Student Performance Dashboard & Ticketing System

Team members

Mohamed Waleed & Youssef Nader:-

-Data Cleaning.
-50% System Design and Application Logic
-Data Visulaistation
-Matplotlip
-50% Sql and DBT Queryes
-File Generation.
-Api Choosing and Testing
-ER Diagram

Youssef Ahmed & Mahmoud Mohamed:-

-Streamlit And Code Aggregation.
-Visualistation of Outputs.
-50% Dbt Queryes
-50% System Design and Application Logic
-Airflow Dags
-Api Categorizing Analysis
-Files And Directory Handling

Project Overview

This is a comprehensive student performance management system built with Streamlit that provides rolebased access for students and administrators. The system integrates data processing, visualization, sentiment analysis, and automated email notifications.

Purpose

The application serves multiple purposes:

1. **Student Portal**: Allows students to view their academic performance and submit feedback
2. **Admin Portal**: Enables administrators to upload and process student data
3. **Automated Feedback System**: Uses sentiment analysis to route student feedback appropriately
4. **Data Pipeline**: Integrates with DBT and Apache Airflow for data transformation and workflow automation

System Architecture

Components

Frontend: Streamlit web application
Data Processing: Pandas for data cleaning and transformation
Sentiment Analysis: Hugging Face Transformers model
Data Pipeline: DBT (Data Build Tool) for SQL transformations **Workflow Automation**: Apache

Airflow for email notifications

Database: CSV files and SQL database integration

Application Structure

1. Main Entry Page (Main.py Purpose: Authentication and role-based routing **Features**:

Single ID-based login system
Validates user credentials against two databases: students_ids.csv - Student records admin.csv -

Administrator records

Stores user information in session state
Routes users to appropriate pages based on role

User Flow:

1. User enters their ID
2. System checks if ID exists in student or admin database
3. Student users gain access to Scores and Feedback pages
4. Admin users are redirected to Admin File Control page

2. Scores Page (1_Scores.py)
Purpose: Display student academic performance and overall statistics **Features for Students**:

Personal score display in tabular format
Subject scores: Math, Physics, Chemistry, Biology, English, History • Individual performance

metrics **Overall Statistics**:

Distribution histograms for all subjects showing:
Number of students per score range
Visual representation of class performance
Attendance heatmap (35x35 grid) showing:
Color-coded attendance patterns
Sorted attendance data visualization
Group-based attendance analysis

Visualizations:
Six histogram charts (one per subject)
Color scheme: Deep sky blue for consistency
Grid layout: Two columns for better organization **Access Control**:

Students: Full access to their data and class statistics
Admins: Restricted access (warning message)
Unauthenticated users: Redirected to login

3. Feedback Page (2_Feedback.py)

Purpose: Collect and analyze student feedback using AI

Features:

Text area for feedback submission
Real-time sentiment analysis using multilingual model
Automated email notification system

Sentiment Analysis:
Model: tabularisai/multilingual-sentiment-analysis
Categories:
Very Negative
Negative
Neutral
Positive
Very Positive

Workflow:

1. Student submits feedback text
2. AI model analyzes sentiment
3. Feedback and sentiment sent to Airflow DAG
4. Email notification triggered based on sentiment
5. User receives confirmation with sentiment classification **Integration**:

Apache Airflow REST API endpoint
Triggers email_feedback_sentiment_dag
Passes feedback text and sentiment as configuration

4. Admin Page (3_Admin.py)
Purpose: Data management and processing for administrators **Features**:

File upload interface (CSV, TXT, XLSX)
Automatic data cleaning and transformation
Database synchronization **Data Processing Pipeline**:

1. **File Upload & Validation**
Accepts CSV, TXT, XLSX formats
Displays preview of uploaded data
Data Cleaning (student_categorize function)
Removes duplicates
Fills missing subject scores with 0
Fills missing attendance with 0
Handles missing names (drops if all null, fills with "Unknown")
Converts birth dates to standard format (YYYY-MM-DD)
3. **Performance Calculation**
Computes average score across all subjects
Categorizes students:
High: Average ≥ 85
Medium: Average ≥ 60
Low: Average < 60
4. **Data Reordering**
Standardized column order for consistency
Places identification fields first, scores next, metrics last
5. **Export & Synchronization**
Saves cleaned data to students_cleaned.csv
Prepares SQL version (without Average_score and Performance)
Exports to DBT seeds directory
Executes DBT commands:
dbt seed : Loads data into database dbt run : Runs transformations

Control:
Admins: Full access to file management
Students: Restricted access (warning message)
Unauthenticated users: Redirected to login

5. Airflow DAG (email_dag.py)

Purpose: Automated email routing based on feedback sentiment **Workflow**:

1. **Recipient Selection** (choose_email function) Retrieves sentiment from DAG configuration

   Routes based on sentiment:
   Very Negative: Urgent email to youssefahmed65ah@gmail.com
   All Others: Standard email to fraction.ff.discord@gmail.com
2. **Email Task** (set_recipient_task) Python operator that determines recipient

   Stores email address in XCom for next task

3. **Send Email** (send_feedback_email)
   Email operator with dynamic recipient
   HTML formatted email including:
   Student information placeholders
   Sentiment classification
   Full feedback text

Configuration:
Manual trigger only (no schedule)
Tagged as "feedback" for organization
Started from Streamlit via REST API

Data Flow

1. Admin uploads student data → 2. Data cleaning & processing. Admin uploads student data 2. Data cleaning & processing
3. Export to students_cleaned.csv → 4. Copy to DBT seeds. Export to students_cleaned.csv 4. Copy to DBT seeds
5. DBT seed command → 6. Data loaded to SQL database. DBT seed command 6. Data loaded to SQL database
7. DBT run command → 8. SQL transformations executed. DBT run command 8. SQL transformations executed
9. Student accesses scores → 10. Data displayed from cleaned CSV. Student accesses scores 10. Data displayed from cleaned CSV
11. Student submits feedback → 12. Sentiment analysis. Student submits feedback 12. Sentiment analysis
13. Airflow DAG triggered → 14. Email sent based on sentiment. Airflow DAG triggered 14. Email sent based on sentiment

Technical Stack

Python 3.x
Streamlit: Web application framework
Pandas: Data manipulation and analysis
Matplotlib/NumPy: Data visualization
PyTorch & Transformers: Sentiment analysis
Apache Airflow: Workflow orchestration
DBT: Data transformation
SQL Database: Data warehouse

File Structure

project/project/
├── Main.py # Entry point and authentication ├── Main.py # Entry point and authentication
├── pages/ ├── pages/
│ ├── 1_Scores.py # Student performance dashboard │ ├── 1_Scores.py # Student performance dashboard
│ ├── 2_Feedback.py # Feedback submission system │ ├── 2_Feedback.py # Feedback submission system
│ └── 3_Admin.py # Admin file management │ └── 3_Admin.py # Admin file management
├── students_ids.csv # Student credentials ├── students_ids.csv # Student credentials
├── admin.csv # Admin credentials ├── admin.csv # Admin credentials
├── students_cleaned.csv # Processed student data ├── students_cleaned.csv # Processed student data
├── email_dag.py # Airflow DAG definition ├── email_dag.py # Airflow DAG definition
└── dbt_folder/ └── dbt_folder/

Key Features Summary

1. **Role-Based Access Control**: Separate interfaces for students and administrators
2. **Real-Time Data Processing**: Immediate cleaning and validation of uploaded files
3. **AI-Powered Sentiment Analysis**: Automatic categorization of student feedback
4. **Intelligent Email Routing**: Priority handling of negative feedback
5. **Comprehensive Visualizations**: Distribution charts and heatmaps for insights
6. **Automated Data Pipeline**: Seamless integration with DBT and SQL databases
7. **Session Management**: Persistent user state across pages

Security Considerations

Session-based authentication
Role-specific page access restrictions
Warning messages for unauthorized access attempts
No direct database credentials in code

Future Enhancements

Encrypt stored credentials
Add password-based authentication
Implement user registration system
Add more detailed analytics and reporting
Enable real-time database connections
Expand sentiment analysis capabilities
Add feedback response system for administrators