction. From the early days of assembly languages to the high-level abstractions of modern programming languages, each step in this journey has been marked by a commitment to making computers more accessible and powerful tools for human expression and problem-solving.

Users found themselves grappling with the nuances of SELECT, FROM, WHERE, and other SQL keywords. The need for a more accessible approach to database queries became evident. Enter the era of Artificial Intelligence (AI), a realm where machines learned to understand and interpret human language. Leveraging Natural Language Processing (NLP) techniques, which enable computers to comprehend, interpret, and generate human-like language, a new paradigm in database interaction emerged. As the journey unfolded into the realms of artificial intelligence and Natural Language Processing (NLP), it became increasingly evident that a transformative breakthrough had occurred in the way humans interacted with databases. The recognition of this breakthrough led to a broader realization — the potential to build systems that could seamlessly integrate NLP to process and convert text to-SQL queries. The implications were profound, as it meant that individuals with diverse backgrounds, including those without a technical or programming acumen, could now effortlessly communicate with databases.

The process became akin to having a conversation with the database, making the interaction more akin to asking a question than executing a command. Consider a scenario where a business analyst, without prior SQL knowledge, needed to extract insights from a sales database. Instead of navigating the complexities of SQL syntax, they could simply articulate their query in plain language, such as "What were the total sales for each product category last month?" The NLP-powered system would process this input, generating a SQL query that accurately captured the user's intent. Behind the scenes, the NLP model would identify the entities in the sentence (total sales, product category, last month) and map them to the corresponding database fields. The resulting SQL query might resemble: "SELECT ProductCategory, SUM(SalesAmount) FROM Sales WHERE Date BETWEEN 'start_of_last_month' AND 'end_of_last_month' GROUP BY Product Category."

The synergy between NLP and SQL created a bridge between technical and non-technical users, fostering a more inclusive approach to data exploration and analysis. As these systems evolved, they found applications in diverse domains, from business intelligence tools and data analytics platforms to educational environments where individuals could effortlessly explore and glean insights from databases without grappling with the intricacies of traditional SQL syntax. In the ever-evolving narrative of communication and technology, the story of WooCommerce, a WordPress plugin tailored for e-commerce, adds a compelling layer to the journey from oral communication to the contemporary innovations of text-to-SQL conversion using NLP.WooCommerce, designed to enhance WordPress functionality for online stores, entered the scene as a user-friendly platform with an open-source nature, contributing to its rapid adoption.

The concept of automated report generation comes to the forefront, streamlining the laborious, manual task of creating reports. This aligns with the principles of user-friendly interactions and the overarching goal of empowering users, regardless of their technical expertise. WooCommerce's success story further accentuates this theme, as it symbolizes the dynamic interplay between language, technology, and the evolution of communication. The continuous evolution of WooCommerce reflects the adaptability required in the digital landscape, mirroring the need for intuitive tools that cater to a broad audience. As the story continues, the integration of NLP into text-to-SQL systems adds a layer of sophistication.

Users of WooCommerce and similar platforms can now leverage NLP-driven systems to not only manage their online stores seamlessly but also interact with databases effortlessly. The combination of intuitive e-commerce solutions like WooCommerce and advanced NLP-driven text-to-SQL technology marks a

paradigm shift towards user-friendly database interactions, enabling businesses to harness the power of data more efficiently .In essence, the journey from oral communication to contemporary innovations like text-to-SQL using NLP is a dynamic saga, where the success stories of platforms like WooCommerce underscore the importance of adapting to technological advancements. As we navigate this evolving landscape, the fusion of language, technology, and user empowerment continues to shape the way we communicate and conduct business in the digital age.

The ability to seamlessly transition from expressing queries in plain language to generating insightful analytics charts and reports exemplifies the convergence of technology and humancentric design. This amalgamation signifies a paradigm shift where data becomes not just a repository of information but a dynamic force that empowers individuals to navigate, understand, and leverage the vast landscape of information at their fingertips. The story persists, with each technological stride enhancing our ability to unlock the potential within the vast troves of data that surround us. The fusion of e-commerce platforms like WooCommerce with advanced technologies underscores the potential for a future where accessing, understanding, and leveraging data becomes an intuitive and empowering process for users across diverse backgrounds. The story continues, resonating with the spirit of progress and the endless possibilities that lie ahead in the realm of technology and communication.

Chapter 2

LITERATURE REVIEW

The research paper “The Study on Natural Language Interface of Relational Databases” presents a WordNet based NLIDBs[1] framework to improve natural language query accuracy and reliability. This framework allows users to interact with databases in natural language, eliminating the need for SQL. It uses WordNet, Ontology, Discourse Representation Structure (DRS)[1], and template techniques to translate queries into SQL. The Ontology-based knowledge base reduces ambiguity, while WordNet enhances search accuracy. DRS ensures precise query transformation, and templates efficiently convert DRS to SQL[1]. Testing with the SQL Server Pubs database shows high efficacy in single-table queries, significantly increasing recognition rates.

This paper has identified a common approach where applications offer a form-based interface, allowing users with limited knowledge of database schemas or structured query language to access and interact with databases. However, due to the lack of understanding of semantic connections among form fields, providing users with a textual explanation of the generated query becomes essential. The paper introduces a novel graph-based solution to the query translation problem. Our experiments demonstrate promising results, especially with an algorithm that captures crucial semantic[2] associations between database entities, offering an abstraction layer over the database schema. This work represents a pioneering effort in the realm of structured query translation, presenting a fertile ground for further research.

This research paper “A Rule Based Approach for NLP Based Query Processing” tackles the formidable challenge of developing Natural Language Interfaces to Databases (NLIDBs) [1] tailored for non expert users. It acknowledges the complexity of translating natural language semantics into the structured query language (SQL) used for database access. The proposed NLIDB system comprises two integral components: the Linguistic Component, responsible for translating natural language into formal queries, and the Database Component, managing traditional database management functions. The authors present a rule based system utilizing Context Free Grammar (CFG) [3] to process natural language queries. The workflow encompasses phases like user interface interaction, word check, removal of excess words, tokenization, and mapping to CFG rules for SQL query generation.

“Domain Specific Query Generation from Natural Language Text” presents an automated method for generating Structure Query Language (SQL) [1] from Natural Language Text, specifically focusing on software requirements specifications. Accurate specifications are crucial in software development, emphasizing the need to address errors during the translation of English text into SQL. The paper identifies challenges such as discourse, semantic intricacies, and negation problems within software specifications. The proposed approach comprises six steps: inputting English queries, preprocessing query text, semantic disambiguation, domain-specific[4] information extraction using ontology, mapping to SQL syntax, and generating SQL queries. The framework utilizes lexical, syntactic, and semantic analyses to extract relevant information and map it to SQL syntax. Performance evaluation metrics include recall, precision, and F-measure, with results indicating promising performance, with an average precision of 84.02% and a recall of 75.4% for Natural Language[4] phrases.

"TiQi: A Natural Language Interface for Querying Software Project Data," introduces TiQi[5] as a groundbreaking solution to the complexities of querying extensive software project data. TiQi[5] operates as a web-based natural language interface.It visualizes project data, transforms natural language queries into SQL, and executes them against centralized or distributed databases.The paper provides insights into TiQi's architecture, comprising TiQiEngine (backend), Web-Server, and Web-Client, with a Traceability Information Model (TIM) facilitating query formulation. Evaluation results show a promising correctness rate of 70.0% for the Isolette dataset and 62.5% for the Easy Clinic dataset. Ongoing challenges, like SQL translation errors, are acknowledged, with proposed future enhancements including machine learning for query classification and an interactive dialog for query disambiguation.

This research paper “A Review Of Different Approaches In Natural Language Interface To Databases” delves into the realm of Natural Language Interface to Database (NLIDB) [1] systems, a pivotal domain in computer science facilitating user-database interactions through natural language. NLIDBs[1] alleviate the complexity of learning database query languages like SQL, enhancing accessibility. Various NLIDB architectures, such as LADDER, CHAT-80, NaLIX, and WASP, are explored, each designed to improve the user experience in querying databases. The components of NLIDBs are outlined, encompassing linguistic and database components. NLIDBs offer user-friendly interactions, simplicity, and fault tolerance, but challenges like limited language understanding and ambiguity persist. The paper concludes by stressing the ongoing research in the NLIDB[1] field, emphasizing the necessity for algorithms to optimize NLIDB queries.

The project outlines the development of an advanced system featuring a three-tier architecture to facilitate communication between non-expert users and database applications, particularly those lacking familiarity with Structured Query Language (SQL) [3]. The system employs pattern matching and semantic techniques to convert natural language into SQL queries. Comparative analysis with an existing system demonstrates the enhanced recall value, accuracy, and precision of the proposed system. Published in BVICAM's International Journal of Information Technology (Volume 14, Issue 1, 2022, pages 437-446), the study contributes to various fields, including Language, Semantics, Dictionaries, Queries, SQL, Pattern Matching, Databases, Natural Language[3], and Data Dictionaries[7].

The paper introduces an innovative solution to simplify the conversion of Natural Language Queries (NLQ) into SQL[1] Queries, addressing the challenges faced by users unfamiliar with SQL. Targeting Training and Placement cell officers, the proposed Natural Language Processing (NLP) [8] model accommodates both text and speech inputs, utilizing speech-to-text conversion for seamless integration of spoken queries into executable SQL queries. The integration of speech recognition enhances the user experience, effortlessly transforming spoken queries into textual format and subsequently into SQL queries. The system's architecture includes a graphical user interface with options for viewing table contents, entering queries in SQL or natural language, and executing or viewing equivalent SQL queries. The proposed system is presented as a significant bridge for non-SQL-proficient[8] users, particularly in the Training and Placement domain.

"A Simple Guide to Implement Data Retrieval through Natural Language Database Query Interface (NLDQ)," introduces a pioneering Natural Language Processing (NLP) system for simplified database querying. The system enables user interaction with relational databases in English without requiring proficiency in formal query languages like SQL[1]. The interaction system involves components like a relational database, a Database Management System (DBMS), and an NLDQ[1] interface, employing tokenization, parsing, syntactic comparison, and SQL generation stages. The algorithm reads input statements, identifies attributes, generates SQL queries, and displays results, utilizing Natural Language Processing to retrieve information and apply filters to rows and columns. In summary, the paper contributes to Natural Language Processing and Human-Computer Interaction[9] by introducing an innovative NLDQ interface, addressing language barriers in database querying, and enhancing user-friendly database interfaces.

This research paper “Natural Language Interfaces to Database (NLIDB): Question Handling and Unit Conversion” focuses on enhancing Natural Language Interfaces to Databases (NLIDB) by overcoming two specific challenges: handling question-type queries and implementing unit conversion [9]. The study builds upon Wibisono's NLIDB system, which used the Stanford Dependency Parser for natural language analysis but was limited to processing declarative queries and lacked unit conversion capabilities. The paper introduces unit conversion capabilities to the NLIDB system, facilitated by recognizing units in the database and those specified by users in natural language queries.Overall, the system improves NLIDB[1] capabilities, addressing limitations in existing systems and providing effective solutions for handling diverse natural language queries and unit conversions in a database context. This research significantly contributes to advancing the field of NLIDB [10].

This paper addresses the challenge faced by users with limited or no knowledge of database languages when accessing information from databases. The proposed system utilizes Natural Language Processing (NLP) [3] to enable users to interact with a railways reservation database through structured natural language questions, converting them into SQL queries. The paper discusses the historical context of NLP applications in database interfaces, referencing examples from as early as 1972. The proposed system in this paper follows a similar path, utilizing semantic grammar, morphological, syntactic[11], and semantic analysis, as well as a data dictionary for SQL generation. The high accuracy achieved indicates the effectiveness of the proposed approach in simplifying the interaction between users and databases, especially for those with limited technical knowledge.

The paper addresses a critical challenge in database management: enabling users without SQL expertise to interact effectively with databases. While SQL proficiency is traditionally required for database manipulation, natural language queries offer a more accessible alternative, allowing users to interact with data intuitively. To tackle this challenge, the paper proposes a novel two-phase approach. In the first phase, component segmentation identifies primary SQL[3] clauses like SELECT, FROM, WHERE, etc. The second phase, slot-filling, extracts sub-components for each primary clause, such as selecting columns or defining aggregation operations.This success highlights the system's potential in democratizing database access and management, benefiting users regardless of their technical expertise[3].

This paper delves into the advantages and drawbacks of existing approaches in the realm of translating text into SQL. We shed light on methods leveraging syntax parsing and those employing deep learning

techniques to predict suitable queries. In summary, despite the deployment of solutions incorporating predefined grammars and linguistic techniques or harnessing the power of deep learning networks, the hurdle of automatically translating natural language into a database language persists [12]. This underscores the need for ongoing improvements to enhance the relevance and quality of the generated queries.

This paper Introduces innovative Natural Language Database Querying (NADAQ) system, a groundbreaking solution addressing the complexity of SQL language and database schemas, making database querying more accessible to human programmers. NADAQ[13] seamlessly integrates deep learning and traditional database parsing techniques to create a robust translation model. Our system not only leverages the latest in deep learning but also draws from mature SQL parsing techniques, resulting in a powerful querying interface. NADAQ proves effective in improving query translation quality, tackling the rejection of meaningless questions, and suggesting query candidates.

The paper introduces a novel approach for the NL2SQL task, enabling users to interact with relational databases using natural language, eliminating the need for SQL. The model, prioritizing data privacy, relies solely on table schema, avoiding the transfer of sensitive data during predictions[15]. Combining RoBERT embeddings[15], data-agnostic knowledge vectors, and LSTM-based sub models, the method achieves a test set execution accuracy of 76.7%, showcasing its zero-shot learning capability for natural language questions and table schema. The related work section references the WikiSQL dataset[15], BERT-based models[5], and the content-enhanced BERT based Text-to-SQL Generation paper.

This research paper “Natural Language Processing Technique for Generation of SQL Queries Dynamically” delves into the realm of Natural Language Processing (NLP) [1] and its applications for database interactions. NLP is underscored as pivotal for enhancing human-machine communication, addressing challenges in comprehending and processing human input through algorithms like Named Entity Recognition, Tokenization, Stemming, Lemmatization[6], Bag of Words, Natural Language Generation, Sentiment Analysis, and Sentence Segmentation. The results indicate that the proposed system performs well in terms of efficiency and query optimization, marking a significant stride in advancing NLP-driven SQL query generation.

Chapter 3

PROPOSED SYSTEM

3.1 Objectives

- Develop a user-friendly system that allows WooCommerce sellers to interact with their database using natural English queries, eliminating the need for SQL expertise.

- Implement a robust language understanding module using the API to accurately interpret and translate English queries into SQL commands for database retrieval.

- Integrate speech recognition capabilities to enable users to interact with the system using spoken Hindi, which is then translated to English for query processing.

- Utilize an API for seamless translation of Hindi voice inputs into English queries, ensuring accessibility for users comfortable with Hindi.

- Design an intuitive web interface to provide a visually appealing platform for users to input queries and view results in both graphical and tabular formats.

- Employ an API to enhance data analysis capabilities, enabling the system to generate insightful visualizations based on user queries and database content.

- To develop our system as a plug-in compatible with WooCommerce.

3.2 Expected Outcomes

- Increased accessibility: Non-technical WooCommerce sellers can easily interact with their database and retrieve data without requiring SQL knowledge.

- Improved efficiency: Users can quickly obtain relevant insights through natural language queries, eliminating the need for manual SQL query construction.

- Enhanced user experience: Integration of speech recognition and translation features allows for seamless interaction, catering to users comfortable with Hindi.

- Data-driven decision-making: Users can visualize query results in graphs and tables, facilitating informed decision-making and business analysis.

- Scalability and flexibility: The system's modular design allows for easy integration of additional features and support for multiple languages in the future.

- Empowerment of users: By democratizing database access and analysis, the system empowers WooCommerce sellers to leverage their data for business growth and optimization.

Chapter 4

METHODOLOGY

4.1 Block Diagram

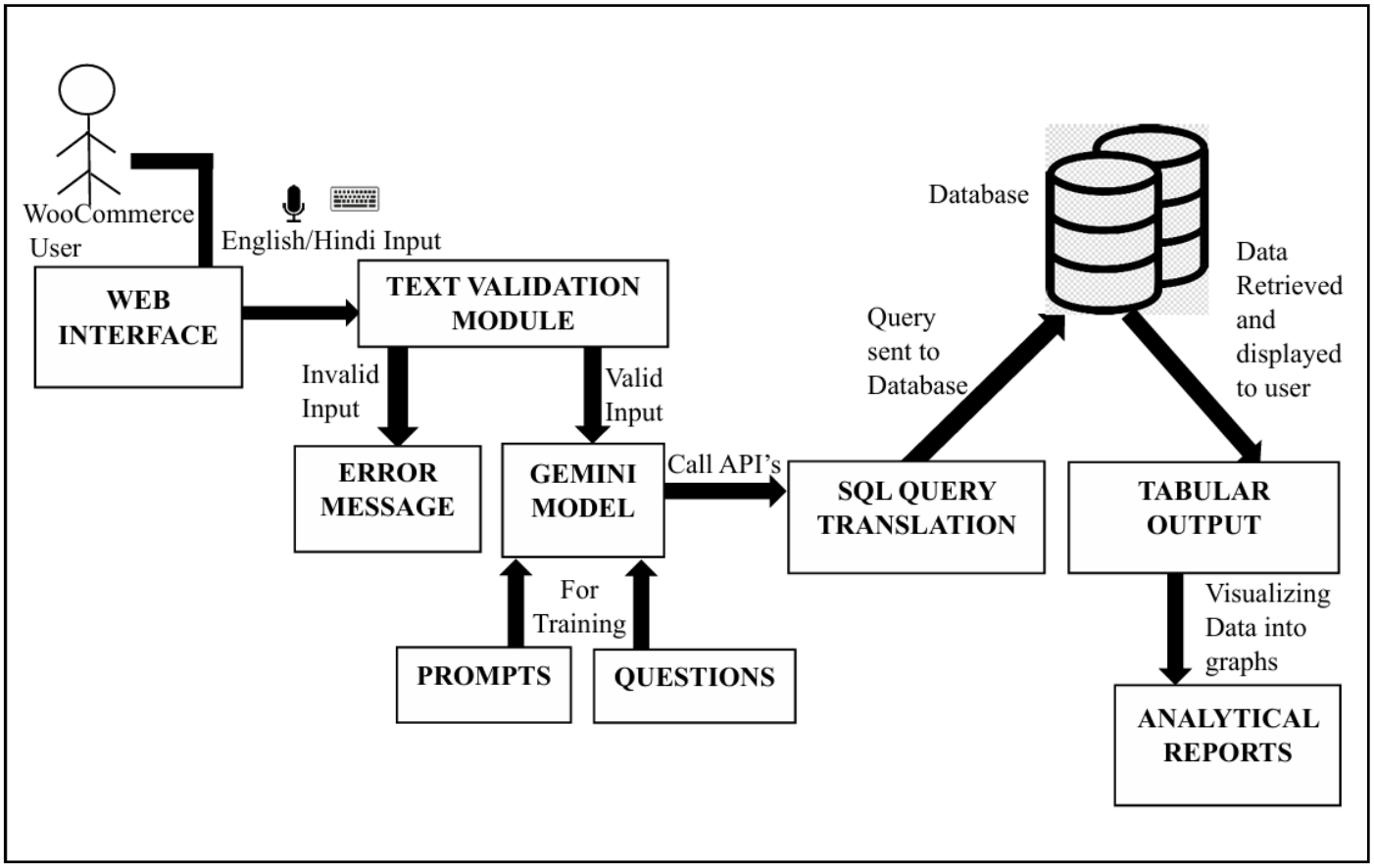


Fig 4.1 Intelligent Bilingual Query Processing Architecture

1. **WooCommerce User Input via Streamlit Web Interface**:

   Users interact with the system through a user-friendly web interface created with Streamlit. They have the flexibility to input their queries in either English or Hindi, providing a convenient user experience.

2. **Text Validation Module**:

   Upon submission, the user input undergoes validation to ensure its integrity and relevance to the system's functionality. If the input is found to be invalid or incomplete, appropriate error messages are displayed, guiding users to rectify their queries.

3. **Gemini Model Processing**:

   Valid inputs are forwarded to the Gemini model, which serves as the core component for refining and interpreting user intent. Leveraging its trained algorithms and knowledge of prompts and questions, the Gemini model enhances the query, making it more precise and actionable.

4. **Translation into SQL Queries**:

   The refined user queries are then translated into SQL queries through dedicated APIs. This translation process ensures compatibility with the database schema and prepares the system for data retrieval.

5. **Data Retrieval from the Database**:

   Executing the SQL queries facilitates the retrieval of relevant data from the underlying database. This data retrieval phase is crucial for accessing the information required to fulfill the user's query effectively.

6. **Tabular Data Presentation**:

   The retrieved data is presented to users in a structured tabular format. This presentation style enhances readability and comprehension, allowing users to easily interpret and analyze the fetched information.

7. **Data Visualization with Matplotlib**:

   Leveraging the capabilities of the Matplotlib Python module, the system generates insightful visualizations such as bar graphs and line graphs based on the retrieved data. These visual representations offer a clear and intuitive understanding of patterns, trends, and relationships within the data.

8. **Analytics Report Generation**:

   The generated graphs serve as the foundation for an in-depth analytics report. This report encompasses comprehensive insights derived from the visualized data, providing valuable information for decision-making and analysis purposes.

9. **PDF Download of Analytics Report**:

   To facilitate sharing and archiving, users have the option to download the analytics report in PDF format. This downloadable document encapsulates the findings, analyses, and visualizations, enabling users to retain a permanent record of the insights gleaned from the system.

4.2 Sequence Diagram

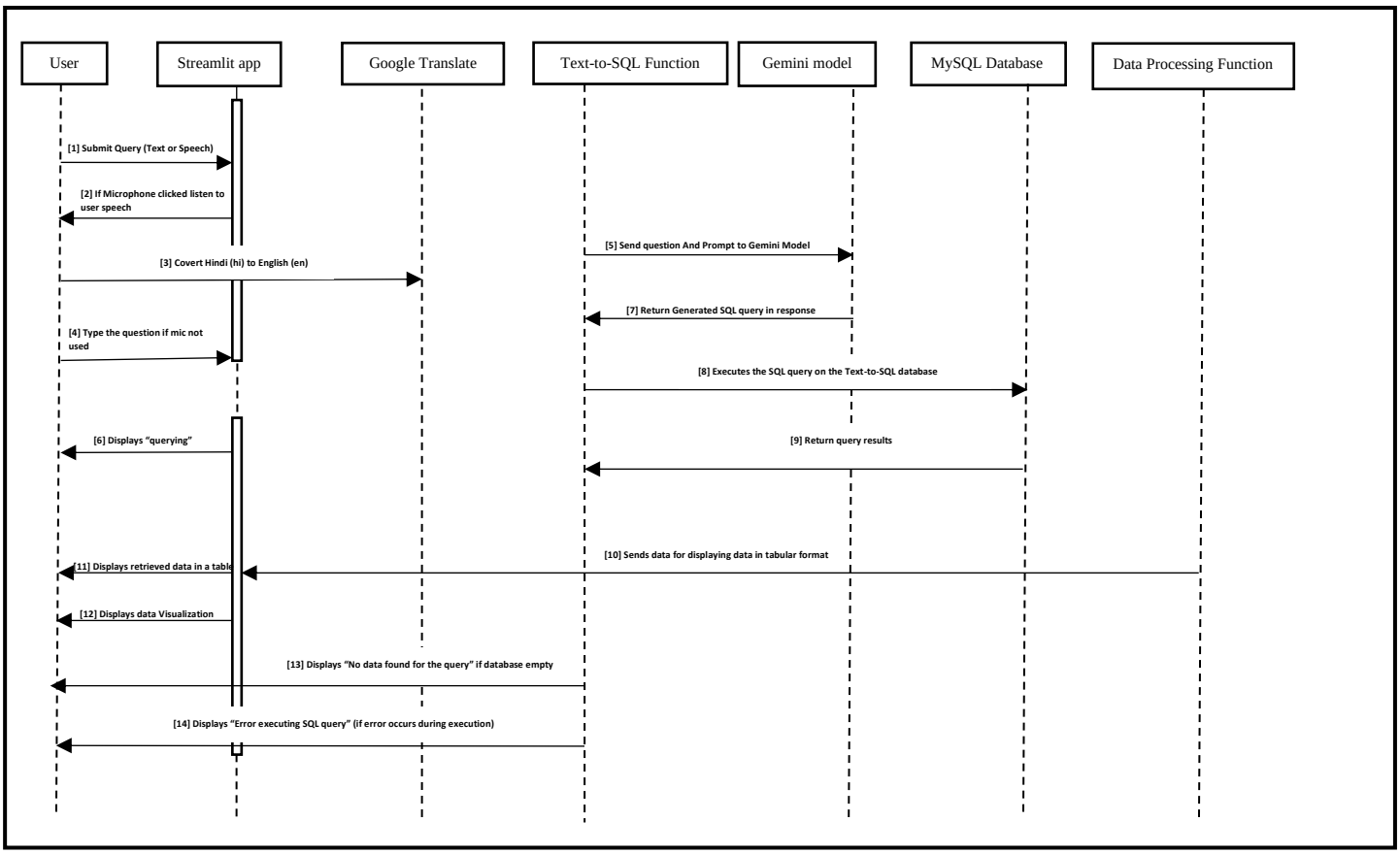


Fig 4.2 Seamless Data Interaction Framework

The sequence diagram outlines the flow of interactions within our system, illustrating how users submit queries, how they are processed, and how results are presented. Initially, users submit queries either through text input or speech input. If speech input is chosen, the system listens for user speech and translates it from Hindi to English using Google Translate before processing. Once the query is received, it is sent to the Gemini Model for text-to-SQL conversion. Upon receiving the generated SQL query, the system displays a "querying..." message to the user and proceeds to execute the query on the MySQL database. If the query is successfully executed, the system processes the data and sends it back to the user interface for display in tabular format. The system may generate data visualizations such as bar charts to further aid in data interpretation. However, if no data is found for the query, the system notifies the user accordingly. If an error occurs during query execution, the system alerts the user about the error. Throughout this process, the Streamlit web application framework serves as the interface between the user and the system. Our system ensures that users can effectively query and analyze data.Overall, this sequence diagram encapsulates the flow of interactions within our system, highlighting its functionality and user-centric design.

Chapter 5

SYSTEM IMPLEMENTATION

Steps involved in our project implementation:

Step1. Text-to-SQL Conversion:

The integration of the Gemini Pro model marks a significant advancement in our text-to-SQL conversion capabilities. By integrating the Gemini Pro model, we have enabled our system to accurately generate SQL queries directly from natural language input.Gemini is a state-of-the-art natural language processing (NLP) model developed by Google. It is specifically designed for generating code from natural language descriptions. This integration streamlines the process of data retrieval and manipulation, empowering users to interact with databases using everyday language seamlessly. To ensure seamless communication between users and the system, we have focused on optimizing the integration process rather than developing separate modules. By leveraging the capabilities of the Gemini Pro model, we eliminate the need for additional custom modules for query prompt handling and response. Instead, we rely on the inherent capabilities of the Gemini Pro model to interpret user inputs, identify query intent, and generate corresponding SQL commands effectively. This approach not only simplifies the system architecture but also ensures high accuracy and reliability in query translation.

We have expanded our system's reach by integrating Google's Speech Recognition API for language detection and translation. By embracing Bilingual support, we have enhanced the inclusivity and accessibility of our platform, catering to a diverse global audience.

Steps involved in Text-to-SQL Conversion using NLP :

1. Text Preprocessing:
    - Tokenization: This initial step involves breaking down the input text into smaller units called tokens. Tokens can be words, subworlds, or characters. For instance, the sentence "How many products are in stock?" might be tokenized into ["How", "many", "products", "are", "in", "stock", "?"].
    - Normalization: Text normalization aims to standardize the text by converting it to a consistent format. This typically involves converting all characters to lowercase, removing punctuation marks, handling contractions, and dealing with special characters. For example, converting "How many products are in stock?" to "how many products are in stock".
2. Semantic Understanding:

- Word Embeddings: In this step, words are mapped to high-dimensional vectors called word embeddings. Word embeddings capture semantic relationships between words based on their contextual usage in a large corpus of text. Techniques like Word2Vec, GloVe, or embeddings learned from transformer models like BERT are commonly used for this purpose.
- Sentence Embeddings: For longer text inputs such as entire questions or queries, sentence embeddings are generated to represent the overall meaning of the text. These embeddings capture the semantic context of the entire sentence, allowing the model to understand the intent behind the text.

3. Syntax Parsing:
    - Part-of-Speech (POS) Tagging: POS tagging involves assigning grammatical categories (nouns, verbs, adjectives, etc.) to each word in the input text. This information helps the model understand the syntactic structure of the text and identify the role of each word in the sentence.
    - Dependency Parsing: Dependency parsing analyses the syntactic structure of the text to identify relationships between words. It determines the dependency relationships between words, such as subject-verb-object relationships, and constructs a parse tree representing the grammatical structure of the sentence.
4. Query Generation:
    - Semantic Parsing: Semantic parsing involves mapping the parsed text into a structured representation that captures the semantics of the input. This structured representation serves as an intermediate step between natural language and SQL. It may involve identifying entities, attributes, conditions, and operations specified in the natural language text.
    - Template Matching: Some systems use predefined templates or rules to map natural language constructs to SQL query components. For example, a template might specify that a question starting with "How many" corresponds to a SQL COUNT query.
5. Machine Learning Models:
    - Supervised Learning: Machine learning models, such as neural networks, can be trained on pairs of natural language questions and their corresponding SQL queries. These models learn to generalize from the training data and generate SQL queries for unseen inputs.
    - Sequence-to-Sequence Models: Sequence-to-sequence (Seq2Seq) models, often based on recurrent neural networks (RNNs) or transformers, are commonly used for text-to-SQL

conversion. These models take the input text sequence and generate the output SQL sequence using encoder-decoder architectures.

6. SQL Query Execution:

   Once the SQL query is generated, it is executed against the database to retrieve the desired information. The database engine processes the query, retrieves the relevant data, and returns the results.

7. Error Handling and Validation:

   The generated SQL query may undergo validation to ensure syntactic correctness and prevent SQL injection attacks. Error handling mechanisms are implemented to handle cases where the natural language input cannot be accurately translated into a valid SQL query.

8. Feedback Loop:

   - In some systems, user feedback on the generated SQL queries may be used to refine and improve the NLP model over time. This iterative process helps enhance the accuracy and usability of the text-to-SQL conversion system by incorporating real-world usage patterns and user preferences.
   - This is generally what goes into NLP processing. Since Gemini is pre-trained on NLP conversions, we did not necessarily train any models further and leveraged Gemini for the Text-to-SQL conversion.

   This is generally what goes into NLP Processing.Since Gemini is pre-trained on NLP Conversions, we did not necessarily train any models further and leveraged Gemini for the Text-to-SQL conversion:

1. Semantic Understanding:

   - Gemini begins by deeply analysing the semantic context of the input text. It employs advanced natural language processing techniques to comprehend the meaning and intent behind the user's query.
   - For example, given the text "Show me all products priced below $50", Gemini recognizes that the user wants to retrieve product data from the database where the price is less than $50.

2. Code Generation:

   - Based on its semantic understanding, Gemini proceeds to generate SQL code that corresponds to the user's query. It draws upon its extensive knowledge base of programming patterns and SQL syntax to construct meaningful SQL queries.

- Continuing with the example, Gemini might generate the SQL query: SELECT * FROM Products WHERE Price < 50.

3. Template Matching and Rule-Based Systems:
    - In addition to its semantic analysis, Gemini employs predefined templates or rules to guide the code generation process. These templates capture common patterns in natural language queries and map them to corresponding SQL structures.
    - For instance, Gemini recognizes the pattern "Show me all [entity] where [condition]" and maps it to the SQL template SELECT * FROM [entity] WHERE [condition].
4. Database Schema Understanding:
    - Gemini leverages its understanding of the database schema to ensure that the generated SQL queries are both syntactically and semantically correct. It analyses the database structure to validate the generated SQL code and ensure compatibility with the database schema.
    - For example, Gemini verifies that the table and column names referenced in the SQL query exist in the database schema and are correctly formatted.
5. Error Handling and Validation:
    - Throughout the conversion process, Gemini incorporates robust error handling mechanisms to address potential issues or ambiguities in the input text. It validates the generated SQL queries to prevent syntactic errors and mitigate security risks such as SQL injection attacks.
    - If the input text contains ambiguous terms or conflicting instructions, Gemini may prompt the user for clarification or provide suggestions to resolve the ambiguity.

Step2. Database Interaction:

Our database interaction system, powered by the PyMySQL library, forms a solid link between our platform and MySQL databases, facilitating smooth communication and efficient data retrieval. With a focus on precision and efficiency, we have meticulously designed specialized functions within our system to execute SQL queries seamlessly and retrieve results for user presentation. These functions are adept at handling a wide range of query types, from basic data retrieval tasks to complex data manipulation operations. Through this integration, our system harnesses the full capabilities of SQL, enabling users to interact with their data effortlessly. Whether users need to fetch specific records, perform intricate joins, or execute aggregate functions, our system's database interaction capabilities ensure a seamless user experience.

Step3. User Interface (UI) and Interaction:

Harnessing the Streamlit web application framework has been instrumental in shaping our user interface into a dynamic and intuitive platform. With Streamlit, we have constructed a web-based interface that seamlessly integrates query input and SQL command display. This framework's versatility enables us to design a user experience that is both responsive and user-friendly, facilitating smooth interaction with data. By leveraging Streamlit's capabilities, we have ensured that users can effortlessly navigate the interface, input queries with ease, and instantly view the corresponding SQL commands. This cohesive integration not only enhances user engagement but also streamlines the process of data querying and analysis. Ultimately, our adoption of Streamlit empowers users to interact with their data in a seamless and efficient manner, fostering a more intuitive and productive user experience.

Step4. Text and voice input handling:

In our pursuit of user-friendly interaction, we have implemented a comprehensive approach to input handling through the integration of both text and voice input functionalities. Users can engage with our system through traditional text input Box as well as innovative voice input capabilities, leveraging the power of Speech Recognition for query submission. This dual-input system provides users with flexibility and accessibility, catering to diverse preferences and enabling efficient interaction with the system.Through Speech Recognition Python module users can articulate their queries verbally, eliminating the need for manual typing and offering a hands-free alternative.

The inclusion of Google Translate adds another layer of versatility to our input handling system, allowing users to interact with the system in their preferred language while still benefiting from the underlying English-based query processing capabilities. This feature is particularly beneficial for users who are more proficient in Hindi or may have limited proficiency in EnglishOverall, our integrated text and voice input handling system represents a significant step forward in user interaction, fostering a more intuitive and efficient user experience.

Chapter 6

RESULTS AND DISCUSSION

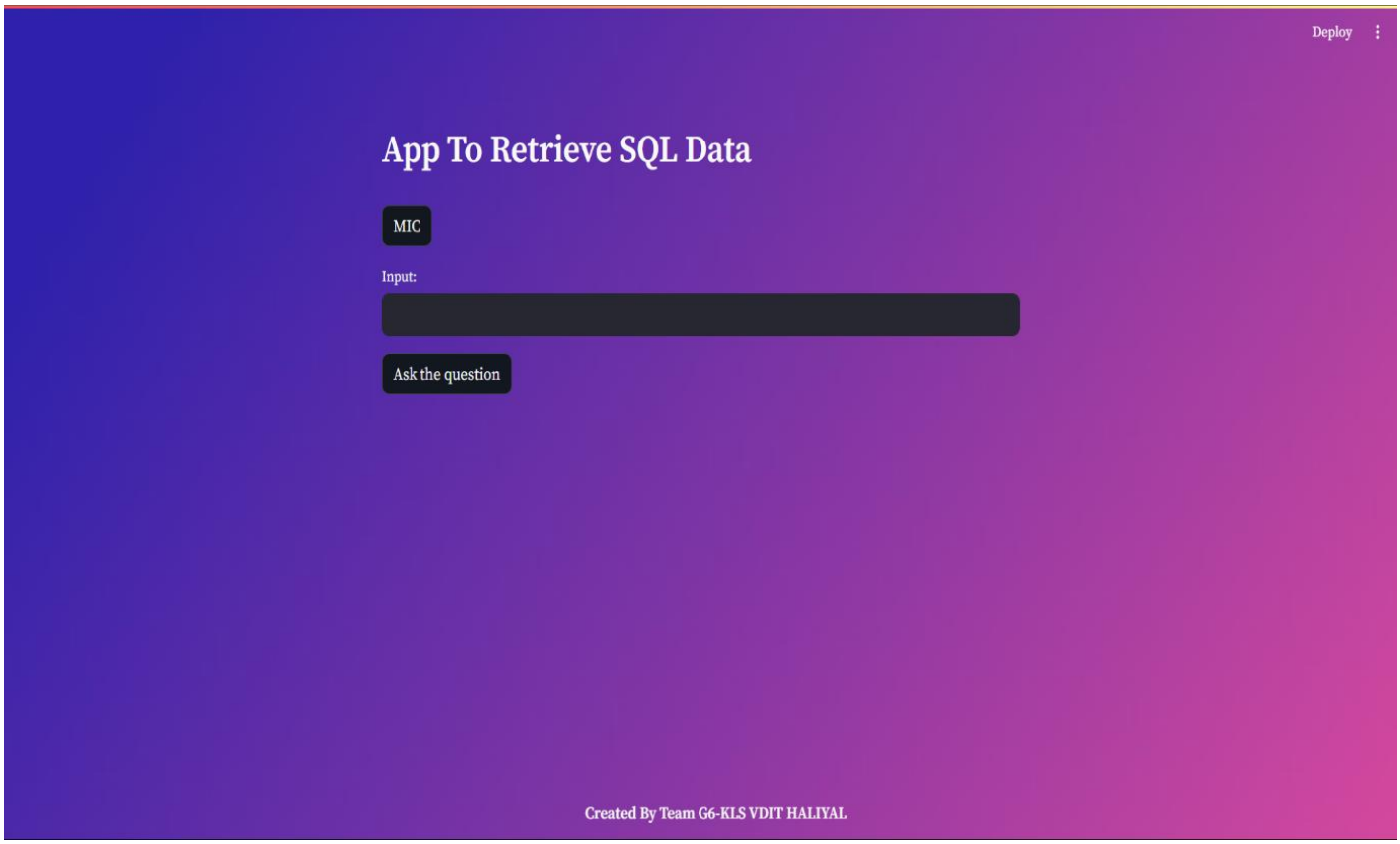


Fig 6.1 User Interface

The Streamlit Report Generator simplifies report creation with its user-friendly interface, developed using the Streamlit Python module. Users input queries through text or voice via a microphone button, then execute them with a single click. Targeting professionals, researchers, and students, it streamlines the process, saving time and enhancing productivity. Future plans include integrating more data sources for enhanced queries and customizable templates to meet specific formatting needs, expanding its utility and versatility.

Fig 6.2 User Input

The flexibility to input queries in the way that suits them best. Once queries are entered, users can execute them with a single click, accelerating the report generation process.

Fig 6.3 Query Generated by generative AI

Upon user request for data, the Streamlit Report Generator executes the generated query, retrieving the relevant information. This seamless process ensures that users receive accurate and up-to-date data tailored to their queries. By dynamically retrieving the queried data, the project enhances user experience and provides real-time insights. Users can trust that the information presented in the report is current and directly relevant to their inquiry, facilitating informed decision-making and analysis.

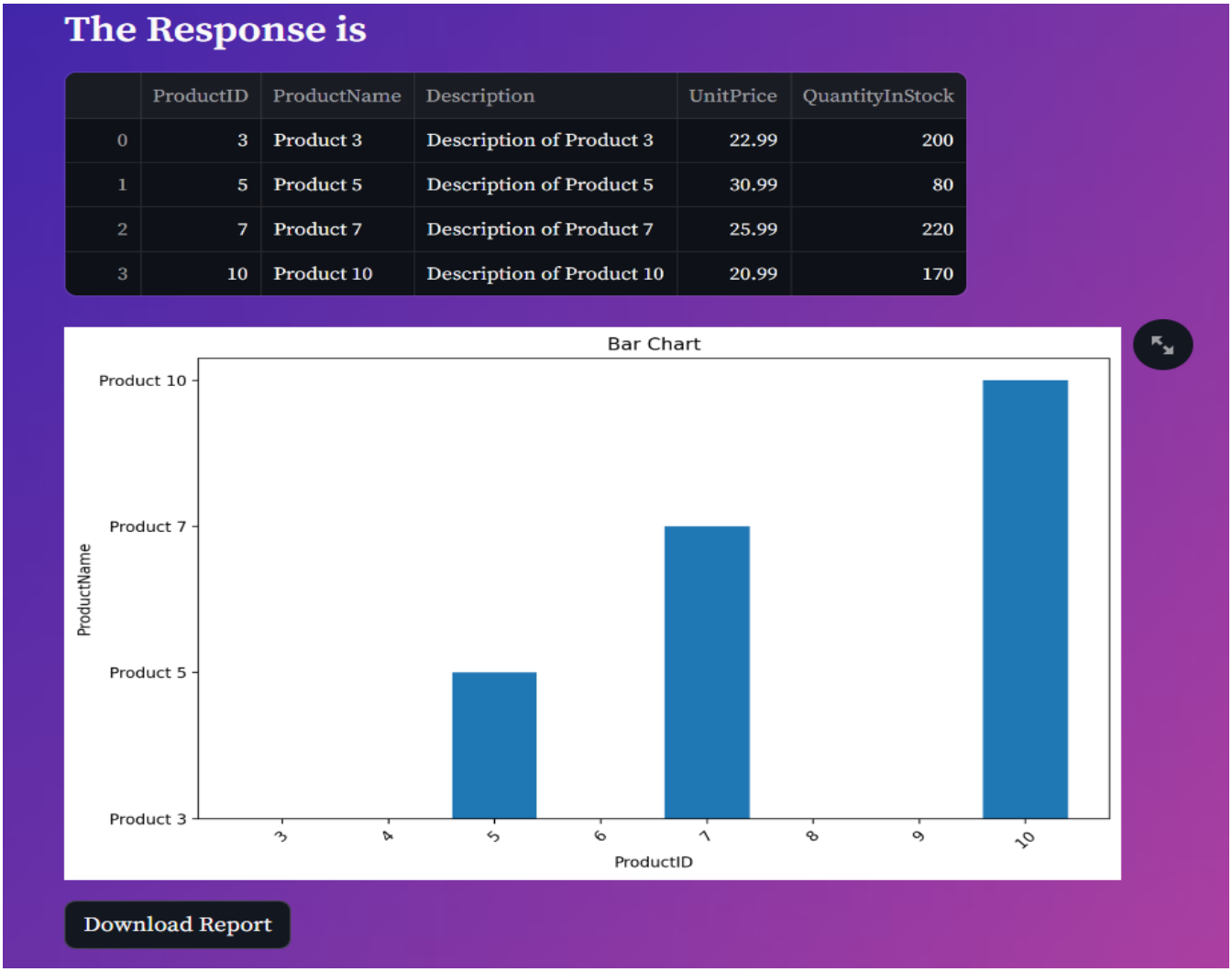


Fig 6.4 Structured Output

The Streamlit Report Generator displays data in the form of a bar chart, presenting information relevant to the user's query. A dedicated download button allows users to download the report in PDF format.This feature enhances user convenience and enables seamless dissemination of insights derived from the data analysis. By offering a streamlined process for both data visualization and report distribution, the project empowers users to efficiently communicate their findings and insights.

```
(.venv) PS E:\FINAL PROJECT\MAIN PROJECT DND\finalproject> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.51:8501

CALLED
Speak
Phase to be Translated :products aur uske description ke bare mein batao
translated:Tell us about products and its description
```sql
SELECT
  ProductName,
  Description
FROM Products;
```
```

Fig 6.5 Speech Recognition and Translation

The Streamlit Report Generator simplifies report creation by seamlessly translating spoken Hindi (HI) queries to English text (EN) during voice input processing. Leveraging sophisticated Natural Language Processing (NLP), the system ensures accurate translations while preserving query integrity. This inclusive approach enhances accessibility for Hindi speakers, fostering a welcoming environment for users from diverse linguistic backgrounds. By prioritizing innovation and user-centric design, the project empowers users to generate reports effortlessly, catering to a wide range of users.

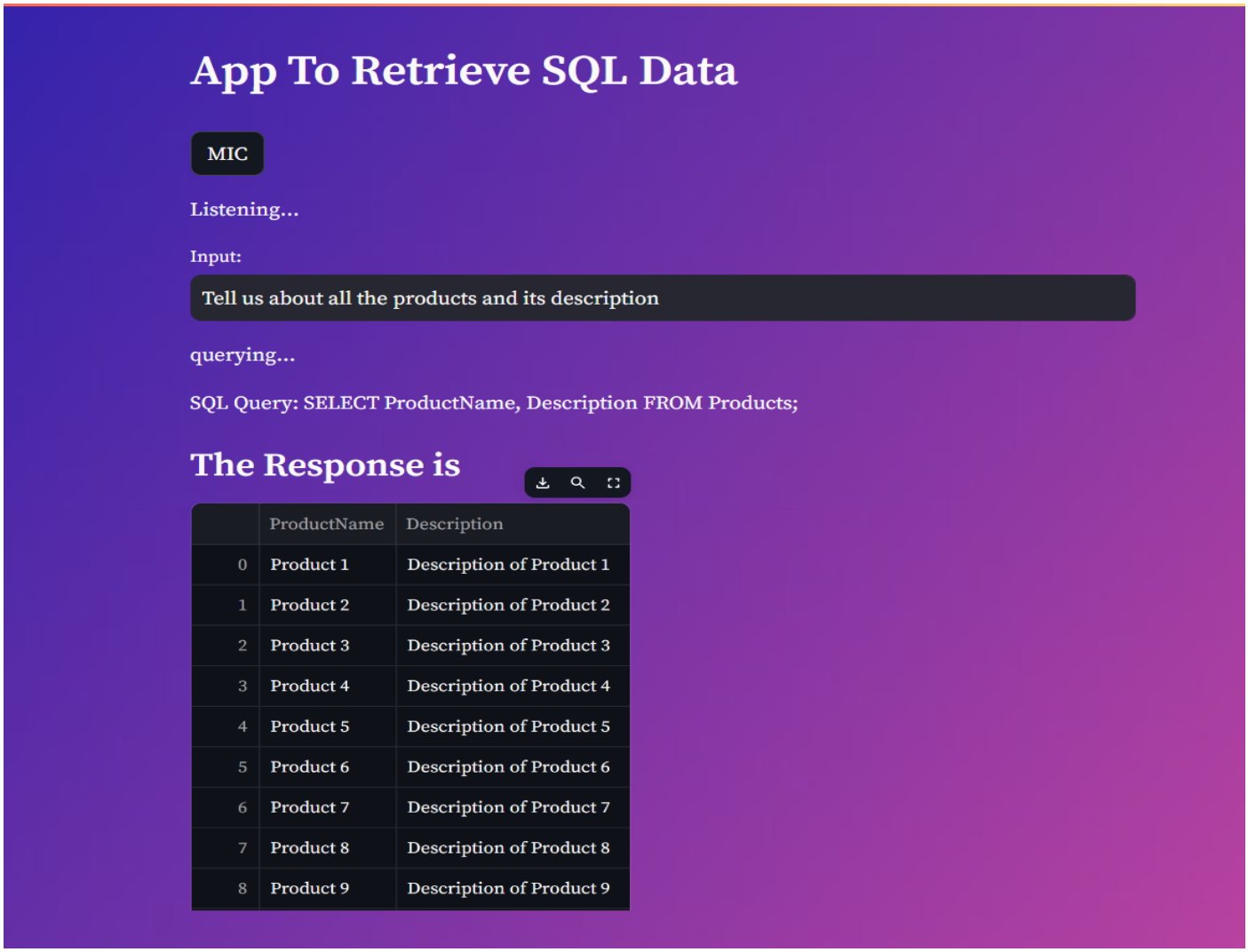


Fig 6.6 Output for the Query through mic

When utilizing the microphone feature, the Streamlit Report Generator interface transitions to a "listening" state upon activation. After users click to initiate the microphone, they are prompted with a "listening" indication. Subsequently, the transcribed English text appears in a designated text box on the interface. Following this, the system displays the corresponding query formulated from the transcribed text in SQL format. This seamless process provides users with a clear and concise representation of their query, ensuring accuracy and facilitating further analysis or refinement as needed.

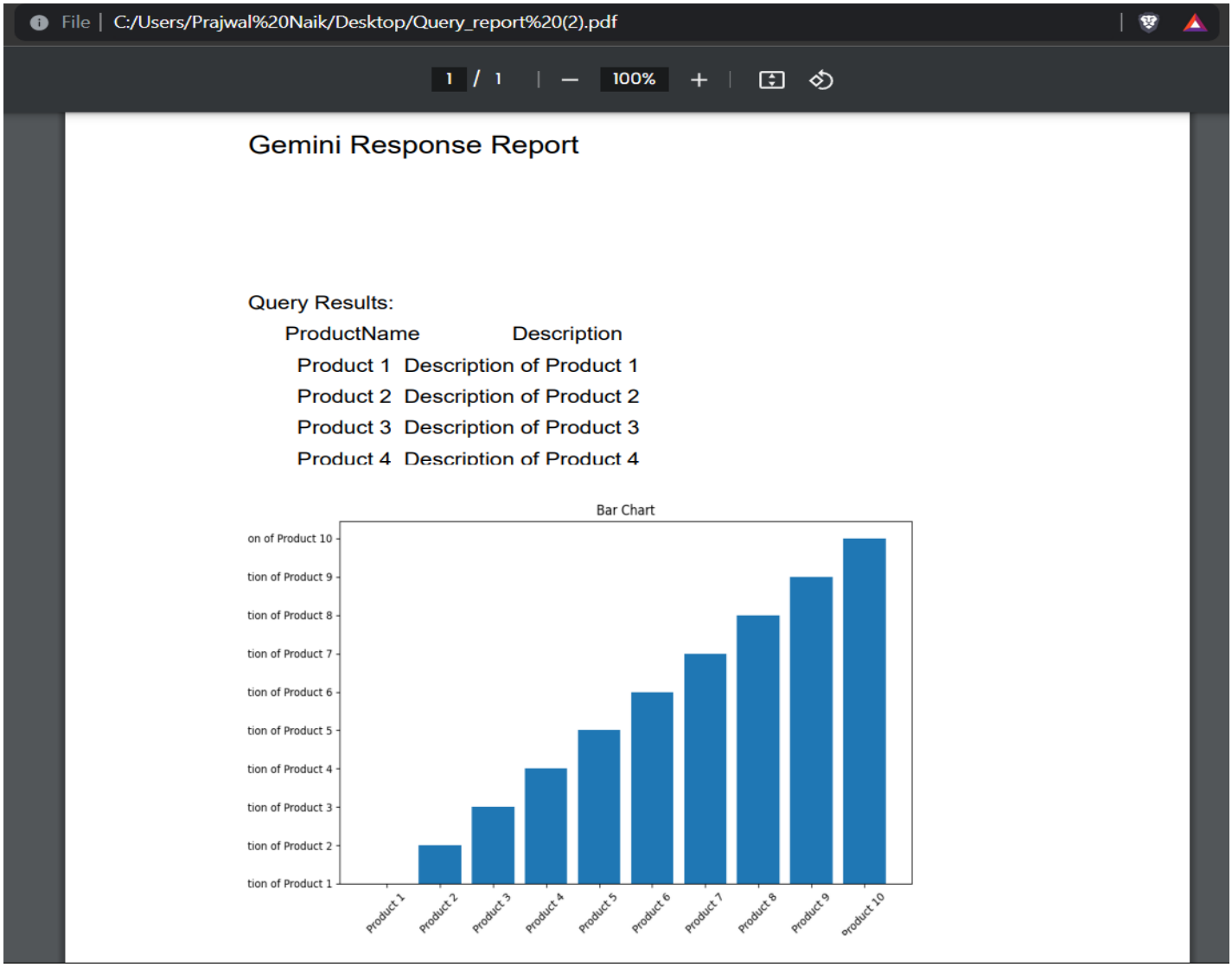


Fig 6.7 Downloaded Report in PDF Format

The retrieved data into a comprehensive report. This report includes visual representations such as bar charts to illustrate key insights derived from the queried data. Additionally, the report incorporates the SQL query used for data retrieval, providing transparency and facilitating reproducibility. Users can download the generated report in PDF format using the dedicated download button, ensuring easy access and sharing of the insights gathered from the query. This streamlined process enables users to effectively communicate their findings and analysis derived from the queried data.

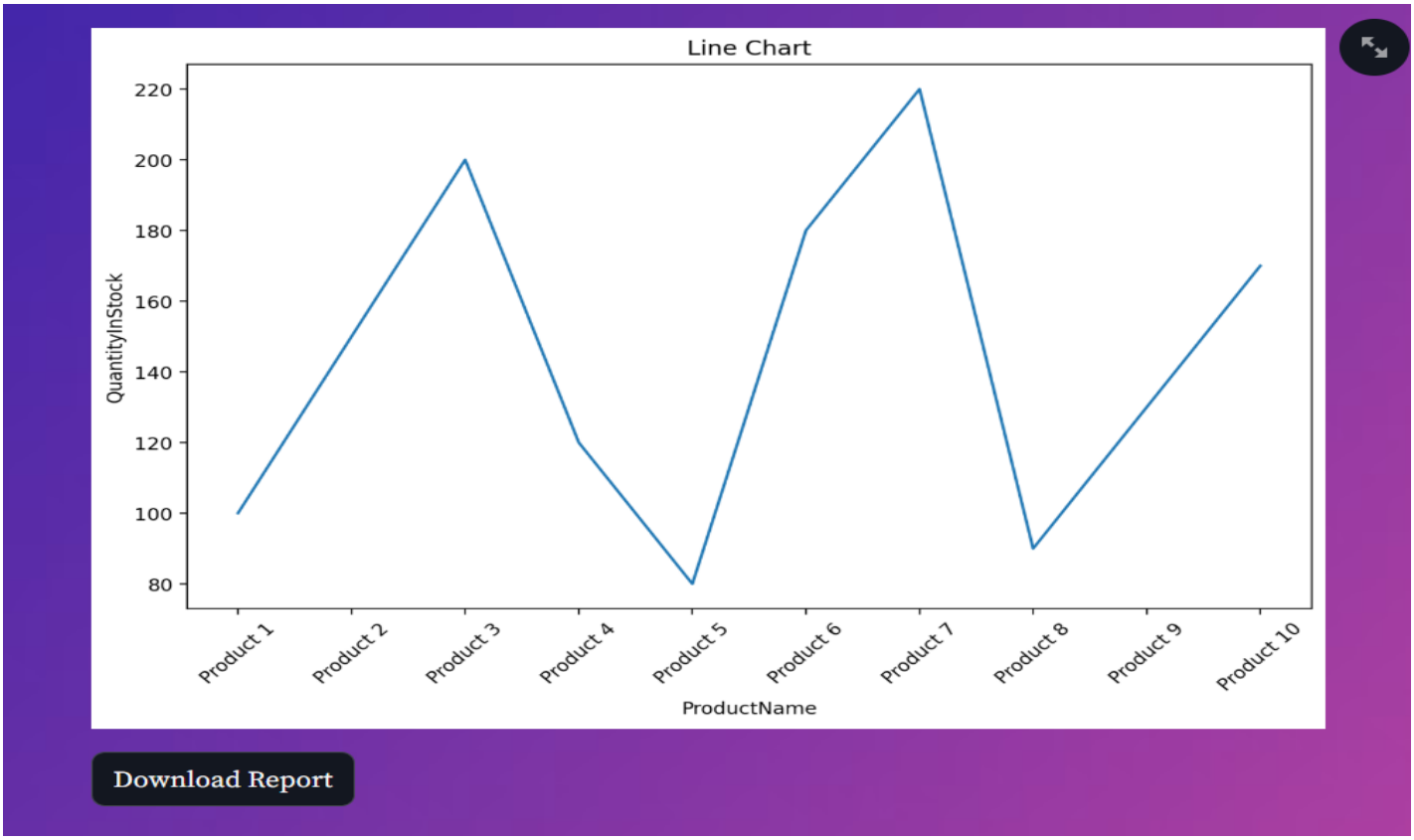


Fig 6.8 Data in the form of Line chart

The Streamlit Report Generator offers users the flexibility to input queries in their preferred manner, whether through manual keyboard entry or voice input via the microphone feature. Once queries are input, users can execute them effortlessly with a single click, expediting the report generation process. This streamlined workflow enhances user efficiency and productivity, allowing for quick and seamless access to insights and analysis. By accommodating diverse query input methods and simplifying the execution process, the project ensures a user-friendly experience tailored to individual preferences, ultimately empowering users to derive meaningful insights with ease.

Chapter 7

CONCLUSION AND SCOPE OF FUTURE WORK

FUTURE SCOPE

1. Scalability:

Implement techniques for horizontal scaling to accommodate increasing data volumes and user traffic. Integrate with cloud-based database solutions like Amazon Aurora or Google Cloud SQL for automatic scalability based on demand. Explore distributed database architectures such as Apache Cassandra or MongoDB for handling large-scale datasets efficiently.

2. Multilingual Support Enhancement:

Expand language support beyond the current languages by incorporating more translation APIs or machine learning models for better language detection and translation accuracy.Enable bidirectional translation to support not only translating queries into English but also translating results back into the user's preferred language.Implement natural language understanding (NLU) capabilities to better interpret and respond to queries in different languages, improving the overall user experience.

3. WordPress Plugin Integration:

Develop a WordPress plugin that seamlessly integrates your system's functionality into WordPress websites.Provide users with an easy-to-use interface within WordPress admin panels to interact with databases and visualize data directly from their websites.Ensure compatibility with various WordPress themes and plugins, offering flexibility and customization options for users.

4. Enhanced Visualization Options:

Expand the range of visualization options beyond bar charts and line charts to include histograms, scatter plots, heatmaps, and more.Integrate with popular visualization libraries such as D3.js or Plotly.js to offer interactive and customizable visualization experiences.Implement real-time updating of visualizations to reflect changes in data dynamically, providing users with up-to-date insights and analysis.

CONCLUSION

Developed system represents a groundbreaking solution for seamless data interaction for WooCommerce users. By integrating advanced text-to-SQL conversion capabilities with comprehensive multilingual support, we have created a versatile platform that caters to diverse user needs with unparalleled accuracy and accessibility.

The ability to convert natural language queries into SQL commands empowers users to interact with databases using everyday language, streamlining the process of data retrieval and manipulation. Through the incorporation of cutting-edge technologies such as the Gemini Pro model, our system ensures high precision and reliability in query translation, enabling users to articulate complex queries with ease.

Furthermore, our commitment to inclusivity is demonstrated through the integration of voice input functionality and multilingual support. Users can submit queries verbally in Hindi and English languages, which are then seamlessly translated into English using Google Translate before being processed by the system. This feature not only enhances accessibility for non-English speakers but also broadens the system's reach to a global audience.

In terms of database interaction, our system excels in efficiently retrieving and presenting data in a user-friendly format. By leveraging the pymysql library and specialized query execution functions, users can effortlessly access the information they need, presented in tabular columns and graphical representations for enhanced visualization and analysis.

Overall, our system represents a harmonious fusion of advanced technology and user-centric design, offering a seamless and intuitive experience for users to interact with data across languages and modalities. It marks a significant step forward in democratizing access to information and empowering users to harness the full potential of their data.

REFERENCES

[1] Xu Yiqiu, Wang Liwei and Yan Shi, "The study on natural language interface of relational databases," 2010 The 2nd Conference on Environmental Science and Information Application Technology, Wuhan, 2010, pp. 596-599, doi: 10.1109/ESIAT.2010.5568401.

[2] G. Koutrika, A. Simitsis and Y. E. Ioannidis, "Explaining structured queries in natural language," 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010), Long Beach, CA, USA, 2010, pp. 333-344, doi: 10.1109/ICDE.2010.5447824.

[3] T. Mahmud, K. M. Azharul Hasan, M. Ahmed and Thwoi Hla Ching Chak, "A rule based approach for NLP based query processing," 2015 2nd International Conference on Electrical Information and Communication Technologies (EICT), Khulna, Bangladesh, 2015, pp. 78-82, doi: 10.1109/EICT.2015.7391926.

[4] A. Iftikhar, E. Iftikhar and M. K. Mehmood, "Domain specific query generation from natural language text," 2016 Sixth International Conference on Innovative Computing Technology (INTECH), Dublin, Ireland, 2016, pp. 502-506, doi: 10.1109/INTECH.2016.7845105.

[5] J. Lin et al., "TiQi: A natural language interface for querying software project data," 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), Urbana, IL, USA, 2017, pp. 973-977, doi: 10.1109/ASE.2017.8115714.

[6] E. U. Reshma and P. C. Remya, "A review of different approaches in natural language interfaces to databases," *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, Palladam, India, 2017, pp. 801-804, doi: 10.1109/ISS1.2017.8389287.

[7] Solanki, Arun & Kumar, Ashutosh. (2018). A system to transform natural language queries into SQL queries. International Journal of Information Technology. 14. 10.1007/s41870-018-0095-2.

[8] A. Kate, S. Kamble, A. Bodkhe and M. Joshi, "Conversion of Natural Language Query to SQL Query," *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, 2018, pp. 488-491, doi: 10.1109/ICECA.2018.8474639.

[9] T. Ahmad and N. Ahmad, "A Simple Guide to Implement Data Retrieval through Natural Language Database Query Interface (NLDQ)," 2019 8th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India, 2019, pp. 37-41, doi: 10.1109/SMART46866.2019.9117501.

[10] D. A. Poetra, T. Esterina Widagdo and F. N. Azizah, "Natural Language Interface to Database (NLIDB) for Query with Temporal Aspect," 2019 International Conference on Data and Software Engineering (ICoDSE), Pontianak, Indonesia, 2019, pp. 1-6, doi: 10.1109/ICoDSE48700.2019.9092618.

[11] M. Uma, V. Sneha, G. Sneha, J. Bhuvana and B. Bharathi, "Formation of SQL from Natural Language Query using NLP," 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), Chennai, India, 2019, pp. 1-5, doi: 10.1109/ICCIDS.2019.8862080.

[12] T. -H. -Y. Vuong, T. -T. -T. Nguyen, N. -T. Tran, L. -M. Nguyen and X. -H. Phan, "Learning to Transform Vietnamese Natural Language Queries into SQL Commands," 2019 11th International Conference on Knowledge and Systems Engineering (KSE), Da Nang, Vietnam, 2019, pp. 1-5, doi: 10.1109/KSE.2019.8919393.

[13] K. Ahkouk and M. Machkour, "Human Language Question To SQL Query Using Deep Learning," 2019 Third International Conference on Intelligent Computing in Data Sciences (ICDS), Marrakech, Morocco, 2019, pp. 1-6, doi: 10.1109/ICDS47004.2019.8942342.

[14] B. Xu et al., "NADAQ: Natural Language Database Querying Based on Deep Learning," in IEEE Access, vol. 7, pp. 35012-35017, 2019, doi: 10.1109/ACCESS.2019.2904720.

[15] D. Pal, H. Sharma and K. Chaudhuri, "Data Agnostic RoBERTa-based Natural Language to SQL Query Generation," 2021 6th International Conference for Convergence in Technology (I2CT), Maharashtra, India, 2021, pp. 1-5, doi: 10.1109/I2CT51068.2021.9417888.

[16] H. Sanyal, S. Shukla and R. Agrawal, "Natural Language Processing Technique for Generation of SQL Queries Dynamically," *2021 6th International Conference for Convergence in Technology (I2CT)*, Maharashtra, India, 2021, pp. 1-6, doi: 10.1109/I2CT51068.2021.9418091.

Appendix A: Hardware Requirements

- Processor Intel Core i3 or an AMD Ryzen 3 processor
- Hard disk 50 GB
- Input Device Keyboard
- Output Device Laptop

Appendix B: Software Requirements

- Operating System 64/32 Bit Operating System
- Database MySQL
- IDE Jupyter