**J.D. TYTLER SCHOOL**



**ACADEMIC YEAR : 2021-22**

**PROJECT REPORT ON**

**SCHOOL MANAGEMENT SYSTEM**

**ROLL NO :**

**NAME : DIVYAM SETHI**

**CLASS : XII - C**

**SUBJECT : COMPUTER SCIENCE**

**PROJECT GUIDE: Mr. Parveen Bhatia**

**Department of Computer Science**

**J.D. TYTLER SCHOOL**

**NEW RAJINDER NAGAR**

**NEW DELHI-110060**

## J.D. TYTLER SCHOOL



# **CERTIFICATE OF COMPLETION**

This is to certify that \_\_DIVYAM SETHI\_\_ Roll No: \_ \_ has successfully completed the project Work entitled **SCHOOL MANAGEMENT SYSTEM** in the subject Computer Science laid down in the regulations of CBSE for the purpose of Practical Examination in Class XII to be held in J.D. Tytler School on\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**Under Guidance Of:**

Mr. Parveen Bhatia

(Dept. of Computer Science)

**Examiner:**

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature:

|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENTS** | | |
| **SER** | **DESCRIPTION** | **PAGE NO** |
| 01 | ACKNOWLEDGEMENT | **04** |
| 02 | INTRODUCTION | **05** |
| 03 | OBJECTIVES OF THE PROJECT | **05** |
| 04 | PROPOSED SYSTEM | **06** |
| 05 | SOURCE CODE | **07** |
| 06 | OUTPUT | **11** |
| 07 | HARDWARE AND SOFTWARE REQUIREMENT | **13** |
| 08 | BIBLIOGRAPHY | **15** |

**ACKNOWLEDGEMENT**

Apart from the efforts of me, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I express deep sense of gratitude to almighty God for giving me strength for the successful completion of the project.

I express my heartfelt gratitude to my parents for constant encouragement while carrying out this project.

I gratefully acknowledge the contribution of the individuals who contributed in bringing this project up to this level, who continues to look after me despite my flaws,

I express my deep sense of gratitude to the luminary The Principal Of J.D. TYTLER SCHOOL who has been continuously motivating and extending their helping hand to us.

I express my sincere thanks to the academician The Vice Principal for constant encouragement and the guidance provided during this project

My sincere thanks to  **Mr. Parveen Bhatia**, A guide, Mentor all the above a friend, who critically reviewed my project and helped in solving each and every problem, occurred during implementation of the project

The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project. I am grateful for their constant support and help.

**PROJECT ON**

**SCHOOL**

**MANAGEMENT SYSTEM**

**INTRODUCTION**

Data management is not only a big task when it comes to storing data automatically and systematically , but it is also a difficult task , so in order to handle this problem , we should have a system for software management , which wont only help to maintain the data but will also help you to store it .

This program will help you keep record of your students , his class batch , fees data , maintain result records and generating report cards and distributing to indivisual students through their mail , reducing human efforts and doing in easiest way possible .

**ENJOY!!!!!!**

­­­

**OBJECTIVES OF THE PROJECT**

The objective of this project is to allow the management to maintain a clean and structured detail of their quiz programme.

Write programs utilizing modern software tools.

1. Apply simple principles effectively when developing small to medium sized projects.
2. Write effective procedural code to store small to medium sized information.
3. Students will demonstrate a breadth of knowledge in computer science, as exemplified in the areas of systems, theory and software development.
4. Students will demonstrate ability to conduct a research or applied Computer Science project, requiring writing and presentation skills which exemplify scholarly style in computer science.

**PROPOSED SYSTEM**

Today one cannot afford to rely on the fallible human beings of be really wants to stand against today’s merciless competition where not to wise saying **“to err is human”** no longer valid, it’s outdated to rationalize your mistake. So, to keep pace with time, to bring about the best result without malfunctioning and greater efficiency so to replace the unending heaps of flies with a much sophisticated hard disk of the computer.

One has to use the data management software. Software has been an ascent in atomization various organisations. Many software products working are now in markets, which have helped in making the organizations work easier and efficiently. Data management initially had to maintain a lot of ledgers and a lot of paper work has to be done but now software product on this organization has made their work faster and easier. Now only this software has to be loaded on the computer and work can be done.

This prevents a lot of time and money. The work becomes fully automated and any information regarding the organization can be obtained by clicking the button. Moreover, now it’s an age of computers of and automating such an organization gives the better look.

**SOURCE CODE**

import os

from PIL import Image

from fpdf import FPDF

from tkinter import \*

import tkinter as tk

from PIL import Image

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.base import MIMEBase

from email import encoders

#=====================================  TIME  =====================================#

import datetime

x=datetime.datetime.now()

year = str(x.year)

month = str(x.month)

date = str(x.day)

#===================================  DATABASE  ===================================#

import mysql.connector as conn

database = conn.connect( host = 'localhost' , user = 'root' , passwd = 'mysql123' )

dbconn = database.cursor(buffered=True)

school\_details = ['School name ' , 'School Address' , 'school code' , 'school Principle' , 'school email' ]

try :

    school\_logo = Image.open('C:/Users/divya/OneDrive/Desktop/PROJECT/LOGO.jpeg')

except :

    pass

try:

    query = 'use school;'

    dbconn.execute(query)

except:

    query = 'create database school;'

    dbconn.execute(query)

    query = 'use  school;'

    dbconn.execute(query)

    query = f'''create table student ( ADMISSION\_NO BIGINT NOT NULL PRIMARY KEY   , NAME VARCHAR(20) NOT NULL

             ,  class\_ varchar(10) NOT NULL , FATHER VARCHAR(20) ,  MOTHER VARCHAR(20)   , ADM\_DATE VARCHAR(15) NOT NULL

             ,  FEES BIGINT NOT NULL , BALANCE BIGINT    , PHONE VARCHAR(13) NOT NULL , EMAIL VARCHAR(30)

             ,  ADDRESS VARCHAR(100) NOT NULL , DOB VARCHAR(15) NOT NULL  ) ; '''

    dbconn.execute(query)

    database.commit()

annonce = '''

enter 'S' OR 's' to get information about school :

enter 'A' OR 'a' to access student details :

enter 'O' or 'o' to get other help :       \n'''

command = input(annonce)

if command == 'S' or command == 's' :

    for detail in school\_details:

        print(detail)

    try :

        school\_logo.show()

    except :

        pass

elif command == 'A' or command == 'a' :

    announcement = '''

    press 'N' or 'n' to show non-payment students :

    press 'S' OR 's' to get student details by admn\_no :

    press 'P' OR 'p' to get student details by phone :

    press 'A' or 'a' to add new student :

    press 'M' or 'm' to modify student details :     \n '''

    command\_2 = input(announcement)

    if command\_2 == 'N' or command\_2 == 'n' :

        query = "SELECT \* FROM student where BALANCE > 0;"

        dbconn.execute(query)

        for y in dbconn :

            print(y)

    elif command\_2 == 'S' or command\_2 == 's' :

        adm\_no = input('enter admission number ')

        query = f"SELECT \* FROM student where ADMISSION\_NO = {adm\_no} ;"

        dbconn.execute(query)

        for y in dbconn :

            print(y)

    elif command\_2 == 'P' or command\_2 == 'p' :

        phn\_no = input('enter phone number ')

        query = f"SELECT \* FROM student where PHONE = {phn\_no} ;"

        dbconn.execute(query)

        for y in dbconn :

            print(y)

    elif command\_2 == 'A' or command\_2 == 'a' :

        query = f"SELECT count(\*) from student ;"

        dbconn.execute(query)

        for y in dbconn :

             total\_no\_students = str(int(y[0]) + 1)

        print(total\_no\_students)

        query = f'desc student;'

        dbconn.execute(query)

        data = []

        data\_record = []

        for y in dbconn :

            data.append(y[0])

            if y[0] == 'ADMISSION\_NO' :

                data\_record.append(str(total\_no\_students+year))

            elif y[0] == 'ADM\_DATE':

                data\_record.append(str(year+'-'+month+'-'+date))

            elif y[0] == 'BALANCE' :

                data\_record.append(0)

            else :

                x = input(f'{y[0] } ;- ')

                data\_record.append(x)

        value = ''

        for xy in data\_record :

            if value == '' :

                value = xy

            else :

                 value = str(value) +'," '+ str(xy) + '"'

        query = f'insert into student values ({value});'

        dbconn.execute(query)

        database.commit()

        print('record added succesfully')

    elif  command\_2 == 'M' or command\_2 == 'm' :

        query = f"USE SCHOOL ;"

        dbconn.execute(query)

        adm\_no = input("enter admission number of student to modify changes :")

        query = f"SELECT \* FROM student where ADMISSION\_NO = {adm\_no} ;"

        dbconn.execute(query)

        for y in dbconn :

            data\_record = y

        data = []

        query = f'desc student;'

        dbconn.execute(query)

        for y in dbconn :

            data.append(y[0])

        edit\_dict = dict()

        for edit in range(len(data)) :

            if data[edit] != 'ADMISSION\_NO' :

                entry = input(f"{data[edit]} = {data\_record[edit]}   :")

                edit\_dict[data[edit]] = str(entry)

        value = ''

        for xy in edit\_dict :

            if (value == '' or value == ' ') == True and  edit\_dict[xy] != '' :

                value =  xy + ' = " ' + edit\_dict[xy] + ' " '

            elif edit\_dict[xy] != '' :

                 value = str(value) + ', ' + xy + '  = " ' + edit\_dict[xy] + ' " '

        query = f"USE SCHOOL ;"

        dbconn.execute(query)

        query = f'''UPDATE student

        SET {value}

        WHERE ADMISSION\_NO = {adm\_no};'''

        dbconn.execute(query)

        database.commit()

        print('changes done successfully')

else :

    batch\_year = int(input("ENTER BATCH YEAR :-"))

    try:

        query = f'use classes\_{batch\_year};'

        dbconn.execute(query)

    except:

        confirm = input('''THIS CLASS BATCH DOES NOT EXIT

        press "Y" OR "y" to create a new batch else press any key ''')

        if confirm == 'Y' or confirm == 'y' :

            query = f'create database classes\_{batch\_year};'

            dbconn.execute(query)

            query = f'create database results\_{batch\_year};'

            dbconn.execute(query)

    annonce = '''

    press 'C' OR 'c' to access class details :

    press 'R' OR 'r' to access report card details :

    press any key to exit :       \n'''

    command = input(annonce)

    #=======================================#  CLASSES #=======================================#

    if command == 'C' or command == 'c' :

        try:

            query = f'use classes\_{batch\_year};'

            dbconn.execute(query)

        except:

            confirm = input('''THIS CLASS BATCH DOES NOT EXIT

            press "Y" OR "y" to create a new batch else press any key ''')

            if confirm == 'Y' or confirm == 'y' :

                query = f'create database classes\_{batch\_year};'

                dbconn.execute(query)

        def get\_class\_student\_detail(class\_sec , admn\_no):

            query = f"USE SCHOOL ;"

            dbconn.execute(query)

            query = f"SELECT \* FROM student where ADMISSION\_NO = {admn\_no} ;"

            dbconn.execute(query)

            data\_record = ''

            for y in dbconn :

                data\_record = y

            data = []

            query = f'desc student;'

            dbconn.execute(query)

            for y in dbconn :

                data.append(y[0])

            edit\_dict = dict()

            for edit in range(len(data\_record)) :

                if data[edit] != 'ADMISSION\_NO' :

                    edit\_dict[data[edit]] = data\_record[edit]

            query = f'use classes\_{batch\_year};'

            dbconn.execute(query)

            query = f'''select \* from  class\_{str(class\_sec)}\_{str(batch\_year)} where ADMISSION\_NO = {admn\_no} ; '''

            dbconn.execute(query)

            data\_record = ''

            for y in dbconn :

                data\_record = y

            if data\_record == '' :

                print(f'{admn\_no} not present in class {class\_sec}')

            data = []

            query = f'desc class\_{str(class\_sec)}\_{str(batch\_year)};'

            dbconn.execute(query)

            for y in dbconn :

                data.append(y[0])

            for edit in range(len(data\_record)) :

                edit\_dict[(data[edit])] = data\_record[edit]

            print(edit\_dict)

        announcement = f'''

        press 'S' or 's' to show classes in batch {batch\_year} :

        press 'A' OR 'a' to add a new class :

        press 'D' OR 'd' to get student details of a class :

        press 'C' OR 'c' to get students of a class :

        press 'W' or 'w' to add new student :

        press 'M' or 'm' to modify student details :

        press 'R' or 'r' to remove a student :  \n '''

        command = input(announcement)

        if command ==  'S' or command == 's' :

            query = f'use classes\_{batch\_year};'

            dbconn.execute(query)

            query = f'show tables;'

            dbconn.execute(query)

            for y in dbconn :

              print(y)

        elif command == 'A' or command == 'a' :

            query = f'use classes\_{batch\_year};'

            dbconn.execute(query)

            class\_sec = input('enter class and section to create a new record :')

            class\_subject = int(input(f'enter subjects allocated in class {class\_sec} :'))

            X = ''

            for i in range(class\_subject):

                X = X +f', SUBJECT\_{i+1} VARCHAR(20) '

            query = f'''create table class\_{str(class\_sec)}\_{str(batch\_year)} ( ADMISSION\_NO BIGINT NOT NULL PRIMARY KEY

             , CLASS\_TEACHER VARCHAR(20) NOT NULL , BEHAVIOUR VARCHAR(20) {X}  ) ; '''

            dbconn.execute(query)

            database.commit()

        elif command == 'D' or command == 'd' :

            class\_sec = input('enter class and section to get student details :')

            admn\_no = input('enter admission number to get student details')

            get\_class\_student\_detail(class\_sec , admn\_no)

        elif command == 'C' or command == 'c' :

            class\_sec = input('enter class and section to get student details :')

            query = f'use classes\_{batch\_year};'

            dbconn.execute(query)

            query = f'SELECT \* FROM class\_{str(class\_sec)}\_{str(batch\_year)};'

            dbconn.execute(query)

            DATA = dbconn.fetchall()

            for y in DATA :

                get\_class\_student\_detail(class\_sec , y[0])

        elif command == 'W' or command == 'w' :

            query = f'use classes\_{batch\_year};'

            dbconn.execute(query)

            class\_sec = input('enter class and section to add student details :')

            query = f'desc class\_{str(class\_sec)}\_{str(batch\_year)};'

            dbconn.execute(query)

            data = []

            data\_record = []

            for y in dbconn :

                data.append(y[0])

                x = input(f'{y[0] } ;- ')

                data\_record.append(x)

            value = ''

            for xy in data\_record :

                if value == '' :

                    value = xy

                else :

                    value = str(value) +'," '+ str(xy) + '"'

            query = f'insert into class\_{str(class\_sec)}\_{str(batch\_year)} values ({value});'

            dbconn.execute(query)

            database.commit()

            print('record added succesfully')

        elif command == 'M' or command == 'm' :

            query = f'use classes\_{batch\_year};'

            dbconn.execute(query)

            class\_sec = input('enter class and section to get student details :')

            adm\_no = input("enter admission number of student to modify changes :")

            query = f"SELECT \* FROM class\_{str(class\_sec)}\_{str(batch\_year)} where ADMISSION\_NO = {adm\_no} ;"

            dbconn.execute(query)

            for y in dbconn :

                data\_record = y

            data = []

            query = f'desc class\_{str(class\_sec)}\_{str(batch\_year)};'

            dbconn.execute(query)

            for y in dbconn :

                data.append(y[0])

            edit\_dict = dict()

            for edit in range(len(data)) :

                if data[edit] != 'ADMISSION\_NO' :

                    entry = input(f"{data[edit]} = {data\_record[edit]}   :")

                    edit\_dict[data[edit]] = str(entry)

            value = ''

            for xy in edit\_dict :

                if (value == '' or value == ' ') == True and  edit\_dict[xy] != '' :

                    value =  xy + ' = " ' + edit\_dict[xy] + ' " '

                elif edit\_dict[xy] != '' :

                    value = str(value) + ', ' + xy + '  = " ' + edit\_dict[xy] + ' " '

            query = f'''UPDATE class\_{str(class\_sec)}\_{str(batch\_year)}

            SET {value}

            WHERE ADMISSION\_NO = {adm\_no};'''

            dbconn.execute(query)

            database.commit()

            print('changes done successfully')

        elif command == 'R' or command == 'r' :

            class\_sec = input('enter class and section to get student details :')

            adm\_no = input("enter admission number of student to remove from class :")

            query = f'DELETE FROM class\_{str(class\_sec)}\_{str(batch\_year)} WHERE ADMISSION\_NO = {adm\_no} ;'

            dbconn.execute(query)

            database.commit()

            print('student removed from class successfully')

    #==============================#  RESULTS #===============================#

    if command == 'R'or command == 'r' :

        try:

            query = f'use classes\_{batch\_year};'

            dbconn.execute(query)

        except:

            confirm = input('''THIS CLASS BATCH DOES NOT HAVE RESULT

            press "Y" OR "y" to create a new else press any key ''')

            if confirm == 'Y' or confirm == 'y' :

                query = f'create database results\_{batch\_year};'

                dbconn.execute(query)

        def get\_class\_student\_detail(class\_sec , result\_type , admn\_no):

            query = f"USE SCHOOL ;"

            dbconn.execute(query)

            query = f"SELECT \* FROM student where ADMISSION\_NO = {admn\_no} ;"

            dbconn.execute(query)

            data\_record = ''

            for y in dbconn :

                data\_record = y

            data = []

            query = f'desc student;'

            dbconn.execute(query)

            for y in dbconn :

                data.append(y[0])

            edit\_dict = dict()

            for edit in range(len(data\_record)) :

                if data[edit] != 'ADMISSION\_NO' :

                    edit\_dict[data[edit]] = data\_record[edit]

            query = f'use results\_{batch\_year};'

            dbconn.execute(query)

            query = f'''select \* from  result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)} where ADMISSION\_NO = {admn\_no} ; '''

            dbconn.execute(query)

            data\_record = ''

            for y in dbconn :

                data\_record = y

            if data\_record == '' :

                print(f'{admn\_no} not present in result of {class\_sec} in {result\_type}')

            data = []

            query = f'desc result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)};'

            dbconn.execute(query)

            for y in dbconn :

                data.append(y[0])

            for edit in range(len(data\_record)) :

                edit\_dict[(data[edit])] = data\_record[edit]

            print(edit\_dict)

        announcement = f'''

        press 'S' or 's' to show results in batch {batch\_year} :

        press 'A' OR 'a' to add a new result class :

        press 'D' OR 'd' to get student details in result :

        press 'C' OR 'c' to get result of students :

        press 'W' or 'w' to add new student :

        press 'M' or 'm' to modify student result :

        press 'R' or 'r' to remove a student :

        press 'G' or 'g' to generate pdf :

        press 'E' or 'e' to email result :  \n '''

        command = input(announcement)

        if command ==  'S' or command == 's' :

            query = f'use results\_{batch\_year};'

            dbconn.execute(query)

            query = f'show tables;'

            dbconn.execute(query)

            for y in dbconn :

              print(y)

        elif command == 'A' or command == 'a' :

            query = f'use results\_{batch\_year};'

            dbconn.execute(query)

            class\_sec = input('enter class and section to create a new record :')

            class\_subject = int(input(f'enter subjects allocated in class {class\_sec} :'))

            result\_type = input('enter result type :')

            X = ''

            for i in range(class\_subject):

                X = X +f', SUBJECT\_{i+1} VARCHAR(20) , MARKS\_{i+1} INT '

            query = f'''create table result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)} ( ADMISSION\_NO BIGINT NOT NULL PRIMARY KEY

             , MAX\_MARKS INT  NOT NULL , EXAM\_TYPE VARCHAR(20) NOT NULL {X}  ) ; '''

            dbconn.execute(query)

            database.commit()

        elif command == 'D' or command == 'd' :

            class\_sec = input('enter class and section to get student details :')

            admn\_no = input('enter admission number to get student details')

            result\_type = input('enter exam type :')

            get\_class\_student\_detail(class\_sec , result\_type , admn\_no)

        elif command == 'C' or command == 'c' :

            class\_sec = input('enter class and section to get student details :')

            result\_type = input('enter exam type :')

            query = f'use results\_{batch\_year};'

            dbconn.execute(query)

  DATA = dbconn.fetchall()

            query = f'SELECT \* FROM result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)};'

            dbconn.execute(query)

            for y in DATA :

                get\_class\_student\_detail(class\_sec , result\_type , y[0])

        elif command == 'W' or command == 'w' :

            query = f'use results\_{batch\_year};'

            dbconn.execute(query)

            class\_sec = input('enter class and section to add student details :')

            result\_type = input('enter exam type :')

            query = f'desc result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)};'

            dbconn.execute(query)

            data = []

            data\_record = []

            for y in dbconn :

                data.append(y[0])

                x = input(f'{y[0] } ;- ')

                data\_record.append(x)

            value = ''

            for xy in data\_record :

                if value == '' :

                    value = xy

                else :

                    value = str(value) +'," '+ str(xy) + '"'

            query = f'insert into result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)} values ({value});'

            dbconn.execute(query)

            database.commit()

            print('record added succesfully')

        elif command == 'M' or command == 'm' :

            query = f'use results\_{batch\_year};'

            dbconn.execute(query)

            class\_sec = input('enter class and section to get student details :')

            adm\_no = input("enter admission number of student to modify changes :")

            result\_type = input('enter exam type :')

            query = f"SELECT \* FROM result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)} where ADMISSION\_NO = {adm\_no} ;"

            dbconn.execute(query)

            for y in dbconn :

                data\_record = y

            data = []

            query = f'desc result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)};'

            dbconn.execute(query)

            for y in dbconn :

                data.append(y[0])

            edit\_dict = dict()

            for edit in range(len(data)) :

                if data[edit] != 'ADMISSION\_NO' :

                    exit = input(f"{data[edit]} = {data\_record[edit]}   :")

                    if exit != '' or exit != ' ' :

                        edit\_dict[data[edit]] = exit

            value = ''

            for xy in edit\_dict :

                if (value == '' or value == ' ') == True and  edit\_dict[xy] != '' :

                    value =  xy + '  = ' + edit\_dict[xy]

                elif edit\_dict[xy] != ''  :

                    value = str(value) + ', ' + xy + '  = " ' + edit\_dict[xy] + ' " '

            query = f'''UPDATE result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)}

            SET {value}

            WHERE ADMISSION\_NO = {adm\_no} ; '''

            dbconn.execute(query)

            database.commit()

            print('changes done successfully')

        elif command == 'R' or command == 'r' :

            query = f'use results\_{batch\_year};'

            dbconn.execute(query)

            class\_sec = input('enter class and section to get student details :')

            adm\_no = input("enter admission number of student to remove from class :")

            result\_type = input('enter exam type :')

            query = f'DELETE FROM result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)} WHERE ADMISSION\_NO = {adm\_no} ;'

            dbconn.execute(query)

            database.commit()

            print('student removed from result successfully')

        elif command == 'G' or command == 'g'  :

            class\_sec = input('enter class and section to get student details :')

            result\_type = input('enter exam type :')

            query = f'use results\_{batch\_year};'

            dbconn.execute(query)

            #=============#  GENERATE PERIODIC TEST REPORT CARD  #====================#

            def generate\_report\_card\_periodic( edit\_dict , edit\_list , class\_teacher , class\_sec ) :

                admission\_no  = edit\_dict['ADMISSION\_NO']

                name          = edit\_dict['NAME']

                date\_of\_birth = edit\_dict['DOB']

                school\_details = ['J.D. TYTLER SCHOOL ' , 'NEW RAJINDER NAGAR , R BLOCK , NEW DELHI 110060' , '85032' , 'Ms NEENA ANDREW' , 'divyamsethi1804@gmail.com' ]

                subjects = dict()

                for xn in range(14 , len(edit\_list) , 2) :

                    subjects[edit\_list[xn][1]] = edit\_list[xn+1][1]

                school   = school\_details[0]

                exam     = edit\_dict['EXAM\_TYPE']

                standard = class\_sec

                maximum\_marks = edit\_dict['MAX\_MARKS']

                total\_marks = 0

                pdf = FPDF('P', 'mm', 'A4')

                pdf.add\_page()

                pdf.set\_font('Arial','B', size = 12)

                try :

                    pdf.image('C:/Users/divya/OneDrive/Desktop/PROJECT/JDTS LOGO.jpeg' , x = 10, y = 10, h = 50  )

                except :

                    pass

                pdf.set\_font('Arial', 'B' , size = 25 )

                pdf.ln(10)

                pdf.set\_text\_color(34,139,34)

                pdf.cell(235,10,f'{school}' , align = 'C' )

                pdf.ln(15)

                pdf.set\_text\_color(255,0,0)

                pdf.cell(235,10,f'{exam}' , align = 'C' )

                pdf.ln(30)

                pdf.set\_text\_color(0,0,0)

                pdf.set\_font('Arial', 'B' , size = 15)

                pdf.cell( 18 , 10 ,f' NAME : { name }'                   , align = 'L' )

                pdf.cell(150 , 10 ,f' ADMISSION NO : { admission\_no } '  , align = 'R')

                pdf.ln(15)

                pdf.cell(18 , 10 ,f' CLASS : { standard }'      , align = 'L' )

                pdf.cell(150 , 10 ,f' DOB : { date\_of\_birth } '  , align = 'R')

                pdf.ln(20)

                pdf.set\_font('Arial', 'B' , size = 12)

                pdf.cell(50 , 10 ,'SUBJECTS'                    , 1 , align = 'L' )

                pdf.cell(45 , 10 ,f'MARKS ( {maximum\_marks} )'  , 1 , align = 'C' )

                pdf.cell(45 , 10 ,f'PERCENTAGE'                 , 1 , align = 'C' )

                pdf.cell(45 , 10 ,'GRADES'                      , 1 , align = 'C' )

                pdf.ln(10)

                to\_percent = 100 / int(maximum\_marks)

                for key in subjects :

                    try :

                        percentage = round(float(subjects[key]) \* float(to\_percent) , 2 )

                        if percentage > 90 :

                            grade = 'A1'

                        elif 90 >= percentage > 80 :

                            grade = 'A2'

                        elif 80 >= percentage > 70 :

                            grade = 'B1'

                        elif 70 >= percentage > 60 :

                            grade = 'B2'

                        elif 60 >= percentage > 50 :

                            grade = 'C1'

                        elif 50 >= percentage > 40 :

                            grade = 'C2'

                        elif 40 >= percentage >= 33 :

                            grade = 'D'

                        else :

                            grade = 'E'

                    except :

                        grade = 'AB'

                        percentage = '-'

                    pdf.cell(50 ,10 ,f'{key}'           , 1 , align = 'L' )

                    pdf.cell(45 ,10 ,f'{subjects[key]}' , 1 , align = 'C' )

                    pdf.cell(45 ,10 ,f'{percentage} %'  , 1 , align = 'C' )

                    pdf.cell(45 ,10 ,f'{grade}'         , 1 , align = 'C' )

                    pdf.ln(10)

                    try :

                        total\_marks += float(subjects[key])

                    except :

                        pass

                marks\_percentage = round(( total\_marks / len(subjects) ) \* to\_percent , 2)

                pdf.cell(50 , 10 ,'GRAND TOTAL'           , 1 , align = 'L' )

                pdf.cell(45 , 10 ,f'{total\_marks}'        , 1 , align = 'C' )

                pdf.cell(45 , 10 ,f'{marks\_percentage} %' , 1 , align = 'C' )

                pdf.cell(45 , 10 ,''    , 1 )

                pdf.ln(20)

                pdf.set\_font('Arial', 'B' , size = 10 )

                pdf.cell(30 , 5 ,'PERCENTAGE'  , 1 )

                pdf.cell(30 , 5 ,'MARKS'       , 1 )

                pdf.cell(30 , 5 ,'GRADE'       , 1 )

                pdf.ln(5)

                pdf.cell(30 , 5 ,'91% - 100%'  , 1 )

                pdf.cell(30 , 5 ,' 23 to 25 '  , 1 )

                pdf.cell(30 , 5 ,'  A1  '      , 1 )

                pdf.ln(5)

                pdf.cell(30 , 5 ,'81% - 90%'   , 1 )

                pdf.cell(30 , 5 ,' 21 to 22 '  , 1 )

                pdf.cell(30 , 5 ,'  A2  '      , 1 )

                pdf.ln(5)

                pdf.cell(30 , 5 ,'71% - 80%'   , 1 )

                pdf.cell(30 , 5 ,' 18 to 20 '  , 1 )

                pdf.cell(30 , 5 ,'  B1  '      , 1 )

                pdf.ln(5)

                pdf.cell(30 , 5 ,'61% - 70%'   , 1 )

                pdf.cell(30 , 5 ,' 16 to 17 '  , 1 )

                pdf.cell(30 , 5 ,'  B2  '      , 1 )

                pdf.ln(5)

                pdf.cell(30 , 5 ,'51% - 60%'   , 1 )

                pdf.cell(30 , 5 ,' 13 to 15 '  , 1 )

                pdf.cell(30 , 5 ,'  C1  '      , 1 )

                pdf.ln(5)

                pdf.cell(30 , 5 ,'41% - 50%'   , 1 )

                pdf.cell(30 , 5 ,' 11 to 12 '  , 1 )

                pdf.cell(30 , 5 ,'  C2  '      , 1 )

                pdf.ln(5)

                pdf.cell(30 , 5 ,'33% - 40%'   , 1 )

                pdf.cell(30 , 5 ,' 8 to 10 '   , 1 )

                pdf.cell(30 , 5 ,'  D  '       , 1 )

                pdf.ln(5)

                pdf.cell(30 , 5 ,'Below 33%'   , 1 )

                pdf.cell(30 , 5 ,' 0 to 7 '    , 1 )

                pdf.cell(30 , 5 ,'  E  '       , 1 )

                pdf.ln(15)

                pdf.set\_font('Arial', 'B' , size = 12)

                pdf.ln(10)

                pdf.cell(20 , 10 ,f'{school\_details[3]} '        , align = 'L' )

                pdf.cell(150 , 10 ,f'{ class\_teacher } '      , align = 'R' )

                pdf.ln(10)

                pdf.cell(25 , 10 ,f'PRINCIPAL '       , align = 'C' )

                pdf.cell(150 , 10 ,f'CLASS TEACHER '   , align = 'R' )

                if os.path.isdir(f"D:/report card") == False :

                    os.mkdir(f"D:/report card")

                if os.path.isdir(f"D:/report card/{exam} {standard}") == False :

                    os.mkdir(f"D:/report card/{exam} {standard}")

                pdf.output(f'D:/report card/{exam} {standard}/{name} {admission\_no} {standard} {exam}.pdf' , 'F')

            query = f'SELECT \* FROM result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)};'

            dbconn.execute(query)

            row = dbconn.fetchall()

            for y in row :

                admn\_no = y[0]

                query = f"USE SCHOOL ;"

                dbconn.execute(query)

                query = f"SELECT \* FROM student where ADMISSION\_NO = {admn\_no} ;"

                dbconn.execute(query)

                data\_record = ''

                row\_1 = dbconn.fetchall()

                for y in row\_1 :

                    data\_record = y

                data = []

                query = f'desc student;'

                dbconn.execute(query)

                for y in dbconn :

                    data.append(y[0])

                edit\_dict = dict()

                for edit in range(len(data\_record)) :

                    if data[edit] != 'ADMISSION\_NO' :

                        edit\_dict[data[edit]] = data\_record[edit]

                query = f'use results\_{batch\_year};'

                dbconn.execute(query)

                query = f'''select \* from  result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)} where ADMISSION\_NO = {admn\_no} ; '''

                dbconn.execute(query)

                row\_2 = dbconn.fetchall()

                data\_record = ''

                for y in row\_2 :

                    data\_record = y

                if data\_record == '' :

                    print(f'{admn\_no} not present in result of {class\_sec} in {result\_type}')

                data = []

                query = f'desc result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)};'

                dbconn.execute(query)

                for y in dbconn :

                    data.append(y[0])

                for edit in range(len(data\_record)) :

                    edit\_dict[(data[edit])] = data\_record[edit]

                edit\_list = [ (k ,v) for  k , v in edit\_dict.items()]

                query = f'use classes\_{batch\_year};'

                dbconn.execute(query)

                query = f'''select \* from  class\_{str(class\_sec)}\_{str(batch\_year)} where ADMISSION\_NO = {admn\_no} ; '''

                dbconn.execute(query)

                row\_3 = dbconn.fetchall()

                data\_record = ''

                for y in row\_3 :

                    data\_record = y

                if data\_record == '' :

                    print(f'{admn\_no} not present in class {class\_sec}')

                data = []

                query = f'desc class\_{str(class\_sec)}\_{str(batch\_year)};'

                dbconn.execute(query)

                for y in dbconn :

                    data.append(y[0])

                edit\_dict\_1 = dict()

                for edit in range(len(data\_record)) :

                    edit\_dict\_1[(data[edit])] = data\_record[edit]

                class\_teacher = edit\_dict\_1['CLASS\_TEACHER']

                generate\_report\_card\_periodic( edit\_dict , edit\_list , class\_teacher , class\_sec )

            print(' report cards are generated pls check "D:\ report card"')

        elif command == 'E' or command == 'e' :

            class\_sec = input('enter class and section to get student details :')

            result\_type = input('enter exam type :')

            school\_details = ['J.D. TYTLER SCHOOL ' , 'NEW RAJINDER NAGAR , R BLOCK , NEW DELHI 110060' , '85032' , 'Ms NEENA ANDREW' , 'divyamsethi1804@gmail.com' ]

            fromaddr = school\_details[4]

            student\_dire  = "C:/Users/divya/OneDrive/Desktop/11 - c email.csv"

            purpose = f'RESULT {result\_type} {class\_sec}'

            def gmail\_content( fromaddr , toaddr , password , admn\_no , name , purpose , class\_sec , exam ) :

                msg = MIMEMultipart()

                msg['From'] = fromaddr

                msg['To'] = toaddr

                msg['Subject'] = f'{name}   {purpose}'

                # string to store the body of the mail

                body = f'{name}   {purpose}'

                msg.attach(MIMEText(body, 'plain'))

                filename = f"{name}.pdf"

                attachment = open( f'D:/report card/{exam} {class\_sec}/{name} {admn\_no} {class\_sec} {exam}.pdf' , "rb")

                p = MIMEBase('application', 'octet-stream')

                p.set\_payload((attachment).read())

                encoders.encode\_base64(p)

                p.add\_header('Content-Disposition', "attachment; filename= %s" % filename)

                msg.attach(p)

                s = smtplib.SMTP('smtp.gmail.com', 587)

                s.starttls()

                s.login(fromaddr, password )

                text = msg.as\_string()

                s.sendmail(fromaddr, toaddr, text)

                s.quit()

            def algorithum(password):

                query = f'use results\_{batch\_year};'

                dbconn.execute(query)

                query = f'SELECT \* FROM result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)};'

                dbconn.execute(query)

                row = dbconn.fetchall()

                for y in row :

                    admn\_no = y[0]

                    query = f"USE SCHOOL ;"

                    dbconn.execute(query)

                    query = f"SELECT \* FROM student where ADMISSION\_NO = {admn\_no} ;"

                    dbconn.execute(query)

                    data\_record = ''

                    for y in dbconn :

                        data\_record = y

                    data = []

                    query = f'desc student;'

                    dbconn.execute(query)

                    for y in dbconn :

                        data.append(y[0])

                    edit\_dict = dict()

                    for edit in range(len(data\_record)) :

                        if data[edit] != 'ADMISSION\_NO' :

                            edit\_dict[data[edit]] = data\_record[edit]

                    query = f'use results\_{batch\_year};'

                    dbconn.execute(query)

                    query = f'''select \* from  result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)} where ADMISSION\_NO = {admn\_no} ; '''

                    dbconn.execute(query)

                    data\_record = ''

                    for y in dbconn :

                        data\_record = y

                    if data\_record == '' :

                        print(f'{admn\_no} not present in result of {class\_sec} in {result\_type}')

                    data = []

                    query = f'desc result\_{str(class\_sec)}\_{str(result\_type)}\_{str(batch\_year)};'

                    dbconn.execute(query)

                    for y in dbconn :

                        data.append(y[0])

                    for edit in range(len(data\_record)) :

                        edit\_dict[(data[edit])] = data\_record[edit]

                    toaddr = edit\_dict['EMAIL']

                    admn\_no = edit\_dict['ADMISSION\_NO']

                    name = edit\_dict['NAME']

                    exam     = edit\_dict['EXAM\_TYPE']

                    try :

                        gmail\_content( fromaddr , toaddr , password , admn\_no , name , purpose , class\_sec , exam )

                        print(f'mailed sent succesfully to {name} admn no { admn\_no}' )

                    except:

                        print(f'failed to send to {name} admn no { admn\_no}' )

            root=tk.Tk()

            root.geometry("400x400")

            passw\_var=tk.StringVar()

            def submit():

                passw\_label = tk.Label(root, text = 'Password entered', font = ('calibre',15,'bold')).grid(row=3,column=1)

                algorithum(passw\_var.get())

            passw\_label = tk.Label(root, text = 'Password', font = ('calibre',15,'bold'))

            passw\_entry=tk.Entry(root, textvariable = passw\_var, font = ('calibre',15,'normal'), show = '\*')

            sub\_btn=tk.Button(root,text = 'Submit', font = ('calibre',15,'normal'), command = submit)

            passw\_label.grid(row=1,column=0)

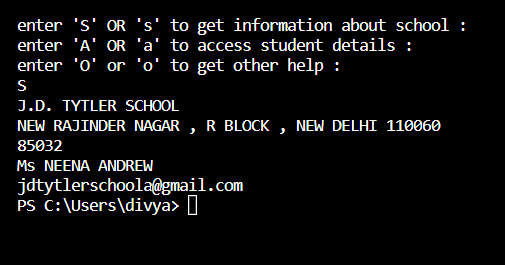
            passw\_entry.grid(row=1,column=1)

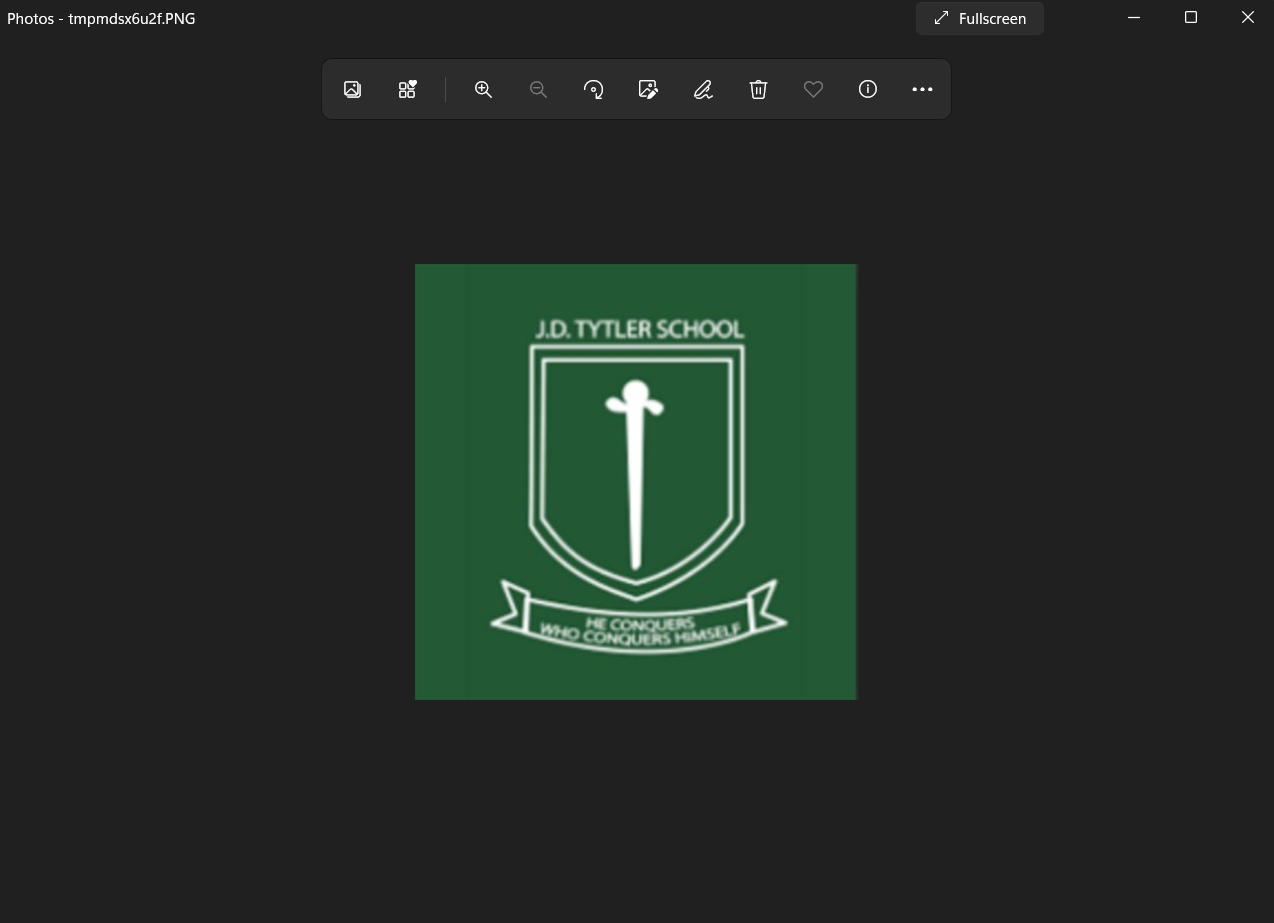
            sub\_btn.grid(row=2,column=1)

            root.mainloop()

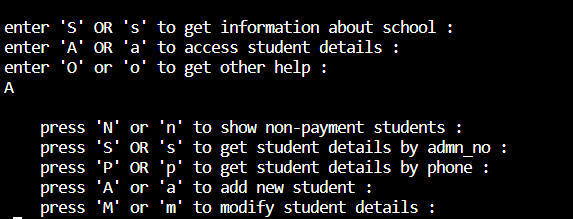
**OUTPUT**

**1 ) viewing school details : -**

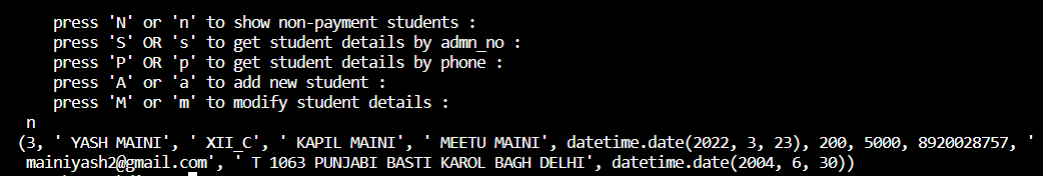




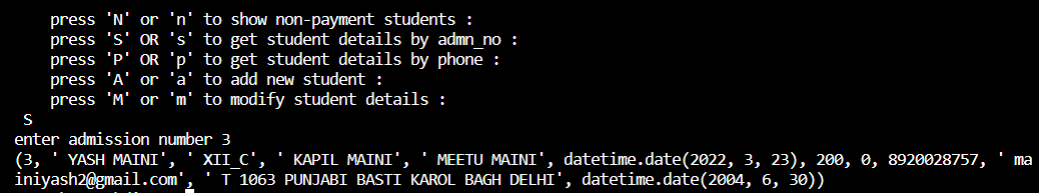
1. **) viewing / EDITING STUDENT DETAILS : -**



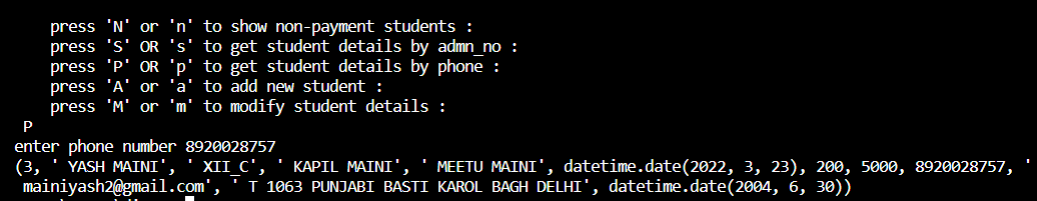
**A ) viewing non payment STUDENTs DETAILS : -**



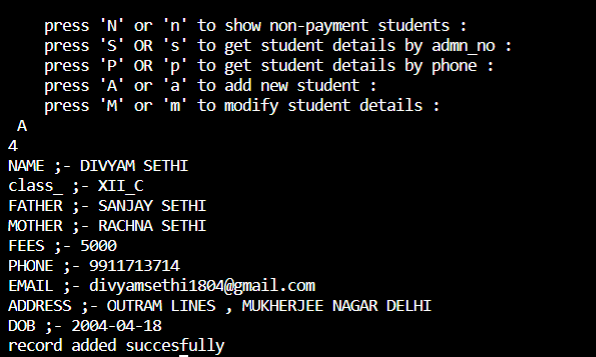
**B ) view STUDENT DETAILS by admission number : -**



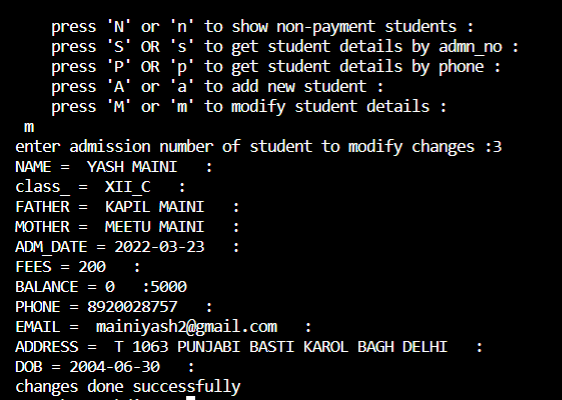
**C ) view STUDENT DETAILS by PHONE number : -**



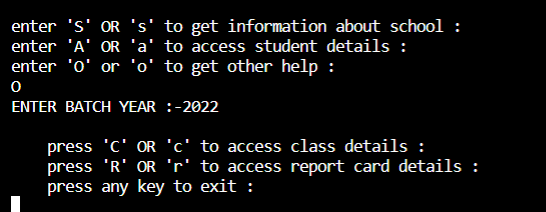
**D ) TO ADD A NEW STUDENT : -**

****

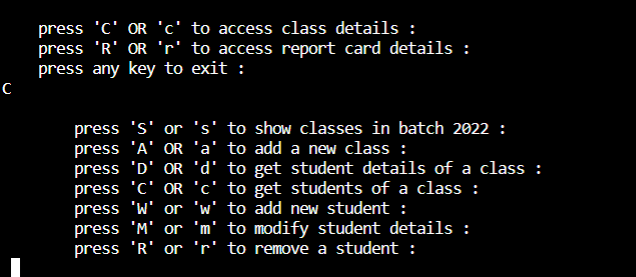
**E ) TO MODIFY STUDENTs DETAILS : -**



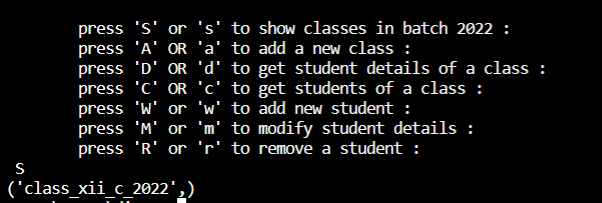
**3 ) viewing CLASS DETAILS AND RESULT : -**

****

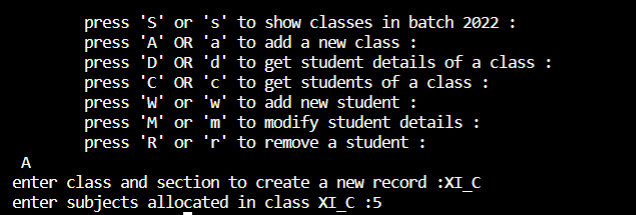
**A ) TO GET CLASS DETAILS : -**



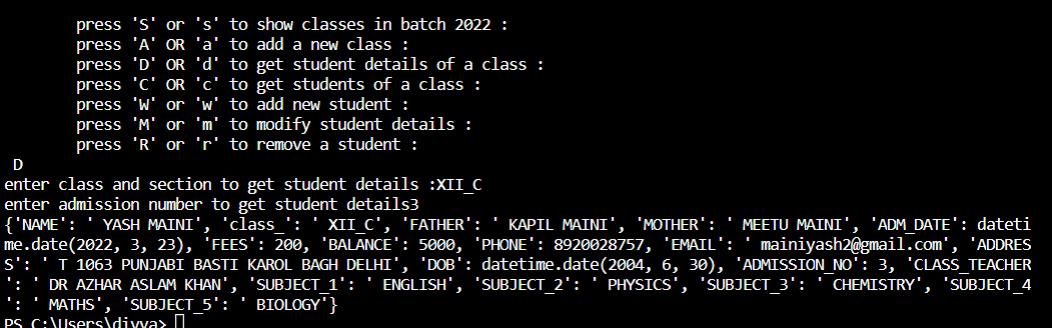
* **TO GET CLASSES IN GIVEN BATCH : -**



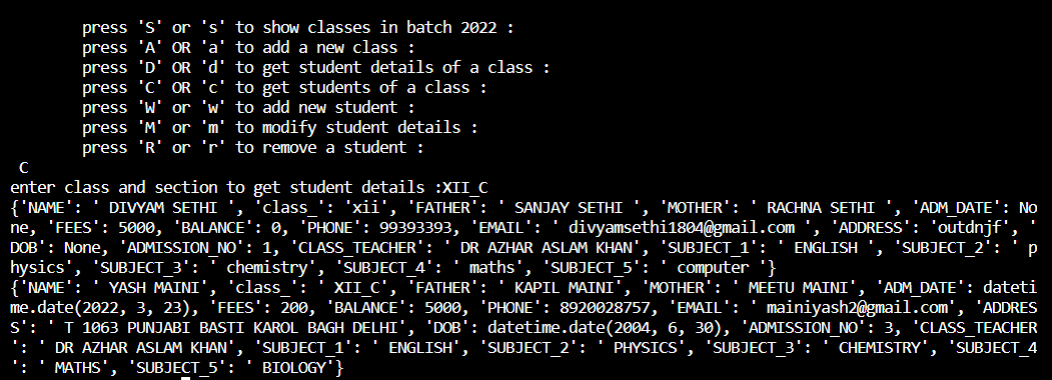
* **TO ADD A NEW CLASS IN GIVEN BATCH : -**



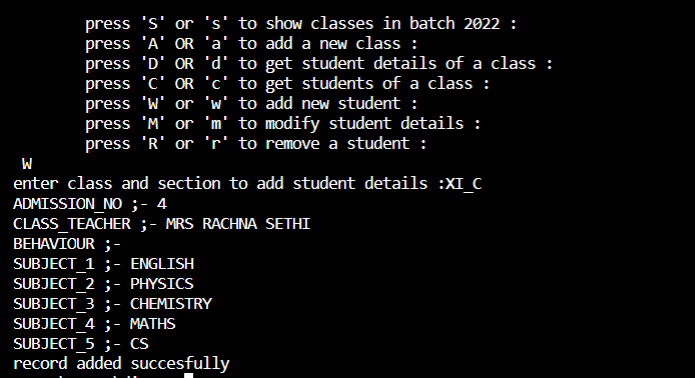
* **TO GET STUDENT DETAILS IN GIVEN CLASS BATCH : -**



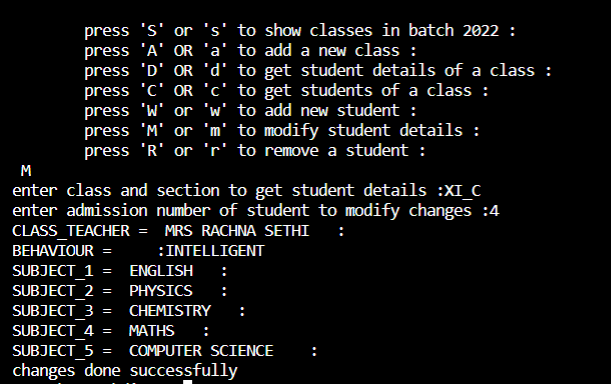
* **TO GET STUDENTS DETAIL IN GIVEN CLASS BATCH : -**



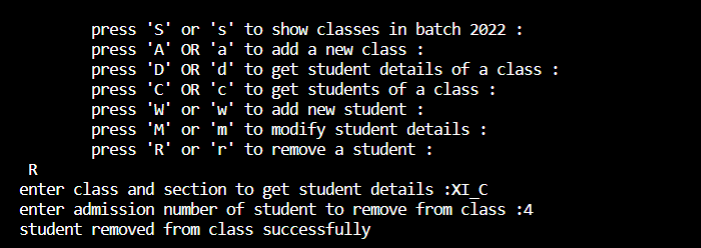
* **TO ADD STUDENT IN GIVEN CLASS BATCH : -**



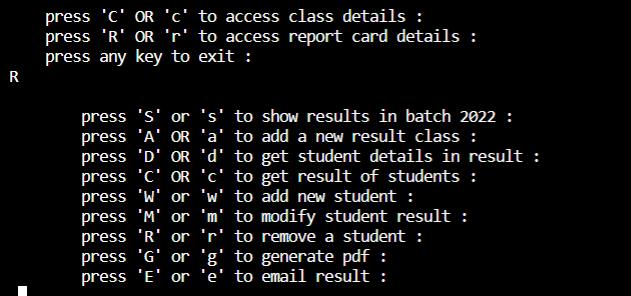
* **TO MODIFY STUDENT DETAILS IN GIVEN CLASS BATCH : =**

****

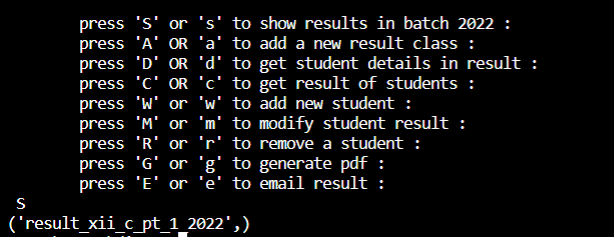
* **TO REMOVE STUDENT IN GIVEN CLASS BATCH : -**



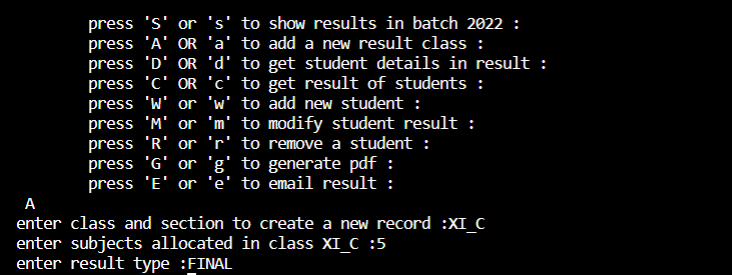
**A ) TO ACCESS RESULT : -**



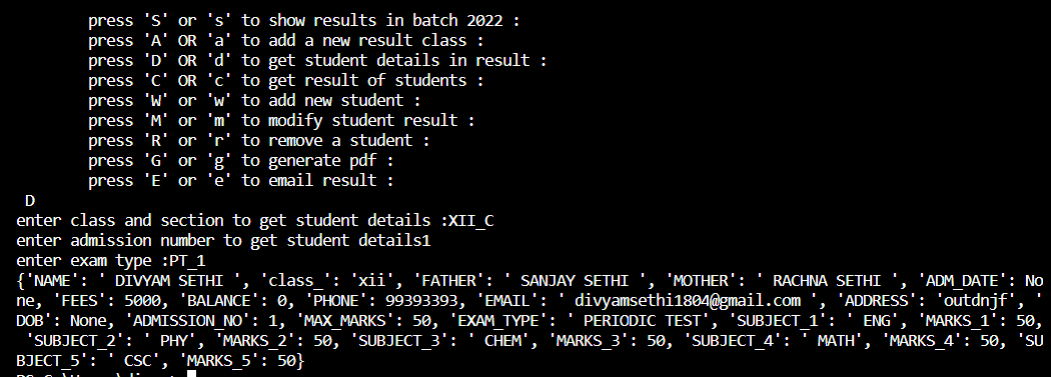
* **TO GET RESULTS IN GIVEN BATCH : -**



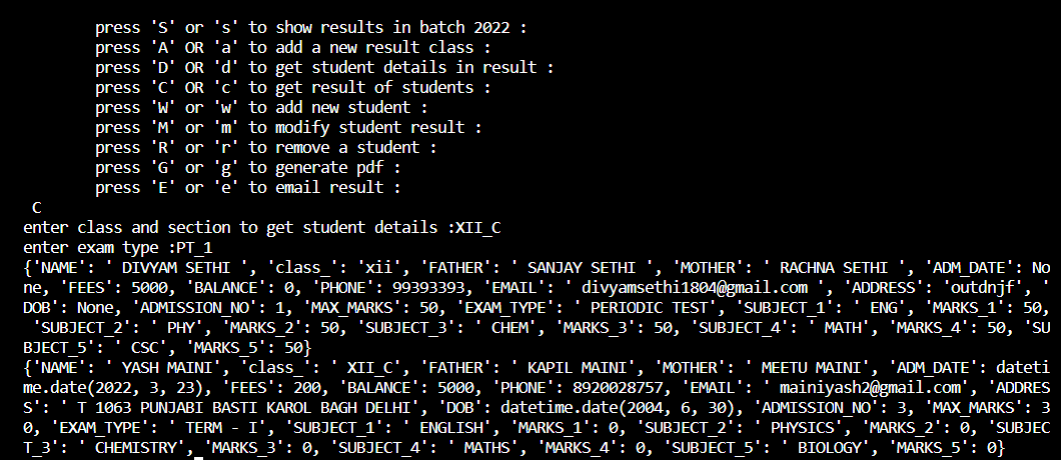
* **TO ADD NEW RESULT RECORD IN GIVEN BATCH : -**

****

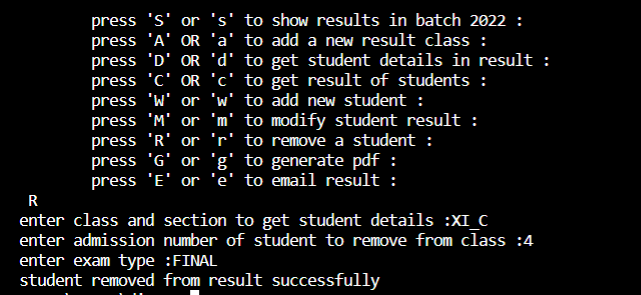
* **TO GET STUDENT RESULT IN GIVEN BATCH : -**



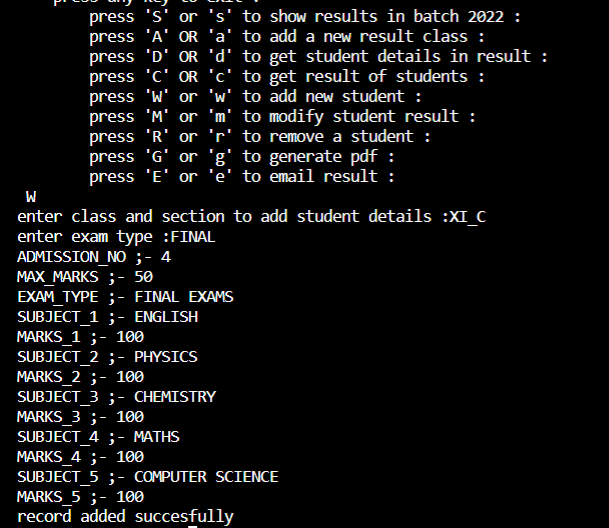
* **TO GET STUDENT RESULT IN GIVEN BATCH : -**



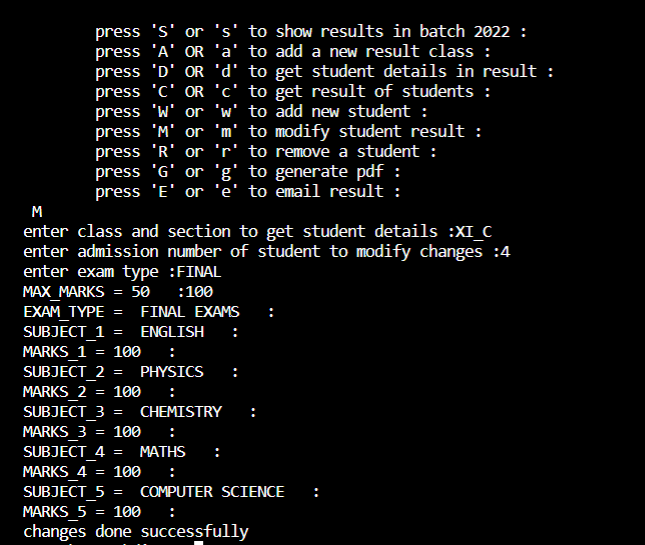
* **TO ADD STUDENT RESULT IN GIVEN RESULT BATCH : -**

****

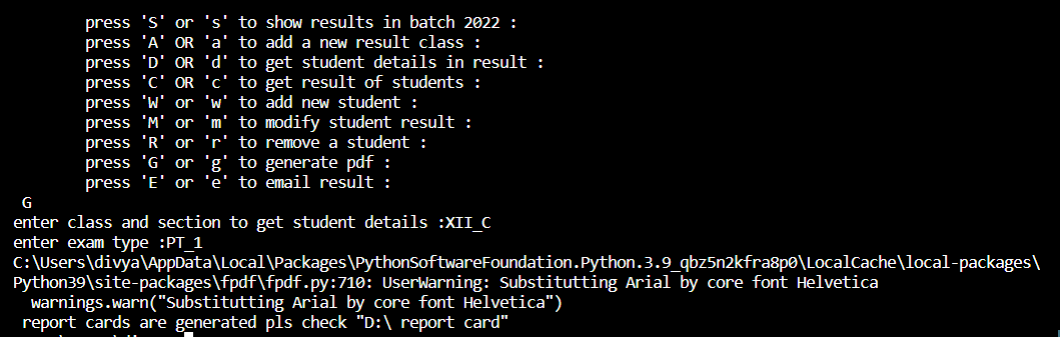
* **TO ADD STUDENT RESULT IN GIVEN RESULT BATCH : -**

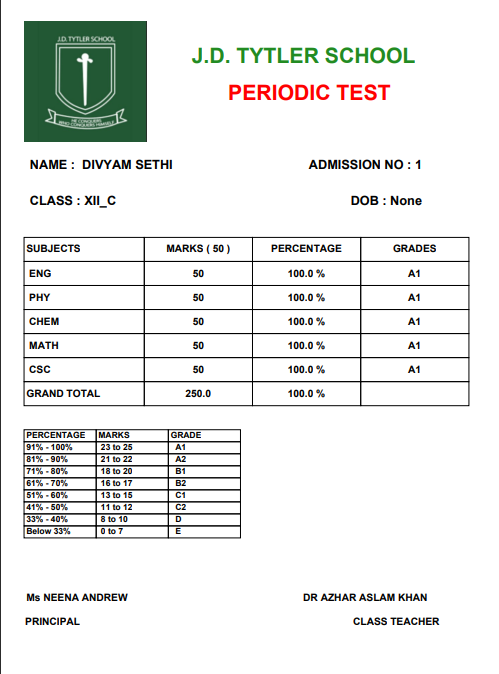


* **TO EDIT STUDENT RESULT IN GIVEN RESULT BATCH : -**

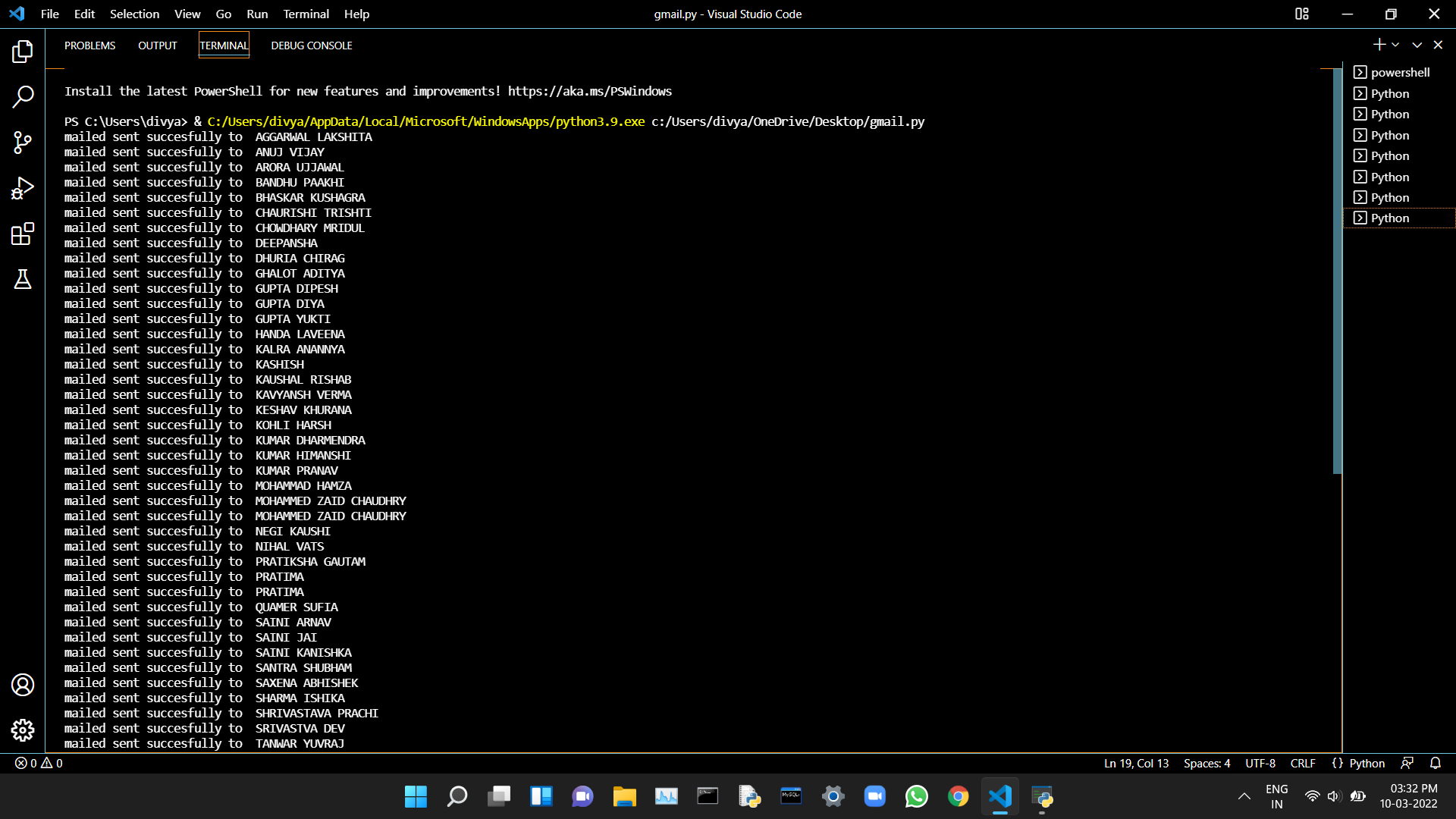


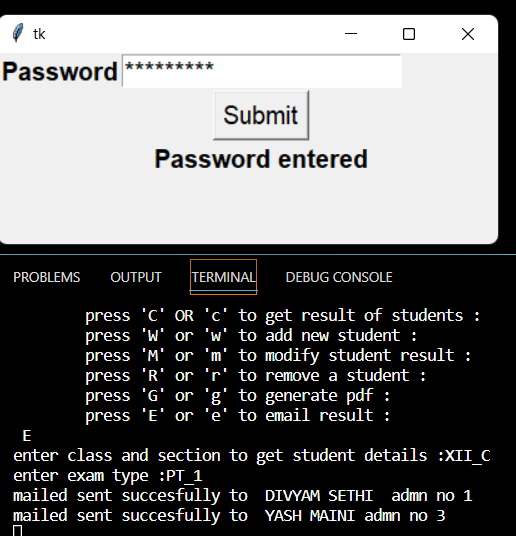
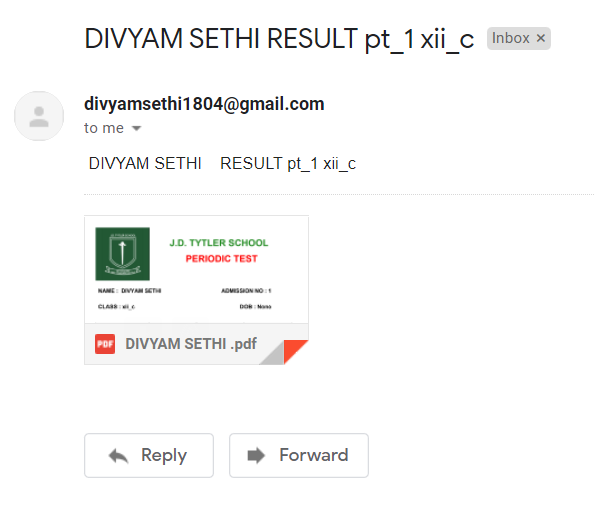
* **TO GENERATE REPORT CARD IN GIVEN RESULT BATCH : -**





* **TO EMAIL REPORT CARD IN GIVEN RESULT BATCH : -**



**HARDWARE AND SOFTWARE REQUIREMENTS**

I.OPERATING SYSTEM : WINDOWS 7 AND ABOVE

II. PROCESSOR : PENTIUM(ANY) OR AMD

ATHALON(3800+- 4200+ DUAL CORE)

III. MOTHERBOARD : 1.845 OR 915,995 FOR PENTIUM 0R MSI

K9MM-V VIA K8M800+8237R PLUS CHIPSET FOR AMD ATHALON

IV. RAM : 512MB+

V. Hard disk : SATA 40 GB OR ABOVE

VI. CD/DVD r/w multi drive combo: (If back up required)

VII. FLOPPY DRIVE 1.44 MB : (If Backup required)

VIII. MONITOR 14.1 or 15 -17 inch

IX. Key board and mouse

X. Printer : (if print is required – [Hard copy])

**SOFTWARE REQUIREMENTS:**

1. Windows OS
2. Python with MySQLconnector module downloaded

**BIBLIOGRAPHY**

1. ***Website:*** [***https://www.geeksforgeeks.org***](https://www.geeksforgeeks.org)
2. ***Website:*** [**https://www.w3resource.com**](https://www.w3resource.com)
3. ***Website:*** [***https://stackoverflow.com***](https://stackoverflow.com)

***\*\*\****