Computer Engineering 4DN4
Laboratory 2
Online Grade Retrieval Application

Group 8
Hengbo Huang - 400241747
Yinwen Xu - 400195279

Lab Contribution: Hengbo Huang : client part
Yinwen Xu : server part

1.1 Server

```
def __init__(self):
    self.create_listen_socket()
    self.student_grades_database = "course_grades_2023.csv"
    self.import_student_grades_database()
    self.process_connections_forever()
```

```
def create_listen_socket(self):
    try:
        # Create an IPv4 TCP socket.
        self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        # Set socket layer socket options. This one allows us to
        # reuse the socket without waiting for any timeouts.
        self.socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

        # Bind socket to socket address, i.e., IP address and port.
        self.socket.bind(Server.SOCKET_ADDRESS)

        # Set socket to listen state.
        self.socket.listen(Server.MAX_CONNECTION_BACKLOG)
        print("Listening on port {} ...".format(Server.PORT))
    except Exception as msg:
        print(msg)
        sys.exit(1)
```

Here I create 4 attributes for the server class. “create_listen_socket()” is used to create a socket which is able to listen to the specific port and IP address.

“student_geades_database” is used to read the filename of the csv file which is used to store the students’ information(student name,id_number, Name, Encryption Key, marks of Lab 1,Lab 2,Lab 3,Lab 4,Midterm,Exam 1,Exam 2,Exam 3,Exam 4)

```
def import_student_grades_database(self):
    """Read in the student database, clean whitespace from each record,
    and parse them .

    """
    # Read in the employee database and clean whitespace from each
    # record.
    self.read_and_clean_database_records()

    # Read each line and parse the student name,id_number, Name, Encryption Key,
    #Marks of Lab 1,Lab 2,Lab 3,Lab 4,Midterm,Exam 1,Exam 2,Exam 3,Exam 4
    self.parse_student_records()
```

“Import_student_grades_database()” is used to read the student database, clean white space and parse them.

```
def read_and_clean_database_records(self):
    """Read a list of people from a database file and add them as
    employees to the company.

    Open the file and read in the lines, stripping off end-of-line
    characters. Ignore blank lines.

    """
    try:
        file = open(self.student_grades_database, "r")
        records = file.readlines()
        print("This is the data I read from CSV \n")
        for line in records:
            print(line)
        records.pop(0)

    except FileNotFoundError:
        print("Creating database: {}". format(self.student_grades_database))
        file = open(self.student_grades_database, "w+")

    # Read and process all non-blank lines in the file. The
    # following uses two list comprehensions.
    self.cleaned_records = [clean_line for clean_line in
                            [line.strip() for line in records]
                            if clean_line != '']

    #print(self.cleaned_records)
    file.close()
```

Here is the specific function implementation in “read_and_clean_database_records”, firstly use file open function to open the file and then use readlines to store every line in the file as a string into variable records. Next, we print each line as required to show that it works well. Then we use a “pop” method to get rid of the first line, which is the meaning of each

column. Lastly, we use two list comprehensions to get rid of white space in each line and close the file.

```
def parse_student_records(self):
    """Split each line into Name,ID Number,Key,Lab 1,Lab 2,Lab 3,Lab 4,Midterm,Exam 1,Exam 2,Exam 3,Exam 4

    self.employee_list is a twelve-tuple containing these
    values. Convert the id number and each grade into an int.

    """
    try:
        self.student_grade_list = [
            (s[0].strip(), int(s[1].strip()), s[2].strip(), int(s[3].strip()), int(s[4].strip()), int(s[5].strip()),
            int(s[6].strip()), int(s[7].strip()),int(s[8].strip()), int(s[9].strip()),
            int(s[10].strip()),int(s[11].strip()) ) for s in
            [s.split(',') for s in self.cleaned_records]]

        self.ID_list = []
        self.GL1_list = []
        self.GL2_list = []
        self.GL3_list = []
        self.GL4_list = []
        self.GM_list = []
        self.GE1_list = []
        self.GE2_list = []
        self.GE3_list = []
        self.GE4_list = []

        self.dict_id_key = {}

           for i in self.student_grade_list:
               self.ID_list.append(i[1])
               self.GL1_list.append(i[3])
               self.GL2_list.append(i[4])
               self.GL3_list.append(i[5])
               self.GL4_list.append(i[6])
               self.GM_list.append(i[7])
               self.GE1_list.append(i[8])
               self.GE2_list.append(i[9])
               self.GE3_list.append(i[10])
               self.GE4_list.append(i[11])
               key = i[1]                               #Student ID
               value = i[2]                             #encryption key
               self.dict_id_key.update({key:value})     #update the dictionary for student ID and encryption key

           self.GL1A = self.get_average(self.GL1_list)
           self.GL2A = self.get_average(self.GL2_list)
           self.GL3A = self.get_average(self.GL3_list)
           self.GL4A = self.get_average(self.GL4_list)
           self.GMA = self.get_average(self.GM_list)
           self.GE1A = self.get_average(self.GE1_list)
           self.GE2A = self.get_average(self.GE2_list)
           self.GE3A = self.get_average(self.GE3_list)
           self.GE4A = self.get_average(self.GE4_list)

           #print(self.student_grade_list)
           #print(self.dict_id_key)


       except Exception as e:
           print("Error: Invalid people name input file.")
           print(str(e))
           exit()
```

```
def get_average(*args):
    total_sum = 0
    total_count = 0
    for lst in args:
        if isinstance(lst, list):
            total_sum += sum(lst)
            total_count += len(lst)

    if total_count == 0:
        return 0
    else:
        return total_sum / total_count
```

The picture above shows how I implement the “parse_student_records ”. Firstly, I use two list comprehensions to get a list of tuples which have 12 elements.(each element in tuple represents one of the information for student) What’s more, I create 9 lists to store the grades of each lab, midterm and exam. After this, I use a get_average function to get the average of each list and use an attribute to store them. Next, I create a list to store the ID number and a dictionary to relate each ID number to the corresponding encryption key.

```python
class Server:
    def connection_handler(self, client):
        # Unpack the client socket address tuple.
        connection, address_port = client
        print("-" * 72)
        print("Connection received from {}.".format(address_port))
        # Output the socket address.
        print(client)

        while True:
            try:
                # Receive bytes over the TCP connection. This will block
                # until "at least 1 byte or more" is available.
                recvd_bytes = connection.recv(Server.RECV_BUFFER_SIZE_for_id)
                recvd_bytes2 = connection.recv(Server.RECV_BUFFER_SIZE_for_command)

                # If recv returns with zero bytes, the other end of the
                # TCP connection has closed (The other end is probably in
                # FIN WAIT 2 and we are in CLOSE WAIT.). If so, close the
                # server end of the connection and get the next client
                # connection.
                if len(recvd_bytes) == 0 & len(recvd_bytes2) ==0:
                    print("Closing client connection ... ")
                    connection.close()
                    break

                # Decode the received bytes back into strings. Then output
                # them.
                try:
                    recvd_id = struct.unpack('!i', recvd_bytes)[0]
                except struct.error as e:
                    print(f"struct error: {e}")
                    recvd_id = None

                print(recvd_id)
                recvd_str = recvd_bytes2.decode(Server.MSG_ENCODING)
                print("Received_id:", recvd_id)
                if recvd_id in self.ID_list:
                    print("User found.")
                else:
                    print("User not found.")
                    print("Closing client connection ... ")
                    connection.close()
                    break

                print("Received_command:", recvd_str)
```

```
Connection received from ('127.0.0.1', 61228).
(<socket.socket fd=7, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM,
 proto=0, laddr=('127.0.0.1', 50000), raddr=('127.0.0.1', 61228)>, ('127.0.0.1',
 61228))
Received_id: 1803933
User found.
Received_command: GG
Already sent the all grades of your labs and exams.
```

Firstly, it will print the IP address and Port it connects to. Here I use 2 receive function to receive student_ id and message separately. For student ID, I use struct.unpack to get the exact integer. Moreover, for the ID we received, I check whether it is in our student_ID list or not. If it is in the list, the server will print “user found”. Otherwise, it will print “user not found” and close the connection.

```
Connection received from ('127.0.0.1', 61508).
(<socket.socket fd=7, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM,
 proto=0, laddr=('127.0.0.1', 50000), raddr=('127.0.0.1', 61508)>, ('127.0.0.1',
 61508))
Received_id: 123
User not found.
Closing client connection ...
```

```
    #GMA, GL1A, GL2A, GL3A, GL4A, GEA and GG GMA/GEA are "get midterm/exam average" and the others are "get lab average" commar
    if recvd_id in self.dict_id_key:
        encryption_key = self.dict_id_key[recvd_id]
        # print("see here after key")
        # rest of your code that uses encryption_key
    else:
        print("error occurs for dict_id_key")
        # handle the case where recvd_id is not found in dict_id_key


    encryption_key_bytes = encryption_key.encode(Server.MSG_ENCODING)
    fernet = Fernet(encryption_key_bytes)
    # print("see here")

    if recvd_str == 'GL1A' :
        message = "The average grade of Lab 1  is :" + str(self.GL1A) + "\n"
        message_bytes = message.encode(Server.MSG_ENCODING)
        connection.sendall(fernet.encrypt(message_bytes))
        print("Already sent the average grade of Lab 1.")

    elif recvd_str == 'GL2A' :
        message = "The average grade of Lab 2  is :" + str(self.GL2A) + "\n"
        message_bytes = message.encode(Server.MSG_ENCODING)
        connection.sendall(fernet.encrypt(message_bytes))
        print("Already sent the average grade of Lab 2.")

    elif recvd_str == 'GL3A' :
        message = "The average grade of Lab 3  is :" + str(self.GL3A) + "\n"
        message_bytes = message.encode(Server.MSG_ENCODING)
        connection.sendall(fernet.encrypt(message_bytes))
        print("Already sent the average grade of Lab 3.")

    elif recvd_str == 'GL4A' :
        message = "The average grade of Lab 4  is :" + str(self.GL4A) + "\n"
        message_bytes = message.encode(Server.MSG_ENCODING)
        connection.sendall(fernet.encrypt(message_bytes))
        print("Already sent the average grade of Lab 4.")
```

```
    elif recvd_str == 'GMA' :
        message = "The average grade of midterm exam is :" + str(self.GMA) + "\n"
        message_bytes = message.encode(Server.MSG_ENCODING)
        connection.sendall(fernet.encrypt(message_bytes))
        print("Already sent the average grade of midterm.")

    elif recvd_str == 'GEA' :
        message = "The average grade of Exam 1  is : " + str(self.GE1A) + "\n" \
                  "The average grade of Exam 2  is : " + str(self.GE2A) + "\n" \
                  "The average grade of Exam 3  is : " + str(self.GE3A) + "\n" \
                  "The average grade of Exam 4  is : " + str(self.GE4A) + "\n"
        message_bytes = message.encode(Server.MSG_ENCODING)
        connection.sendall(fernet.encrypt(message_bytes))
        print("Already sent the average grade of 4 exams.")

    elif recvd_str == 'GG' :
        index = 0
        for i in range(len(self.ID_list)):
            if self.ID_list[i] == recvd_id:
                index = i

        message = "Your grade of Lab 1  is :" + str(self.student_grade_list[index][3]) +"\n" \
                  "Your grade of Lab 2  is :" + str(self.student_grade_list[index][4]) +"\n" \
                  "Your grade of Lab 3  is :" + str(self.student_grade_list[index][5]) +"\n" \
                  "Your grade of Lab 4  is :" + str(self.student_grade_list[index][6]) +"\n" \
                  "Your grade of midterm  is :" + str(self.student_grade_list[index][7]) +"\n" \
                  "Your grade of exam 1  is :" + str(self.student_grade_list[index][8]) +"\n" \
                  "Your grade of exam 2  is :" + str(self.student_grade_list[index][9]) +"\n" \
                  "Your grade of exam 3  is :" + str(self.student_grade_list[index][10]) +"\n" \
                  "Your grade of exam 4  is :" + str(self.student_grade_list[index][11]) +"\n"
        message_bytes = message.encode(Server.MSG_ENCODING)
        connection.sendall(fernet.encrypt(message_bytes))
        print("Already sent the all grades of your labs and exams.")

    # print("see after if")

except KeyboardInterrupt:
    print()
    print("Closing client connection ... ")
    connection.close()
    break
```

After receiving the student_ID, server will check which encryption key it will use for this student. Next, it will check the command it received to determine what message it will send using the corresponding encryption key.

1.2 Client

```
class Client:
    command = "";
    student_id = "";
    encryption_key = "none";
```

Three variables are declaration to store command , student id and encryption key.

```
def __init__(self):
    while(1):
        print(" Please enter student ID and command")
        print(" user commands list : GMA, GL1A, GL2A, GL3A, GL4A, GEA, GG")

        self.get_console_input();
        if(self.command != ""):
            self.get_socket()
            self.connect_to_server()
            self.send_console_input_forever()
```

```
def get_console_input(self):
    # In this version we keep prompting the user until a non-blank
    # line is entered, i.e., ignore blank lines.
    while True:
        self.student_id = int(input("Student ID: "))
        self.command = input("Command: ")
        if self.command != "":
            print("Command entered:",self.command)
            if (self.command == "GMA" ):
                print("Fetching Midterm average")
            elif(self.command == "GEA"):
                print("Fetching Exam average")
            elif(self.command == "GL1A"):
                print("Fetching Lab1 average")
            elif(self.command == "GL2A"):
                print("Fetching Lab2 average")
            elif(self.command == "GL3A"):
                print("Fetching Lab3 average")
            elif(self.command == "GL4A"):
                print("Fetching Lab4 average")
            elif(self.command == "GG"):
                print("Fetching Grade of student", self.student_id)
            else:
                print("command not find")
            break
```

After the client part is called , it will first print two sentences to let the user enter student id and command, with the list of commands. Once the user enters the student id and command, it will connect to the server and wait for the response. Also it will print the meaning of the command, if there not such command, it will print command not find.

```
    def read_csv(self):
        self.student_list = list(csv.reader(open('decode_key.csv')))
    def find_key(self):
        for i in range(21):
            if( str(self.student_id) == str(self.student_list[i][1]) ):
                self.encryption_key = self.student_list[i][2]
```

After receiving the response from the server, it will find encryption_key according to the student. decode_key.csv file is the file that only saves the name, student id and encryption key file. It’s more convenient and faster for the client to find the key by saving them in a csv file.

	A	B	C	D	E	F	G	H
1	Name	ID Numbe	Key					
2	Kacie Stepl	1803933	M7E8erO15CIh902P8DQsHxKbOADTgEPGHdiY0MplTuY=					
3	Yassin Jorc	1884159	PWMKkdXW4VJ3pXBpr9UwjefmllxYwPzk11Aw9TQ2wZQ=					
4	Lowri Matl	1853847	UVpoR9emlZDrpQ6pCLYopzE2Qm8bCrVyGEzdOOo2wXw=					
5	Tiya Sherid	1810192	bHdhydsHzwKdb0RF4wG72yGm2a2L-CNzDl7vaWOu9KA=					
6	Nikola Forı	1891352	iHsXoe_5Fle-PHGtgZUCs5ariPZT-LNCUYpixMC3NxI=					
7	Veer Blair	1811313	IR_IQPnIM1TI8h4USnBLuUtC72cQ-u4Fwvlu3q5npA0=					
8	Isabelle Mc	1804841	kE8FpmTv8d8sRPlswQjCMaqunLUGoRNW6OrYU9JWZ4w=					
9	Samir Grea	1881925	_B__AgO34W7urog-thBu7mRKj3AY46D8L26yedUwf0I=					
10	Zander Ker	1877711	dLOM7DyrEnUsW-Q7OM6LXxZsbCFhjmyhsVT3P7oADqk=					
11	Shahzaib B	1830894	aM4bOtearz2GpURUxYKW23t_DlljFLzbfgWS-IRMB3U=					
12	Morgan Bu	1855191	-lieSn1zKJ8P3XOjyAlRcD2KbeFl_BnQjHyCE7-356w=					
13	Amaan Ro	1821012	Lt5wWqTM1q9gNAgME4T5-5oVptAstg9llB4A_iNAYMY=					
14	Theodore I	1844339	M6glRgMP5Y8CZIs-MbyFvev5VKW-zbWyUMMt44QCzG4=					
15	Ace Branch	1898468	SS0XtthxP64E-z4oB1lsdrzJwu1PUq6hgFqP_u435AA=					
16	Anthony B	1883633	0L_o75AEsOay_ggDJtOFWkgRpvFvM0snlDm9gep786I=					
17	Tobey Bell	1808742	9BXraBysqT7QZLBjegET0e52WklQ7BBYWXvv8xpbvr8=					
18	Jannat Cas	1863450	M0PgiJutAM_L9jvyfrGDWnbfJOXmhYt_skL0S88ngkU=					

decode_key

```
else:
    print("Received: ", recvd_bytes_decryption.decode(Server.MSG_ENCODING))
    print("Closing server connection ... ")
    #self.socket.close()
    self.command = ""
```

After receiving the content from the server and printing it in the terminal, the client will close the connection with the server until the user enters a new student id and command.

1.3 Termial output:

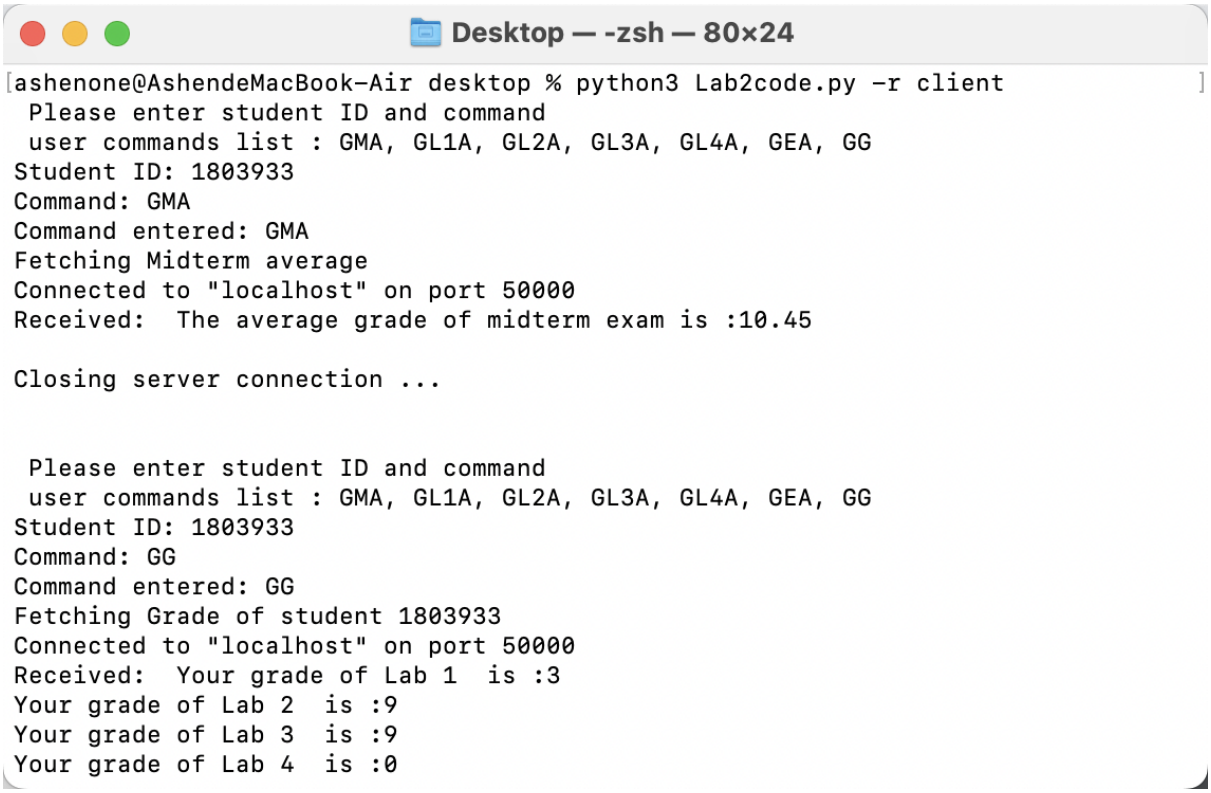
client after connection

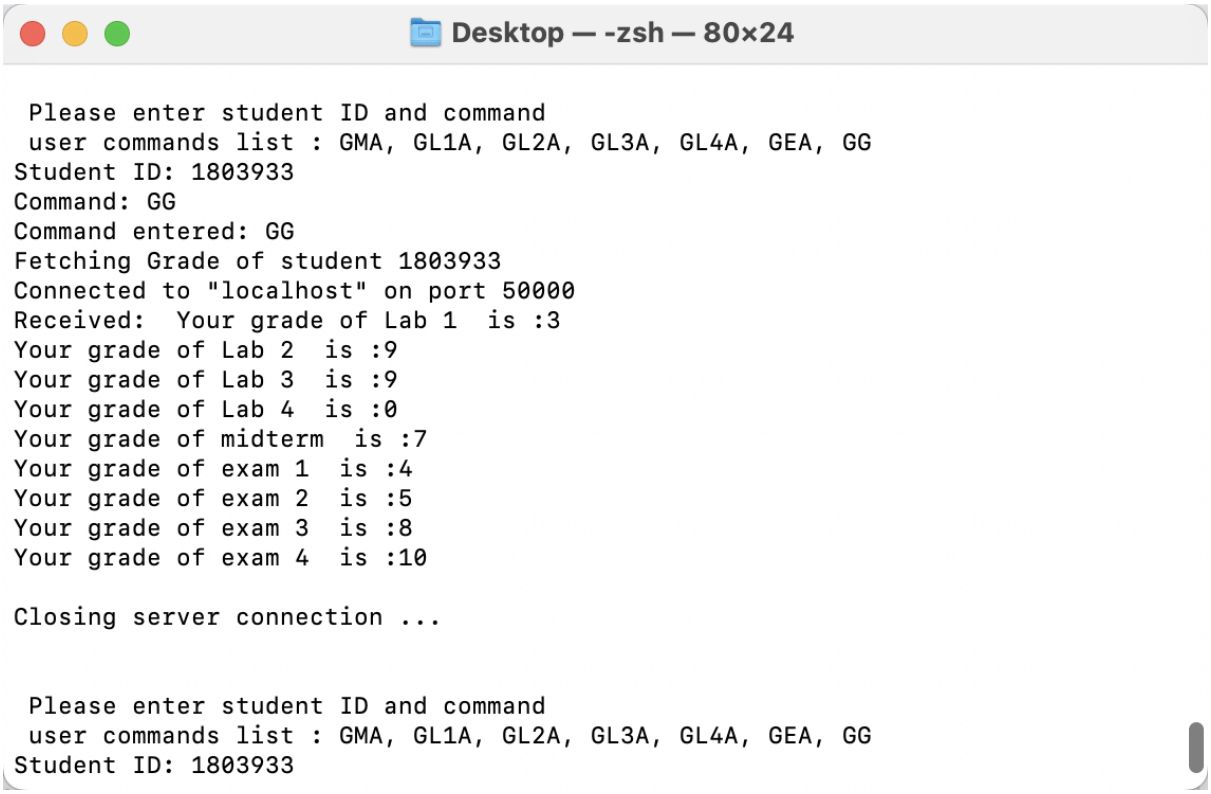
```
PS C:\Users\22749\Desktop\Year4\DN\lab2> python DNlab2.p
y -r client
 Please enter student ID and command
 user commands list : GMA, GL1A, GL2A, GL3A, GL4A, GEA,
GG
Student ID:
```

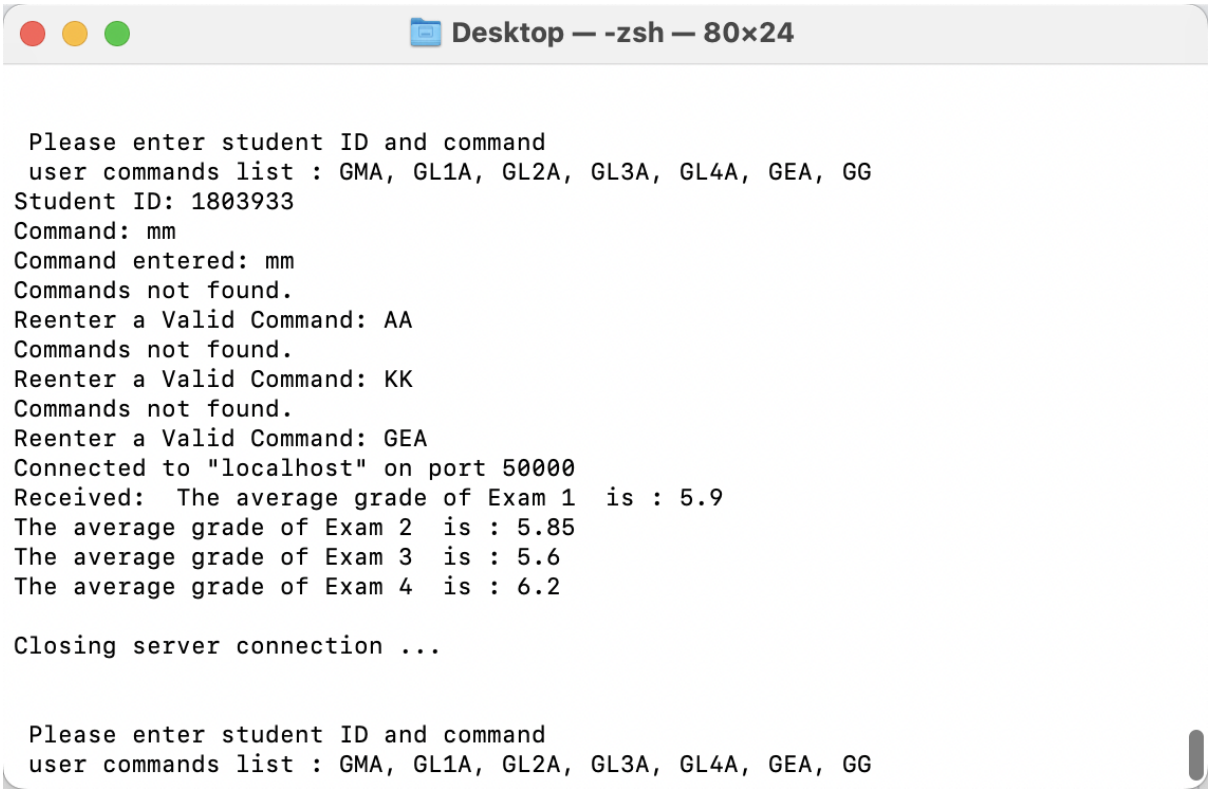
server after connection

```
rectory                                    python DNlab2.py
 -r server   2749\Desktop\Year4\DN\lab2>
Listening on port 50000 ...
This is the data I read from CSV

Name,ID Number,Key,Lab 1,Lab 2,Lab 3,Lab 4,Midterm,Exam 1
,Exam 2,Exam 3,Exam 4

Kacie Stephenson,1803933,M7E8erO15CIh902P8DQsHxKbOADTgEPG
HdiY0MplTuY=,3,9,9,0,7,4,5,8,10

Yassin Jordan,1884159,PWMKkdXW4VJ3pXBpr9UwjefmlIxYwPzk11A
w9TQ2wZQ=,9,2,10,3,8,3,9,5,7

Lowri Mathews,1853847,UVpoR9emIZDrpQ6pCLYopzE2Qm8bCrVyGEz
dOOo2wXw=,9,0,0,2,17,6,10,7,4

Tiya Sheridan,1810192,bHdhydsHzwKdb0RF4wG72yGm2a2L-CNzDl7
vaWOu9KA=,10,1,0,6,15,8,7,6,6

Nikola Forrest,1891352,iHsXoe_5Fle-PHGtgZUCs5ariPZT-LNCUY
pixMC3NxI=,4,7,0,6,5,0,5,5,10

Veer Blair,1811313,IR_IQPnIM1TI8h4USnBLuUtC72cQ-u4Fwvlu3q
5npA0=,4,8,5,3,12,7,4,0,2

Isabelle Mcgrath,1804841,kE8FpmTv8d8sRPIswQjCMaqunLUGoRNW
6OrYU9JWZ4w=,1,7,4,0,13,8,9,6,0

Samir Greaves,1881925,_B__AgO34W7urog-thBu7mRKj3AY46D8L26
yedUwf0I=,6,3,7,1,6,4,6,5,9
```

Some examples to prove it works as required.
Client Part:


Desktop — -zsh — 80×24
ashenone@AshendeMacBook-Air desktop % python3 Lab2code.py -r client
Please enter student ID and command
user commands list : GMA, GL1A, GL2A, GL3A, GL4A, GEA, GG
Student ID: 1803933
Command: GMA
Command entered: GMA
Fetching Midterm average
Connected to "localhost" on port 50000
Received: The average grade of midterm exam is :10.45
Closing server connection ...
Please enter student ID and command
user commands list : GMA, GL1A, GL2A, GL3A, GL4A, GEA, GG
Student ID: 1803933
Command: GG
Command entered: GG
Fetching Grade of student 1803933
Connected to "localhost" on port 50000
Received: Your grade of Lab 1 is :3
Your grade of Lab 2 is :9
Your grade of Lab 3 is :9
Your grade of Lab 4 is :0


Desktop — -zsh — 80×24
Please enter student ID and command
user commands list : GMA, GL1A, GL2A, GL3A, GL4A, GEA, GG
Student ID: 1803933
Command: GG
Command entered: GG
Fetching Grade of student 1803933
Connected to "localhost" on port 50000
Received: Your grade of Lab 1 is :3
Your grade of Lab 2 is :9
Your grade of Lab 3 is :9
Your grade of Lab 4 is :0
Your grade of midterm is :7
Your grade of exam 1 is :4
Your grade of exam 2 is :5
Your grade of exam 3 is :8
Your grade of exam 4 is :10
Closing server connection ...
Please enter student ID and command
user commands list : GMA, GL1A, GL2A, GL3A, GL4A, GEA, GG
Student ID: 1803933


Desktop — -zsh — 80×24
Please enter student ID and command
user commands list : GMA, GL1A, GL2A, GL3A, GL4A, GEA, GG
Student ID: 1803933
Command: mm
Command entered: mm
Commands not found.
Reenter a Valid Command: AA
Commands not found.
Reenter a Valid Command: KK
Commands not found.
Reenter a Valid Command: GEA
Connected to "localhost" on port 50000
Received: The average grade of Exam 1 is : 5.9
The average grade of Exam 2 is : 5.85
The average grade of Exam 3 is : 5.6
The average grade of Exam 4 is : 6.2
Closing server connection ...
Please enter student ID and command
user commands list : GMA, GL1A, GL2A, GL3A, GL4A, GEA, GG

Server Part


Desktop — Python Lab2code.py -r server — 80×24
ashenone@AshendeMacBook-Air Desktop % python3 Lab2code.py -r server
Listening on port 50000 ...
This is the data I read from CSV
Name,ID Number,Key,Lab 1,Lab 2,Lab 3,Lab 4,Midterm,Exam 1,Exam 2,Exam 3,Exam 4
Kacie Stephenson,1803933,M7E8erO15CIh902P8DQsHxKbOADTgEPGHdiY0MplTuY=,3,9,9,0,7,4,5,8,10
Yassin Jordan,1884159,PWMKkdXW4VJ3pXBpr9UwjefmlIxYwPzk11Aw9TQ2wZQ=,9,2,10,3,8,3,9,5,7
Lowri Mathews,1853847,UVpoR9emIZDrpQ6pCLYopzE2Qm8bCrVyGEzdOOo2wXw=,9,0,0,2,17,6,10,7,4
Tiya Sheridan,1810192,bHdhydsHzwKdb0RF4wG72yGm2a2L-CNzDl7vaWOu9KA=,10,1,0,6,15,8,7,6,6
Nikola Forrest,1891352,iHsXoe_5Fle-PHGtgZUCs5ariPZT-LNCUYpixMC3NxI=,4,7,0,6,5,0,5,5,10
Veer Blair,1811313,IR_IQPnIM1TI8h4USnBLuUtC72cQ-u4Fwvlu3q5npA0=,4,8,5,3,12,7,4,0,2

Desktop — Python Lab2code.py -r server — 80×24

```
,2

Isabelle Mcgrath,1804841,kE8FpmTv8d8sRPIswQjCMaqunLUGoRNW6OrYU9JWZ4w=,1,7,4,0,13
,8,9,6,0

Samir Greaves,1881925,_B__AgO34W7urog-thBu7mRKj3AY46D8L26yedUwf0I=,6,3,7,1,6,4,6
,5,9

Zander Kendall,1877711,dLOM7DyrEnUsW-Q7OM6LXxZsbCFhjmyhsVT3P7oADqk=,8,10,5,4,17,
4,8,10,2

Shahzaib Buckley,1830894,aM4bOtearz2GpURUxYKW23t_DlljFLzbfgWS-IRMB3U=,4,5,7,9,8,
5,7,0,6

Morgan Bush,1855191,-IieSn1zKJ8P3XOjyAlRcD2KbeFl_BnQjHyCE7-356w=,1,6,7,10,1,5,7,
2,8

Amaan Robbins,1821012,Lt5wWqTM1q9gNAgME4T5-5oVptAstg9llB4A_iNAYMY=,2,8,4,4,8,0,9
,5,8

Theodore Lawson,1844339,M6glRgMP5Y8CZIs-MbyFvev5VKW-zbWyUMMt44QCzG4=,0,7,10,7,14
,9,2,2,9

Ace Branch,1898468,SS0XtthxP64E-z4oB1IsdrzJwu1PUq6hgFqP_u435AA=,10,1,3,7,11,9,9,
```

Desktop — Python Lab2code.py -r server — 80×24

```
Anthony Bernard,1883633,0L_o75AEsOay_ggDJtOFWkgRpvFvM0snlDm9gep786I=,4,1,10,8,19
,10,9,9,9

Tobey Bell,1808742,9BXraBysqT7QZLBjegET0e52WklQ7BBYWXvv8xpbvr8=,2,10,8,2,10,9,0,
8,6

Jannat Cassidy,1863450,M0PgiJutAM_L9jvyfrGDWnbfJOXmhYt_skL0S88ngkU=,5,2,4,5,10,4
,5,9,3

Imran Marquez,1830190,v-5GfMaI2ozfmef5BNO5hI-fEGwtKjuI1XcuTDh-wsg=,9,9,1,6,17,10
,0,7,5

Amani Castro,1835544,LI14DbKGBfJExlwLodr6fkV4Pv4eABWkEhzArPbPSR8=,5,9,5,7,3,7,6,
8,4

Blanka Holt,1820930,zoTviAO0EACFC4rFereJuc0A-99Xf_uOdq3GiqUpoeU=,9,5,2,0,8,6,0,7
,10

--------------------------------------------------------------------------
Connection received from ('127.0.0.1', 62145).
(<socket.socket fd=7, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM,
 proto=0, laddr=('127.0.0.1', 50000), raddr=('127.0.0.1', 62145)>, ('127.0.0.1',
 62145))
```

Desktop — Python Lab2code.py -r server — 80×24

```
--------------------------------------------------------------------------------
Connection received from ('127.0.0.1', 62145).
(<socket.socket fd=7, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM,
 proto=0, laddr=('127.0.0.1', 50000), raddr=('127.0.0.1', 62145)>, ('127.0.0.1',
 62145))
Received_id: 1803933
User found.
Received_command: GMA
Already sent the average grade of midterm.
Closing client connection ...
--------------------------------------------------------------------------------
Connection received from ('127.0.0.1', 62153).
(<socket.socket fd=7, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM,
 proto=0, laddr=('127.0.0.1', 50000), raddr=('127.0.0.1', 62153)>, ('127.0.0.1',
 62153))
Received_id: 1803933
User found.
Received_command: GG
Already sent the all grades of your labs and exams.
Closing client connection ...
--------------------------------------------------------------------------------
Connection received from ('127.0.0.1', 62156).
(<socket.socket fd=7, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM,
 proto=0, laddr=('127.0.0.1', 50000), raddr=('127.0.0.1', 62156)>, ('127.0.0.1',
```

Desktop — Python Lab2code.py -r server — 80×24

```
Received_id: 1803933
User found.
Received_command: GG
Already sent the all grades of your labs and exams.
Closing client connection ...
--------------------------------------------------------------------------------
Connection received from ('127.0.0.1', 62156).
(<socket.socket fd=7, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM,
 proto=0, laddr=('127.0.0.1', 50000), raddr=('127.0.0.1', 62156)>, ('127.0.0.1',
 62156))
Received_id: 1803933
User found.
Received_command: GEA
Already sent the average grade of 4 exams.
Closing client connection ...
--------------------------------------------------------------------------------
Connection received from ('127.0.0.1', 62199).
(<socket.socket fd=7, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM,
 proto=0, laddr=('127.0.0.1', 50000), raddr=('127.0.0.1', 62199)>, ('127.0.0.1',
 62199))
Received_id: 123
User not found.
Closing client connection ...
```