

## ABSTRACT

Modern organizations increasingly depend on large amounts of data, often stored in common formats like CSV, Excel, JSON, and PDF files. However, getting useful insights from these raw datasets remains a challenge, especially for non-technical users without programming skills or knowledge of data analysis tools. "Axnos AI - Talk to Your Data" fills this gap by offering an AI chatbot platform that allows users to upload data files and ask questions in plain language, removing the need for coding or complex interfaces. By using advanced AI models, the proposed system turns user questions into executable Pandas operations that filter, group, sort, and summarize data as needed. All queries, results, and file uploads are securely logged, letting users check their analysis history whenever they like. The proposed system uses a range of modern technologies, including asynchronous task handling, RESTful APIs and a user-friendly frontend made with Tailwind CSS and shadcn UI. By making data analysis accessible to everyone, Axnos AI helps business users, researchers, and anyone working with data make informed decisions quickly, promoting a data-driven culture across various fields.

All queries, results, and file uploads are securely logged, letting users check their analysis history whenever they like. The proposed system uses a range of modern technologies, including asynchronous task handling, RESTful APIs and a user-friendly frontend made with Tailwind CSS and shadcn UI. By making data analysis accessible to everyone, Axnos AI helps business users, researchers, and anyone working with data make informed decisions quickly, promoting a data-driven culture across various fields.

# INTRODUCTION

## a. Need of the Work

In today's data-driven world, individuals and organizations collect and store vast amounts of information in formats such as CSV, Excel, JSON, and PDF. However, extracting meaningful insights from this data remains difficult for non-technical users who lack programming or analytical experience. Traditional tools like spreadsheets, SQL queries, or coding environments often create barriers, leading to dependency on technical experts and delayed decisions.

Therefore, there is a strong need for an intelligent, easy-to-use platform that enables users to interact with their data naturally and efficiently. Axnos AI fulfills this need by using advanced Large Language Models (LLMs) to allow users to perform data analysis simply by asking questions in plain language, making data exploration faster, easier, and accessible to everyone.

## b. Limitations of Existing System

1. **Technical Barriers:** Most existing data analysis tools require coding knowledge or expertise in software like Python, SQL, or Power BI, which limits accessibility for general users.
2. **Limited Multi-Format Support:** Current systems often handle only specific file formats and cannot seamlessly process data from varied sources such as CSV, Excel, JSON, and PDF together.
3. **Lack of Conversational Interaction:** Traditional tools do not provide a natural, question-based interface for exploring data, making analysis time-consuming and complex for non-technical users.

### c. Proposed System

The proposed system, **Axnos AI – Talk to Your Data**, introduces an intelligent chatbot platform that allows users to analyze and visualize data by communicating through simple text or voice queries. Users can upload their datasets in multiple formats (CSV, Excel, JSON, PDF), and the system leverages **Large Language Models (LLMs)** to interpret user queries and generate relevant **Python (Pandas-based)** operations automatically. The system then executes this code securely and presents clear results in tabular or visual form, along with the corresponding generated code for transparency.

#### Key Features:

- **LLM-Based Query Interpretation:** Converts user prompts into executable data operations without requiring programming knowledge.
- **Multi-Format Data Support:** Efficiently handles CSV, Excel, JSON, and PDF files with automatic validation and preprocessing.
- **Secure & Transparent Execution:** Runs code in a sandboxed environment while displaying results and generated code to users.
- **User History Management:** Maintains logs of uploaded files, queries, and outputs for future reference.
- **User-Friendly Interface:** Built with React, Tailwind CSS, and shadcn UI to ensure an intuitive and responsive experience.

## LITERATURE SURVEY

Wang et al. proposed an NLI for interactive data analysis. The system recommends context-aware next-step queries to guide novice analysts during step-bystep exploration. It also organizes query histories and results into dashboards, showing improved effectiveness compared to a non-recommendation baseline [1]. However, it focuses more on query suggestions than on multi-format ingestion or remembering analyses across sessions.

Quamar et al. surveyed NLID approaches, including rule-based systems, text-toSQL, and hybrids. They outlined challenges in identifying entities, linking semantics, and generating structured queries. They also highlighted progress toward conversational multi-turn interfaces and evaluation benchmarks. The monograph maps out the design space and open issues but does not include heterogeneous file analysis or persistent interaction trails [2].

The IEEE work linked via 9699035 addresses conversational analytics for multiturn natural language interaction in analytical workflows. It demonstrates accessibility gains for non-technical stakeholders but often relies on predefined schemas or visualization grammars. This creates gaps in generalized onboarding for various file types and in-session provenance unless such features are specifically designed [3].

Talk2Data allows for exploratory visual analysis through question decomposition. It breaks down complex intents into sub-questions that correspond to visual operations and uses multi-hop reasoning. However, its focus on decomposition and visual answerability does not cover broader uploads for PDF or JSON files or track longitudinal histories of user queries and insights beyond specific study settings [4].

The ACL N19-1423 paper presents techniques for multi-hop or context-aware question processing to support decomposition, disambiguation, and retrieval for conversational data analysis. However, these methods are tested on text-focused datasets and do not offer end-to-end interfaces for heterogeneous uploads, Pandas-backed code generation, or secure execution for non-technical users [5].

The literature highlights several approaches to natural language interfaces for data. These include query recommendation, text-to-SQL, hybrid parsing, conversational context, and visual exploration through question decomposition. However, there are still common gaps. These include challenges in onboarding multi-format data, maintaining cross-session histories, and providing secure execution that is accessible to non-technical users.

## **Motivation:**

Today, many people and businesses keep their data in formats like CSV, Excel, PDF, and JSON. While saving this data is simple, understanding and using it can be difficult. A lot of individuals do not know how to use programming or advanced software to explore what their data means. Because of this, important information often goes unused. This challenge gave rise to Axnos AI.

### **Making Data Simple for Everyone**

Most data tools are built for people who already know how to code or work with complex software. This makes it hard for non-technical users such as teachers, store managers, office workers, and students to use their data. Many of them want helpful insights but don't have the required skills. Axnos AI makes it possible for them to simply type a question in everyday English and get meaningful answers. No coding or special software is needed.

### **Talking Naturally with Data**

People are now familiar with talking to devices using natural language. Axnos AI applies this idea to working with data. Instead of scrolling through spreadsheets or writing formulas, users can just ask something like, "What were the most sold items last month?" and get a clear answer. This way of asking makes data work feel easier and more human.

### **Helping People Use Data to Decide Better**

When people can easily understand their data, they make better decisions. Axnos AI supports this by removing the need for technical knowledge. It helps users rely on real facts instead of guesses. This leads to clearer planning, fewer mistakes, and more confident choices in everyday work. Axnos AI is made to be easy to understand and use. It works with many types of files and keeps a record of every question and answer.

## **PROBLEM STATEMENT & OBJECTIVES**

### **a. Problem Statement:**

“To develop a system for Data Science Tools that enables users to perform data analysis, preprocessing, and visualization using simple text or voice prompts.”

### **b. Objectives:**

1. To design a user-friendly interface that allows data upload and interaction through natural language or voice commands.
2. To implement a natural language processing (NLP) engine that interprets user queries and maps them to appropriate data operations.
3. To automate common data science tasks such as data visualization, cleaning, and summary statistics generation through prompt-based interaction.
4. To provide real-time feedback and responses to user queries by integrating backend data processing with the front-end interface.
5. To support multiple data formats (e.g., CSV, Excel, JSON, PDF) and abstract away technical complexities for users with minimal data science knowledge.

## REQUIREMENT ANALYSIS

### a. Dataset Description

(ADD DETAILED DESCRIPTION ABOUT DATASET LIKE ATTRIBUTES IN DATASET, FEATURES, NUMBER OF RECORDS IN DATASET, SOURCE LINK)

### b. Hardware Requirements

- Processor: Intel i3 or higher/ Ryzen 3 or higher
- Memory: Min 4GB RAM

### c. Software Requirements

- Operating System: Linux/Windows.
- AI Tools: OpenRouter, Gemini.
- Programming Languages: Python, JavaScript, Typescript, HTML, CSS.
- Frameworks: React, Node.js, FastAPI, Nest.js, Django.
- Databases: Neon, Supabase, MongoDB, PostgreSQL, Redis.
- Tools: Docker, Git & Github, Postman, Transformer, Vercel, Render.
- Styling: Tailwind CSS, Shaden UI.

### d. Libraries Used

#### Python Libraries:

1. Pandas - A Python library that provides high-performance, easy- to- use data structures like DataFrame for drawing, transubstantiating, and assaying labeled, irregular data.
2. NumPy - The main scientific computing package in Python, offering fast n- dimensional arrays and vectorized operations for calculation, direct algebra, and arbitrary slice.
3. Seaborn - A statistical visualization library erected on Matplotlib. It provides pandas- friendly, dataset- acquainted APIs with seductive defaults for exploratory plots.
4. Matplotlib - A plotting library for creating static, animated, and interactive visualizations with both py-plot and object-oriented interfaces.

- **JavaScript Libraries:**

1. React - An element- grounded JavaScript library for erecting dynamic stoner interfaces with JSX and a virtual DOM. It uses one- way data inflow and hooks- grounded state operation.
2. Redux - A state vessel that keeps app state consolidated in a single store.

# SYSTEM DESIGN & IMPLEMENTATION

## a. System Architecture

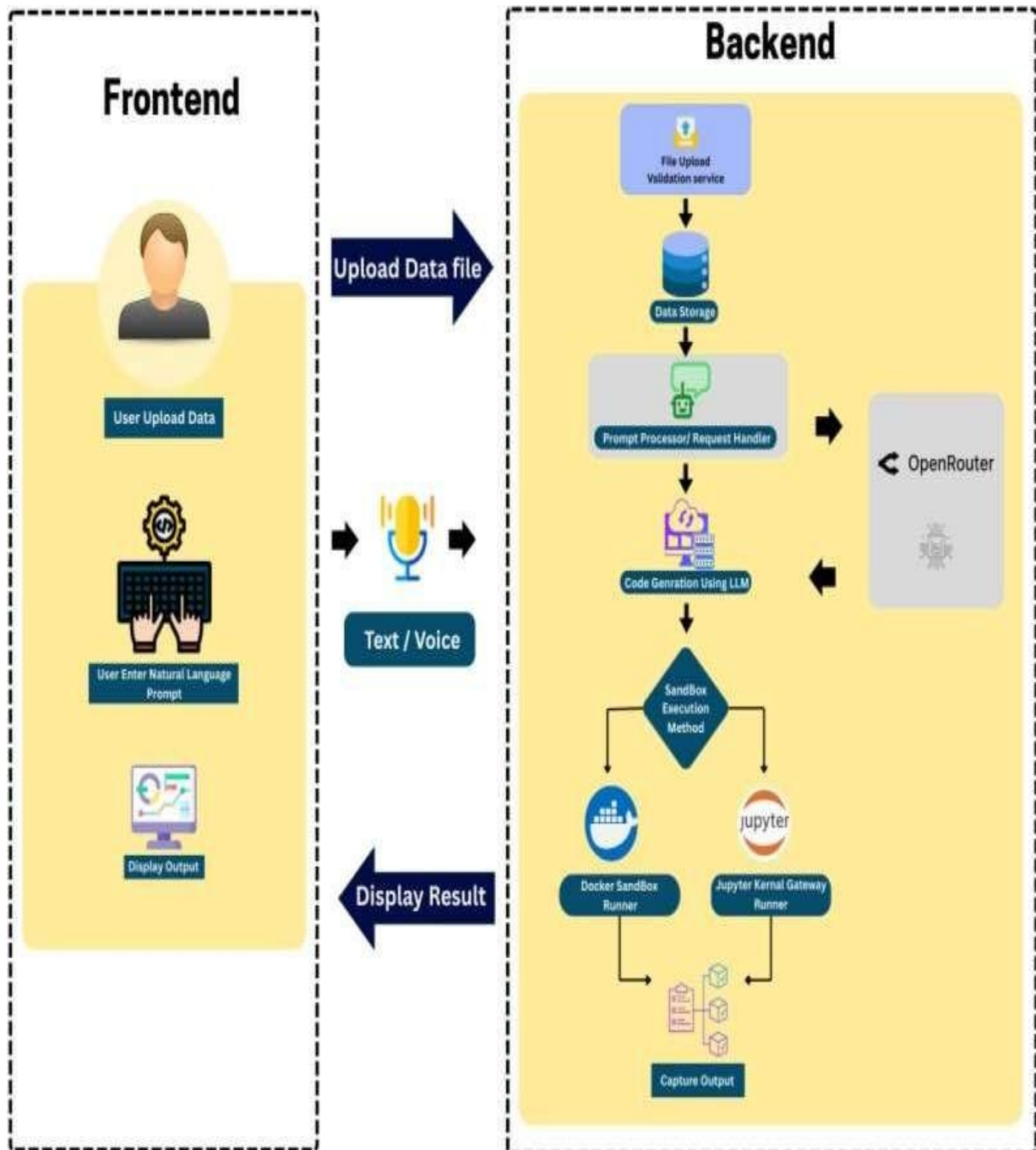


Fig.1 Proposed Architecture Diagram

- **User Uploads Data:**

Users select and upload their data files via a simple web interface. The frontend uses React with file input components for smooth user interaction. An upload API on the backend receives the file for processing.

- **File Validates & Saves:**

Uploaded files are checked to ensure they match supported formats like CSV, JSON or XLSX. Backend validation is performed using Python to verify file integrity. Files are stored securely on cloud storage environments.

- **User Inputs Prompt:**

Users can provide queries via natural language prompts either by entering or using voice to describe their desired data tasks. The frontend chat or user interface captures these input requests. This interaction initiates the data analysis workflow.

- **Prompt Sent to AI:**

The user's prompt is sent from the backend to an AI language model API. The AI interprets the request, determines analysis goals, and drafts a plan. Python backend manages communication with external LLM APIs like OpenAI.

- **Code Generated**

The AI generates executable Python code tailored for analysis or visualization. Code can include data processing steps, statistical calculations, or charting commands. LLMs such as GPT-4 drive the automated code authoring process.

- **Code Execution**

The generated code is executed within a secure, sandboxed environment. Common sandbox options include Docker containers, cloud VMs, or notebook converters. This ensures safe handling of arbitrary code without compromising system integrity.

- **Return Results & Code**

The system compiles analysis output, charts, and the executed code into a user-friendly format. Results are delivered back to the frontend for easy display and review. Users see both visualizations and the underlying code for transparency.

## **b. Data Flow Diagrams**

### **Data Flow Diagrams (Level 0 & 1)**

## **c. Implementation Details**

- **User Authentication & Registration Module**
  1. Implemented secure user registration and login functionality with password hashing and salting for enhanced data security.
  2. Managed user sessions using JWT-based token authentication ensuring safe and persistent login handling.
  3. Integrated PostgreSQL via NeonDB for storing user credentials and authentication logs with proper data validation.
- **Chat Configuration & History Management**
  1. Designed chat management modules that handle chat creation, history storage, and retrieval per user.
  2. Added automatic chat title generation based on message context for better organization of chat records.
  3. Established database relations between USER\_AUTH, CHAT, and PROMPT tables to support real-time chat history and prompt tracking.
- **Code Generation**
  1. Integrated OpenAI-based code generation APIs to process user prompts and generate corresponding code snippets dynamically.
  2. Updated and optimized backend models and serializers for handling prompt requests and storing results efficiently.
  3. Implemented result management by linking generated code outputs to user chat sessions within the database.
- **Frontend Integration**
  1. Developed user interface modules for registration, login, and chat dashboard using responsive frontend technologies.
  2. Connected frontend components with backend APIs to enable real-time chat communication and code result display.
  3. Ensured proper error handling and form validation for all authentication and chat interactions on the client side.

## **RESULT DESCRIPTION**

**It should include the results of system modules (till current implementation) with statistical analysis and visualization (if any).**

## REFERENCES

- [1] Wang, X., et al. (2022). Interactive Data Analysis with Next-step Natural Language Query Recommendation.
- [2] Quamar, A., Efthymiou, V., Lei, C., & Özcan, F. (2020). Natural Language Interfaces to Data.  
Shen, L., Shen, E., Luo, Y., Yang, X., Hu, X., Zhang, X., Tai, Z., & Wang, J. (2021). Towards Natural Language Interfaces for Data Visualization: A Survey. IEEE.
- [3] Guo, Y., Shi, D., Guo, M., Wu, Y., Chen, Q., & Cao, N. (2022). Talk2Data: A Natural Language Interface for Exploratory Visual Analysis via Question Decomposition.
- [4] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT*, 1(1), 4171–4186.

