

Student Data Analysis

A Python program for academic data analysis of students

Student:

Abhilash Reddy Peram (10532966)

Sabitha Maram (10533048)

Srikanth Shilesh Pasam (10387794)

Teacher:

Paul Laird

Course:

Programming for Information Systems (B9IS123)

Table of Contents

<i>Table of Figures</i>	<i>3</i>
<i>Introduction.....</i>	<i>4</i>
<i>Requirements</i>	<i>5</i>
Functional Requirements.....	5
Non-Functional Requirements	5
<i>Student Data</i>	<i>6</i>
<i>Program Features</i>	<i>6</i>
Stage 1 – Login/Register	6
Stage 2 – Loading files	6
Stage 3 – Data Analysis.....	7
<i>Implementation</i>	<i>8</i>
<i>Individual Contributions.....</i>	<i>16</i>
Srikanth Shilesh Pasam	16
Reflection.....	16
Sabitha Maram	17
Abhilash Reddy Peram.....	18
<i>Appendix 1</i>	<i>19</i>
<i>Appendix 2</i>	<i>20</i>

Table of Figures

Figure 1 Login Section	8
Figure 2 Creating new user.....	8
Figure 3 Login of existing user	9
Figure 4 Selecting folder location and opening file	9
Figure 5 Calculating the class average.....	10
Figure 6 Subject wise marks sorted display	11
Figure 7 Searching for a student.....	11
Figure 8 Plotting graph of student progress.....	12
Figure 9 Plotting graph of class progress.....	12
Figure 10 Adding or editing student details	13
Figure 11 New student details entry	13
Figure 12 Editing existing student details.....	14
Figure 13 Successfully edited student details.....	14
Figure 14 Fernet key used in encryption	15
Figure 15 Encrypted and encoded user login details.....	15
Figure 16 Saving generated graphs to local memory	15

Introduction

The Student Data Analysis is a program which is designed to analyze a students or classrooms academic performance. It has been designed with the purpose of addressing the needs of Teach For India. Teach For India is a non-government non-profit organization and is part of Teach For All network. The organizations aim is to tackle the problems of education inequity in the country by providing excellent education to every child. As a part of this organization, qualified and motivated Fellows are placed in low-income private schools or under resourced government schools to drive change. During this time, the fellows collect large volumes of student and school data ranging from academic performance, culture and values of the class, social and economic levels of the parents, teacher resources, training modules, infrastructure of the school, school income and expenditure, external stakeholder investments and much more. All this data is used by the fellows to analyze and create action plans for the improvement of the quality of education received. Currently the organization has 900 fellows, 250 staff and 3000 alumni across 7 cities in India impacting 260 schools and 32000 students directly. As of this day, the organization only uses spreadsheets in order to record and analyze all the student data collected. This task gets increasingly difficult every year as the organization keeps expanding and reaches out to more and more schools and students. The efficiency with which all this data can be analyzed and used is severely limited to the scope and extent of what a spreadsheet can do.

This project aims to tackle a small portion of this issue as a part of this assignment. The program takes in the raw student data from the existing spreadsheets (.xlsx or .csv) runs them through the program to provide an analysis of the student or classrooms academic performance. The current program only focuses on academic analysis within the scope of this CA but plans are made to integrate other aspects like deployment in the Cloud, a backend database system, web interfacing along with a mobile app for GUI in the future.

Requirements

Functional Requirements

1. As the School Leader, I need to see a summary of the marks of each subject for each grade so that I can identify the areas of development for the respective grade. (Reporting feature)
2. As a School Leader, I need to see a summary of the marks scored by each student for a subject so that I can identify the areas of development for the respective subject teacher. (Reporting feature)
3. As a Class Teacher, I need to see a summary of the marks scored by each student for all subjects so that I can give feedback to students individually. (Search feature)
4. As a Class Teacher, I would like to be able to add or edit students data into the system. (Entry/update feature)
5. As a Subject Teacher, I want to see a summary of the marks scored by students in my subject so that I can identify which student requires extra classes scheduled.
6. As a Parent, I need to see a summary of the marks scored by my child so that I can keep a track of his/her academic progress. (Search feature)
7. As an external stakeholder, I need to see a summary of the growth of a classroom over 3 years in the form of graph so that I can decide if I should continue funding Teach For India. (Reporting feature)
8. As the City Director of Teach For India, I want access to the Student Data Analysis System to be provided only after proper authentication has taken place because it contains sensitive information of school and students. (Authentication feature)

Non-Functional Requirements

1. As a fellow who frequently accesses the system, I want the program to load up quickly so that I can finish up my work faster.
2. As the School Relations Manager of Teach For India, I want the graphs to use the Teach For India's blue color so that it matches the impact growth document shared with the stakeholders.

Student Data

The organization uses spreadsheets to store their data. For the purpose of the CA, the organization agreed to let us use few of their historic data sets. They have been included in the CA submission under the 'Source' folder. From this data sets, eight CSV files have been created which will act as the raw data file for the program demonstration. These CSV files have the student roll, student name, gender, RC (reading comprehension), listening, writing and math columns. The grading system for each subject is different. RC levels range from - 0.5 to 5.5, writing starts at 0 and goes all the way to 8, listening is from 0 to 5 and math is percentage based. There is much more data in the source files but for demonstration purposes we will be limiting the scope of the program to only these elements.

Program Features

Stage 1 – Login/Register

The program uses CLI as the interaction tool with the user. On executing the program, the user will be asked to either login or register.

- In both the login and register phases the user details are encoded and encrypted for security.
- The password input is taken using the GetPass library function which hides the password entry.
- The encryption is done using the cryptography library Fernet which uses symmetric encryption.
- The Fernet key is stored locally in the project folder under the file name 'Fernet_key.txt' for the sake of demonstration.
- The registered user details are encoded into byte code format. This is then encrypted using the Fernet Key created. All this data is stored locally under the file name 'User_details.txt' for the sake of demonstration.

Stage 2 – Loading files

Once the user logs in successfully they are granted access to the program.

- To access the files on the system the program uses Tkinter library functions.

- The Tkinter provides a dialogue box for the user to select the directory where the files are located.
- Once the user selects the directory, all the files in that directory are listed using the OS library function.
- The user then needs to select a file from that list which needs to be processed.
- After the file is selected, the program loads the CSV file into a DataFrame using the Pandas library functions.

Stage 3 – Data Analysis

Once the files are loaded, the program can perform data analysis on it depending on what the user wants to do.

- The user has the option to:
 - ◆ Calculate the class averages of each grade.
 - ◆ Calculate individual subject average for each grade.
 - ◆ Search for a student using their full name or part of their name and display their corresponding subject-wise marks.
 - ◆ Using the Matplotlib library function to plot the graph which show the classroom growth for each subject over the span of three years.
 - ◆ Plot the graph to show an individual student growth for each subject over the span of three years.
- The graphs which are generated post the analysis is saved locally in the 'Graphs' folder using a certain naming convention for user access.
- The user can add new student details or edit existing ones.

Implementation

The working code is included in Appendix 2 for reference. The implementation of the program is shown using screenshots.

```
Welcome to School Data Analysis Tracker!  
Are you an existing user or a new user?  
1) Existing user  
2) New user
```

Figure 1 Login Section

```
Welcome to School Data Analysis Tracker!  
Are you an existing user or a new user?  
1) Existing user  
2) New user  
2
```

```
Create a user name: sri  
Create your password: .....
```

```
User created successfully!
```

Figure 2 Creating new user


```
Welcome to School Data Analysis Tracker!
Are you an existing user or a new user?
1) Existing user
2) New user
1

Username: sri
Password: .....

Login Successful

Pick a choice to process data in the way required:
1) Claculate the class average
2) Subject-wise marks
3) Student-wise marks
4) Plot graph of class growth
5) Add/edit student data

```




Figure 3 Login of existing user

```
Welcome to School Data Analysis Tracker!
Are you an existing user or a new user?
1) Existing user
2) New user
1

Username: sri
Password: .....

Login Successful

Pick a choice to process data in the way required:
1) Claculate the class average
2) Subject-wise marks
3) Student-wise marks
4) Plot graph of class growth
5) Add/edit student data
1

Select a directory from the pop up window to begin

The directory path you selected is:
/Users/srikanthshileshpasam/OneDrive - Dublin Business School (DBS)/Python/CA2/Data

The available files are:
['2_2014.csv', '2_2015.csv', '2_2016.csv', '.DS_Store', 'Source Folder', '1_2016.csv', '1_2014.csv', '1_2015.csv',
'Graphs']

Select a file from above:

```

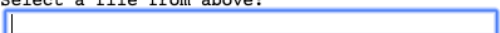


Figure 4 Selecting folder location and opening file

Opening file...

Order	Name	Gender	RC	Listening	Writing	Math
0	1	Ahmed	Male	0.2	0.0	0.10
1	2	Ayan	Male	0.2	0.0	0.09
2	3	Ayesha Fatima 1	Female	0.2	1.1	0.19
3	4	Ayesha Fatima.	Female	0.0	0.6	0.21
4	5	Dinesh	Male	0.2	1.0	0.55
5	6	Faisal Pasha	Male	0.5	1.9	0.33
6	7	Firdous Banu	Female	0.5	2.5	0.23
7	8	Javeen	Male	0.5	2.9	0.38
8	9	Kuhrshid	Male	1.0	2.4	0.58
9	10	Md.Mukram	Female	0.5	2.6	0.41
10	11	Mirza Baig	Female	0.0	2.0	0.43
11	12	Mohd. Azhar	Male	0.5	3.0	0.11
12	13	Mohd. Saadiq	Male	0.5	2.4	0.54
13	14	Mohammed Ali	Male	0.2	0.7	0.37
14	15	Moinuddin	Male	0.0	0.6	0.15
15	16	Nayyar Fatima	Female	0.2	0.6	0.13
16	17	Nazreen Begum	Female	0.5	1.0	0.26
17	18	Nida Fatima	Female	0.5	2.9	0.44
18	19	Nikith Tabasuman	Female	0.2	1.0	0.15
19	20	Saba Begum	Female	0.5	1.2	0.18
20	21	Sana Fimal	Female	0.2	1.1	0.15
21	22	Sayyum	Female	0.5	1.9	0.46
22	23	Shaik Ikram	Male	0.2	1.2	0.40
23	24	Sidra Begum	Female	0.5	0.9	0.27
24	25	Syed Ashfaq	Male	0.2	2.9	0.38
25	26	Tabassam Begum	Female	0.5	2.6	0.49
26	27	Arifa	Female	0.0	0.5	0.27
27	28	Saniya	Female	0.5	2.7	0.35
28	29	Neha Begum	Female	1.0	2.1	0.54
29	30	Mahejabeen	Female	0.0	1.4	0.19
30	31	Sana Begum 2	Female	0.5	2.6	0.58
31	32	Mehak	Female	0.2	0.6	0.17
32	33	Farheen	Female	0.5	0.7	0.21
33	34	Mohd Ibrahim	Male	0.5	2.0	0.44
34	35	Saba 1	Female	1.5	2.9	0.42
35	36	Imran Ali	Male	0.5	2.0	0.24
36	37	Imtiaz	Male	0.5	1.1	0.16
37	38	Aquaib	Male	1.0	2.9	0.28
38	39	Ghouse	Male	0.2	1.5	0.23

The class average is as below:

RC: 0.41

Listening: 0.94

Writing: 1.64

Math: 30.92

Figure 5 Calculating the class average

The class RC marks from the highest to lowest are:

Roll Order	Name	RC
35	Saba 1	1.5
38	Aquaib	1
29	Neha Begum	1
9	Kuhrshid	1
20	Saba Begum	0.5
34	Mohd Ibrahim	0.5
26	Tabassam Begum	0.5
22	Savvum	0.5

Figure 6 Subject wise marks sorted display

23	24	Sidra Begum	Female	0.5	0.5	0.9	0.27
24	25	Syed Ashfaq	Male	0.2	2.0	2.9	0.38
25	26	Tabassam Begum	Female	0.5	1.5	2.6	0.49
26	27	Arifa	Female	0.0	0.5	0.5	0.27
27	28	Saniya	Female	0.5	1.0	2.7	0.35
28	29	Neha Begum	Female	1.0	1.5	2.1	0.54
29	30	Mahejabeen	Female	0.0	0.5	1.4	0.19
30	31	Sana Begum 2	Female	0.5	2.5	2.6	0.58
31	32	Mehak	Female	0.2	0.2	0.6	0.17
32	33	Farheen	Female	0.5	1.0	0.7	0.21
33	34	Mohd Ibrahim	Male	0.5	0.5	2.0	0.44
34	35	Saba 1	Female	1.5	3.0	2.9	0.42
35	36	Imran Ali	Male	0.5	1.0	2.0	0.24
36	37	Imtiaz	Male	0.5	0.5	1.1	0.16
37	38	Aquaib	Male	1.0	2.5	2.9	0.28
38	39	Ghouse	Male	0.2	0.5	1.5	0.23

Enter name of student
begum

Marks for begum are as follows:

Roll Order	Name	RC	Listening	Writing	Math
17	Nazreen Begum	0.5	0.5	1	0.26
20	Saba Begum	0.5	0.5	1.2	0.18
24	Sidra Begum	0.5	0.5	0.9	0.27
26	Tabassam Begum	0.5	1.5	2.6	0.49
29	Neha Begum	1	1.5	2.1	0.54
31	Sana Begum 2	0.5	2.5	2.6	0.58

Figure 7 Searching for a student

Enter name of student
arifa

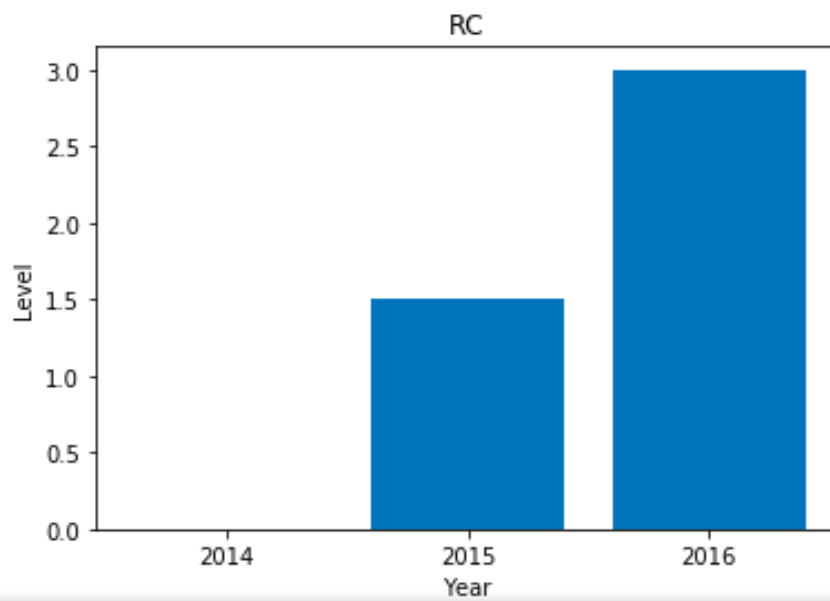


Figure 8 Plotting graph of student progress

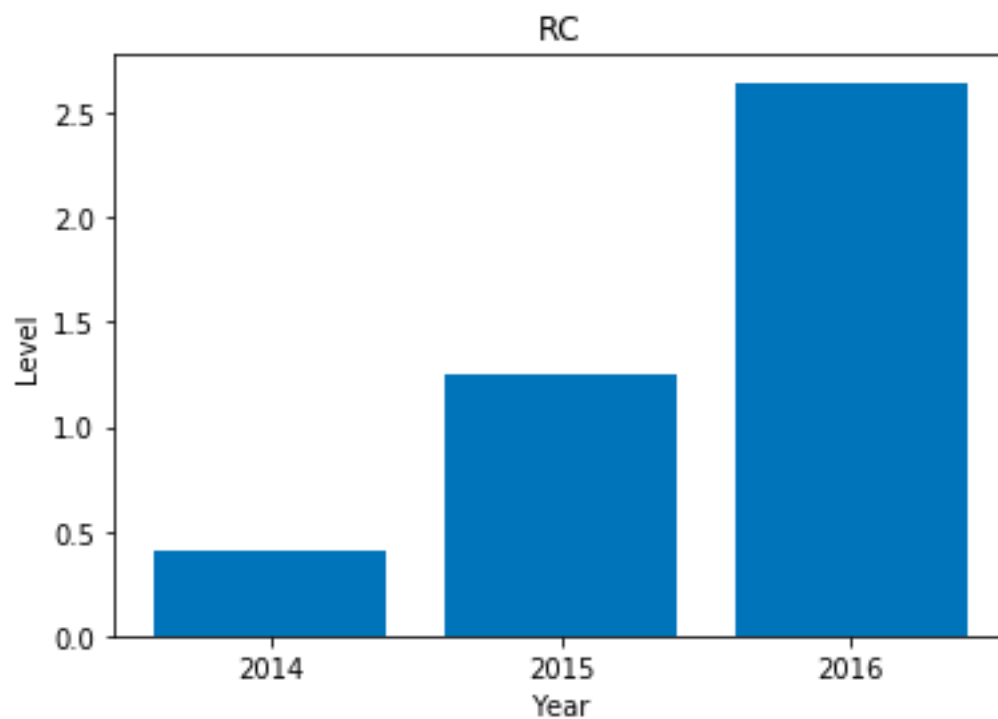


Figure 9 Plotting graph of class progress

32	33	Farheen	Female	0.5	1.0	0.7	0.21
33	34	Mohd Ibrahim	Male	0.5	0.5	2.0	0.44
34	35	Saba 1	Female	1.5	3.0	2.9	0.42
35	36	Imran Ali	Male	0.5	1.0	2.0	0.24
36	37	Imtiaz	Male	0.5	0.5	1.1	0.16
37	38	Aquaib	Male	1.0	2.5	2.9	0.28
38	39	Ghouse	Male	0.2	0.5	1.5	0.23

- 1) New student entry
- 2) Edit existing student data

Figure 10 Adding or editing student details

32	33	Farheen	Female	0.5	1.0	0.7	0.21
33	34	Mohd Ibrahim	Male	0.5	0.5	2.0	0.44
34	35	Saba 1	Female	1.5	3.0	2.9	0.42
35	36	Imran Ali	Male	0.5	1.0	2.0	0.24
36	37	Imtiaz	Male	0.5	0.5	1.1	0.16
37	38	Aquaib	Male	1.0	2.5	2.9	0.28
38	39	Ghouse	Male	0.2	0.5	1.5	0.23
39	123	Testing	Male	1.0	1.0	1.0	1.00

Would you like to add another entry?

- 1) Yes
- 2) No

Figure 11 New student details entry

37	38	Aquaib	Male	1.0	2.5	2.9	0.28
38	39	Ghouse	Male	0.2	0.5	1.5	0.23
39	123	Testing	Male	1.0	1.0	1.0	1.00

1)New student entry
 2)Edit existing student data
 2

Enter the roll number of the student to begin editing

39

Enter new details at roll number 39

Enter name of student:
 editing test

Enter gender:
 male

Enter RC level:
 1

Enter Listening level:
 1

Enter Writing level:
 1

Enter Math level:

1

Figure 12 Editing existing student details

34	35	Saba 1	Female	1.5	3.0	2.9	0.42
35	36	Imran Ali	Male	0.5	1.0	2.0	0.24
36	37	Imtiaz	Male	0.5	0.5	1.1	0.16
37	38	Aquaib	Male	1.0	2.5	2.9	0.28
38	39	Editing Test	Male	1.0	1.0	1.0	1.00
39	123	Testing	Male	1.0	1.0	1.0	1.00

Update saved successfully!

Figure 13 Successfully edited student details

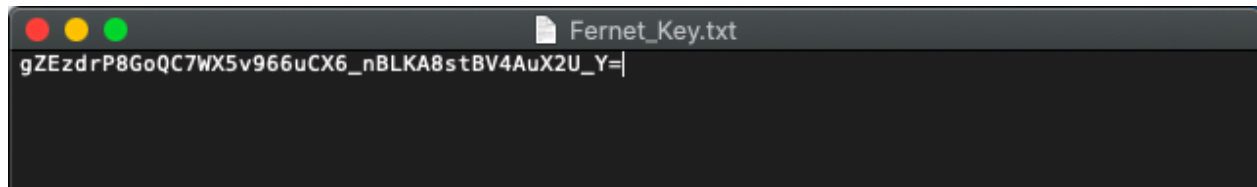


Figure 14 Fernet key used in encryption

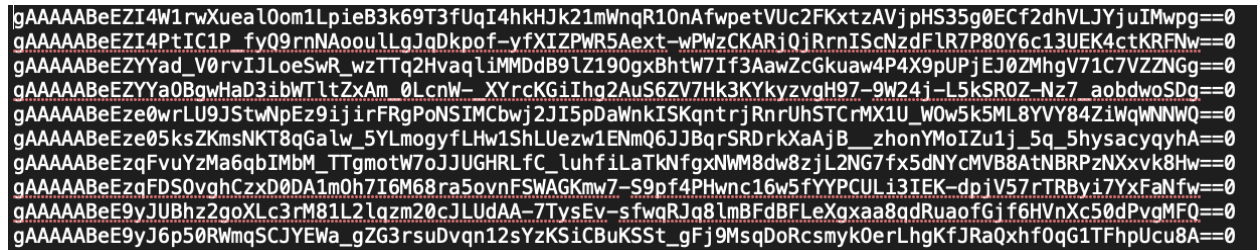


Figure 15 Encrypted and encoded user login details

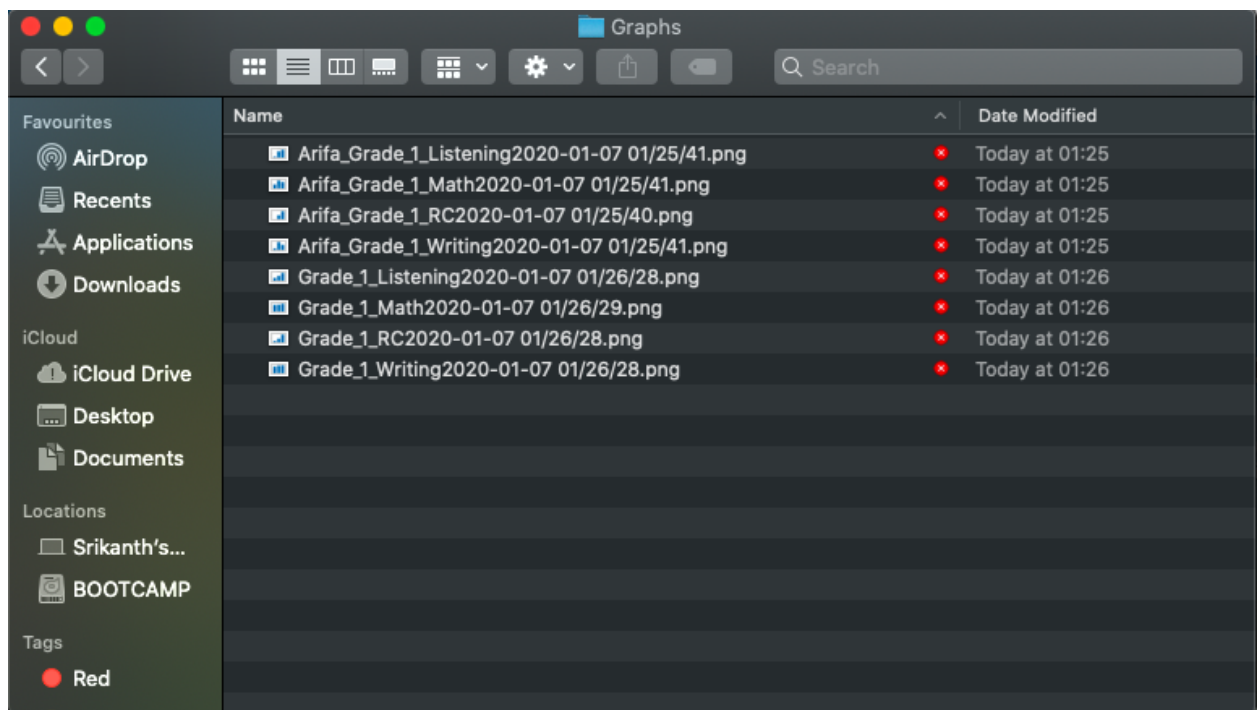


Figure 16 Saving generated graphs to local memory

Individual Contributions

Srikanth Shilesh Pasam

The three of us as a team contributed towards all the elements of the code. Each aspect has been discussed and brainstormed. This collaboration has helped us to learn from each other. Though inputs were taken from all team members, my major contribution for the CA is as follows:

1. Functional and non-functional requirements
2. User choice logic flow
3. LoadData() class
4. Avg() class
5. SubjectMarks() class
6. StudentMarks() class
7. Auth() class
8. Keys() class
9. Report writing
10. Power point presentation

Reflection

I come from Electronics and Communication Engineering background with no prior knowledge of the subject. My professional experience has been majorly in Teach for India, a non-government organization within the education sector. So, I have no professional experience working with Python or any other form of programming language either. Because of this I had to put in extra efforts to learn everything from the scratch. This proved extremely difficult during the initial days of the course where I had to learn a new language for each subject in the module. Nonetheless I had to push myself as this was a challenge I willingly signed up for. A major setback I faced was during the first in-class test. I performed poorly and this was my wakeup call to roll up my sleeves. I took it upon myself to study and improve my command over Python as much as possible. The time I got during reading week proved very useful for this. I signed up for courses on Udemy and spent major of my time practicing python and trying to perfect the program I failed to do during the in-class test. Through a lot of hard work and perseverance I slowly but steadily started getting more comfortable and fluent with Python. I was able to complete the program and submit it nine

days early on Moodle. This built a good sense of accomplishment within myself and motivated me to push further and harder.

Most of us in class had difficulty during the initial days with the subject due to various reasons and many a times we have let him down. But professor Paul has been extremely patient with us. He taught us from the scratch, helped us practice, shared resources for self-study and even followed up with our progress on Boot Camp. All this constant push and effort is what kept me motivated to try harder. Being a teacher myself I can personally relate to the amount of efforts that go into balancing time crunch in covering the syllabus and making sure the students are able to keep up with the pace. This only made me push myself and put in that extra hour every night. I am especially grateful to professor Paul for introducing me to GitHub. Coming from non tech background I had absolutely no awareness about it. But since then I have been very active on GitHub. I have even used it for all my other modules. In fact, I even based part of my CA in Network and System Administration module on GitHub using the Continuous Integration and Continuous Deployment concept.

This project idea has an association with me on a very personal level. I have been part of Teach For India for many years. My deep desire to impact change in the society by being an advocate for ed-equity is what made me choose this line of work. During my time in the organization I have noticed that one particular area of development here is the requirement of a proper data analysis system. Even in this current age of technology, the organizations vision was severely hampered by the lack of this system. This realization is also one of the major reasons for me to choose this particular course. I intend to do my final year thesis project around developing a system that can address these issues for Teach For India and donate the finished application. To this end I am confident that the module – Programming for Information Systems, the programming language – Python and the professor – Mr. Paul Laird has been a tremendous help in helping me get closer to achieving my goals.

Sabitha Maram

It was really good experience working in group where we can share ideas and knowledge of different people and put them together. It was really a tough and challenging experience for me to learn python programming as I am not into programming in my previous Education.

For this Assignment we discussed and decided to create Student Data Analysis. The key features in this work are User Authentication, Calculating Class averages and student averages, Visualizing student and class average in the form of pictorial representation in the form of graphs and Student data entry. The area where i worked individually is Graph plots and tried to do as much I learned. Srikanth in my group helped me to go through this to the end.

Abhilash Reddy Peram

This is very good learning experience with a new group of people who are our fellow members in class. Sitting all together and sharing knowledge on whole new subject with each other is very useful. We learned python it through Udemmy classes and worked on the assignment whenever we got time while completing the other assignment challenges. And even data camp. Initially it's a little bit slow to learn and catch up with each other eventually we got along with the assignment.

We have gone through the requirements and figured out what to be done with the data files we have. So, we chose to deal with student data files based on their academic performance. The actual data sets are gathered from the existing data base of a school by our group member Srikanth because has worked in teaching profession for a while in the past. We tried all the possible ways to meet the given requirement criteria in the module. After getting all the data files we all sorted out our work with each other. I have taken the authentication and data entry for the data files. Working on authentication is quite fun and interesting thing to learn like giving access to the users so that only they can have the access with the decryption side. And data entry is like updating the new student data if any student profile gets missed out. While I got struck with some problem in using keys and encryption my group members helped me with the problems and we worked together if any problem raises for any of the team members.

Finally, while coming to the module learning its very good experience to interact with the lecturer. This is the part I like the most as we came from the places where we have little bit interaction with the lecturer but, here MR. Paul keep on persist us to speak up and ask for the doubts and even ask us to share our ideas with him. We are very much great full for having MR. Paul as our module lecturer.

Appendix 1

The GitHub link for our project is:

https://github.com/sabithamaram/PIS_B9IS123_CA2

We have not realized until it's too late that all our individual commits were being made to the Master folder directly where instead we needed to fork the Master directory, commit to the forked directory and later on merge it to the Master. Hence, we all have the same GitHub link to share though each of our individual commits are visible in the 'Insights' section of the GitHub folder shared above.

Appendix 2

Importing required libraries

```
import pandas as pd
from tabulate import tabulate as tb
import matplotlib.pyplot as plt
import numpy as np
import os
import tkinter
from tkinter import filedialog
from datetime import datetime
from getpass import getpass
from cryptography.fernet import Fernet
```

Pandas to use DataFrames, reading and writing external files
Tabulate to display data in a neat tabular form
Matplotlib to plot graphs for student data analysis
OS to list the files in a directory
Tkinter to allow user to select a directory from where to load the external files
Datetime used as part of naming convention when saving generated graph files

Creating a class to read the external CSV data files

class LoadData:

```
    def __init__(self, choice):
        self.choice = choice
```

Creating a method to allow user to select the directory and location of file to open

```
    def file_select(self):
        # Initializing the tkinter function
        root = tkinter.Tk()
```

Getting the directory location of the data files from the user. Initially providing a default location

```
        dir_name = filedialog.askdirectory(parent=root, initialdir="/Users/srikanthshileshpasam/OneDrive - Dublin Business School (DBS)/Python/CA2/Data/",
        title='Please select a directory')
```

```
        root.quit() # Reference Link - https://stackoverflow.com/questions/28590669/tkinter-tkfiledialog-doesnt-exist/28590707
```

```
        print(f'\nThe directory path you selected is:\n{dir_name}')
```

```
        # Listing all the files in the directory selected by the user
        files_list = os.listdir(dir_name) # Reference Link - https://stackove
```

rflow.com/questions/3207219/how-do-i-list-all-files-of-a-directory

```
print(f'\nThe available files are:\n {files_list}')

# Choice 4 needs multiple files to be loaded. Hence checking for this
first
if self.choice == 4:
    grade_choice = int(input('\nChoose a grade\n'))

    if grade_choice == 1:
        # Loading the chosen grade data files from the directory path
        # selected by the user
        first_data_file = pd.read_csv(dir_name + '/' + '1_2014.csv')
        sec_data_file = pd.read_csv(dir_name + '/' + '1_2015.csv')
        third_data_file = pd.read_csv(dir_name + '/' + '1_2016.csv')

        return (first_data_file, sec_data_file, third_data_file, dir_
name, grade_choice)

    elif grade_choice == 2:
        first_data_file = pd.read_csv(dir_name + '/' + '2_2014.csv')
        sec_data_file = pd.read_csv(dir_name + '/' + '2_2015.csv')
        third_data_file = pd.read_csv(dir_name + '/' + '2_2016.csv')

        return (first_data_file, sec_data_file, third_data_file, dir_
name, grade_choice)

    else:
        print('Grade data does not exist!')

# If any other choice then only a single file needs to be opened
else:
    # Getting the name of the file to be opened from the user
    file_name = input('\nSelect a file from above:\n')

    return (pd.read_csv(dir_name + '/' + file_name + '.csv'), dir_nam
e, file_name)

# Creating a class to allow for manual entry/edit of student details
class DataEntry:

    def __init__(self, dir_path, data_file_name, s_data, s_name, s_gender, ro
ll_num, s_rc, s_lis, s_wri, s_math):
        self.dir_path = dir_path
        self.data_file_name = data_file_name
        self.s_data = s_data
        self.s_name = s_name
        self.s_gender = s_gender
        self.roll_num = roll_num
        self.s_rc = s_rc
        self.s_lis = s_lis
```

```

        self.s_wri = s_wri
        self.s_math = s_math

    # Creating a method to collect new student entry from user and saving to
    file
    def data_entry(self):
        # appending a new row to the existing DataFrame file. Using key-value
        pairs of dictionary to write the data to the appropriate columns
        self.s_data = self.s_data.append({'Order':self.roll_num, 'Name':self.
s_name, 'Gender':self.s_gender, 'RC':self.s_rc, 'Listening':self.s_lis, 'Writ
ing':self.s_wri, 'Math':self.s_math }, ignore_index=True)

        print(f'\nData entered successfully.\nSaving file..\n{self.s_data}')

        # Saving the data to the file in the directory loaction and file name
        given by the user previously when opening the file
        self.s_data.to_csv(self.dir_path + '/' + self.data_file_name + '.csv'
, index=False)

        return self.s_data

    # Creating a method to collect and edit existing student data from user a
    nd saving to file
    def data_edit(self):

        self.s_data.loc[self.roll_num, 'Name'] = self.s_name
        self.s_data.loc[self.roll_num, 'Gender'] = self.s_gender
        self.s_data.loc[self.roll_num, 'RC'] = self.s_rc
        self.s_data.loc[self.roll_num, 'Listening'] = self.s_lis
        self.s_data.loc[self.roll_num, 'Writing'] = self.s_wri
        self.s_data.loc[self.roll_num, 'Math'] = self.s_math

        # Saving the data to the file in the directory loaction and file name
        given by the user previously when opening the file
        self.s_data.to_csv(self.dir_path + '/' + self.data_file_name + '.csv'
, index=False)

        return self.s_data

# Creating a class to calculate subject-wise averages
class Avg:

    def __init__(self, avg_data):
        self.avg_data = avg_data

    # Creating separate methods for each subject because in real world the ac
    tual data format of each subject will be different and hence calculating aveg
    aes for each of them will require different approaches
    # Creating a method for collecting the RC data from the DataFrame

```

```

def rc(self):
    rc_data = []

    # Creating a for loop to iterate over all values of 'RC' row and appending them to a list 'rc_data'
    for index, row in self.avg_data.iterrows(): # Reference link - https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas
        reading_row_rc = row['RC']
        rc_data.append(reading_row_rc)

    # Claculating the RC average
    return sum(rc_data)/len(rc_data)

# Creating a method for collecting the Listening data from the DataFrame
def listening(self):
    lis_data = []

    # Creating a for loop to iterate over all values of 'Listening' row and appending them to a list 'lis_data'
    for index, row in self.avg_data.iterrows():
        reading_row_lis = row['Listening']
        lis_data.append(reading_row_lis)

    # Claculating the Listening average
    return sum(lis_data)/len(lis_data)

# Creating a method for collecting the Writing data from the DataFrame
def writing(self):
    wri_data = []

    # Creating a for loop to iterate over all values of 'Writing' row and appending them to a list 'wri_data'
    for index, row in self.avg_data.iterrows():
        reading_row_wri = row['Writing']
        wri_data.append(reading_row_wri)

    # Claculating the Writing average
    return sum(wri_data)/len(wri_data)

# Creating a method for collecting the Math data from the DataFrame
def math(self):
    math_data = []

    # Creating a for loop to iterate over all values of 'Math' row and appending them to a list 'math_data'
    for index, row in self.avg_data.iterrows():
        reading_row_math = row['Math']
        math_data.append(reading_row_math)

```

```

        # Claculating the Math average
        return (sum(math_data)/len(math_data)) * 100

# Creating a class to tabulate subject-wise marks of a class during a particular year
class SubjectMarks:

    def __init__(self, sub_data):
        self.sub_data = sub_data

    # Creating a method to collect 'RC' data of all students from the DataFrame
    def rc(self):
        # Selecting the columns from the DataFrame
        data_df = self.sub_data[['Order', 'Name', 'RC']]
        data_df.sort_values(by=['RC'], inplace = True, ascending=False)

        # Returning tabulated data with the column headings
        return tb(data_df, headers=["Roll Order", "Name", "RC"], tablefmt='grid', showindex='never') #Reference Link - https://pypi.org/project/tabulate/

    # Creating a method to collect 'Listening' data of all students from the DataFrame
    def listening(self):
        # Selecting the columns from the DataFrame
        data_df = self.sub_data[['Order', 'Name', 'Listening']]
        data_df.sort_values(by=['Listening'], inplace = True, ascending=False)

        # Returning tabulated data with the column headings
        return tb(data_df, headers=["Roll Order", "Name", "Listening"], tablefmt='grid', showindex='never')

    # Creating a method to collect 'Writing' data of all students from the DataFrame
    def writing(self):
        # Selecting the columns from the DataFrame
        data_df = self.sub_data[['Order', 'Name', 'Writing']]
        data_df.sort_values(by=['Writing'], inplace = True, ascending=False)

        # Returning tabulated data with the column headings
        return tb(data_df, headers=["Roll Order", "Name", "Writing"], tablefmt='grid', showindex='never')

    # Creating a method to collect 'Math' data of all students from the DataFrame

```



```

    rame
    def math(self):
        # Selecting the columns from the DataFrame
        data_df = self.sub_data[['Order', 'Name', 'Math']]
        data_df.sort_values(by=['Math'], inplace = True, ascending=False)

        # Returning tabulated data with the column headings
        return tb(data_df, headers=["Roll Order", "Name", "Math"], tablefmt='
grid', showindex='never')

# Creating a class to tabulate student marks individually
class StudentMarks:

    def __init__(self, stu_data, stu_name):
        self.stu_data = stu_data
        self.stu_name = stu_name

        # Creating a method to search for a student name in the DataFrame and tab
        ulate his/her details
        def marks(self, class_call=None):
            self.class_call = class_call

            # Selecting the columns to be displayed
            data_df = pd.DataFrame(self.stu_data, columns = ['Order', 'Name', 'RC
', 'Listening', 'Writing', 'Math'])

            # Searching for the entire or part of student name in the 'Name' colu
            mn of the DataFrame using the name entered by the user
            data_df = data_df[data_df['Name'].str.contains(self.stu_name)] # Refe
            rence link - https://davidhamann.de/2017/06/26/pandas-select-elements-by-stri
            ng/

            if data_df.empty == True: # Reference Link - https://pandas.pydata.or
            g/pandas-docs/version/0.18/generated/pandas.DataFrame.empty.html
                return 'No student found!'

            # Checking to see if data is requested from another class or for tabu
            lating and displaying the DataFrame
            elif class_call == None:
                return tb(data_df, headers=["Roll Order", "Name", "RC", "Listenin
g", "Writing", "Math"], tablefmt='grid', showindex='never')

            # If data requested from another class then DataFrame is sent directl
            y without tabulating it
            else:
                return data_df

```

```

# Creating a class to display the data in the form of graphs
class GraphPlot:

    def __init__(self, graph_data_1, graph_data_2, graph_data_3, dir_loc, graph_grade):
        self.graph_data_1 = graph_data_1
        self.graph_data_2 = graph_data_2
        self.graph_data_3 = graph_data_3
        self.dir_loc = dir_loc
        self.graph_grade = graph_grade

    # Creating a method to collect the average RC values of a class over all the years
    def class_plot_rc(self):
        # Calling the 'Avg' class within this class and sending data to process
        class_call = Avg(self.graph_data_1)

        # Calling a method of a different class within this class to calculate the average levels of sent data
        rc_1 = class_call.rc()

        class_call = Avg(self.graph_data_2)
        rc_2 = class_call.rc()

        class_call = Avg(self.graph_data_3)
        rc_3 = class_call.rc()

        # Sending the calculated average levels to a different method within this class to plot the graph
        self.plot_graph(rc_1, rc_2, rc_3, 'RC', self.dir_loc, self.graph_grade)

    # Creating a method to collect the average Listening values of a class over all the years
    def class_plot_lis(self):
        class_call = Avg(self.graph_data_1)
        lis_1 = class_call.listening()

        class_call = Avg(self.graph_data_2)
        lis_2 = class_call.listening()

        class_call = Avg(self.graph_data_3)
        lis_3 = class_call.listening()

        self.plot_graph(lis_1, lis_2, lis_3, 'Listening', self.dir_loc, self.graph_grade)

    # Creating a method to collect the average Writing values of a class over

```

all the years

```
def class_plot_writing(self):
    class_call = Avg(self.graph_data_1)
    wri_1 = class_call.writing()

    class_call = Avg(self.graph_data_2)
    wri_2 = class_call.writing()

    class_call = Avg(self.graph_data_3)
    wri_3 = class_call.writing()

    self.plot_graph(wri_1, wri_2, wri_3, 'Writing', self.dir_loc, self.graph_grade)
```

Creating a method to collect the average Math values of a class over all the years

```
def class_plot_math(self):
    class_call = Avg(self.graph_data_1)
    math_1 = class_call.math()

    class_call = Avg(self.graph_data_2)
    math_2 = class_call.math()

    class_call = Avg(self.graph_data_3)
    math_3 = class_call.math()

    self.plot_graph(math_1, math_2, math_3, 'Math', self.dir_loc, self.graph_grade)
```

Creating a method to collect individual students levels over the years

```
def stu_plot(self, student_name):
    self.student_name = student_name

    # Collecting year_1 marks of a student
    year_1 = StudentMarks(self.graph_data_1, student_name)

    # Collecting all subject marks for a student
    marks_1 = year_1.marks(True)

    # Checking to see if student data exists for user entry before proceeding further
    # If data exists, the 'StudentMarks' class will return a DataFrame which when checked by 'isinstance' will result in true
    # If data does not exist then the 'StudentMarks' class will return a string and the if condition will result in false thus not executing further
    if isinstance(marks_1, pd.DataFrame):

        # Looping through the individual student data to collect subject data individually
        for index, row in marks_1.iterrows():
```

```

        rc_1 = row['RC']
        lis_1 = row['Listening']
        wri_1 = row['Writing']
        math_1 = row['Math']

    year_2 = StudentMarks(self.graph_data_2, student_name)
    marks_2 = year_2.marks(True)

    for index, row in marks_2.iterrows():
        rc_2 = row['RC']
        lis_2 = row['Listening']
        wri_2 = row['Writing']
        math_2 = row['Math']

    year_3 = StudentMarks(self.graph_data_3, student_name)
    marks_3 = year_3.marks(True)

    for index, row in marks_3.iterrows():
        rc_3 = row['RC']
        lis_3 = row['Listening']
        wri_3 = row['Writing']
        math_3 = row['Math']

    # Sending the individual student subject-wise data to another method within the class to plot the graph
    self.plot_graph(rc_1, rc_2, rc_3, 'RC', self.dir_loc, self.graph_grade, student_name)
    self.plot_graph(lis_1, lis_2, lis_3, 'Listening', self.dir_loc, self.graph_grade, student_name)
    self.plot_graph(wri_1, wri_2, wri_3, 'Writing', self.dir_loc, self.graph_grade, student_name)
    self.plot_graph(math_1, math_2, math_3, 'Math', self.dir_loc, self.graph_grade, student_name)

    else:
        print('No student data found!')

    # Creating a method to plot graphs
    def plot_graph(self, data_1, data_2, data_3, sub, loc, grade_num, student_name=None):
        self.data_1 = data_1
        self.data_2 = data_2
        self.data_3 = data_3
        self.sub = sub
        self.loc = loc
        self.grade_num = grade_num
        self.stud_name = student_name

        x = [2014, 2015, 2016]
        y = [data_1, data_2, data_3]

```

```

# Creating bar graph
plt.bar(x, y)
plt.xlabel('Year')
plt.ylabel('Level')
plt.title(sub) # Reference Link - https://www.geeksforgeeks.org/graph
-plotting-in-python-set-1/

# Setting the x-axis label frequency
plt.xticks(np.arange(min(x), max(x)+1, 1.0)) # Reference Link - https
://stackoverflow.com/questions/12608788/changing-the-tick-frequency-on-x-or-y
-axis-in-matplotlib

# Saving the generated graphs to the location selected by the user wi
th a certain naming convention
if stud_name == None:
    plt.savefig(loc + '/Graphs/' + 'Grade_' + str(grade_num) + '_' +
sub + datetime.now().strftime('%Y-%m-%d %H:%M:%S') + '.png') # Reference Link
- https://stackoverflow.com/questions/415511/how-to-get-the-current-time-in-p
ython

else:
    plt.savefig(loc + '/Graphs/' + stud_name + '_Grade_' + str(grade_
num) + '_' + sub + datetime.now().strftime('%Y-%m-%d %H:%M:%S') + '.png')

    return plt.show()

# Creating a class to authenticate user access
class Auth:

    def __init__(self, user=None, pwd=None, fer_key=None):
        self.user = user
        self.pwd = pwd
        self.fer_key = fer_key

# Creating a method to validate existing user credentials
def credentials(self):
    with open("User_Details.txt", "rb") as file:
        data = file.readlines()
        file.close()

        x = False
        y = 0
        while x == False and y < len(data):
            if (self.fer_key.decrypt(self.user) == self.fer_key.decrypt(d
ata[y])) and (self.fer_key.decrypt(self.pwd) == self.fer_key.decrypt(data[y+1
])):
                x = True
                return True
            else:

```

```

        y += 2

    if x == False:
        return False

# Creating a method to allow for new user registrations
def new_user(self):
    file = open("User_Details.txt", "ab")

    file.write(self.user)
    line = str(0) + "\n"
    file.write(line.encode('utf-8'))

    file.write(self.pwd)
    line = str(0) + "\n"
    file.write(line.encode('utf-8'))

    file.close()

    return 'User created successfully!'

# Creating a class to generate and read encryption keys
class Keys:

    # Creating a method for generating new encryption key
    def key_gen():
        key = Fernet.generate_key()

        file = open("Fernet_Key.txt", "wb")
        file.write(key)
        file.close()

        print('New key generated successfully!')

    # Creating a method for accessing the encryption key in order to decrypt
    the login credentials
    def read_key():
        with open('Fernet_Key.txt', 'rb') as file:
            data = file.read()
            file.close()

        return data

# Keys.key_gen()

```

```

login = int(input('\nWelcome to School Data Analysis Tracker!\nAre you an existing user or a new user?\n1) Existing user\n2) New user\n'))

if login == 1:
    user_id = input('\nUsername: ')
    user_id = user_id.encode()
    pwd = getpass('Password: ')
    pwd = pwd.encode() # Reference Link - https://stackoverflow.com/questions/9202224/getting-command-line-password-input-in-python

    key_code = Keys
    code = key_code.read_key()

    key = Fernet(code)

    encrypted_pwd = key.encrypt(pwd)
    encrypted_id = key.encrypt(user_id)

    user_auth = Auth(encrypted_id, encrypted_pwd, key)

    login_status = user_auth.credentials()

    if login_status == True:
        print('\nLogin Successful\n')

        # User choice for data processing
        choice = int(input('\n\nPick a choice to process data in the way required:\n1) Calculate the class average\n2) Subject-wise marks\n3) Student-wise marks\n4) Plot graph of class growth\n5) Add/edit student data\n'))

        load_data = LoadData(choice)

        # If user wants to see the growth in the form of graphs
        if choice == 4:
            comp_choice = input('Pick the growth type:\n1) Student Growth\n2) Class Growth\n')

            print('\nSelect the data directory from the pop up window to load the classroom data files from\n')

            # Collecting all the CSV files as a DataFrame for a user chosen grade level
            first_file, sec_file, third_file, d_loc, grade = load_data.file_select()

            print(f'\nThe raw data of Grade {grade} is shown below:\nYEAR 1\n{first_file}\nYEAR 2\n{sec_file}\nYEAR 3\n{third_file}')

```

```

graph
# Sending all the DataFrame files to the GraphPlot class to plot
graph_plot = GraphPlot(first_file, sec_file, third_file, d_loc, g
rade)

# For individual student growth over the years
if comp_choice == '1':
    student = input('\nEnter name of student\n')
    stu_growth = graph_plot.stu_plot(student.title())

# For overall class growth over the years
elif comp_choice == '2':
    class_growth_rc = graph_plot.class_plot_rc()
    class_growth_lis = graph_plot.class_plot_lis()
    class_growth_writing = graph_plot.class_plot_writing()
    class_growth_math = graph_plot.class_plot_math()

elif choice == 1 or choice == 2 or choice == 3 or choice == 5:
    print('\nSelect a directory from the pop up window to begin\n')

    master_data_file, d_loc, f_name = load_data.file_select()

    print(f'\nOpening file...\n{master_data_file}')

# If user wants overall class averages subject-wise for a particu
lar year
if choice == 1:
    class_average = Avg(master_data_file)

    class_average_rc = class_average.rc()
    class_average_lis = class_average.listening()
    class_average_writing = class_average.writing()
    class_average_math = class_average.math()

    print('\n\nThe class average is as below:\n\nRC: %.2f' %cla
ss_average_rc)
    print('\nListening: %.2f' %class_average_lis)
    print('\nWriting: %.2f' %class_average_writing)
    print('\nMath: %.2f' %class_average_math)

# If user wants subject-wise marks of a class for a particular ye
ar
elif choice == 2:
    subject_wise_marks = SubjectMarks(master_data_file)

    class_rc = subject_wise_marks.rc()
    class_lis = subject_wise_marks.listening()
    class_writing = subject_wise_marks.writing()
    class_math = subject_wise_marks.math()

```



```

        print(f'\n\n\nThe class RC marks from the highest to lowest a
re:\n{class_rc}')
        print(f'\n\n\nThe class Listening marks from the highest to l
owest are:\n{class_lis}')
        print(f'\n\n\nThe class Writing marks from the highest to low
est are:\n{class_writing}')
        print(f'\n\n\nThe class Math marks from the highest to lowest
are:\n{class_math}')

    # If user wants individual student marks
    elif choice == 3:
        student = input('\nEnter name of student\n')

        student_wise_marks = StudentMarks(master_data_file, student.t
itle())

        student_marks = student_wise_marks.marks()

        print(f'\nMarks for {student} are as follows:\n{student_marks
}')

    # If user wants to add/edit student data
    elif choice == 5:
        entry_choice = int(input('\n1)New student entry\n2)Edit exist
ing student data\n'))

        if entry_choice == 1:
            more_entries = True

            # Using while loop to check if user wants to enter multip
le student details
            while more_entries == True:
                student = input('\nEnter name of student\n')
                roll_entry = int(input(f'\nEnter Roll number of {stud
ent.title()}\n'))

                gender = input('\nEnter gender:\n')
                rc_entry = float(input('\nEnter RC level:\n'))
                lis_entry = float(input('\nEnter Listening level:\n'))

                wri_entry = float(input('\nEnter Writing level:\n'))
                math_entry = float(input('\nEnter Math level:\n'))

                enter_data = DataEntry(d_loc, f_name, master_data_fil
e, student.title(), gender.capitalize(), roll_entry, rc_entry, lis_entry, wri
_entry, math_entry)

                appending_data = enter_data.data_entry()

                print(f'\nThe updated data file is shown below:\n{app
ending_data}\n')

                entries = input('\nWould you like to add another entr

```

```

y?\n1) Yes\n2) No\n')

        if entries == '1' or entries.upper() == 'Y' or entries.upper() == 'YES':
            more_entries = True

        else:
            more_entries = False
            print('\nExiting...')

    elif entry_choice == 2:
        student_number = int(input('\nEnter the roll number of the student to begin editing\n'))
        name_entry = input(f'Enter new details at roll number {student_number}\n\nEnter name of student:\n')

        student_number -= 1
        name_entry = name_entry.title()

        gender = input('\nEnter gender:\n')
        rc_entry = float(input('\nEnter RC level:\n'))
        lis_entry = float(input('\nEnter Listening level:\n'))
        wri_entry = float(input('\nEnter Writing level:\n'))
        math_entry = float(input('\nEnter Math level:\n'))

        enter_data = DataEntry(d_loc, f_name, master_data_file, name_entry.title(), gender.capitalize(), student_number, rc_entry, lis_entry, wri_entry, math_entry)

        edit_data = enter_data.data_edit()

        print(f'\nUpdated file is shown below:\n{edit_data}\n\nUpdate saved successfully!\n')

    else:
        print('Invalid choice!')

    else:
        print('\nLogin failed!')

elif login == 2:
    user_id = input('\nCreate a user name: ')
    user_id = user_id.encode()
    pwd = getpass('Create your password: ')
    pwd = pwd.encode() # Reference Link - https://stackoverflow.com/questions/9202224/getting-command-line-password-input-in-python

    key_code = Keys
    code = key_code.read_key()

    key = Fernet(code)

```

```
encrypted_pwd = key.encrypt(pwd)
encrypted_id = key.encrypt(user_id)

user_auth = Auth(encrypted_id, encrypted_pwd)

login_status = user_auth.new_user()

print(f'\n{login_status}')

else:
    print('Invalid choice!')
```