PROJECT MARSHALAR

**By**:

**Team 4**

Y Jaikanth - 12A711

M Ravi Prasath - 12A718

Sanjiv Kannaa J - 12A724

Thushar Nandam -12A731

**Class**- XII A7

BONAFIDE CERTIFICATE

Certified that this project is a bonafide work

of Master/Miss

Roll No. of Class XII of **Maharishi Vidya Mandir Sr. Sec. School**, Chetpet, Chennai during the year 2020 – 2021.

Date: 04/01/2021 Teacher – in – charge

Submitted for AISSCE Practical Examination held in the **COMPUTER SCIENCE** Laboratory at **Maharishi Vidya Mandir, Sr. Sec. School**,Chetpet, Chennai.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DATE : |  |  |  | INTERNAL EXAMINER |
| PRINCIPAL |  |  |  | EXTERNAL EXAMINER |
| SEAL |  |  |  |  |

**ACKNOWLEDGEMENT**

I express my gratitude to our **Principal, Shri. G. HARIBABU** of our institution for his continuous support and encouragement.

I express my sincere thanks to my Computer Science Teachers,

**Smt. V. SUNDARI**

**Smt. G. ANITHA**

**Smt. H. NAZREEN BANU**

for helping me to complete this project successfully.

I would also like to thank our Laboratory Assistant **Smt. A. VIJAYA** for all the help extended to us.

**INDEX**

1. Problem Definition

2. Problem Analysis

3. Hardware and Software Requirements

4. Future Enhancement

5. Source Code

6. Output

7. Bibliography

**PROBLEM DEFINITION**:

An innovative idea to assist the teachers with organizing the multitude of projects submitted in hand as well as through the internet because of the novel corona virus. In this situation the challenges faced by the teachers have diversified, to cope up with the situation this project helps us to adapt with new “normal” situation. The project submission work and even more hectic work of tracking the submitted project has got more intense with this current scenario.

Project Marshalar proves to be an efficient solution for this problem faced by many teachers and professors. First of all, it is highly secure and reliable. Project Marshalar uses a Login mechanism which only allows the registered users to access the database, hence makes it difficult for intruders and solves the problem of security. It is very simple and user friendly yet highly efficient. Project Marshalar efficiently accepts data from the user and stores them in a pre-created database. Project Marshalar is created in such a manner that it reduces data redundancy and hence reduces the risk of duplication of data. All the data are presented in the form of table, in a very simple manner. Thus, Project Marshalar can be defined as the ideal solution for the problems faced by the teacher while recording project works. Project Marshalar is built using Python 3.8 and SQLite3. Project Marshalar’s interface is created using tkinter in python. While the data base is stored using SQLite3.

**PROBLEM ANALYSIS**:

Source Code Explanation:

* Initially the program creates a table, PythonMarshal
* The program then checks whether the user has logged out or not.
* If the user has not logged out, the user is directed to the main screen
* If the user has logged out or if the program is being run for the first time the program calls for the account login screen.
* The account login screen displays two options:
  + - * Login- The user can use this option to log in if the user has already registered.
      * Register- The user can use this option to register.
* The program asks the user to register by creating user name and password and uploading details about the classes the user handles.
* The program proceeds by storing the user’s credentials as a file, which is used to verify later. Then the user is directed to the Account login screen.
* The user now can login. Login screen gets the credentials and then verifies them with the stored credentials.
* If the login is successful the user is directed to the menu screen If the login is not successful the user is shown the exception (User not found or Incorrect password) and then is asked to login again.
* Menu screen displays five options namely UPLOAD PROJECT, VIEW TABLE, SEARCH TABLE, CLASS STATUS, LOGOUT.
* Choosing UPLOAD PROJECT directs the user to the upload screen where the user is asked to enter the details such as class, section, team no, topic, short note, names of team members, roll Nos of the team members, grade, link of the documentation, remarks, and status.
* By uploading, the above given details are inserted into the table PYTHON MARSHALAR. The user is then directed to the menu screen
* If any exception occurs, the user is notified and directed to the menu screen.
* Choosing VIEW TABLE, the user can view a table created by using tkinter, containing important attributes such as class, date of submission, team no, topic, description of the topic, names of team members and grade by extracting data from the database
* Closing VIEW TABLE directs the user to menu screen.
* Choosing SEARCH TABLE, the user is directed to the search table screen where the user is asked to input the value for team number which the user wants to search for.
* The user is directed to the result screen where all the details of the team are extracted from the database and displayed.
* Closing the result screen directs the user to menu screen.
* Exceptions if any are notified and then user is directed to the menu screen.
* Choosing CLASS STATUS directs the user to the status screen where the user is asked to input the value for the class to be checked.
* The user is shown the result in the form of table which contains the details of the total number of teams, No. SUBMITTED, ONTIME, LATE, NOT SUBMITTED.
* Exceptions if any are notified and then the user is asked to re-enter appropriate inputs.
* By closing the table the user is then directed to the menu screen.
* By choosing LOGOUT the user is logged out and is directed to the Account login screen.

Module by Module Explanation:

1. **Database Connection and Table creation Module**:

Initially a database is connected and created using database connectivity sqlite3. To avoid duplication of table, except statement is used which passes the program to the next line.

2.**Account Login Module:**

This module gives two options to the user:

* 1. **Register**: The user is asked to register in which the user must create a username and password along with classes which the user handles. The program retrieves this information and stores them as a file for future reference hence, successfully registering.
  2. **Login:** The user is asked to login with the created credentials. The given credentials are verified with the respective stored files. If the verification is successful, the user can enter the menu screen and toggle. If exceptions occur, the program notifies the exception and asks the user to login again.

3 **Menu Module:**

The user is provided with five features in the menu screen, namely UPLOAD PROJECT, VIEW TABLE, SEARCH TABLE, CLASS STATUS and LOGOUT using a user interface built with tkinter.

4.**Upload Project Module:**

This module enables the user to upload the details of a team’s project work along with grade and remarks. To avoid redundancy Team No is set as a primary key. If exception occur the program notifies the user and returns to the menu screen.

5.**View Table Module:**

Displays the important attributes of all the teams in the form of table designed using tkinter.

6. **Search Table Module:**

Displays all the contents of a specific team which is inputted by the user. Exceptions are notified and the user is led back the menu screen.

7.**Class Status Module:**

Displays the submission status of a specific class. Exceptions are notified, the user is asked to enter the correct inputs.

8.**Logout Module:**

The user can logout using this module.

MODULES USED FROM PYTHON LIBRARY

**tkinter:**

The [tkinter](https://docs.python.org/3/library/tkinter.html#module-tkinter) package (“Tk interface”) is the standard Python interface to the Tk GUI toolkit. Both Tk and [tkinter](https://docs.python.org/3/library/tkinter.html#module-tkinter) are available on most Unix platforms, as well as on Windows systems. In computing, an interface is a shared boundary across which two or more separate components of a computer system exchange information. The exchange can be between software, computer hardware, peripheral devices, humans, and combinations of these. This program’s user interface is built using tkinter. The program’s user interface is built in such a way to make the program more user friendly by using buttons.

**SQLite3**:

SQLite is a C library that provides a lightweight disk-based database that doesn’t require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It’s also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.The sqlite3 module was written by Gerhard Häring. It provides a SQL interface compliant with the DB-API 2.0 specification described by [PEP 249](https://www.python.org/dev/peps/pep-0249). To use the module, you must first create a [Connection](https://docs.python.org/3/library/sqlite3.html#sqlite3.Connection) object that represents the database Once you have a [Connection](https://docs.python.org/3/library/sqlite3.html#sqlite3.Connection), you can create a [Cursor](https://docs.python.org/3/library/sqlite3.html#sqlite3.Cursor) object and call its [execute()](https://docs.python.org/3/library/sqlite3.html#sqlite3.Cursor.execute) method to perform SQL commands. SQLite3 is used to create database and to perform operations in the database like Select etc.

# **OS - Miscellaneous operating system interfaces:**

This module provides a portable way of using operating system dependent functionality. Like read or write a file ([open()](https://docs.python.org/3/library/functions.html#open)), manipulate paths, ([os.path](https://docs.python.org/3/library/os.path.html#module-os.path) module), read all the lines in all the files on the command line ( [fileinput](https://docs.python.org/3/library/fileinput.html#module-fileinput) ) module and for creating temporary files and directories. The program uses this module to create files to store registration details and to verify them.

**SYS MODULE:**

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. This program imports a user defined module created using sys.pycache\_prefix

sys.pycache\_prefix:,

If this is set (not None), Python will write bytecode-cache .pyc files to (and read them from) a parallel directory tree rooted at this directory, rather than from pycache\_\_ directories in the source code tree. Any \_\_pycache\_\_ directories in the source code tree will be ignored and new *.pyc* files written within the pycache prefix. Thus, if you use [compileall](https://docs.python.org/3/library/compileall.html#module-compileall) as a pre-build step, one must ensure one runs it with the same pycache prefix (if any) that one will use at runtime. A relative path is interpreted relative to the current working directory.

**Hardware and Software Requirements:**

OS - Microsoft Windows 8 & above

System Type - x86-64bit based processor

Processor - Intel(R) core(TM) i3-8250

CPU - 2.4GHz & above

Memory (RAM) - 3.0 GB & above

Front End - Python 3.8.0 & above

Back End - SQLite3

Future Enhancement:

* Project Marshalar has very good scope in the future as the educational system is taking up the online platform. Project Marshalar can used as a very effective platform for storing student’s project related data and documentation in institutional level and with more enhancements, modifications and updates Project Marshal can be use in national level .
* The present scope of Project Marshalar enables the user to store the data in a local database with required modification Project Marshalar can store the data in web based database. This can enables the user to access the data from anywhere.
* Project Marshalar can be created as a website or a web based application for multiple user access and easy accessibility of the data.
* A built- in documentation up loader and viewer can be created to view documentation instead of uploading and viewing documentation through links.

SOURCE CODE:

# T**his code is to be created as module and then imported in the main program.**

#import modules

import time

from tkinter import \*

import os

# Designing window for registration

Name = ''

def register():

global register\_screen

register\_screen = Toplevel(main\_screen)

register\_screen.title("Register")

register\_screen.geometry("700x700")

global Name

global username

global password

global Class1

global Class2

global Class3

global Class4

global Class5

global Name\_entry

global username\_entry

global password\_entry

global Class1\_entry

global Class2\_entry

global Class3\_entry

global Class4\_entry

global Class5\_entry

Name= StringVar()

username = StringVar()

password = StringVar()

Class1= StringVar()

Class2= StringVar()

Class3= StringVar()

Class4= StringVar()

Class5= StringVar()

Label(register\_screen, text="Please enter details below").pack()

Label(register\_screen, text="").pack()

Name\_lable = Label(register\_screen, text="Name")

Name\_lable.pack()

Name\_entry = Entry(register\_screen, textvariable=Name)

Name\_entry.pack()

username\_lable = Label(register\_screen, text="Username \* ")

username\_lable.pack()

username\_entry = Entry(register\_screen, textvariable=username)

username\_entry.pack()

password\_lable = Label(register\_screen, text="Password \* ")

password\_lable.pack()

password\_entry = Entry(register\_screen, textvariable=password, show='\*')

password\_entry.pack()

Class1\_label=Label(register\_screen, text="Class1 your Handling \*")

Class1\_label.pack()

Class1\_entry=Entry(register\_screen, textvariable=Class1)

Class1\_entry.pack()

Class2\_label=Label(register\_screen, text="Class2 your Handling 'if not type none'")

Class2\_label.pack()

Class2\_entry=Entry(register\_screen, textvariable=Class2)

Class2\_entry.pack()

Class3\_label=Label(register\_screen, text="Class3 your Handling 'if not type none'")

Class3\_label.pack()

Class3\_entry=Entry(register\_screen, textvariable=Class3)

Class3\_entry.pack()

Class4\_label=Label(register\_screen, text="Class4 your Handling 'if not type none'")

Class4\_label.pack()

Class4\_entry=Entry(register\_screen, textvariable=Class4)

Class4\_entry.pack()

Class5\_label=Label(register\_screen, text="Class5 your Handling 'if not type none'")

Class5\_label.pack()

Class5\_entry=Entry(register\_screen, textvariable=Class5)

Class5\_entry.pack()

Label(register\_screen, text="").pack()

Button(register\_screen, text="Register", width=10, height=1, command = register\_user).pack()

# Designing window for login

def login():

global login\_screen

login\_screen = Toplevel(main\_screen)

login\_screen.title("Login")

login\_screen.geometry("300x250")

Label(login\_screen, text="Please enter details below to login").pack()

Label(login\_screen, text="").pack()

global username\_verify

global password\_verify

username\_verify = StringVar()

password\_verify = StringVar()

global username\_login\_entry

global password\_login\_entry

Label(login\_screen, text="Username \* ").pack()

username\_login\_entry = Entry(login\_screen, textvariable=username\_verify)

username\_login\_entry.pack()

Label(login\_screen, text="").pack()

Label(login\_screen, text="Password \* ").pack()

password\_login\_entry = Entry(login\_screen, textvariable=password\_verify, show= '\*')

password\_login\_entry.pack()

Label(login\_screen, text="").pack()

Button(login\_screen, text="Login", width=10, height=1, command = login\_verify).pack()

#Label(login\_screen, text="").pack()

#Button(login\_screen, text="Close", height=2, width=30, command=delete\_login\_screen).pack()

# Implementing event on register button

def register\_user():

Name\_info=Name.get()

username\_info = username.get()

password\_info = password.get()

Class1\_info= Class1.get()

Class2\_info= Class2.get()

Class3\_info= Class3.get()

Class4\_info= Class4.get()

Class5\_info= Class5.get()

file = open(username\_info, "w")

file.write(Name\_info + "\n")

file.write(username\_info + "\n")

file.write(password\_info+"\n")

file.write(Class1\_info+"\n")

file.write(Class2\_info+"\n")

file.write(Class3\_info+"\n")

file.write(Class4\_info+"\n")

file.write(Class5\_info)

file.close()

Name\_entry.delete(0,END)

username\_entry.delete(0, END)

password\_entry.delete(0, END)

register\_screen.destroy()

# Implementing event on login button

def login\_verify():

username1 = username\_verify.get()

password1 = password\_verify.get()

username\_login\_entry.delete(0, END)

password\_login\_entry.delete(0, END)

list\_of\_files = os.listdir()

if username1 in list\_of\_files:

file1 = open(username1, "r")

verify = file1.read().splitlines()

if password1 in verify:

l = open('Name.txt', 'w')

l.write(verify[0])

login\_sucess()

else:

password\_not\_recognised()

else:

user\_not\_found()

# Designing popup for login success

def login\_sucess():

''' global login\_success\_screen

login\_success\_screen = Toplevel(login\_screen)

login\_success\_screen.title("Login Success")

login\_success\_screen.geometry("150x100")

Label(login\_success\_screen, text="Welcome").pack()

Button(login\_success\_screen, text="OK", command=delete\_login\_success).pack()

'''

global Name

Name=str(Name)

Name=Name[:-4]

login\_screen.destroy()

main\_screen.destroy()

f=open('loginTrue.txt', 'w')

f.write('True')

rav = open('Name.txt','w')

rav.write(Name)

# Designing popup for login invalid password

def password\_not\_recognised():

global password\_not\_recog\_screen

password\_not\_recog\_screen = Toplevel(login\_screen)

password\_not\_recog\_screen.title("Success")

password\_not\_recog\_screen.geometry("150x100")

Label(password\_not\_recog\_screen, text="Invalid Password ").pack()

Button(password\_not\_recog\_screen, text="OK", command=delete\_password\_not\_recognised).pack()

# Designing popup for user not found

def user\_not\_found():

global user\_not\_found\_screen

user\_not\_found\_screen = Toplevel(login\_screen)

user\_not\_found\_screen.title("Success")

user\_not\_found\_screen.geometry("150x100")

Label(user\_not\_found\_screen, text="User Not Found").pack()

Button(user\_not\_found\_screen, text="OK", command=delete\_user\_not\_found\_screen).pack()

# Deleting popups

def delete\_login\_success():

login\_success\_screen.destroy()

def delete\_password\_not\_recognised():

password\_not\_recog\_screen.destroy()

def delete\_user\_not\_found\_screen():

user\_not\_found\_screen.destroy()

def delete\_main\_account\_screen():

main\_screen.destroy()

def delete\_login\_screen():

login\_screen.destroy()

# Designing Main(first) window

def main\_account\_screen():

global main\_screen

main\_screen = Tk()

main\_screen.geometry("300x250")

main\_screen.title("Account Login")

Label(main\_screen,text="Select Your Choice", width="300", height="2", font=("Calibri", 13)).pack()

Label(main\_screen,text="").pack()

Button(main\_screen,text="Login", height="2", width="30", command = login).pack()

Label(main\_screen,text="").pack()

Button(main\_screen,text="Register", height="2", width="30", command=register).pack()

#Label(text="").pack()

#Button(text="Close", height="2", width="30", command=delete\_main\_account\_screen).pack()

main\_screen.mainloop()

# **Main Program**

from tkinter import \*

import sqlite3

import os

import time

import sys

import login

connection7=sqlite3.connect('PythonMarshal.db')

c7=connection7.cursor()

try:

c7.execute('create table PythonMarshal(teacher varchar(10), DOS varchar(20), class varchar(10), team\_no varchar(10) primary key, topic varchar(100), descrip varchar(200), team\_members1 varchar(10), team\_no1 int(4), team\_members2 varchar(10), team\_no2 int(2), team\_members3 varchar(10), team\_no3 int(2), team\_members4 varchar(10), team\_no4 int(2), grade float(10), documentation varchar(30), highlights varchar(20))')

except:

pass

Name = ''

def menu():

global menu\_screen

global upload

global view\_table

global search

global Name

name1= " Logined as : " + Name

menu\_screen=Tk()

menu\_screen.title("PROJECT MARSHALAR")

menu\_screen.geometry('640x400')

menu\_screen.geometry('+320+90')

L=Label(menu\_screen,text="Menu")

L.config(font=("Arial",20))

L.pack()

L1=Label(menu\_screen,text=name1)

L1.pack()

Label(menu\_screen,text="").pack()

Label(text="").pack()

Button(menu\_screen,text="UPLOAD PROJECT",width=15,height=1,command=upload).pack()

Label(text="").pack()

Button (menu\_screen,text="VIEW TABLE",width=15,height=1,command=dashboard).pack()

Label(text="").pack()

Button(menu\_screen,text="SEARCH TABLE",width=15,height=1,command=search\_table).pack()

Label(text="").pack()

Button(menu\_screen,text="CLASS STATUS",width=15,height=1,command=class\_status).pack()

Label(text="").pack()

Button(menu\_screen,text="LOGOUT",width=15,height=1,command=Log\_out).pack()

'''def update\_project():

global Name

update\_project1(Name)

'''

'''def update\_project1(name):

print('\n'\*30)

data = []

data.append(name)

data.append(time.ctime(time.time())[-4:]) #year

data.append(str(input('Enter class and section : '))) #class

data.append(str(input('Enter team number : '))) #team number

data.append(str(input('Enter topic : '))) #topic

data.append(str(input('Enter topic short note : '))) #short note on topic

for i in range(4):

data.append(str(input('Enter name of team member : '))) #name of member

data.append(str(input('Roll number : '))) #roll number of member

data.append(str(input('Enter grade : ')))

data.append(str(input("Enter documentation link: ")))

#data.append(str(data[1] + '\_' + data[2] + '\_' + data[3] + '\_' + '\_documentation.txt'))# refer this for the type of saving of the documentation.txt file

data.append(str(input('Enter remarks : '))) #highlights

sas = ''

for i in data:

sas = sas + '\"' + i + '\"' + ','

sas = sas[:-1]

query = 'INSERT INTO PythonMarshal VALUES(' + sas + ')'

c7.execute(query)

connection7.commit()

print('\n\nDataUpdated')

time.sleep(4)

rep = input('another Team? (y/n): ')

if rep in ['y', 'Y', 'Yes', 'yes']:

update\_project(mame)

else:

menu()

'''

def upload():

menu\_screen.destroy()

global upload\_screen

upload\_screen=Tk()

upload\_screen.title("UPLOAD")

upload\_screen.geometry("1600x900")

global Class\_section

global Team\_no

global Topic

global Short\_Note

global Name\_Member1

global Name\_Member2

global Name\_Member3

global Name\_Member4

global Name\_Member1

global Roll\_no\_Member1

global Roll\_no\_Member2

global Roll\_no\_Member3

global Roll\_no\_Member4

global Grade

global Doc

global Remarks

#global DOS

Class\_section= StringVar()

Team\_no = StringVar()

Short\_Note = StringVar()

Name\_Member1= StringVar()

Name\_Member2= StringVar()

Name\_Member3= StringVar()

Name\_Member4= StringVar()

Roll\_no\_Member1= StringVar()

Roll\_no\_Member2= StringVar()

Roll\_no\_Member3= StringVar()

Roll\_no\_Member4= StringVar()

Grade= StringVar()

Doc= StringVar()

Remarks= StringVar()

Label(upload\_screen, text="Please enter details below").pack()

Label(upload\_screen, text="").pack()

Class\_section = Label(upload\_screen, text="Class and section (format:12a7)")

Class\_section.pack()

Class\_section = Entry(upload\_screen,textvariable=Class\_section)

Class\_section.pack()

Team\_no = Label(upload\_screen, text="Team\_no (format:<yearclassteamno>)")

Team\_no.pack()

Team\_no = Entry(upload\_screen,textvariable=Team\_no)

Team\_no.pack()

Topic= Label(upload\_screen, text="Topic")

Topic.pack()

Topic= Entry(upload\_screen, textvariable=Topic)

Topic.pack()

Short\_Note = Label(upload\_screen, text="Short note on topic ")

Short\_Note.pack()

Short\_Note = Entry(upload\_screen, textvariable=Short\_Note)

Short\_Note.pack()

Name\_Member1=Label(upload\_screen, text="Name of Member1")

Name\_Member1.pack()

Name\_Member1=Entry(upload\_screen, textvariable=Name\_Member1)

Name\_Member1.pack()

Roll\_no\_Member1=Label(upload\_screen, text="Roll\_no\_Member1")

Roll\_no\_Member1.pack()

Roll\_no\_Member1=Entry(upload\_screen, textvariable=Roll\_no\_Member1)

Roll\_no\_Member1.pack()

Name\_Member2=Label(upload\_screen, text="Name of Member2")

Name\_Member2.pack()

Name\_Member2=Entry(upload\_screen, textvariable=Name\_Member2)

Name\_Member2.pack()

Roll\_no\_Member2=Label(upload\_screen, text="Roll no of Member2")

Roll\_no\_Member2.pack()

Roll\_no\_Member2=Entry(upload\_screen, textvariable=Roll\_no\_Member2)

Roll\_no\_Member2.pack()

Name\_Member3=Label(upload\_screen, text="Name of Member3")

Name\_Member3.pack()

Name\_Member3=Entry(upload\_screen, textvariable=Name\_Member3)

Name\_Member3.pack()

Roll\_no\_Member3=Label(upload\_screen, text="Roll no of Member3")

Roll\_no\_Member3.pack()

Roll\_no\_Member3=Entry(upload\_screen, textvariable=Roll\_no\_Member3)

Roll\_no\_Member3.pack()

Name\_Member4=Label(upload\_screen, text="Name of Member4")

Name\_Member4.pack()

Name\_Member4=Entry(upload\_screen, textvariable=Name\_Member4)

Name\_Member4.pack()

Roll\_no\_Member4=Label(upload\_screen, text="Roll no of Member4")

Roll\_no\_Member4.pack()

Roll\_no\_Member4=Entry(upload\_screen, textvariable=Roll\_no\_Member4)

Roll\_no\_Member4.pack()

Grade= Label(upload\_screen, text="Grade")

Grade.pack()

Grade= Entry(upload\_screen, textvariable=Grade)

Grade.pack()

Doc= Label(upload\_screen, text="Doc")

Doc.pack()

Doc= Entry(upload\_screen, textvariable=Doc)

Doc.pack()

Remarks= Label(upload\_screen, text="Remarks")

Remarks.pack()

Remarks= Entry(upload\_screen, textvariable=Remarks)

Remarks.pack()

Label(upload\_screen, text="").pack()

Button(upload\_screen, text="UPLOAD", width=10, height=1, command = up).pack()

def up():

Date=str(time.ctime(time.time()))

month = Date[4:7]

if 'Jan' in month:

month = 1

if 'Fe' in month:

month = 2

if 'Mar' in month:

month = 3

if 'Ap' in month:

month = 4

if 'May' in month:

month = 5

if 'Jun' in month:

month = 6

if 'Jul' in month:

month = 7

if 'Au' in month:

month = 8

if 'S' in month:

month = 9

if 'Oc' in month:

month = 10

if 'N' in month:

month = 11

if 'Dec' in month:

month = 12

Date = Date[8:10] + ':' + str(month) + ':' + Date[-4:]

Classec=Class\_section.get()

Teamno = Team\_no.get()

Topicinfo = Topic.get()

Shortnote= Short\_Note.get()

NameMember1= Name\_Member1.get()

RollnoMember1= Roll\_no\_Member1.get()

NameMember2= Name\_Member2.get()

RollnoMember2= Roll\_no\_Member2.get()

NameMember3= Name\_Member3.get()

RollnoMember3= Roll\_no\_Member3.get()

NameMember4= Name\_Member4.get()

RollnoMember4= Roll\_no\_Member4.get()

Gradeinfo= Grade.get()

Docinfo= Doc.get()

Remarksinfo= Remarks.get()

uplist=[Name,Date,Classec,Teamno,Topicinfo,Shortnote,NameMember1,RollnoMember1, NameMember2,RollnoMember2,NameMember3,RollnoMember3,NameMember4,RollnoMember4,Gradeinfo,Docinfo,Remarksinfo]

#print(uplist)

qery=""

for i in uplist:

qery = qery + '\"' + i + '\"' + ','

qery = qery[:-1]

query = 'INSERT INTO PythonMarshal VALUES(' + qery + ')'

c7.execute(query)

connection7.commit()

upload\_screen.destroy()

menu()

def dashboard():

# Python program to create a table

class Table:

def \_\_init\_\_(self,root):

# code for creating table

for i in range(total\_rows):

for j in range(total\_columns):

self.e = Entry(root, width=13, fg='blue',

font=('Arial',16,'bold'))

self.e.grid(row=i, column=j)

self.e.insert(END, lst[i][j])

# take the data

c7.execute('select class,DOS,team\_no,topic,descrip,team\_members1,team\_members2,team\_members3,team\_members4,grade,documentation from PythonMarshal') #class, team\_no, topic, short note, namex4, grade

data = c7.fetchall()

lst = [("Class","DOS","Team No","Topic","description","member1","member2","member3","member4","Grade","Documentation") ]

lst=lst+data

# find total number of rows and

# columns in list

total\_rows = len(lst)

total\_columns = len(lst[0])

# create root window

root = Tk()

t = Table(root)

root.mainloop()

menu()

def search\_table():

menu\_screen.destroy()

global search\_screen

search\_screen=Tk()

search\_screen.title("SEARCH TABLE")

global Team\_no

Team\_no= StringVar()

Team\_no = Label(search\_screen, text="Team\_no (format(eg):<202012A704>)")

Team\_no.pack()

Team\_no = Entry(search\_screen,textvariable=Team\_no)

Team\_no.pack()

Label(text="").pack()

Button (search\_screen,text="SEARCH TABLE", width=15,height=1,command=search).pack()

def search():

search\_screen.destroy

Teamno = Team\_no.get()

in\_put="SELECT \* FROM PythonMarshal "

c7.execute(in\_put)

out\_put=c7.fetchall()

data1=''

for i in out\_put:

if i[3]== Teamno:

data1=i

print(data1)

global out\_screen

out\_screen=Tk()

out\_screen.title("Result")

for i in range(0,len(data1)):

if i==0:

Label(out\_screen,text="").pack()

Label(out\_screen,text="").pack()

elif i==1:

Label(out\_screen,text="").pack()

Label(out\_screen,text="DOS: "+data1[i]).pack()

else:

Label(out\_screem,text="No suck team exist").pack()

def class\_status():

print('Under Development')

time.sleep(2)

menu()

screen = Tk()

def Log\_out():

time.sleep(1)

rav = open('Name.txt', 'w')

rav.write('')

loginTrue = open('loginTrue.txt', 'w')

loginTrue.write('')

menu\_screen.destroy()

ini()

def ini():

global Name

loginTrue = open('loginTrue.txt', 'r')

if loginTrue.read() == 'True':

jai = open('Name.txt', 'r')

Name = jai.read()

menu()

else:

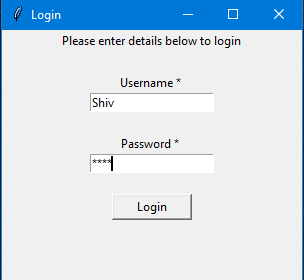
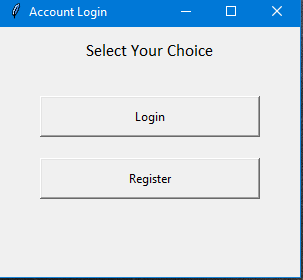
print('Please Login')

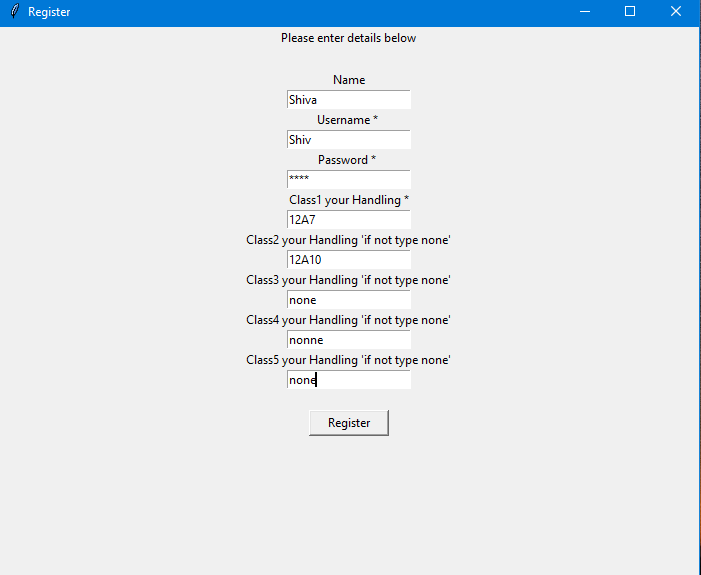
login.main\_account\_screen()

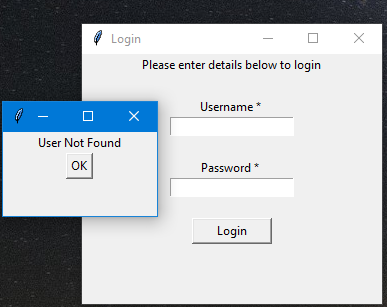
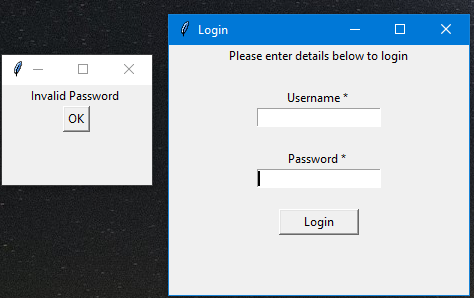
ini()

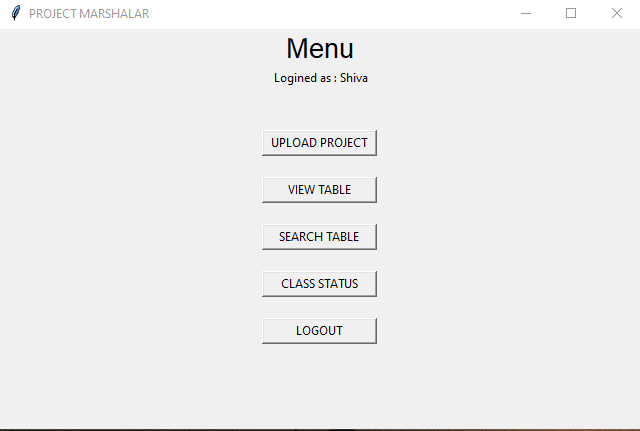
ini()

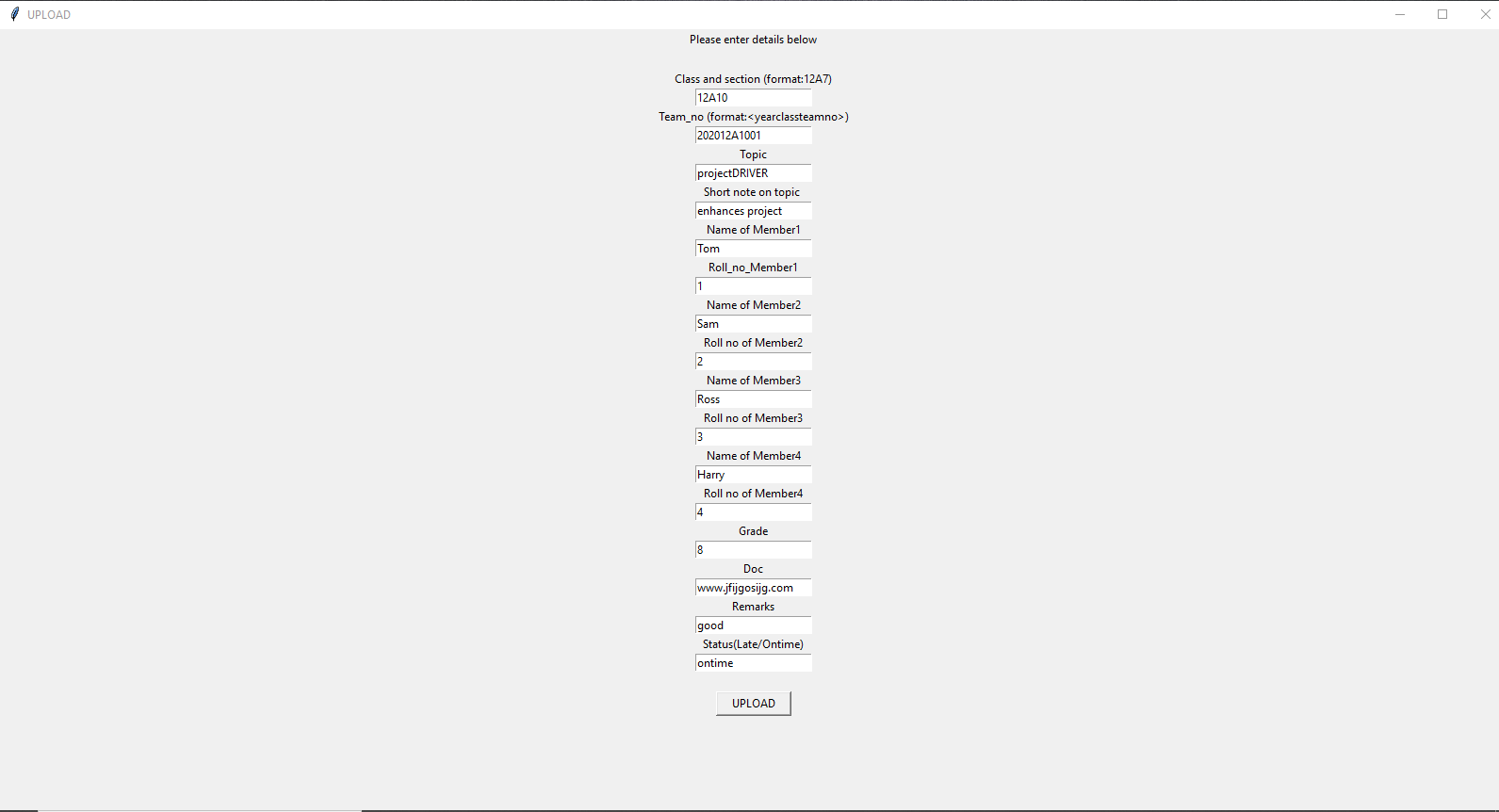
OUTPUT:

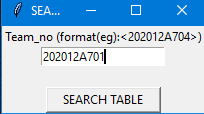
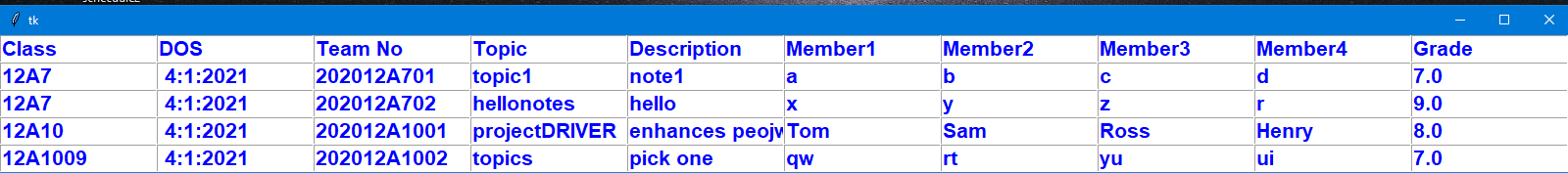
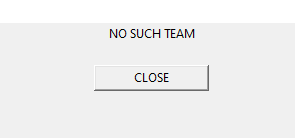
 

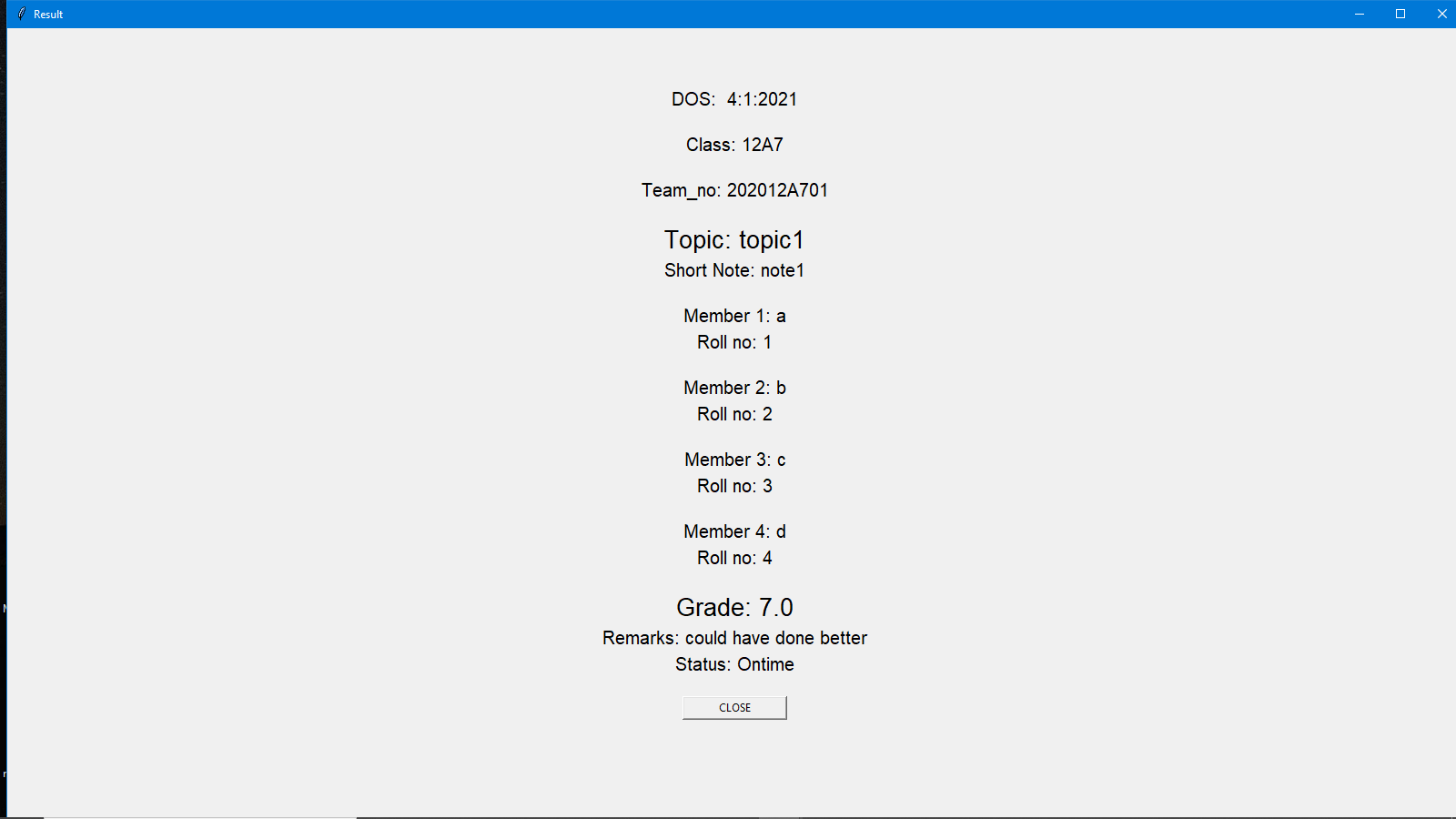


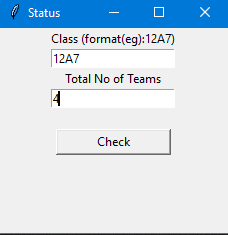
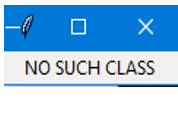
 









BIBLIOGRAPHY:

* The complete reference by Martin C Brown from Mr Grawhill publications.
* Computer science with Sumita Arora from Dhanpatrai publications.
* [www.pythonforgeeks.org](http://www.pythonforgeeks.org)
* [www.python.org](http://www.python.org)
* [www.Coursera.com](http://www.Coursera.com)