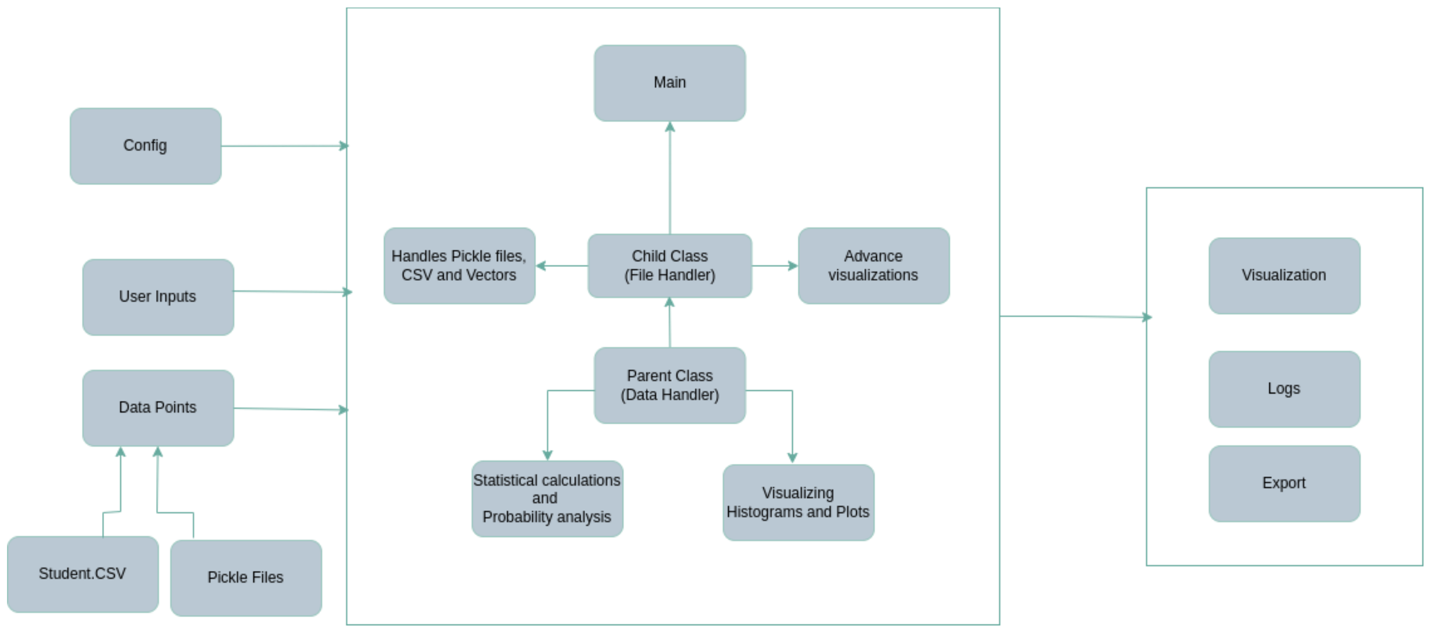
Module Communication Flow


Config
User Inputs
Data Points
Student.CSV
Pickle Files
Main
Handles Pickle files, CSV and Vectors
Child Class (File Handler)
Advance visualizations
Parent Class (Data Handler)
Statistical calculations and Probability analysis
Visualizing Histograms and Plots
Visualization
Logs
Export

Module Outline

Module list:

1. **Config.py**: Holds common configuration constants and global settings.
2. **main.py**: Acts as the entry point of the application, orchestrating the flow between modules and coordinating data handling, processing, visualization, and exporting tasks.
3. **LoggingModule.py**: Provides utility functions for logging progress and handling errors.

Project File Structure:

Project_root/

- Config.py – Configuration module with global settings
- main.py – Entry point of the application
- LoggingModule.py – Logging and error handling utilities

- **modules/** – Core data handling modules
    - DataHandler.py – Parent class for generic data operations
    - FileHandler.py – Child class for handling CSV and Pickle files
    - StatisticsHandler.py – Specialized class for statistical calculations
    - __init__.py – Init file to make this a package

- **utilities/** – Supporting modules for exporting and visualization
    - ExportModule.py – Module for exporting data and visualizations
    - VisualizationModule.py – Module for creating visualizations
    - __init__.py – Init file to make this a package

- **Input/** – Folder for input data files
    - Student.csv – Example CSV data file
    - example_data.pkl – Example Pickle data file

- **Output/** – Folder for exported results and images
    - results.csv – Exported CSV file
    - results.pkl – Exported Pickle file
    - visualizations/ – Folder for saved visualization images

- **logs/** – Folder for log files

- app.log – Log file for tracking progress and errors

Parent and Child Class

Parent Class: DataHandler

- **Responsibilities:**
  - Manages configuration constants for data processing and visualization.
  - Provides basic data handling functions, including data visualization and querying.
  - Acts as the base class for more specialized file handling classes.

- **Methods:**
  - __init__(self, config): Initializes the class with configuration constants.
  - read_file(self, file_path): Abstract method; intended to be overridden by child classes to read data.
  - query(self, condition): Queries data based on a simple condition (e.g., filtering specific values).
  - visualize(self, column, plot_type): Visualizes data with basic plot types, such as histograms and line plots.

Child Class 1.1: CSVHandler

- **Responsibilities:**
  - Inherits from DataHandler to handle CSV-specific file operations.
  - Reads data from a CSV file and stores it in a DataFrame.

- Provides enhanced visualization capabilities for CSV data, such as violin plots, box plots, and scatter plots.
- Queries the dataset using Boolean indexing to filter data based on multiple conditions.

- **Methods:**
  - __init__(self, config, file_path): Initializes the class with configuration constants and CSV file path.
  - read_file(self): Reads CSV data into a DataFrame.
  - visualize(self, column, plot_type): Provides enhanced visualization options specific to CSV data.
  - query(self, conditions): Uses Boolean indexing to query data with multiple conditions.

Parent Class: ProbabilityCalculator

- **Responsibilities:**
  - Manages statistical calculations for data, including vector operations and categorical data analysis.
  - Serves as the base class for statistical processing, extending support to various types of calculations.

- **Methods:**
  - __init__(self, config): Initializes the class with configuration constants.
  - calculate_statistics(self, data): Computes basic statistical values such as mean, median, and standard deviation.
  - calculate_probabilities(self, data): Calculates joint and conditional probabilities for the data.
  - vector_operations(self, vector1, vector2): Performs vector-based operations, such as dot product and angle calculations.

- analyze_categorical(self, data): Analyzes categorical attributes, generating unique values, permutations, and combinations.

Child Class 2.1: PickleHandler

- **Responsibilities:**
  - Inherits from ProbabilityCalculator to handle data processing for Pickle files.
  - Reads data from a Pickle file and processes it within a DataFrame.
  - Performs advanced statistical calculations and generates visualizations.

- **Methods:**
  - __init__(self, config, file_path): Initializes the class with configuration constants and Pickle file path.
  - read_file(self): Reads data from a Pickle file into a DataFrame.
  - calculate_joint_probabilities(self, data): Calculates joint and conditional probabilities.
  - vector_operations(self, vector1, vector2): Performs specific vector operations related to statistical analysis.

Functionalities and Features

- **Logging**
  - Provides global logging functionality for tracking progress and errors throughout the application.
  - Uses try-except blocks within functions to capture and handle errors.

- **File Handling**
  - Supports reading data from both CSV and Pickle files.
  - The CSVHandler manages CSV files, while the PickleHandler manages Pickle files.

- **Querying**
  - Allows querying of data based on specific conditions using methods in the DataHandler.
  - CSVHandler offers Boolean indexing for complex condition-based filtering.

- **Data Calculation**
  - ProbabilityCalculator provides functions for calculating statistical values such as mean, median, and standard deviation.
  - Calculates joint and conditional probabilities within datasets.
  - Vector operations include dot product calculations, unit vector generation, and angle determination.

- **Categorical Analysis**
  - Analyzes categorical data attributes to identify unique values.
  - Generates permutations and combinations for categorical attributes as required.

- **Data Visualization**
  - Basic visualization capabilities in DataHandler, including histograms and line plots.
  - Enhanced visualization options in CSVHandler, such as violin plots, box plots, and scatter plots.
  - VisualizationModule supports saving of visualizations as image files for easy reference.

- **Export**
  - Exports results to various formats, including CSV, Pickle, and image files.
  - ExportModule includes functions for saving processed data and visualizations to specified output directories.

- **Configuration Management**
  - Stores application-wide configuration settings, such as file paths, plot styles, and logging options, in Config.py.
  - Ensures consistent settings across modules, supporting ease of maintenance and flexibility.

Main.py

```
from Config import CONFIG
from modules.FileHandler import CSVHandler, PickleHandler
from utilities.ExportModule import export_csv
from utilities.VisualizationModule import create_visualizations


def main():
    # Load and process CSV data
    csv_handler = CSVHandler(config=CONFIG,
file_path="./Input/input_data.csv")
    csv_data = csv_handler.read_file()


    # Visualize attendance data
    csv_handler.visualize(column="Attendance", plot_type="histogram")
    csv_handler.visualize(column="Category", plot_type="box")


    # Load and process Pickle data
    pickle_handler = PickleHandler(config=CONFIG,
file_path="./Input/input_data.pkl")
```

```
    pickle_data = pickle_handler.read_file()


    # Export a sample of CSV data
    export_csv(csv_data.head(), "./Output/sample_data.csv")


    # Additional visualizations for comparison
    create_visualizations(csv_data, columns=["Attendance",
"District"], save_path="./Output/visualizations/")


if __name__ == "__main__":
    main()
```

Config

```
CONFIG = {
    "plot_style": "seaborn",
    "log_file": "app.log",
    "output_folder": "./Output/"
}
```

module_tmp.py

# Version: Draft

# Date Last Updated: 2024-11-10

#%% MODULE BEGINS

module_name = 'DataHandler'

'''

Version: Draft

Description: Provides base classes and utilities for handling CSV and Pickle files, as well as logging and basic error handling.

Authors: Sofiat Adeyemi, Jayden Stewart, Sarseej Shrestha

Date Created: 2024-11-13

'''

```
#%% IMPORTS
from Config import CONFIG
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import pickle

#%% USER INTERFACE VARIABLES
ALLOWED_PLOT_TYPES = ['violin', 'box', 'histogram']
DEFAULT_PLOT_STYLE = CONFIG.get("plot_style", "seaborn")

#%% INITIALIZE PLOT STYLE
plt.style.use(DEFAULT_PLOT_STYLE)

#%% LOGGING FUNCTION
def log_progress(message):
    """Simple log function for tracking progress."""
    with open(CONFIG.get("log_file", "app.log"), "a") as log_file:
        log_file.write(f"{message}\n")
```

```
#%% ERROR HANDLING DECORATOR

def catch_errors(func):

    """Decorator for catching and logging errors."""

    def wrapper(*args, **kwargs):

        try:

            return func(*args, **kwargs)

        except Exception as e:

            log_progress(f"Error in {func.__name__}: {e}")

    return wrapper


#%% BASE CLASS

class BaseHandler:

    """Base class for file handling operations."""

    def __init__(self, config, file_path):

        """Initialize with config and file path."""

        self.config = config

        self.file_path = file_path

        self.data = None


    @catch_errors

    def read_file(self):

        """Abstract method for file reading (to be implemented in
subclasses)."""

        raise NotImplementedError("Subclasses should implement this
method.")


#%% CSV HANDLER CLASS
```

```
class CSVHandler(BaseHandler):
    """Handles CSV file operations."""

    @catch_errors
    def read_file(self):
        """Reads CSV file into a DataFrame."""
        self.data = pd.read_csv(self.file_path)
        return self.data

    @catch_errors
    def visualize(self, column, plot_type='violin'):
        """Creates basic visualizations for a specified column."""
        if plot_type not in ALLOWED_PLOT_TYPES:
            raise ValueError(f"Invalid plot type. Allowed types are:
{ALLOWED_PLOT_TYPES}")

        plt.figure(figsize=(10, 6))
        if plot_type == 'violin':
            sns.violinplot(y=self.data[column])
        elif plot_type == 'box':
            sns.boxplot(y=self.data[column])
        else:
            sns.histplot(self.data[column])

        plt.title(f"{plot_type.capitalize()} Plot of {column}")

plt.savefig(f"{self.config['output_folder']}/{column}_{plot_type}.png"
)
```

```
        plt.close()


#%% PICKLE HANDLER CLASS
class PickleHandler(BaseHandler):
    """Handles Pickle file operations."""


    @catch_errors
    def read_file(self):
        """Reads Pickle file into a DataFrame or dictionary."""
        with open(self.file_path, 'rb') as f:
            self.data = pickle.load(f)
        return self.data


    @catch_errors
    def calculate_statistics(self):
        """Calculates basic statistics if data is a DataFrame."""
        if isinstance(self.data, pd.DataFrame):
            return self.data.describe()
        raise TypeError("Data must be a pandas DataFrame to calculate
statistics.")


#%% TEST MAIN FUNCTION
def main():
    """Basic testing function for CSV and Pickle handlers."""
    # Test CSV handler
    csv_handler = CSVHandler(CONFIG, "./Input/input_data.csv")
    csv_handler.read_file()
    csv_handler.visualize(column="Attendance", plot_type="histogram")

    # Test Pickle handler
    pickle_handler = PickleHandler(CONFIG, "./Input/input_data.pkl")
    pickle_handler.read_file()
    stats = pickle_handler.calculate_statistics()
    print(stats)

#%% SELF-RUN
if __name__ == "__main__":
    main()
```

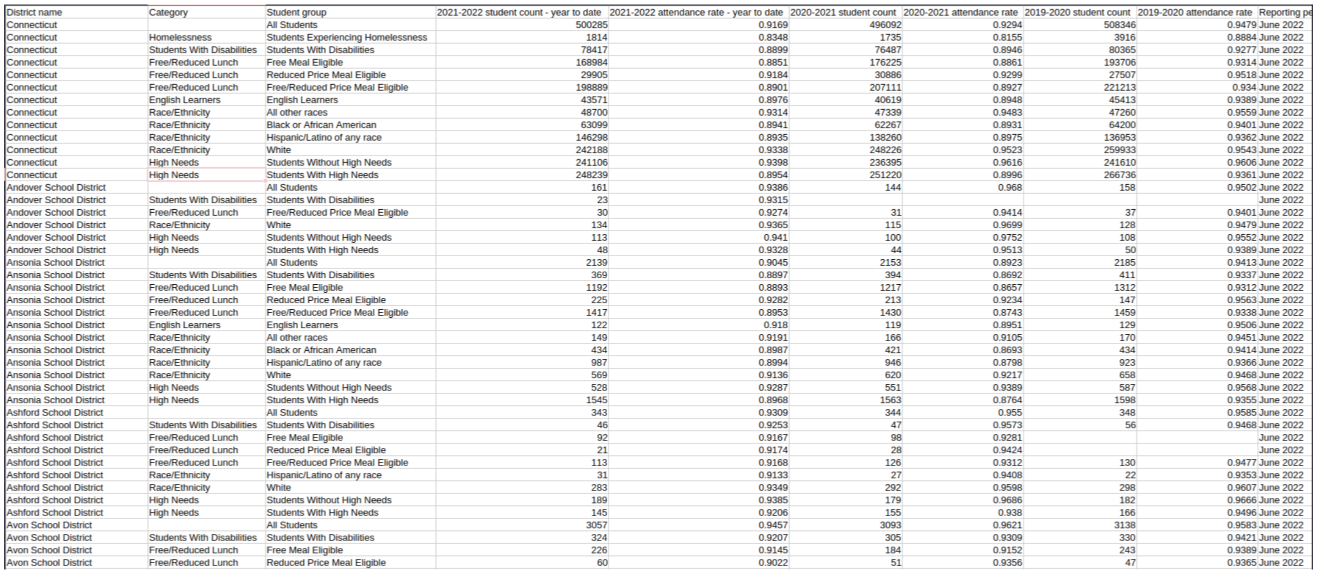
Input Data Format

“Student Attendance Records-2022"

An In-depth analysis of student issues

Sofiat Adeyemi. Jayden Stewart, Sarseej Shrestha

Example of the data format:



District name	Category	Student group	2021-2022 student count - year to date	2021-2022 attendance rate - year to date	2020-2021 student count	2020-2021 attendance rate	2019-2020 student count	2019-2020 attendance rate	Reporting pe
Connecticut		All Students	500285	0.9169	496092	0.9294	508346	0.9479	June 2022
Connecticut	Homelessness	Students Experiencing Homelessness	1814	0.8348	1735	0.8155	3916	0.8884	June 2022
Connecticut	Students With Disabilities	Students With Disabilities	78417	0.8899	76487	0.8946	80365	0.9277	June 2022
Connecticut	Free/Reduced Lunch	Free Meal Eligible	168984	0.8851	176225	0.8861	193706	0.9314	June 2022
Connecticut	Free/Reduced Lunch	Reduced Price Meal Eligible	29905	0.9184	30886	0.9299	27507	0.9518	June 2022
Connecticut	Free/Reduced Lunch	Free/Reduced Price Meal Eligible	198889	0.8901	207111	0.8927	221213	0.934	June 2022
Connecticut	English Learners	English Learners	43571	0.8976	40619	0.8948	45413	0.9389	June 2022
Connecticut	Race/Ethnicity	All other races	48700	0.9314	47339	0.9483	47260	0.9559	June 2022
Connecticut	Race/Ethnicity	Black or African American	63099	0.8941	62267	0.8931	64200	0.9401	June 2022
Connecticut	Race/Ethnicity	Hispanic/Latino of any race	146298	0.8935	138260	0.8975	136953	0.9362	June 2022
Connecticut	Race/Ethnicity	White	242188	0.9338	248226	0.9523	259933	0.9543	June 2022
Connecticut	High Needs	Students Without High Needs	241106	0.9398	236395	0.9616	241610	0.9606	June 2022
Connecticut	High Needs	Students With High Needs	248239	0.8954	251220	0.8996	266736	0.9361	June 2022
Andover School District		All Students	161	0.9386	144	0.968	158	0.9502	June 2022
Andover School District	Students With Disabilities	Students With Disabilities	23	0.9315					June 2022
Andover School District	Free/Reduced Lunch	Free/Reduced Price Meal Eligible	30	0.9274	31	0.9414	37	0.9401	June 2022
Andover School District	Race/Ethnicity	White	134	0.9365	115	0.9699	128	0.9479	June 2022
Andover School District	High Needs	Students Without High Needs	113	0.941	100	0.9752	108	0.9552	June 2022
Andover School District	High Needs	Students With High Needs	48	0.9328	44	0.9513	50	0.9389	June 2022
Ansonia School District		All Students	2139	0.9045	2153	0.8923	2185	0.9413	June 2022
Ansonia School District	Students With Disabilities	Students With Disabilities	369	0.8897	394	0.8692	411	0.9337	June 2022
Ansonia School District	Free/Reduced Lunch	Free Meal Eligible	1192	0.8893	1217	0.8657	1312	0.9312	June 2022
Ansonia School District	Free/Reduced Lunch	Reduced Price Meal Eligible	225	0.9282	213	0.9234	147	0.9563	June 2022
Ansonia School District	Free/Reduced Lunch	Free/Reduced Price Meal Eligible	1417	0.8953	1430	0.8743	1459	0.9338	June 2022
Ansonia School District	English Learners	English Learners	122	0.918	119	0.8951	129	0.9506	June 2022
Ansonia School District	Race/Ethnicity	All other races	149	0.9191	166	0.9105	170	0.9451	June 2022
Ansonia School District	Race/Ethnicity	Black or African American	434	0.8987	421	0.8693	434	0.9414	June 2022
Ansonia School District	Race/Ethnicity	Hispanic/Latino of any race	987	0.8994	946	0.8798	923	0.9366	June 2022
Ansonia School District	Race/Ethnicity	White	569	0.9136	620	0.9217	658	0.9468	June 2022
Ansonia School District	High Needs	Students Without High Needs	528	0.9287	551	0.9389	587	0.9568	June 2022
Ansonia School District	High Needs	Students With High Needs	1545	0.8968	1563	0.8764	1598	0.9355	June 2022
Ashford School District		All Students	343	0.9309	344	0.955	348	0.9585	June 2022
Ashford School District	Students With Disabilities	Students With Disabilities	46	0.9253	47	0.9573	56	0.9468	June 2022
Ashford School District	Free/Reduced Lunch	Free Meal Eligible	92	0.9167	98	0.9281			June 2022
Ashford School District	Free/Reduced Lunch	Reduced Price Meal Eligible	21	0.9174	28	0.9424			June 2022
Ashford School District	Free/Reduced Lunch	Free/Reduced Price Meal Eligible	113	0.9168	126	0.9312	130	0.9477	June 2022
Ashford School District	Race/Ethnicity	Hispanic/Latino of any race	31	0.9133	27	0.9408	22	0.9353	June 2022
Ashford School District	Race/Ethnicity	White	283	0.9349	292	0.9598	298	0.9607	June 2022
Ashford School District	High Needs	Students Without High Needs	189	0.9385	179	0.9686	182	0.9666	June 2022
Ashford School District	High Needs	Students With High Needs	145	0.9206	155	0.938	166	0.9496	June 2022
Avon School District		All Students	3057	0.9457	3093	0.9621	3138	0.9583	June 2022
Avon School District	Students With Disabilities	Students With Disabilities	324	0.9207	305	0.9309	330	0.9421	June 2022
Avon School District	Free/Reduced Lunch	Free Meal Eligible	226	0.9145	184	0.9152	243	0.9389	June 2022
Avon School District	Free/Reduced Lunch	Reduced Price Meal Eligible	60	0.9022	51	0.9356	47	0.9365	June 2022

Explanation of the Data structure:

- District name: Specifies the district
- Category: Specifies the category of the students
- Student group: Specifies the group of student under a specific category
- S_Count22: Student count from 2021 to 2022
- S_Count21: Student count from 2020 to 2021
- S_Count20: Student count from 2019 to 2020
- S_Attendance22: Student attendance rate from 2021 to 2022
- S_Attendance21: Student attendance rate from 2020 to 2021
- S_Attendance20: Student attendance rate from 2019 to 2020
- R_Period: Time when the report was made
- D_Update: Time when report was last updated

Objectives of the dataset:

- To identify key issues that students are facing in different districts.
- To understand student groups based on ethnic and social conditions such that their issues are to be alleviated.

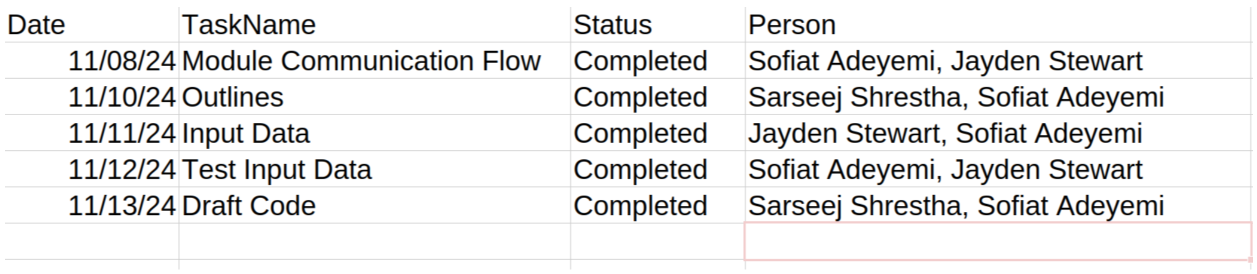
Source:

https://catalog.data.gov/dataset/school-attendance-by-student-group-and-district-2021-2022

Github URL:

https://github.com/sofiaunnie/CS340_-S24-_-ByteBee-.

Team Progress Report:



Date	TaskName	Status	Person
11/08/24	Module Communication Flow	Completed	Sofiat Adeyemi, Jayden Stewart
11/10/24	Outlines	Completed	Sarseej Shrestha, Sofiat Adeyemi
11/11/24	Input Data	Completed	Jayden Stewart, Sofiat Adeyemi
11/12/24	Test Input Data	Completed	Sofiat Adeyemi, Jayden Stewart
11/13/24	Draft Code	Completed	Sarseej Shrestha, Sofiat Adeyemi