Solent University

Department of Science and Engineering

Student Documentation Software

Author : **Q102543890 - Anandhakrishnan Madathil Remesh**

Course Title : **Programming for problem- solving (COM728)**

Module Leader : **Jarutas Andritsch**

Date : **24-08-2024**

Table of contents

Table of figures

Index of Tables

1 Overview

Analyzing student performance is vital in education. Technology has made software for managing student records crucial globally. This introduction emphasizes the benefits of using student records to assess performance. Software has changed how we collect, process, and analyze student data, providing insights for personalized learning and better decisions. Using technology for student records improves education, helping students succeed and stay healthy.

Table 1: Requirement Completion

Requirement	**Status**
Users need to input the file path or filename for file access	**COMPLETED**
Load the data from a CSV file into memory using the CSV module reader function() and store it in list	**COMPLETED**
Retrieve the sex, age, number of relative in a family, state, and race based on the ID	**COMPLETED**
Retrieve the sex, school support, access internet, attendance rate, and parental involvement associated with a specific race	**COMPLETED**
Retrieve the ID, free time, math score, reading score and writing score of students whose absences are less than 50 based on the parental involvement	**COMPLETED**
Retrieve information from your chosen columns and apply a specific condition that relates to an individual student.	**COMPLETED**
system loads data from a CSV file into memory using the read_csv() function, utilising the file path or filename obtained	**COMPLETED**
Analyse the average math score of students with an attendance rate greater than 80%, based on race	**COMPLETED**
Analyse the average number of absences among students with a particular level of parental involvement.	**COMPLETED**
Identify the top 3 levels of mother's education for a specific race of students based on the parental involvement levels.	**COMPLETED**

Analyse the data to derive meaningful insights about academic performance based on your unique selection and condition, distinct from the previous requirements.	**COMPLETED**
use the data frame which is already loaded	**COMPLETED**
Create a chart to illustrate the proportion of students in this school based on their race	**COMPLETED**
Create a chart to visually compare the average writing scores among students in each race group	**COMPLETED**
Create a chart to illustrate the relationship between students' reading and writing scores.	**COMPLETED**
Create a visualisation of your selection to showcase information related to student performance that can reveal trends, different, behaviours, or patterns, ensuring it is distinct from previous requirements.	**COMPLETED**

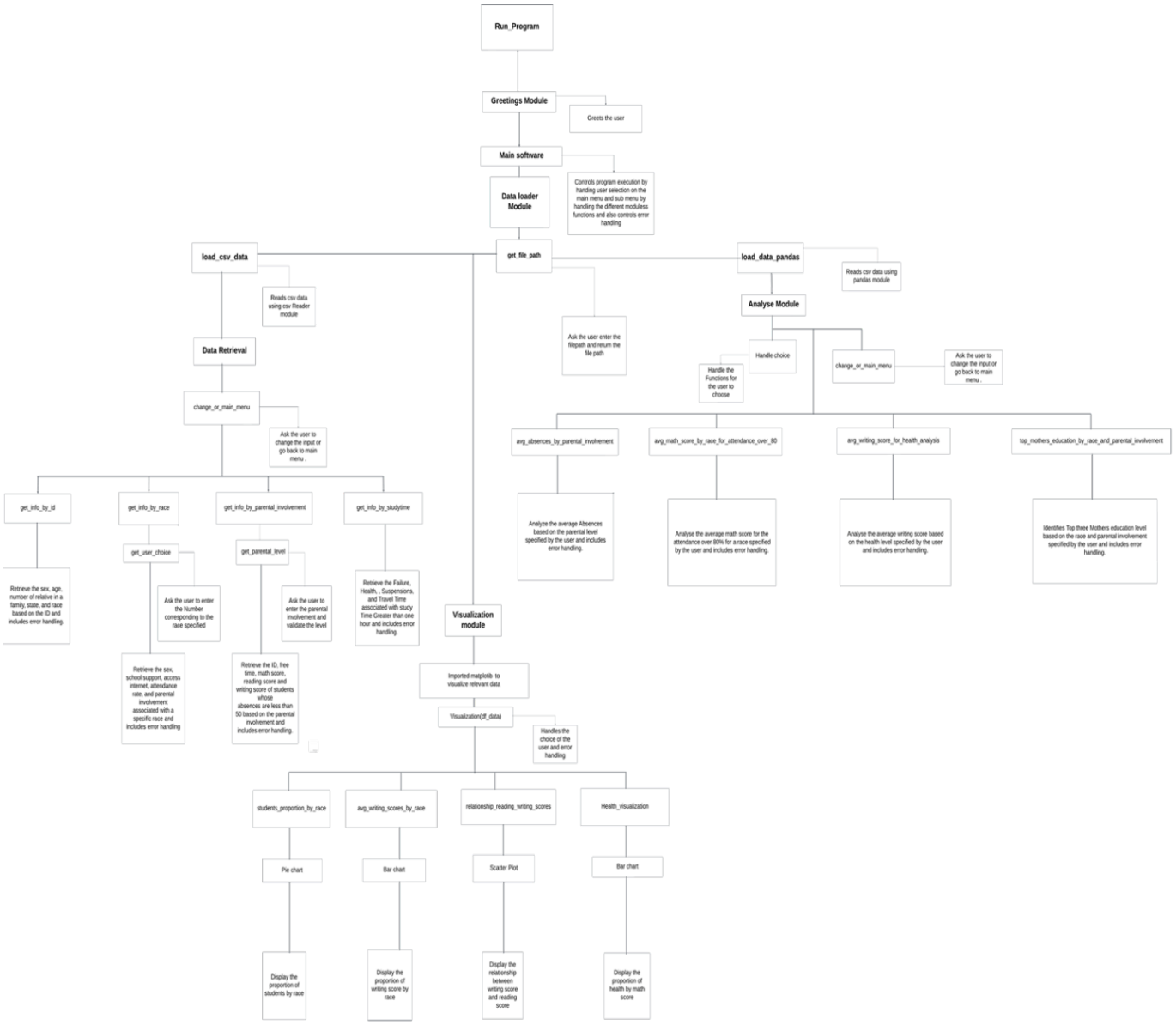
Status options: Completed/ Partially Completed/ Not Attempted

2 Project Implementation

The aim of this project is to create an interactive system for handling, organizing, and analyzing student data. Built with Python, the system includes a program that imports data from a CSV file and offers multiple functions for accessing, analyzing, and visualizing that data.

2.1 Project structure

The program is structured in a way that various modules and functions make it easier to navigate and use. The program is coded in such a way that an error in one module will not hinder the operation of the entire program. The main program file is Main Software code.py which contains the main program loop and user interface. The program allows a user to enter the path to the CSV file containing the student's data, students_data.csv.


Run_Program
Greetings Module
Greets the user
Main software
Data loader Module
Controls program execution by handing user selection on the main menu and sub menu by handling the different modulesss functions and also controls error handling
load_csv_data
Reads csv data using csv Reader module
get_file_path
Ask the user enter the filepath and return the file path
load_data_pandas
Reads csv data using pandas module
Data Retrieval
change_or_main_menu
Ask the user to change the input or go back to main menu .
Analyse Module
Handle choice
Handle the Functions for the user to choose
change_or_main_menu
Ask the user to change the input or go back to main menu .
avg_absences_by_parental_involvement
avg_math_score_by_race_for_attendance_over_80
avg_writing_score_for_health_analysis
top_mothers_education_by_race_and_parental_involvement
Analyze the average Absences based on the parental level specified by the user and includes error handling.
Analyse the average math score for the attendance over 80% for a race specified by the user and includes error handling.
Analyse the average writing score based on the health level specified by the user and includes error handling.
Identifies Top three Mothers education level based on the race and parental involvement specified by the user and includes error handling.
get_info_by_id
get_info_by_race
get_info_by_parental_involvement
get_info_by_studytime
Retrieve the sex, age, number of relative in a family, state, and race based on the ID and includes error handling.
get_user_choice
Ask the user to enter the Number corresponding to the race specified
Retrieve the sex, school support, access internet, attendance rate, and parental involvement associated with a specific race and includes error handling
get_parental_level
Ask the user to enter the parental involvement and validate the level
Retrieve the ID, free time, math score, reading score and writing score of students whose absences are less than 50 based on the parental involvement and includes error handling.
Retrieve the Failure, Health, , Suspensions, and Travel Time associated with study Time Greater than one hour and includes error handling.
Visualization module
Imported matplotlib to visualize relevant data
Visualization(df_data)
Handles the choice of the user and error handling
students_proportion_by_race
avg_writing_scores_by_race
relationship_reading_writing_scores
Health_visualization
Pie chart
Bar chart
Scatter Plot
Bar chart
Display the proportion of students by race
Display the proportion of writing score by race
Display the relationship between writing score and reading score
Display the proportion of health by math score

2.2 Modules and function

The software program is divided into various modules and functions that perform specific tasks. The Modules can be broadly classified into 8 categories, and here are the details of each module/function within these categories and its task:

2.2.1 Greetings Module

The greetings module provides the function for greetings.

2.2.1.1 Greetings- generate_student_documentation_greeting()

This function just has a greetings message for the software to greet the user and return greeting

```
    greeting = """
............Welcome to the Student Documentation Software!.......

"""
```

Figure 1 Greetings Message

2.2.2 Data Loading Modules

Data loading modules load-read data from various file formats, such as CSV, into Python.

2.2.2.1 *Load data - csv module{ load_csv_data}*

The ‘load_csv_data’ function reads a CSV file retrieved from the specified file path and stores the data in a list using the `csv.reader` module in Python.

```
try:
    with open(file_path, mode='r', encoding='utf-8') as file:
        reader = csv.reader(file)
        data = list(reader)
    return data
except FileNotFoundError:
    print("Error: CSV file not found. Please restart the software!")
    return None
except Exception as e:
    print(f"Error: {e}")
    return None
```

Figure 2 error handling and reading the csv file

2.2.2.2 *Load data - Pandas Module {load_data_pandas()}*

The load_data_pandas() function uses the pandas module in Python to load a CSV file Retrieved from the file path and create a pandas data frame.

```
try:
    df_data = pd.read_csv(file_path, header=0, encoding='utf-8')
    df_data['Race'] = df_data['Race'].str.lower()
    df_data['Parental_involvement'] = df_data['Parental_involvement'].str.lower()
    df_data['Mother_education_level '] = df_data['Mother_education_level '].str.lower()



    return df_data
except Exception as e:
    print(f"Error loading data: {e}")
    return None
```

Figure 3 creating a data frame using pandas

2.2.3 Get File path from User Module {get_file_path()}

The `get_file () ` function asks the user to enter the file path or the file name that is `students_data.csv `. Once a file name is entered, the function returns the file path.

```
def get_file():
    file_path = input("Enter the CSV file path: ")
    return file_path
```

Figure 4 Retrieve the file path

2.2.4 Main software code Module

This module consists of two functions one is the display menu which shows the user the display menu and other is the main consisting of lines of code used to run the whole program

In this Student Documentation software this module contains the main function code. That contains various modules containing the functions for Retrieving, analyzing, and visualizing data.

The program displays the user with a menu option: Retrieve Relevant Information, Analyze Data and Visualize Data. Depending on the user's choice, the program displays a sub-menu of options that allow the user to select the specific data they want to retrieve, analyze, or visualize. Once the user selects an option, the program calls the relevant function to carry out the selected operation. The program provides the outputs on the status of the operation to the user and then returns the user to the main menu to select another option or exit the program.

```
# Call the function to get the greeting message
greeting_message = generate_student_documentation_greeting()


print("1. Retrieve Data")
print("2. Analyse Data")
print("3. Visualization")
print("4. Exit")
```

Figure 5 Greetings and Display Menu

```
#entering a loop for handling the user choice
while True:
    #display the main menu
    display_menu()
    #get the user's choice
    choice = input("Please select an option: ")
    #if the user chose "retrieve data
    if choice == '1':
        while True:
            #load data from the csv file
            data = load_csv_data(file_path)
            print("\nWhich of the following would you like to do? Make your selection from the options shown:")
            print("1. Look up the data records using the provided Student ID number")
            print("2. Retrieve data based on Race")
            print("3. Retrieve data on Parental Involvement and Absences where the number of absences is less than 50")
            print("4. Retrieve data based on study time")
            print("5. Exit")
            #get user's sub-choice
            sub_choice = input("Enter your choice: ")
            #handle the user's sub-choice
            if sub_choice == '1':
                student_id = input("Enter the student ID: ")
                #get the function
                get_info_by_id(data, student_id)
```

Figure 6 While loop For Main menu user interface

```
    #chose to exist
    elif choice == '4':
        print("Exiting the Application. Thank you!")
        break # Exit the outer loop, ending the program

    #Handle invalid main menu choices
    else:
        print("Invalid choice. Please try again.")
```

Figure 7 Use of Break and continue for menu option

2.2.5 Data Retrieval Modules

The data retrieval module provides the functions for retrieving specific data related to students' data.

2.2.5.1 User change or main menu { change_or_main_menu()}

Allowing the user to change the input or go back top main menu, it loops until a valid choice is made True if the user opts to change and false for the main men Invalid inputs prompts a retry.

```
while True:
    # Display the options to the user.
    print("Would you like to:")
    print("1. Change input and get a new output")
    print("2. Go back to the main menu")
    # Ask the user to make a choice.
    choice = input("Enter your choice (1 or 2): ")
    # If the user chooses '1', return True to indicate continuation.
    if choice == '1':
        return True

    # If the user chooses '2', return False to indicate returning to the main menu.
    elif choice == '2':
        return False

    # If the input is invalid, notify the user and ask for input again.
    else:
        print("Invalid choice. Please try again.")
```

Figure 8 While Loop Change Menu

2.2.5.2 Data Retrieval using Host ID {get_info_by_id(data, student_id) }

The `get_info_by__id()` is a function that allows the user to retrieve information about an Students data by specifying the Student ID using the while loop. If the user selects option 1, the function asks the user to enter a Students ID and searches the information by ID from a dataset. It prints details like sex, age, number of relatives, state, and race if the ID is found. If not, it notifies the user and prompts for a new ID or allows returning to the main menu using the change_or_main_menu() function.

```
#starting the loop
while True:
    found = False #intialize the flag for student's ID if found
    for row in data:
        # Check if the current row's first element (ID) matches the input student ID
        if row[0] == student_id:
            # Print the relevant information for students of the specified ID
            print(f"Sex: {row[1]}\tAge: {row[2]}\tNumber of Relatives: {row[21]}\tState: {row[26]}\tRace: {row[27]}")
            found = True # Set the flag to True to indicate the ID was found
            break
```

Figure 9 For loop iteration of specified ID

2.2.5.3 Get the user choice {get_user_choice()}

This Function allows the user to enter the choice of the race to be utilized for the next function get_info_by_race(data, race) for retrieving the data that is based on race

```
choice = input("Enter the number corresponding to your choice: ").strip()

race_options = {
    "1": "Asian",
```

Figure 10 input and choice option for Race

2.2.5.4 Data Retrieval using Race {get_info_by_race(data, race) }

This module allows users to retrieve information from the student's dataset based on Race and prints their information as listed in the code. if no matching race is found

It prints an error message the function then prompts to either search again or turn to main menu using change_or_main_menu() function. If continuing, it asks for new race using get_user_choice().

```
while True:
    found = False
    for row in data:
        if row[27] == race:
            print(f"Sex: {row[1]}\tSchool Support: {row[14]}\tAccess Internet: {row[19]}\tAttendance Rate: {row[31]}\tParental Involveme
            found = True
```

Figure 11 For Loop Literation for Specified Race

2.2.5.5 Get the parental level { get_parental_level()}

Allowing the user to enter the level of parental involvement, it repeatedly asks for input until a valid option is provided, returning the valid input if the input is invalid, it displays an error message and retries.

```
while True:
    #ask the user to enter the parental involvement level
    level = input("Enter parental involvement level (high, medium, low): ").strip().lower()
    #check if the input is valid or not
    if level in ['high', 'medium', 'low']:
        return level#return the valid level
    else:
        #inform the user the input is invalid
        print("Invalid input. Please enter 'high', 'medium', or 'low'.")
```

Figure 12 Checking input is present in valid level

2.2.5.6 Data Retrieval using parental involvement {get_info_by_parental_involvement(data, parental_involvement)}

This code defines a function that allows users to retrieve information on based on parental level using the function get_parental_level() this function retrieves the parental level then it filters the a value below 50 in a certain field then it prints the information as listed If no matches are found, it displays an error message. The user can choose to search again or return to the main menu using change_or_main_menu(), and it prompts for new input with get_parental_level().

```
while True:
    found = False
    #iterate over each row in the data
    for row in data:
        if row[37].strip().lower() == parental_involvement.lower() and int(row[25]) < 50:
            #print the relevant student information
            print(f"ID: {row[0]}\tFree Time: {row[22]}\tMath Score: {row[28]}\tReading Score: {row[29]}\tWriting Score: {row[30]}")
            found = True
```

Figure 13 For loop iteration for parental _involvement

2.2.5.7 Data Retrieval using Study Time {get_info_by_studytime(data, Studytime)}

This code defines a function called ` get_info_by_studytime(data, Studytime) ` that asks the user to input and returns additional information. The Function searches for students with a specified amount of study time greater than 1 hour. Prints the specified rows of information on the dataset If no matches are found, it displays an error message. The user can choose to search again or return to the main menu using change_or_main_menu(). It prompts for new input if continuing.

```
while True:
    found = False
    # Iterate over each row in the data starting from the second row
    for row in data[1:]:
        # Check if the study time matches and is greater than 1 hour
        if row[12] == Studytime and int(row[12]) > 1:
            #print the relevant information of the student
            print(f"Failures: {row[13]}\tHealth: {row[25]}\tSuspensions: {row[32]}\tTravel Time: {row[11]}")
            found = True
```

Figure 14 For iteration for study time with specified condition

2.2.6 Data Analysis Modules

The data analysis modules provide functions for retrieving specific data-related analysis that can be done with the CSV file and returns the results.

2.2.6.1 Average Absentees {avg_absences_by_parental_involvement(df_data)}

The Function analyses student absence data and calculates and displays the average number of student absences based on levels of parental involvement (low, medium, high) from a dataset. It repeatedly asks the user for a parental involvement level and shows the corresponding average absences, it continues to prompt the user until they choose to exit.

```
    #calculate the mean absences grouped by parental involvement
    avg_absences_by_involvement = df_data.groupby('Parental_involvement')['Absences'].mean()
    #define the valid levels in parental involvement
    valid_levels = ['low', 'medium', 'high']
```

Figure 15 pandas' group by and mean function

2.2.6.2 *Average math score by race {avg_math_score_by_race_for_attendance_over_80(df_data)}*

This function calculates and displays the average math scores of students with attendance over 80%, grouped by race. It then asks the user to select a race to view the corresponding average score.

```
print("4. African American")
print("5. Other")

choice = input("Enter your choice (1, 2, 3, 4, or 5): ")
```

Figure 16 Choice option for corresponding race

```
# Filter the DataFrame to include only students with attendance rate over 80%
high_attendance = df_data[df_data['Attendance_rate'] > 80]
#calculating the mean
avg_math_scores_by_race = high_attendance.groupby(df_data['Race'].str.lower())['Math_score'].mean()
```

Figure 17 Mean value calculation by pandas groupby function

2.2.6.3 *Average Writing score {avg_writing_score_for_health_analysis(df_data)}*

The code defines a Function that calculates and displays the average writing scores of students, grouped by their health status. It prompts users to input a health level to view the corresponding average score. It repeatedly asks the user for a health level and shows the corresponding average, it continues to prompt the user until they choose to exit.

```
valid_levels = ['good', 'poor', 'fair', 'peak', 'excellent']
avg_writing_scores_by_health = df_data.groupby(df_data['Health'].str.lower())['Writing_score'].mean()
```

Figure 18 valid levels and Calculating Mean

```
while True:
    #asks the user to enter a valid health level
    health_level = input("Enter the Health (good, poor, fair, peak, or excellent): ").lower()
    #check the entered health level is valid or not
    if health_level not in valid_levels:
        print("Not valid. Please check again and re-enter the Health.")
        continue
    # Retrieve the average writing score for the specified health level
    avg_writing_score = avg_writing_scores_by_health.get(health_level)
```

Figure 19 while and continue for validating the input

2.2.6.4 Get Top Mothers Education level

{top_mothers_education_by_race_and_parental_involvement(df_data)}

The function analyzes a dataset to identify the top three levels of mother's education based on race and parental involvement.

```
# Group data by Race, Parental Involvement, and Mother's Education Level, and count occurrences
grouped_data = df_data.groupby(['Race', 'Parental_involvement', 'Mother_education_level ']).size().reset_index(name='count')
# Filter the data based on the specified race and parental involvement level
filtered_group = grouped_data[(grouped_data['Race'].str.lower() == race) &
                              (grouped_data['Parental_involvement'].str.lower() == parental_involvement)]
# Sort the filtered data by count and return the top 3 education levels
top_3_education_levels = filtered_group.sort_values(by='count', ascending=False).head(3)
```

Figure 20 Analyzing Education Levels and Parental Involvement Using Pandas Groupby

2.2.6.4.1 Internal Function: {get_top_mother_education_levels}

Groups and filters data to find the top three education levels for specified criteria. This function enables users to interactively analyze education levels by race and parental involvement within a dataset. It groups and sorts the data, then prompts the user to select options to display the relevant information.

```
user_parental_involvement = input("Enter the specific parental involvement level: ").strip().lower()

if user_parental_involvement in unique_parental_involvement:
    top_3_levels = get_top_mother_education_levels(user_race, user_parental_involvement)
    print(f"\nTop 3 levels of mother's education for race '{user_race.title()}' and parental involvement '{user_parental_involvement.title()
    print(top_3_levels[['Mother_education_level ', 'count']])
else:
    print(f"Parental involvement level '{user_parental_involvement}' is not found in the dataset.")
```

Figure 21 Exploring the relationship between parental involvement and mother's education

2.2.6.5 Handle choice { handle_choice(choice, df_data)}

The handle choice function directs the flow of the program based on user input, executing specific analysis functions on a dataset.

```
    # Check the user's choice and call the appropriate function
    if choice == '1':
        avg_absences_by_parental_involvement(df_data)
    elif choice == '2':
        avg_math_score_by_race_for_attendance_over_80(df_data)
```

Figure 22 choice option for handling the user input

2.2.7 Data Visualization Modules

The data visualization module provides functions for visualizing the data in various forms. These functions use the matplotlib.pyplot module to create various types of charts and graphs.

2.2.7.1 Student proportion by Race

{ students_proportion_by_race(df_data)}

The purpose of this code is to visualize the distribution of students by race in a dataset. It creates a pie chart that shows the proportion of each race, providing a clear and immediate understanding of the racial composition within the data.

```
# Set the size of the figure
plt.figure(figsize=(10, 6))
# Create a pie chart with the race labels
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
#  the title of the chart
plt.title('Proportion of Students by Race')

# Display the pie chart
plt.show()
```

Figure 23 Plotting pie chart

2.2.7.2 Average writing score by race

{ avg_writing_scores_by_race(df_data)}

The function avg_writing_scores_by_race visualizes the average writing scores for each race in a dataset. It creates a bar chart that provides an overview of how writing scores vary across different races.

```
# Calculating average writing score for each race
avg_scores = {race: sum(scores)/len(scores) for race, scores in race_scores.items()}
```

Figure 24 Calculating Avg Writing scores

```
# Plotting the data
plt.figure(figsize=(10, 6))
# Create a bar plot
plt.bar(avg_scores.keys(), avg_scores.values(), color='skyblue')
# Label for the x-axis
plt.xlabel('Race')
# Label for the y-axis
plt.ylabel('Average Writing Score')
#title
plt.title('Average Writing Scores by Race')

# Display the plot
plt.show()
```

Figure 25 Plotting Bar Chart

2.2.7.3 Relationship Between Reading and Writing Score {relationship_reading_writing_scores(df_data)}

The function relationship_reading_writing_scores visualize the relationship between reading and writing scores in a dataset. The function extracts 'Reading_score' and 'Writing_score' columns from the provided Data Frame df_data. It creates a scatter plot to show how these two scores correlate.

```
reading_scores = df_data['Reading_score']
writing_scores = df_data['Writing_score']

# Set the size of the figure
plt.figure(figsize=(10, 6))
# Create a scatter plot with reading scores on the x-axis and writing scores on the y-axis
plt.scatter(reading_scores, writing_scores, alpha=0.5)
```

Figure 26 Plotting scatter plot

2.2.7.4 Health Visualization { Health_visualization(df_data)}

The defined function visualizes the average math scores for each health status using a bar chart. It groups data by health status, calculates the averages, and plots them with labeled axes and a title.

```
# Calculate the average scores
avg_scores = {health: sum(scores) / len(scores) for health, scores in avg_math_scores_health.items()}

# Plotting
plt.figure(figsize=(10, 6))
#plot the bar chart
plt.bar(avg_scores.keys(), avg_scores.values(), color='red')
plt.xlabel('Health')
plt.ylabel('Average Math Score')
plt.title('Average Math Scores by Health')
#show the plot
plt.show()
```

Figure 27 Calculating the average score and plotting bar chart

2.2.7.5 Visualization Main Module { Visualization(df_data)}

Main function to display the visualization menu and operate user input using choice

```
# Operates the user's choice
if choice == '1':
    students_proportion_by_race(df_data)
elif choice == '2':
    avg_writing_scores_by_race(df_data)
elif choice == '3':
    relationship_reading_writing_scores(df_data)
elif choice == '4':
    Health_visualization(df_data)
```

Figure 28 while loop for handling the choice of user

2.2.8 Run_program Module

This module is used to run the whole program. This code runs the Main_software_code.Main() function from the Main_software_code Module.

```
import Main_software_code

# Run the Main function from main_software
if __name__ == "__main__":
    Main_software_code.Main()
```

Figure 29 Executes the entire program

2.3 GitHub Repository Evidence

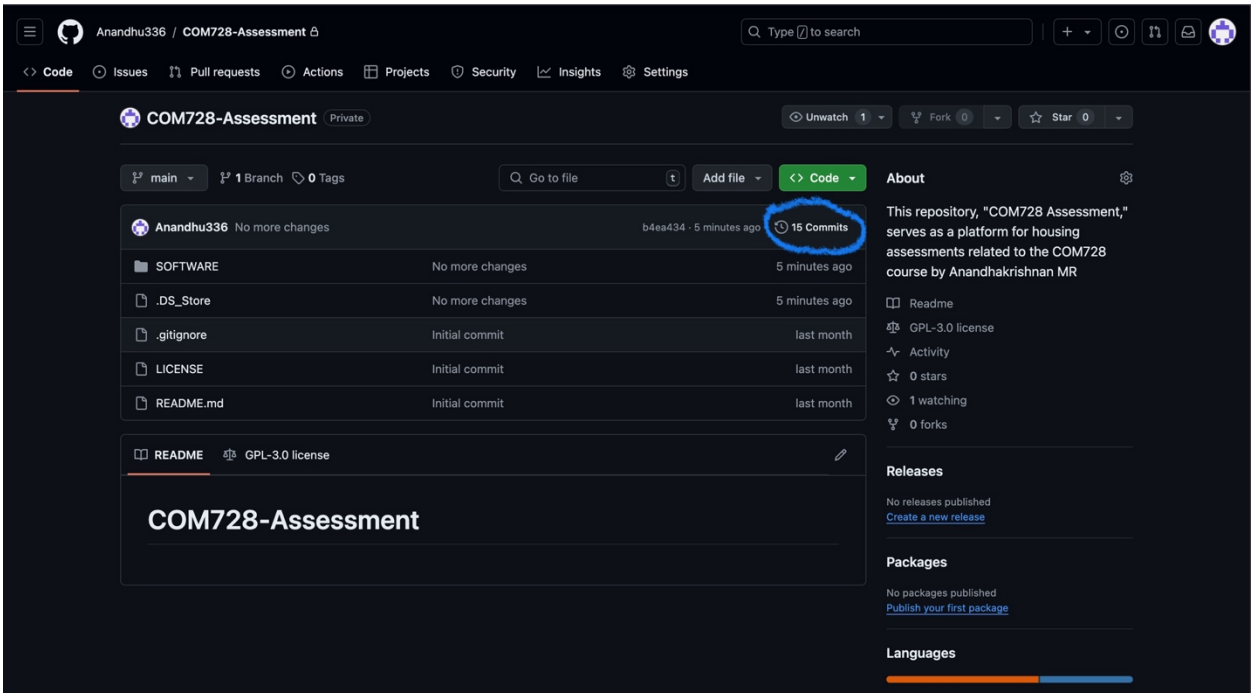


Figure 30 GitHub Repository Main Page

Sample screen shot of your commit history: