AI DATA CLEANING ASSISTANT

Author Name: G SIVA SAI

Email: gmsivasai@gmail.com

Github link: https://github.com/gmsivasai/ai-cleaning-agent.git

ABSTRACT

Data quality plays a crucial role in accurate data analysis and decision-making. However, real-world datasets frequently contain missing values, duplicate records, and inconsistent formats. Manual data cleaning is time-consuming and error-prone. This paper presents an AI Data Cleaning Assistant, a web-based application that enables users to securely upload CSV files, automatically detect common data issues, apply selected cleaning operations, and download cleaned datasets. The system is implemented using a FastAPI backend, Streamlit frontend, and PostgreSQL database, with secure JWT-based authentication. The proposed solution ensures efficiency, transparency, and secure access to cleaned data.

Keywords*: Data Cleaning, FastAPI, Streamlit, JWT Authentication, CSV Processing, Data Quality*

I. INTRODUCTION

Data preprocessing is a fundamental step in data analysis and machine learning pipelines. Most datasets obtained from real-world sources contain inconsistencies such as missing values, duplicate records, and incorrect data types. Without proper cleaning, these issues can significantly impact analytical results.

The AI Data Cleaning Assistant aims to automate and simplify the data cleaning process by providing a secure and user-friendly platform where users can upload datasets, analyze issues, apply cleaning steps, and download the cleaned files.

II. PROBLEM STATEMENT

Existing data cleaning tools either require extensive manual intervention or lack transparency in applied cleaning operations. Additionally, many systems do not provide user-specific security and file history management. There is a need for a system that combines automation, user control, and security.

III. OBJECTIVES

The main objectives of this project are:

- To design a secure API-based data cleaning system
- To automatically detect common data quality issues
- To allow users to selectively apply cleaning steps
- To ensure secure access to original and cleaned files
- To maintain a user-specific file history

IV. SYSTEM ARCHITECTURE

A. Architecture Overview

The system follows a client–server architecture consisting of:

- Frontend: Streamlit web interface
- Backend: FastAPI RESTful APIs
- Database: PostgreSQL
- File Storage: Local file system

B. Architecture Flow

1. User interacts with the Streamlit frontend
2. Requests are sent to the FastAPI backend
3. Backend authenticates the user using JWT
4. CSV files are processed using Pandas
5. Metadata is stored in PostgreSQL
6. Cleaned files are returned to the user

IV.I STREAMLIT SCREENSHOT

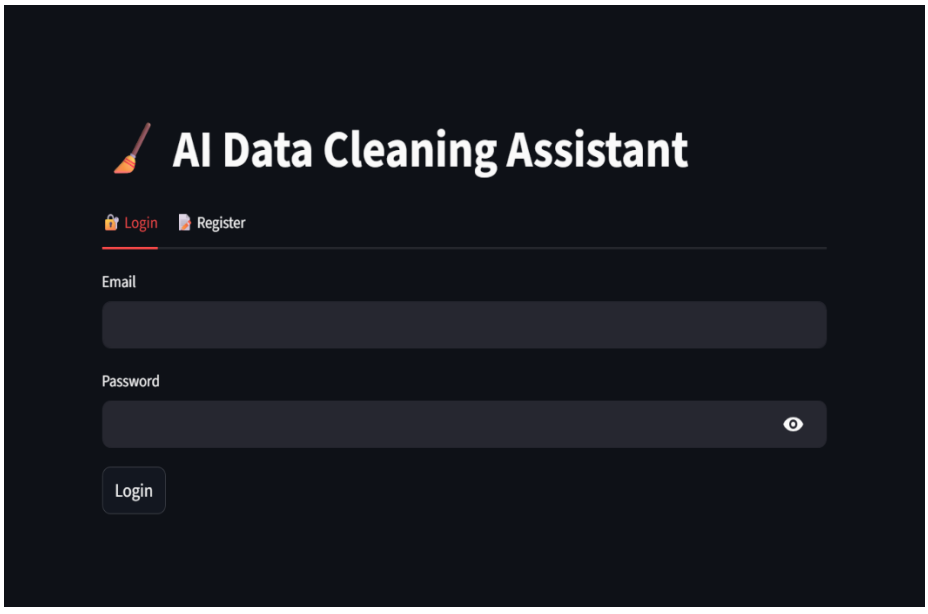

Fig. 1. Streamlit Login Interface

Context:
This figure shows the login interface of the AI Data Cleaning Assistant developed using Streamlit. Registered users can securely log in by providing their email and password credentials. The interface serves as the entry point to the application and ensures that only authenticated users can access data upload, cleaning, and download functionalities.

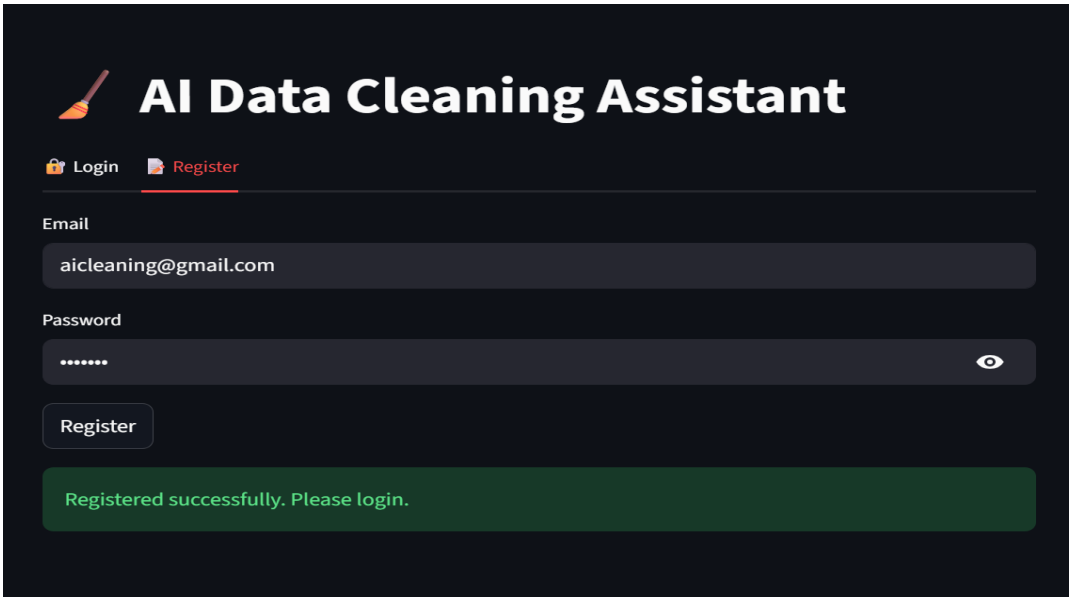

Fig. 2. User Registration Interface

Context:
This screenshot illustrates the user registration interface, where new users can create an account by providing valid credentials. Upon successful registration, the system displays a confirmation message and prompts the user to log in. User credentials are securely processed and stored using password hashing mechanisms.

Fig. 3. Successful User Login

Context:
This figure demonstrates a successful login operation in the Streamlit frontend. Once authenticated, the user gains access to the core functionalities of the application, including CSV upload, analysis, and cleaning modules. Authentication tokens obtained from the backend are used for secure communication.

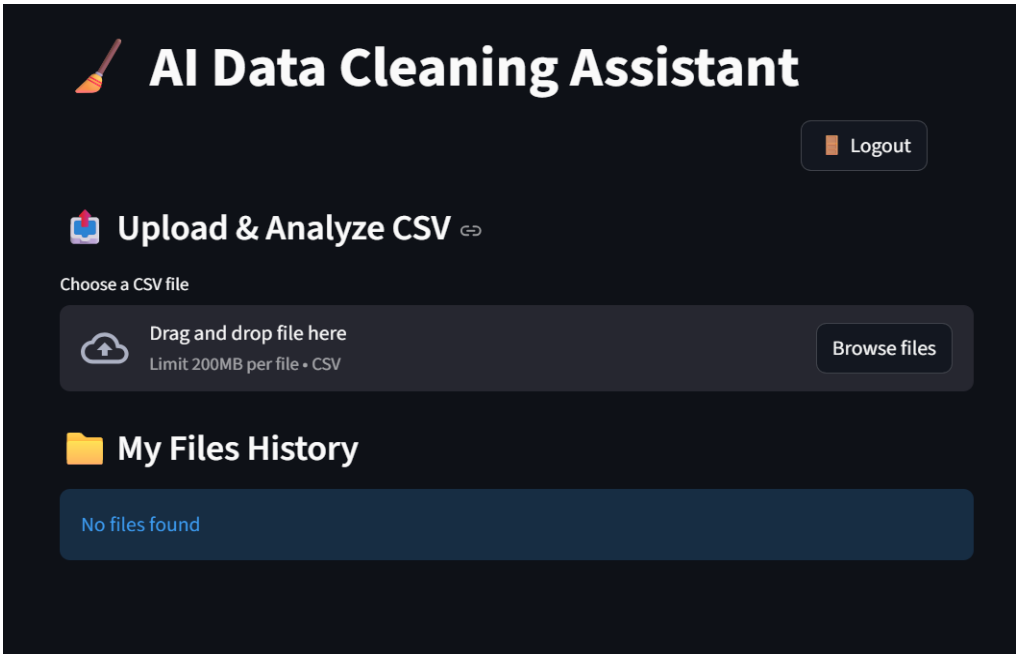

Fig. 4. CSV Upload Interface

Context:
This screenshot shows the CSV upload interface, where users can upload datasets using a drag-and-drop mechanism or by browsing local files. The system validates the file format and size before allowing further processing, ensuring only valid CSV files are accepted.

Fig. 5 CSV File Analysis Completion

Context:
This figure illustrates the successful analysis of an uploaded CSV file. After analysis, the system confirms completion and prepares the dataset for cleaning by identifying issues such as missing values and duplicate records. This step provides transparency before applying cleaning operations.

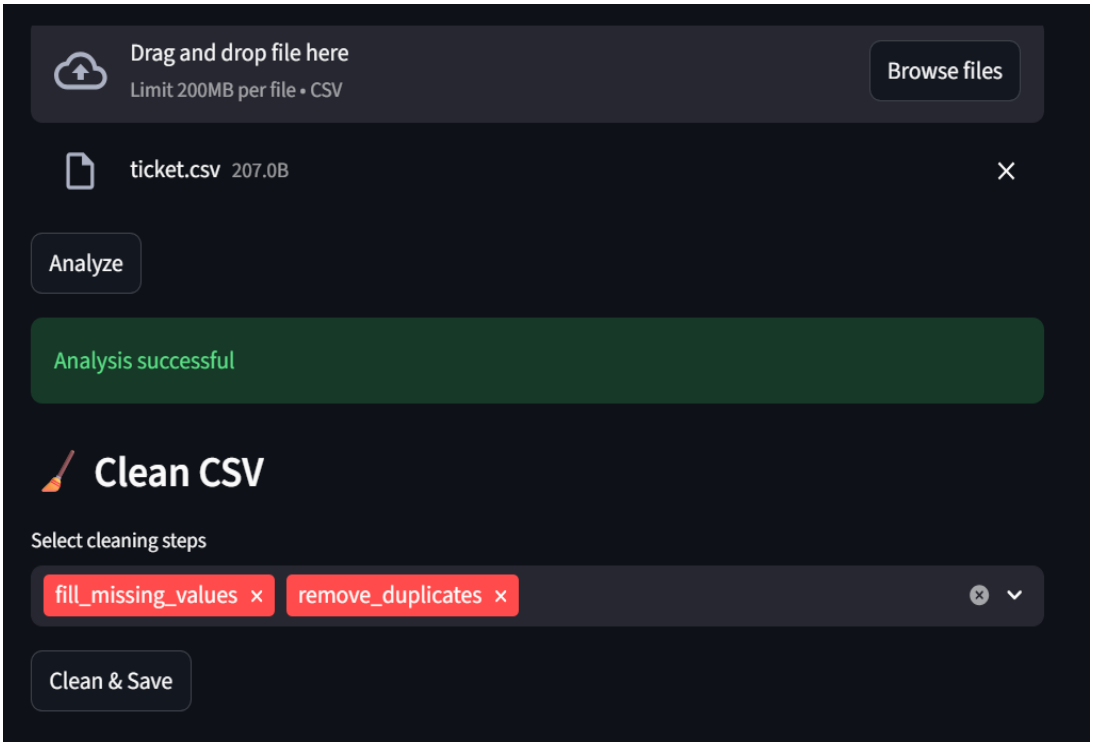

Fig. 5. Selection of Data Cleaning Operations

Context:
This screenshot displays the data cleaning module, where users can select specific cleaning steps such as filling missing values and removing duplicate rows. The modular design allows users to control how the dataset is cleaned instead of applying fixed transformations.

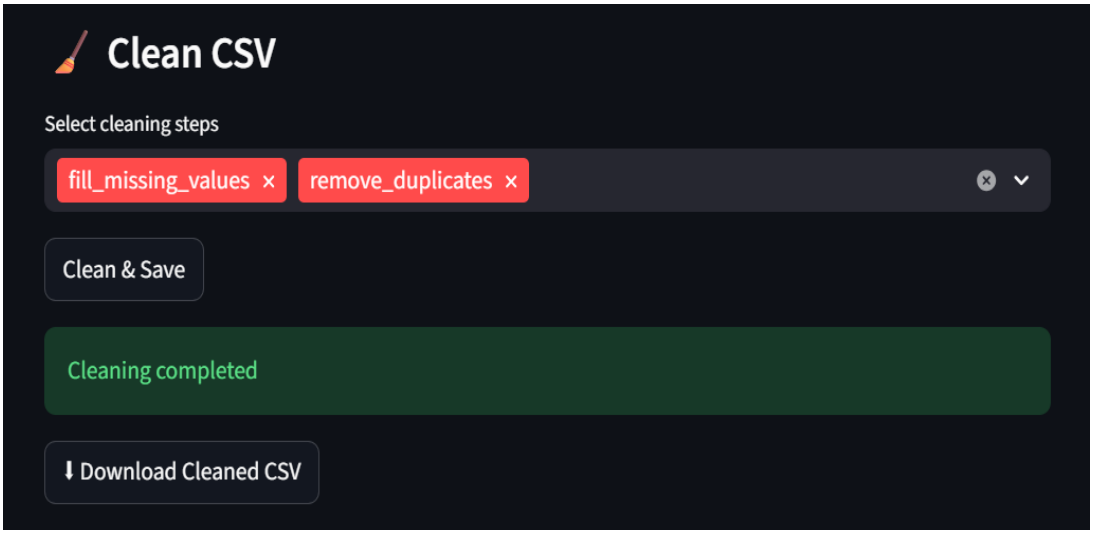

Fig. 6. Data Cleaning Execution and Completion

Context:
This figure shows the execution and successful completion of the selected data cleaning operations. Once processed, the system generates a cleaned CSV file and provides an option to download it. This confirms the effectiveness of the cleaning pipeline.

Fig. 7. Cleaning Summary Generation

Context:
This screenshot presents the cleaning summary section, which displays details such as the original file name, cleaned file name, applied cleaning steps, and analysis metadata. The summary can be exported in CSV or PDF format, enabling documentation and reporting.

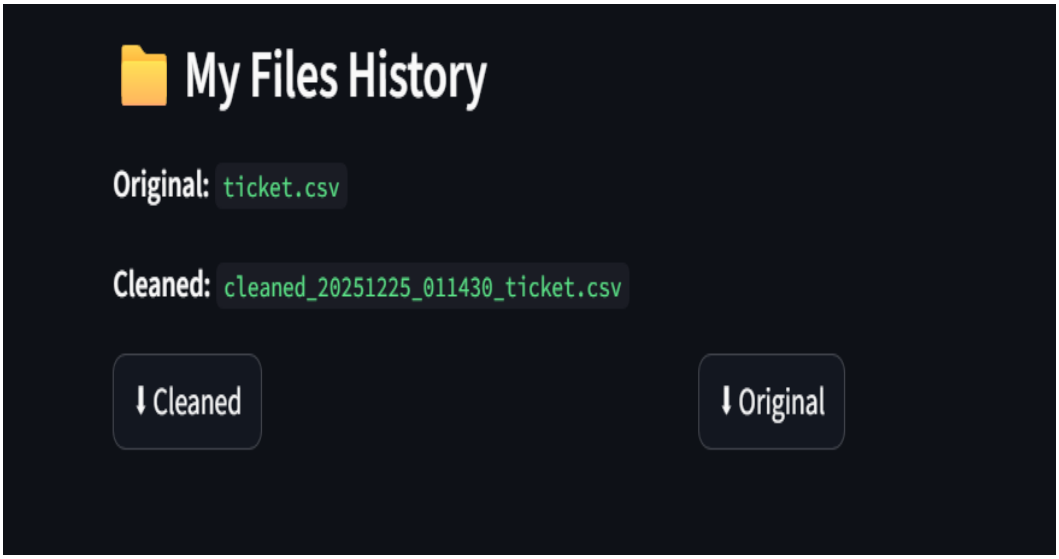

Fig. 8. User File History and Download Options

Context:
This figure shows the user-specific file history interface. It allows users to view previously processed files and download either the original or cleaned versions. This feature ensures traceability, accountability, and secure access to user data.

V. TECHNOLOGY STACK

A. Backend Technologies

- Python
- FastAPI
- SQLAlchemy
- PostgreSQL
- Pandas
- JSON Web Tokens (JWT)
- Passlib (bcrypt)
- Sklearn

B. Frontend Technologies

- Streamlit
- Requests library

VI. PROJECT STRUCTURE

```
ai-data-cleaning-backend/
│
├── app/
│   ├── main.py
│   ├── database.py
│   ├── models.py
│   │
│   ├── auth/
│   │   ├── router.py
│   │   ├── schemas.py
│   │   ├── deps.py
│   │   └── security.py
│   │
│   ├── upload/
│   │   ├── router.py
│   │   ├── temp_store.py
│   │   └── history_router.py
│   │
│   └── cleaning/
│       ├── analyze.py
│       └── agent.py
│
├── uploads/
├── cleaned/
├── frontend_streamlit.py
├── requirements.txt
├── run_app.bat
├── stop_app.bat
└── README.md
```

VII. FUNCTIONAL MODULES

A. Authentication Module

- User registration and login
- Password hashing using bcrypt
- JWT token generation upon successful login
- Token validation for all protected endpoints

Authentication Mechanism:

The system uses JWT-based stateless authentication. After logging in, a JWT token is issued and passed via the Authorisation header for protected API calls. The backend supports direct token usage and optionally accepts Bearer-prefixed tokens.

B. CSV Upload and Analysis Module

- Accepts CSV file uploads
- Detects:
    - Missing values
    - Duplicate rows
    - Dataset structure
- Handles empty and invalid CSV files

C. Data Cleaning Module

- User-selected cleaning operations:
    - Filling missing values
    - Removing duplicate rows
- Cleaned CSV generation
- Temporary data management using unique identifiers

D. Download Module

- Secure download of cleaned CSV files
- Secure download of original CSV files
- User-specific authorization enforced

E. History and Reporting Module

- Per-user file history storage
- Cleaning summary generation
- Export summary as CSV and PDF

VIII. API ENDPOINTS

Endpoint	Method	Description
/auth/register	POST	Register a new user
/auth/login	POST	User authentication
/upload/analyze	POST	Analyse CSV file
/upload/clean/{temp_id}	POST	Apply cleaning steps
/files/my	GET	Retrieve file history
/files/download/{filename}	GET	Download cleaned file
/files/download/original/{filename}	GET	Download original file

VIII.I FASTAPI SCREENSHOT

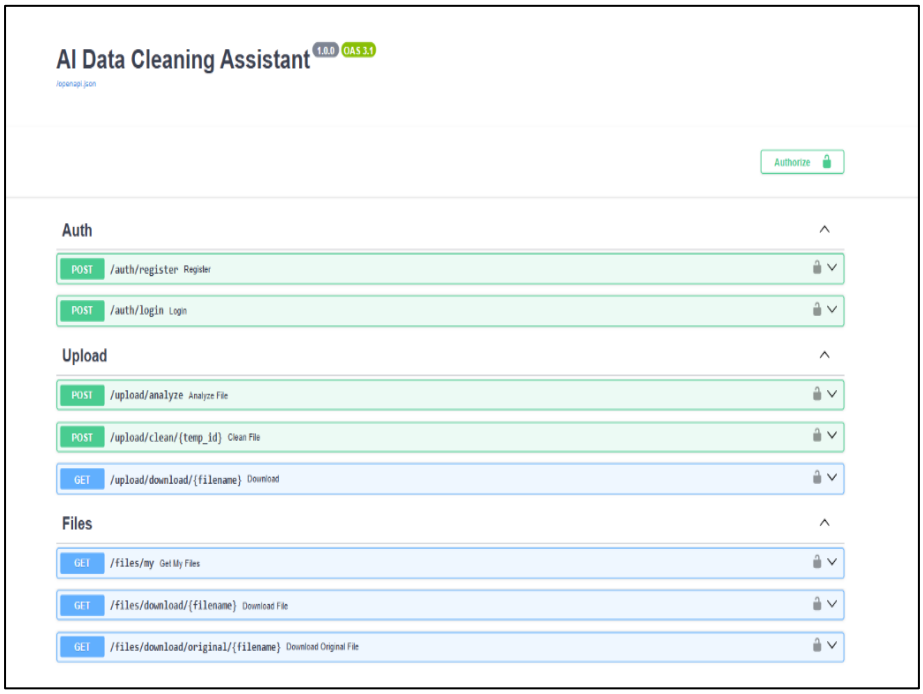

Fig. 10. FastAPI Swagger UI for AI Data Cleaning Assistant

Context:
This figure shows the Swagger UI automatically generated by the FastAPI backend for the AI Data Cleaning Assistant. It provides an interactive interface to explore and test all REST API endpoints, including authentication, file upload, data analysis, cleaning, and download operations. The Swagger UI helps in validating API functionality and ensures transparency in backend operations during development and testing.

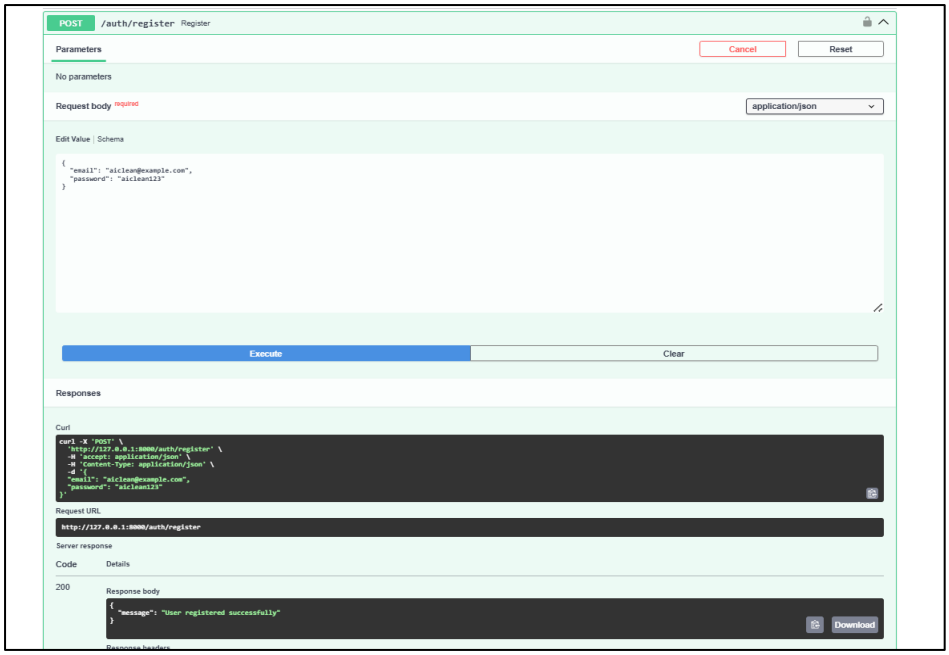

Fig. 11. User Registration API Execution

Context:
This screenshot demonstrates the execution of the /auth/register endpoint, where a new user is registered by submitting credentials in JSON format. Upon successful execution, the system returns a confirmation message indicating successful user registration. Passwords are securely hashed before being stored in the database, ensuring user security.

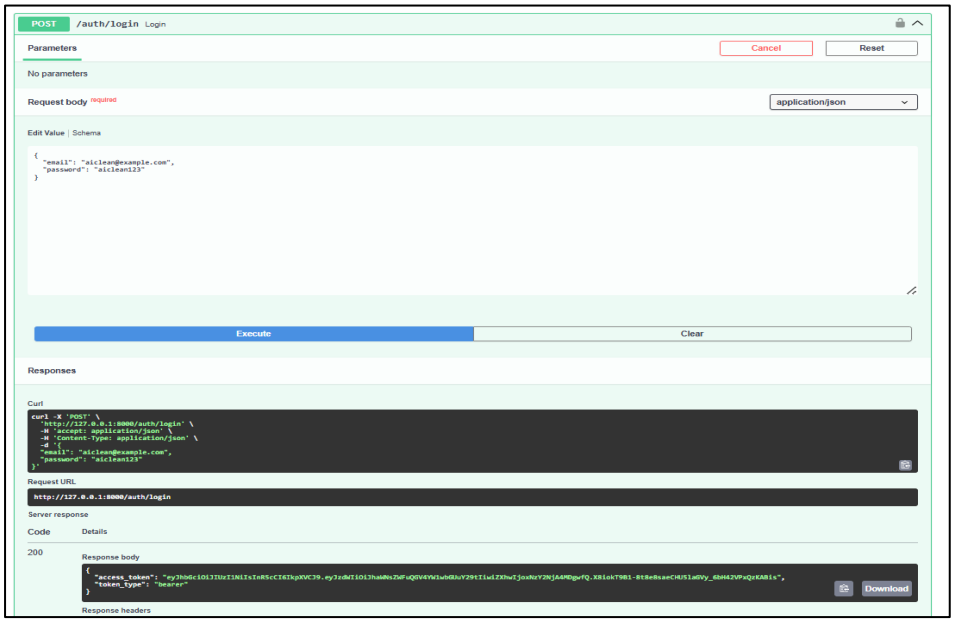

Fig. 12. User Login API and JWT Token Generation

Context:
This figure illustrates the /auth/login endpoint execution. After providing valid credentials, the system generates a JSON Web Token (JWT), which is used for authenticating subsequent protected API requests. This confirms the implementation of stateless authentication in the backend

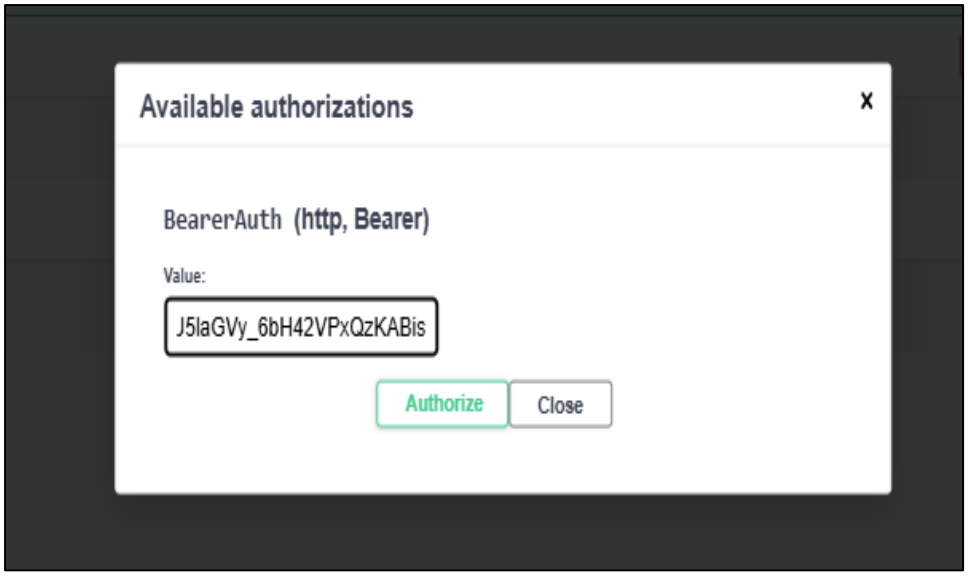

Fig. 13. JWT Authorization Using Bearer Token

Context:
This figure illustrates the authorization mechanism in the Swagger UI, where the generated JSON Web Token (JWT) is provided using the Bearer authentication scheme. Once authorized, the user gains secure access to protected endpoints, including CSV upload, data cleaning, and file download operations.

Authentication Mechanism:
The system implements JWT-based stateless authentication. After successful login, a JWT token is issued to the user and passed through the Authorization header for all protected API requests. The backend validates the token for each request and supports both direct token usage and Bearer-prefixed tokens, ensuring secure and flexible authentication.

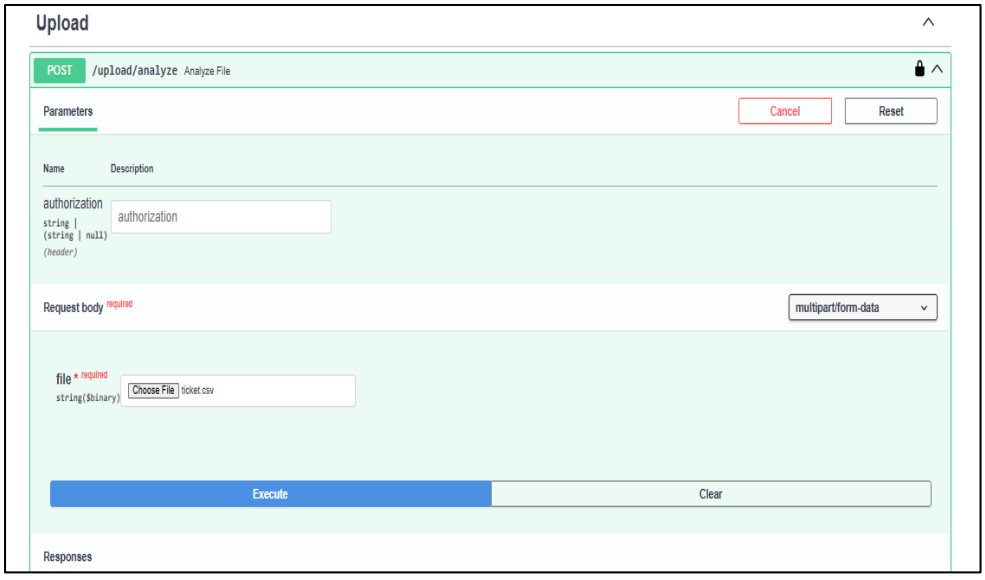

Fig. 14. CSV File Upload and Analysis Endpoint

Context:
This figure displays the /upload/analyze endpoint in action, where a CSV file is uploaded for analysis. The backend processes the file and identifies data quality issues such as missing values, duplicate rows, column data types, and suggests appropriate cleaning steps. The analysis results are returned in JSON format along with a temporary file identifier.

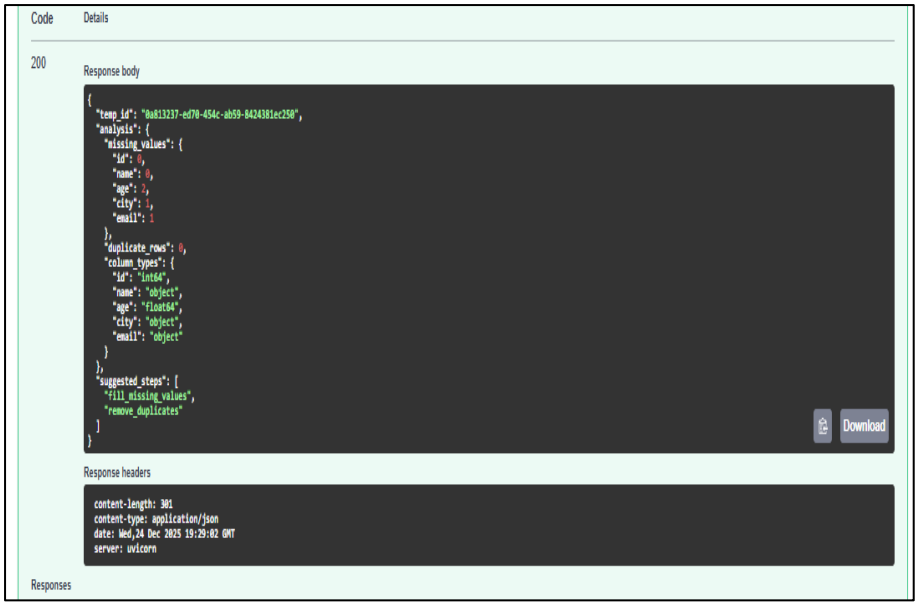

Fig. 15. Response: CSV Analyse

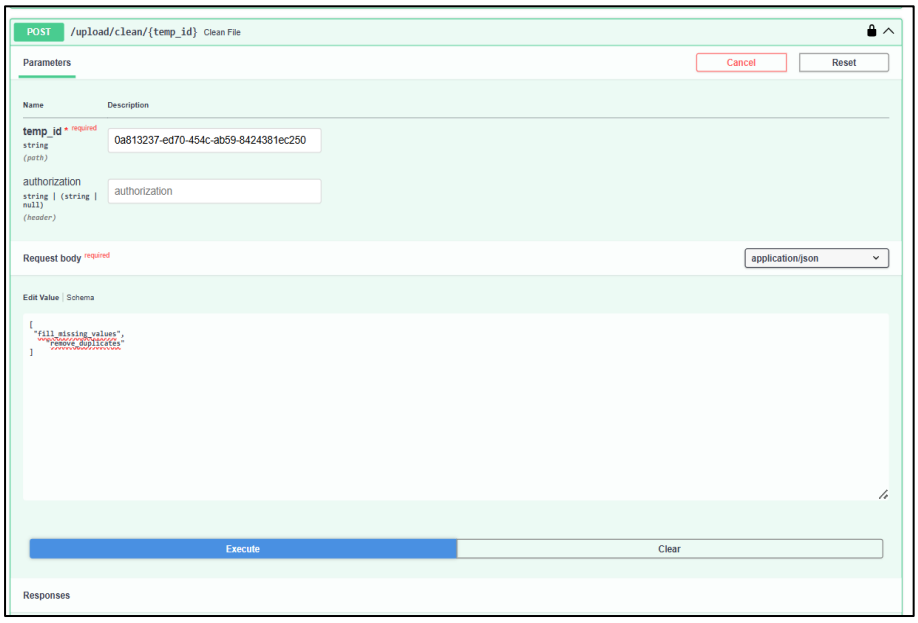

Fig. 16. Data Cleaning Operation Execution

Context:
This screenshot demonstrates the /upload/clean/{temp_id} endpoint, where user-selected cleaning operations such as filling missing values and removing duplicates are applied to the uploaded dataset. The backend processes the file based on the provided temporary ID and returns a downloadable cleaned CSV file path.

Fig. 17. Response: downloadable cleaned CSV file

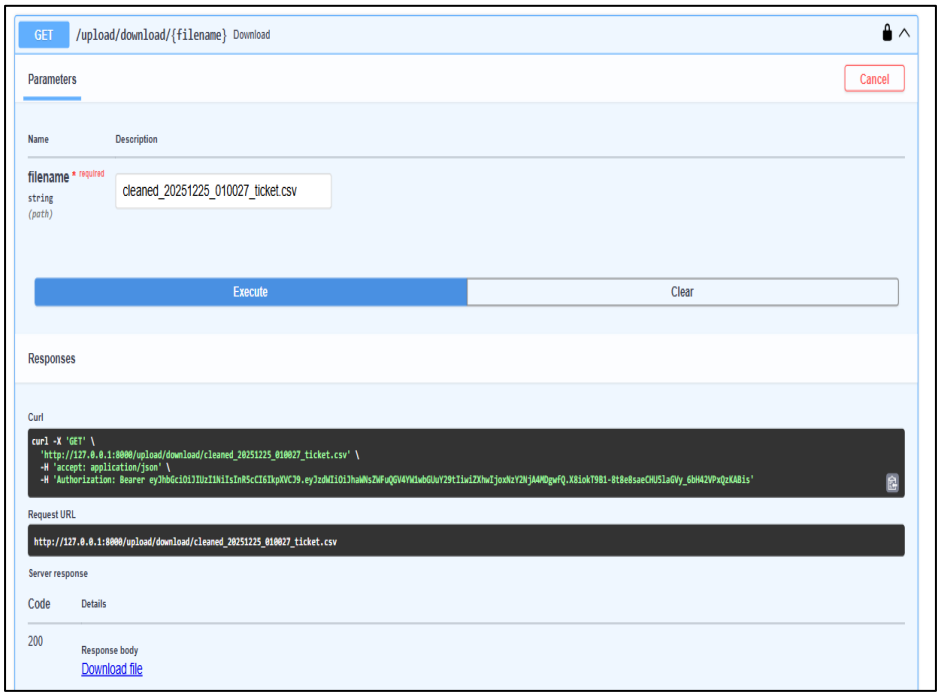

Fig. 18. Download of Cleaned CSV File

Context:
This figure shows the execution of the download endpoint for retrieving the cleaned CSV file. The API ensures that only authorized users can download their processed files, enforcing strict access control and data security.

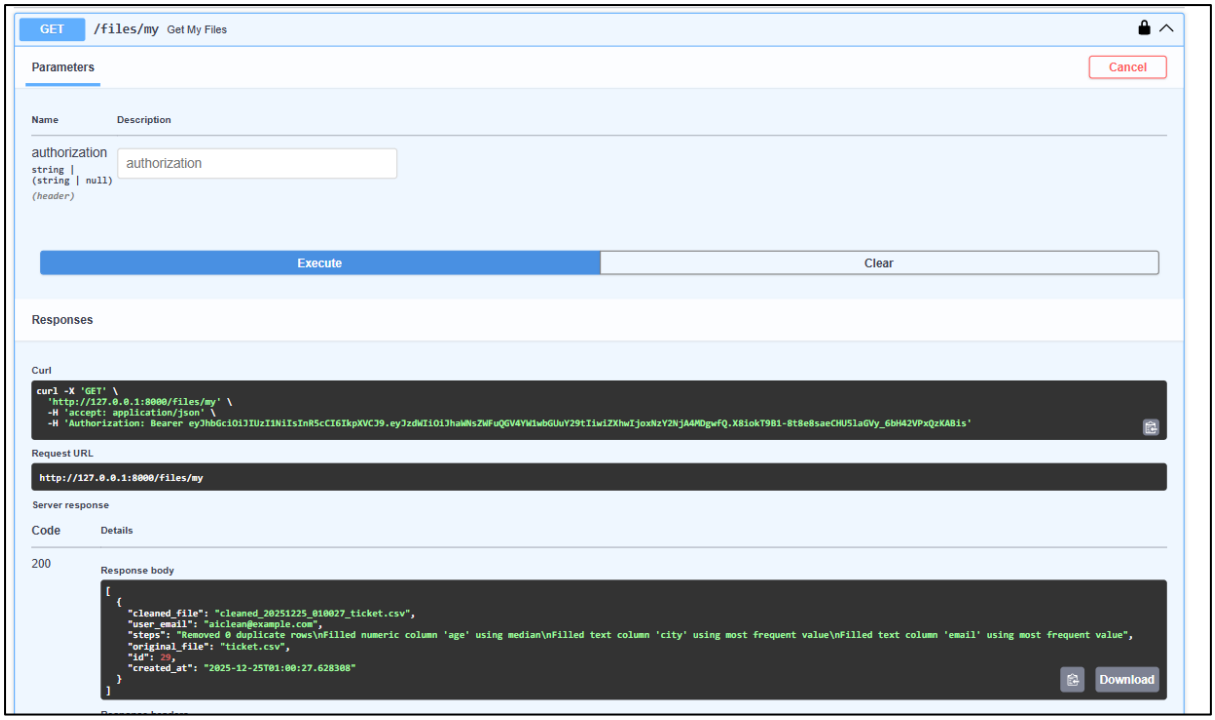

Fig. 19. User File History Retrieval

Context:

This screenshot illustrates the /files/my endpoint, which retrieves the authenticated user's file history. It displays metadata such as original file name, cleaned file name, applied cleaning steps, and timestamp. This feature provides transparency and traceability of all cleaning operations performed by the user.

IX. WORKFLOW DESCRIPTION

The system consists of the following workflows:

1. User authentication workflow
2. CSV upload and analysis workflow
3. Data cleaning workflow
4. Secure file download workflow
5. History and reporting workflow

X. EDGE CASE HANDLING

The system handles various edge cases, including:

1. Empty CSV files
2. Header-only CSV files
3. Invalid or expired JWT tokens
4. Unauthorized file access
5. Duplicate user registrations
6. Invalid input formats
7. History and reporting workflow

XI. SECURITY MEASURES

- Password hashing using bcrypt
- JWT-based stateless authentication
- Authorization checks for every API request
- User-specific file access control

XII. RESULTS

The system successfully detects data issues, applies cleaning operations accurately, and securely manages file access. The frontend and backend integration ensures smooth user experience and reliable data processing.

XIII. CONCLUSION

The AI Data Cleaning Assistant provides an efficient and secure solution for automated CSV data cleaning. The system combines modern API design, authentication mechanisms, and user-centric workflows to address real-world data preprocessing challenges.

XIV. FUTURE WORK

- Advanced data quality rules
- LangGraph-based multi-step orchestration
- Cloud storage integration
- Stronger password policies
- Data visualization dashboards

XV. GITHUB REPOSITORY

Github link: https://github.com/gmsivasai/ai-cleaning-agent.git