

PERSONAL PLANNER



A project by

VICTOR BIJU

CLASS 12 C

Roll No:10

St. Columba's School

New Delhi

Board Roll No:_____

CERTIFICATE

*THIS IS TO CERTIFY THAT VICTOR BIJU OF
CLASS 12 C HAS WORKED ON THE PROJECT
PERSONAL PLANNER UNDER MY SUPERVISION AND
HAS COMPLETED IT TO MY SATISFACTION.*

*I FURTHER CERTIFY THAT THIS PROJECT IS UP TO
MY EXPECTATIONS AND AS PER THE GUIDLINES
ISSUED BY CBSE*

*R. Nagpal
(Teacher)*

Acknowledgment

I am sincerely grateful to Br. E.V. Miranda, the Principal of St. Columba's School and Mr. Naresh Chopra, Administrator, Senior School for the whole hearted support and encouragements.

It is my privilege to express my sincere regards to my Computer Teacher Mrs. R. Nagpal for her valuable guidance and cooperation which helped me to learn Python and to carry out the project in a systematic manner. I am grateful to my teacher for her help.

I am also thankful to my parents, brother and my friends who have guided and helped me in carrying out this project.

Index

1. Aim of the Project	5
2. Introduction to Personal Planner	6
3. Graphical User Interface (Tkinter)	7
4. File handling	11
5. Account creation and Login	12
6. Database Connectivity	13
7. Modules	17
8. Outputs	17
9. One day and Multi-Day Planning	18
10. Add reminders	21
11. View and Search Plans	22
12. Delete Projects/reminders	24
13. Program (Complete Code)	27
14. Testing of Project	54
15. Conclusion and Future Enhancements	56
16. Bibliography	57

Aim of the Project

Planning is the process of thinking about the activities required to achieve a desired goal. It is the first and foremost activity to acquire desired results. It involves the creation and maintenance of a plan, such as psychological aspects that require conceptual skills. Planner increases productivity by organizing our life to focus on our goals.

Inspiration for the project came from my strong urge to learn Python. Moreover, I always like to lead an organized life with proper plan. This project is my humble attempt to implement these aspirations in an automated manner with planning and organizing my time and my day to day life.

*This project uses all basic features and functions of Python. Further, **file handling** and **database connectivity** with various database manipulations are demonstrated here. Additionally, the project uses **Tkinter module**, which is **Graphical User Interface (GUI)** package of python. Thus, Personal Planner has become more user friendly.*

INTRODUCTION TO PERSONAL PLANNER

Daily habit of written planning has seen from early 16th century. Since then, planners have seen a huge transformation from daily diary to today's advanced artistic design layouts. Gustav Grossmann, a great industrialist and Economist, published a leather-bound version of this predecessor to our modern organizers and time management systems in the 1930s and wrote a 200-page user's guide to go along with it, detailing how buyers were supposed to use the binder book in the most efficient way. Grossmann conceived the day planner as an instrument that was supposed to support its user on his path to growth – that is, so long as it was used in accordance with the user's guide. It did this by helping him lead his life in a "rational" way. And part of leading one's life in a rational way was the "right" kind of emotional self-conduct. Users were supposed to write down their "goals," "wishes," and daily "tasks" in the book at set times throughout the day. They had to distinguish between "life-long goals," "wishes" for certain phases of life, "plans" for the year, and "tasks" for the month, the week, and the day. The "goals," "plans," "wishes," and "tasks" were divided up into their own different sections: "life-plans," "plans for a phase of life," "plans for the year," "plans for the month," "plans for the week," "plans for the day.

Planner increases productivity by organizing our life to focus on our goals. This project helps to organize life by planning day to day activities and manage our time efficiently. Mainly it has two planning modes namely; i) single day tasks and ii) Multi-day tasks.

GRAPHICAL USER INTERFACE (TKINTER)

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Importing tkinter is same as importing any other module in the Python code. To create a tkinter app following steps are involved:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets

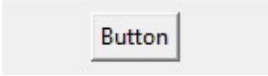


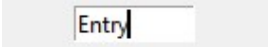
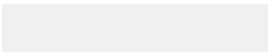
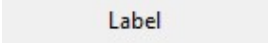
Code from project


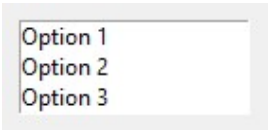

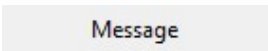
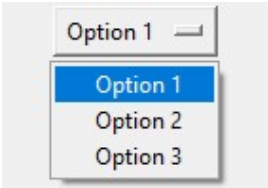

```
from tkinter import *  
from tkinter import ttk  
from tkcalendar import *  
root=Tk()  
root.title('Personal Planner')  
root.mainloop() # mainloop
```

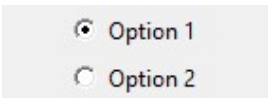


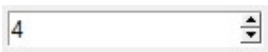
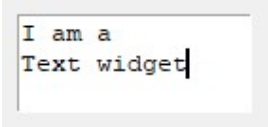
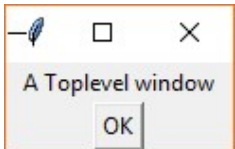
There is a method known by the name `mainloop()` is used when your application is ready to run. `Mainloop()` is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

Widgets

A widget is a controllable element within the GUI, and all Tkinter widgets have a shared set of methods. The following table shows all the core widgets with an example screenshot. Some of these widgets have been used in my project.

Widget	Description	Example
<u>Button</u>	Clickable area with text that calls an associated function whenever the user clicks in the area.	
<u>Canvas</u>	General purpose widget that can display simple graphics or provide an area to implement a custom widget.	
<u>Checkbutton</u>	Allows a user to read and select between two distinct values (e.g. on/off). Also known as a "checkboxbox."	
<u>Entry</u>	Area that allows the user to enter one line of text. For entering multiple lines, use the <i>Text</i> widget.	
<u>Frame</u>	A container for other widgets. A Frame can be useful for grouping other widgets together in a complex layout.	
<u>Label</u>	Used to display text or an image that may not be edited by the user.	

<u>LabelFrame</u>	A rectangular area that contains other widgets, but unlike a <i>Frame</i> , a border and label may be added to help group widgets together.	
<u>Listbox</u>	Displays a list of text alternatives. The user can choose (highlight) one or more options.	
<u>Menu</u>	Used to create menus and submenus within an interface. Can be used to create the always-shown "menu bar" popular in many GUI applications.	
<u>Menubutton</u>	Obsolete as of Tk 8.0. Use <i>Menu</i> widget instead.	See <i>Menu</i>
<u>Message</u>	Used to display static text, like <i>Label</i> , but allows for multiple lines, text wrapping, and maintaining aspect ratios.	
<u>OptionMenu</u>	Drop-down (or pop-up) menu that allows users to select one option from several options.	
<u>PanedWindow</u>	Container for one or more widgets split into multiple "panes." These panes can be resized by the user by dragging the separator line(s) (known as "sashes").	

<u>Radiobutton</u>	Several Radiobuttons can be used together to allow the user to select one option out of a group of options.	
<u>Scale</u>	User can select a numerical value by moving a slider.	
<u>Scrollbar</u>	Paired with a <i>Canvas</i> , <i>Entry</i> , <i>Listbox</i> , or <i>Text</i> widget to allow for scrolling within that widget.	
<u>Spinbox</u>	Allows the user to select only one option out of a list of options.	
<u>Text</u>	Area used to display and edit multiple lines of text. Can be used as a full text editor by the user.	
<u>Toplevel</u>	A container for other widgets (much like the <i>Frame</i> widget) that appears in its own window. It can be useful for creating other application windows or pop-up notifications.	

FILE HANDLING

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but unlike other concepts of Python, this concept here is also easy and short. Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let's start with Reading and Writing files.

We use **open ()** function in Python to open a file in read or write mode. As explained above, **open ()** will return a file object. To return a file object we use **open()** function along with two arguments, that accepts file name and the mode, whether to read or write. So, the syntax being: **open(filename, mode)**. There are three kinds of mode, that Python provides and how files can be opened:

- “ **r** “, for reading.
- “ **w** “, for writing.
- “ **a** “, for appending.
- “ **r+** “, for both reading and writing

Code from project

```
file=open('g.txt','a+')  
file.seek(0)  
d=file.readlines()  
file.close()
```

ACCOUNT CREATION AND LOGIN

In my project file handling is demonstrated for user account management. User accounts are created and its credentials are being stored in the file in a **custom encrypted** format. File open, reading, writing, searching (seeking) and file close operations are demonstrated in the project.

For the first time users, an account has to be created with a unique username and password. After account creation, user can login to view, create tasks or projects. File name used is **g.txt**



Account creation :

```
file=open('g.txt','a+')
    k='{},{}'.format(name,pas)
    file.write(k+'\n')
    file.close()
```

Account Login :

```
file=open('g.txt','a+')
    file.seek(0)
    d=file.readlines()
```

Custom encryption :

```
change=1
```

```
s=""
for i in range(0,len(pas)):
    k=chr(ord(pas[i])+i*change)
    change=change*-1
    s=s+k
pas=s
```

DATABASE CONNECTIVITY

A **database** is an organized collection of data, generally stored and accessed electronically from a computer system. The databases are more complex and they are often developed using formal design and modeling techniques.

The database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.

MYSQL

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data.

PYTHON MYSQL DATABASE CONNECTION

Following Arguments are required to connect MySQL from Python :

- **Username** – i.e., the username that you use to work with MySQL Server. The default username for the MySQL database is a **root**

- **Password** – Password is given by the user at the time of installing the MySQL database. If you are using root then you won't need the password.
- **Host Name** – is the server name or Ip address on which MySQL is running. if you are running on localhost, then you can use localhost, or it's IP, i.e. 127.0.0.0
- **Database Name** – Database name to which you want to connect

STEPS TO CONNECT MYSQL DATABASE IN PYTHON USING MYSQL CONNECTOR PYTHON

1. Install MySQL Connector Python using pip.
2. Use the `mysql.connector.connect()` method of MySQL Connector Python with required parameters to connect MySQL.
3. Use the connection object returned by a `connect()` method to create a `cursor` object to perform Database Operations.
4. The `cursor.execute()` to execute SQL queries from Python.
5. Close the Cursor object using a `cursor.close()` and MySQL database connection using `connection.close()` after your work completes.
6. Catch Exception if any that may occur during this process.

Here, database is used to store tasks/projects of each user. Each user has separate database and each kind of tasks/reminders have separate tables. Database is created with username and each user has three tables namely **rem, man and one**.

As an example a user is created with username 'V' database is created with same name and tables named **rem, man and one** are created. A screenshot of the MySQL database structure of the user V is shown below :

```

MySQL 8.0 Command Line Client - Unicode
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

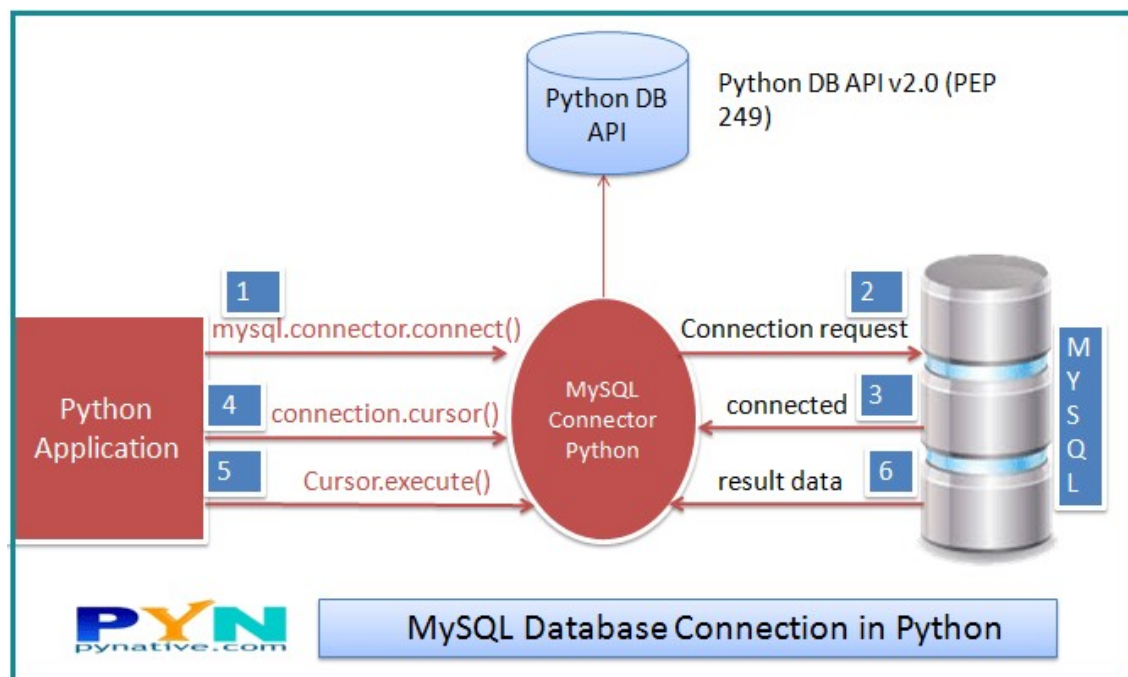
mysql> use v
Database changed
mysql> desc rem;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name  | char(20) | YES  |     | NULL    |       |
| date  | date   | YES  |     | NULL    |       |
| time  | char(5) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.10 sec)

mysql> desc man;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| sno   | int(20) | YES  |     | NULL    |       |
| name  | char(20) | YES  |     | NULL    |       |
| sdate | date   | YES  |     | NULL    |       |
| stime | char(5) | YES  |     | NULL    |       |
| edate | date   | YES  |     | NULL    |       |
| etime | char(5) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> desc one
-> ;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| sno   | int(20) | YES  |     | NULL    |       |
| name  | char(20) | YES  |     | NULL    |       |
| date  | date   | YES  |     | NULL    |       |
| stime | char(5) | YES  |     | NULL    |       |
| etime | char(5) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>

```

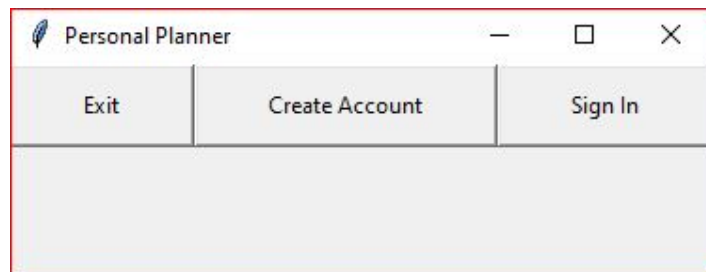


MODULES USED

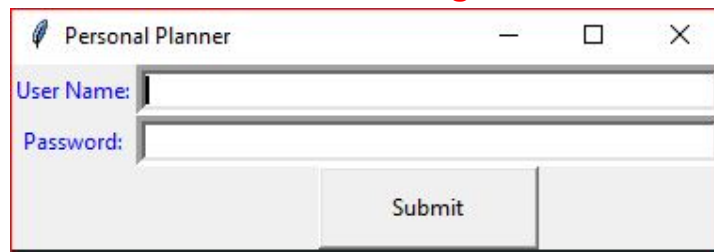
```
from tkinter import *  
from tkinter import ttk  
from tkcalendar import *  
from datetime import date  
from datetime import datetime
```

OUTPUT SCREENS

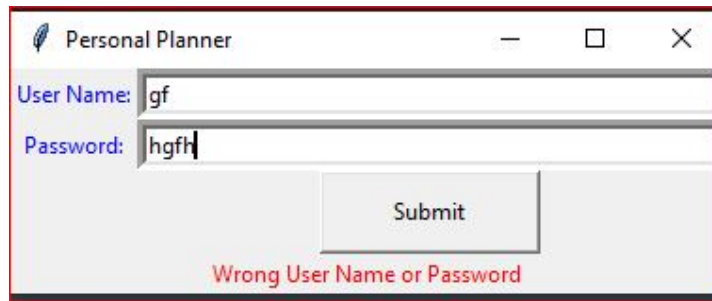
Main Screen



Account Creation/Login Screen

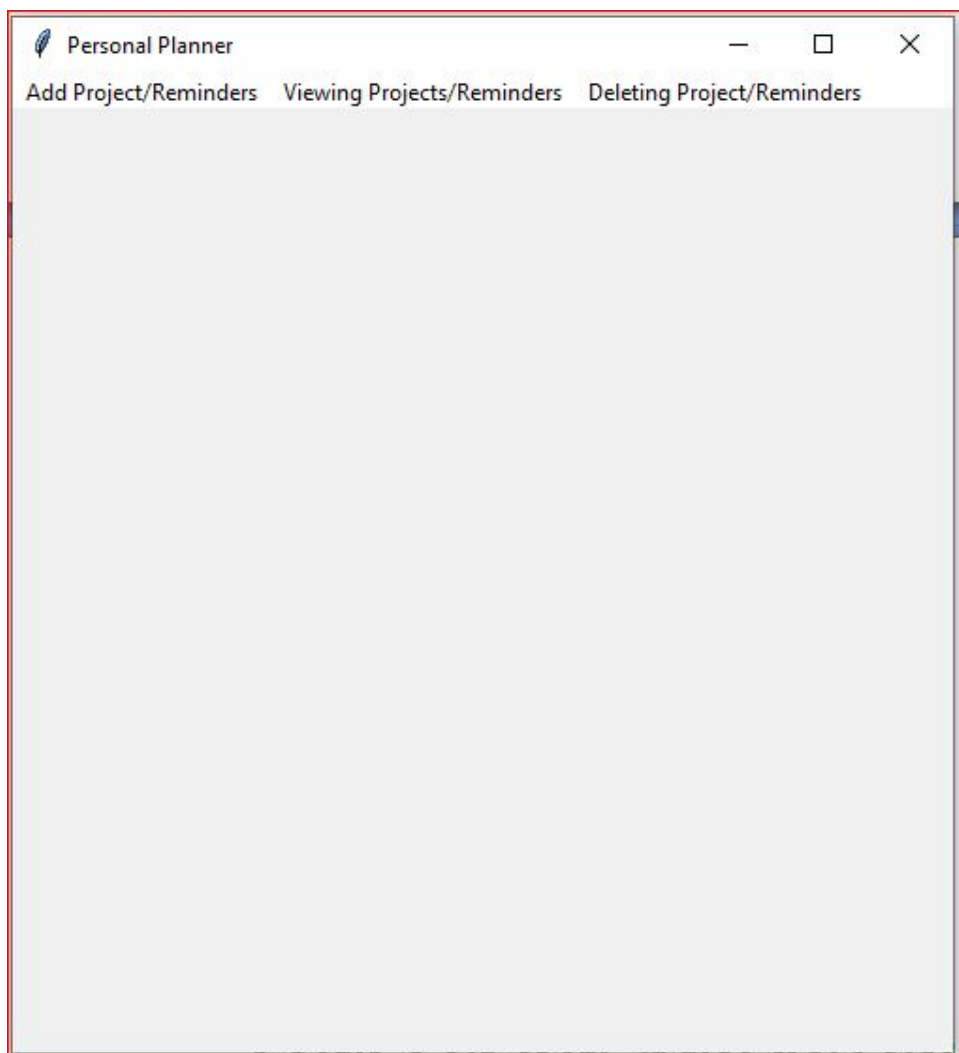


A valid username and password has to be entered to login to the planner. If user name or password is left blank an error message will appeared. If username and password is not correct user cannot log into the planner and error message wrong user name /password appears

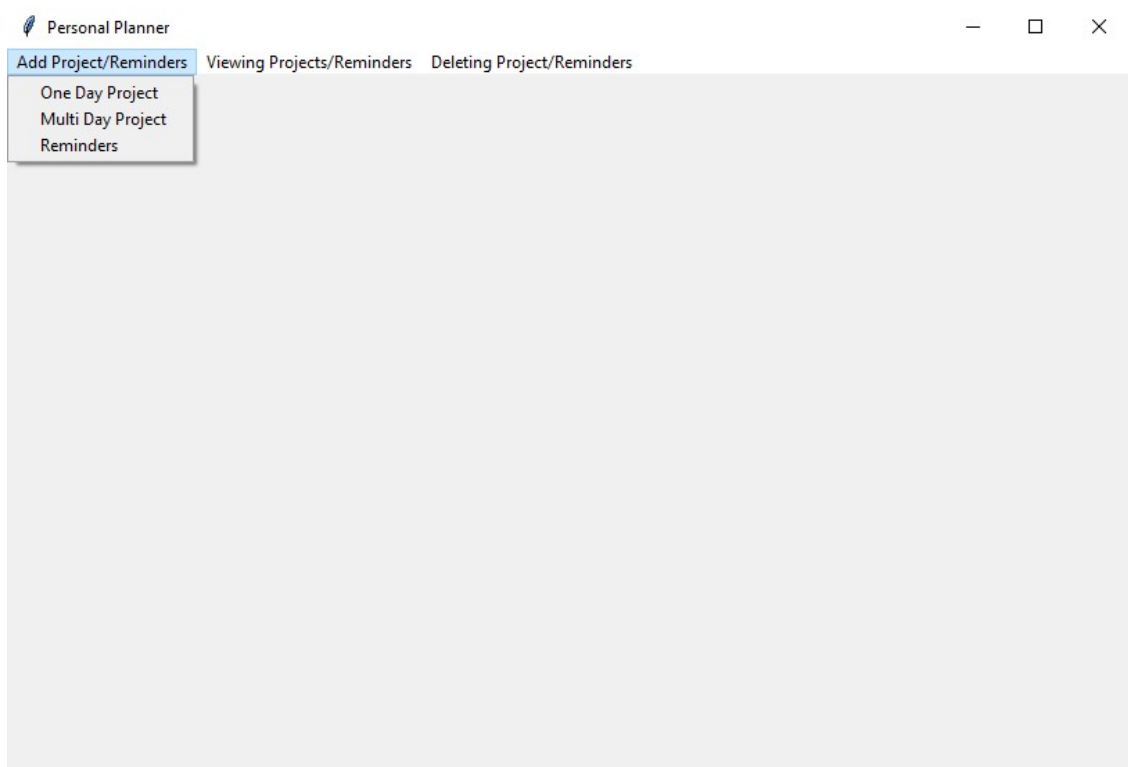


After user id is created, user can login to the planner manage his tasks according to the requirements :

Screen after Login to Manage projects/tasks



ONE DAY , MULTI-DAY PLANNING AND REMINDERS



The app is designed to schedule tasks for single day or for multiple days. User can assign tasks for a single day by entering calendar values with start and end time. Similarly, tasks can be assigned for multiple days starting from a date to end date which can be used to repeat tasks everyday at particular time for scheduled durations.

User has option to add projects, add reminders or view projects as shown in the figure below :

Screen below shows adding new single day projects/tasks. Each task has a unique name, date start time and end time.

Single Day Projects/tasks

Personal Planner

Add Project/Reminders
Viewing Projects/Reminders
Deleting Project/Reminders

Name of Project:

Date for The Project

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
44	26	27	28	29	30	31	1
45	2	3	4	5	6	7	8
46	9	10	11	12	13	14	15
47	16	17	18	19	20	21	22
48	23	24	25	26	27	28	29
49	30	1	2	3	4	5	6

The Starting Time for The Project

HH MM
00 00

The Ending Time for The Project

HH MM
00 00

Submit

Multiple Day Projects/tasks

Personal Planner

Add Project/Reminders
Viewing Projects/Reminders
Deleting Project/Reminders

Name of Project:

Starting Date for The Project

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
44	26	27	28	29	30	31	1
45	2	3	4	5	6	7	8
46	9	10	11	12	13	14	15
47	16	17	18	19	20	21	22
48	23	24	25	26	27	28	29
49	30	1	2	3	4	5	6

Ending date for the Project

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
44	26	27	28	29	30	31	1
45	2	3	4	5	6	7	8
46	9	10	11	12	13	14	15
47	16	17	18	19	20	21	22
48	23	24	25	26	27	28	29
49	30	1	2	3	4	5	6

Starting Time for The Project

HH MM
00 00

Ending Time for The Project

HH MM
00 00

Submit

ADD REMINDERS

User can add reminders, birthdays, marriage anniversary, meetings etc..

Personal Planner

Add Project/Reminders Viewing Projects/Reminders Deleting Project/Reminders

Reminders:

Date

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
44	26	27	28	29	30	31	1
45	2	3	4	5	6	7	8
46	9	10	11	12	13	14	15
47	16	17	18	19	20	21	22
48	23	24	25	26	27	28	29
49	30	1	2	3	4	5	6

Time

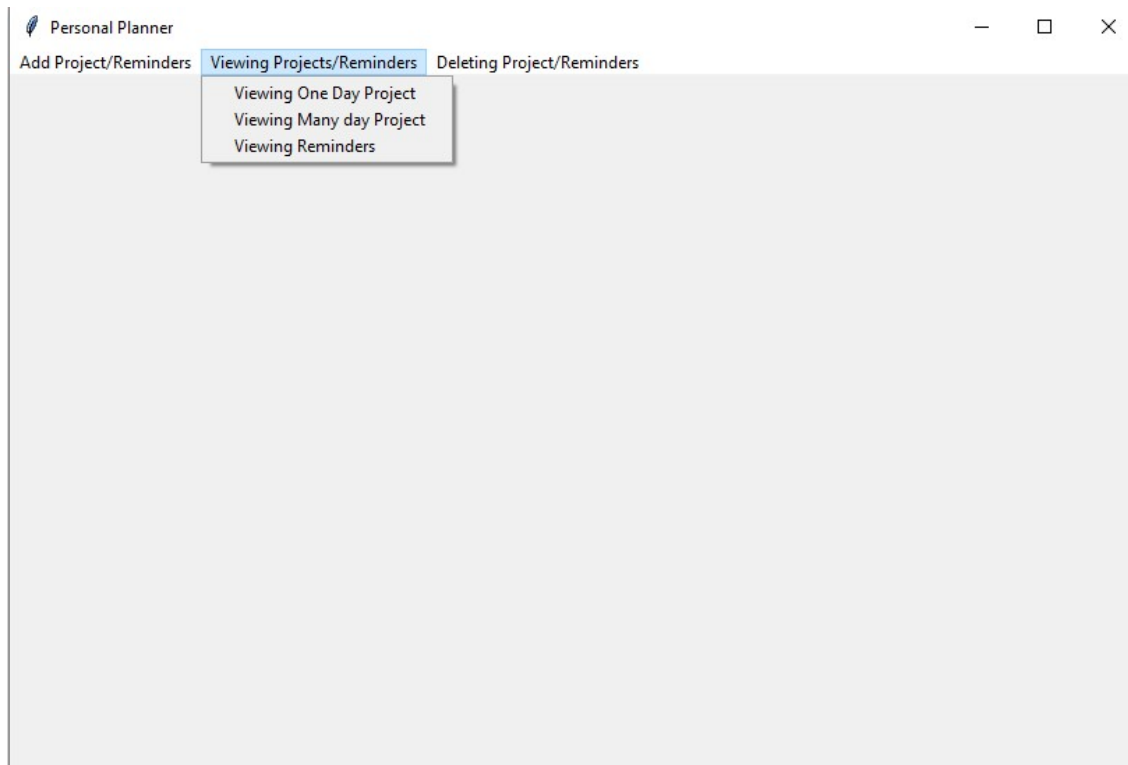
HH MM

00 00


Submit

VIEW AND SEARCH PLANS

Viewing and searching is essential part of any database management systems. Plans/tasks created have to be viewed and searched in the database and it should be viewed to user. Screen below shows search capability of the App. User can view single day, Multiple day or reminders.



Search can be made by name of the task or by date. Also user can view his complete tasks/reminders. An example is as below :


Personal Planner
—
□
×

Add Project/Reminders
Viewing Projects/Reminders
Deleting Project/Reminders

Search By Name

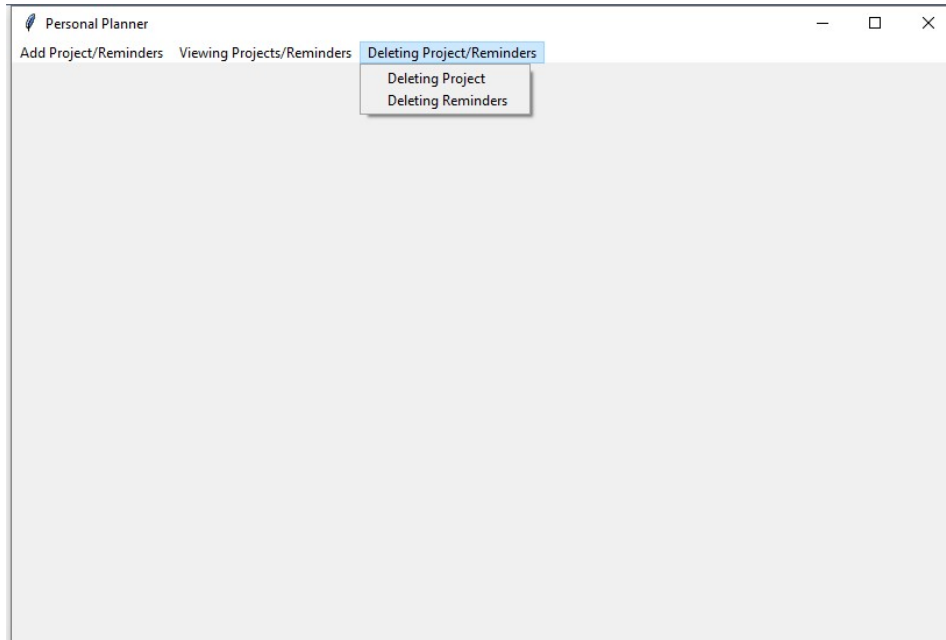
Search By Date

Search All

Name	Start Date	Start Time	End Date	End Time
Study	2018-02-06	00:00	2018-02-07	04:45
Practical(Comp)	2018-02-06	00:30	2018-02-09	00:45
dfsdfg	2020-11-21	17:30	2020-11-21	20:00
Jogging	2020-11-23	00:00	2020-11-23	00:15

DELETE PROJECT OR REMINDERS

Reminders that have already passed by are automatically deleted by the code after successfully login. Further, user can delete any project or reminders of his choice.



A list of one day and multiple day projects are displayed on clicking delete project menu as shown below. Then select option button one day or Many day project , then type name of the project to be deleted. On clicking Submit button, the project gets deleted.

Personal Planner

—

□

×

Add Project/Reminders

Viewing Projects/Reminders

Deleting Project/Reminders

One Day Pro

Names	Dates
Foot ball	2018-02-05
ytjyiy	2020-11-21
	2020-11-21
	2020-11-22
basket	2020-11-22

Many Day Pro

Names	Start Dates
Study	2018-02-06
Practical(Comp)	2018-02-06
dfsdfg	2020-11-21
Jogging	2020-11-23

Name of project to be deleted

☒ One Day Project

☐ Many Day Project

Submit

COMPLETE CODE

```
# -*- coding: utf-8 -*-  
"""
```

Created on Sun Oct 18 11:47:32 2020

```
@author: Peter  
"""
```

```
# l46 b19 e11 cal4 d8 m3 (l22)
```

```
def clear1():  
    for widget in top3.winfo_children():  
        if type(widget)!=Menu :  
            widget.destroy()  
def clear2():  
    for widget in top1.winfo_children():  
        if type(widget)==Label :  
            widget.destroy()  
def clear3():  
    for widget in top2.winfo_children():  
        if type(widget)==Label :  
            widget.destroy()  
def clear4():  
    for widget in top3.winfo_children():  
        if type(widget)==Label or type(widget)==LabelFrame:  
            widget.destroy()  
def clear5():  
    for widget in top3.winfo_children():  
        widget.destroy()  
  
def man():  
    def st2():  
  
        nam=e6.get()  
        date1=cal2.selection_get()  
        date2=cal3.selection_get()
```

```

stime=d5.get()+':'+d6.get()
etime=d7.get()+':'+d8.get()
#print(nam,sdate,edate,stime,etime)
if date1<d:
    l47=Label(top3,text='Choose a future start date',fg='red')
    l47.place(x=300,y=450)
    l47.after(2000, lambda: (l47.destroy())) )
elif date2<date1:
    l48=Label(top3,text='Choose a future end date',fg='red')
    l48.place(x=300,y=450)
    l48.after(2000, lambda: (l48.destroy())) )
elif date1==d and int(d5.get())<t.hour:
    l49=Label(top3,text='Choose a future start time',fg='red')
    l49.place(x=300,y=450)
    l49.after(2000, lambda: (l49.destroy()))
elif date1==d and int(d5.get())==t.hour and
int(d6.get())<=t.minute:
    l50=Label(top3,text='Choose a future start time',fg='red')
    l50.place(x=300,y=450)
    l50.after(2000, lambda: (l50.destroy()))
elif date2==date1 and d5.get()>d7.get():
    l51=Label(top3,text='Choose a future end time',fg='red')
    l51.place(x=300,y=450)
    l51.after(2000, lambda: (l51.destroy()))
elif date2==date1 and d5.get()==d7.get() and d6.get()>=d8.get():
    l53=Label(top3,text='Choose a future end time',fg='red')
    l53.place(x=300,y=450)
    l53.after(2000, lambda: (l53.destroy()))
elif nam=="":
    l52=Label(top3,text='Do enter name',fg='red')
    l52.place(x=300,y=450)
    l52.after(2000, lambda: (l52.destroy()))
else:

    c.execute("select max(sno) from man ")
    f=c.fetchall()
    for i in f:

```

```

        for e in i:

            if e==None:
                sno=1
            else:
                sno=int(e)+1
            c.execute("insert into man
values({}, '{}', '{}', '{}', '{}', '{}')".format(sno,nam,date1,stime,date2,etime
))
            sql.commit()
            clear1()
            top3.geometry('500x500')

clear1()
top3.geometry('1200x500')
t=datetime.now()
l15=Label(top3,text='Name of Project:',fg='blue')
l15.grid(row=0,column=0,pady=10)
e6=Entry(top3,width=50,borderwidth=5)
e6.grid(row=0,column=1,pady=10)

l16=Label(top3,text='Starting Date for The Project',fg='blue')
l16.grid(row=1,column=0,pady=10)
cal2 = Calendar(top3,
                font="Arial 14", selectmode='day',
                cursor="hand2", year=t.year, month=t.month, day=t.day)
cal2.grid(row=1,column=1,columnspan=2)

l17=Label(top3,text='Starting Time for The Project',fg='blue')
l17.place(x=0,y=320)

l18=Label(top3,text='HH      MM',fg='blue')
l18.grid(row=2,column=1)
l18.place(x=175,y=300)

```

```

d5=ttk.Combobox(top3, width =
3,value=['00','01','02','03','04','05','06','07','08','09','10','11','12','13','14'
,'15','16','17','18','19','20','21','22','23','24'])
d5.grid(row=3,column=1,pady=100)
d5.place(x=175,y=320)
d5.current(0)

```

```

d6=ttk.Combobox(top3, width = 3,value=['00','15','30','45'])
d6.grid(row=3,column=1,padx=100)
d6.place(x=225,y=320)
d6.current(0)

```

```

l21=Label(top3,text='Ending date for the Project',fg='blue')
l21.grid(row=1,column=3)

```

```

cal3 = Calendar(top3,
                font="Arial 14", selectmode='day',
                cursor="hand2", year=t.year, month=t.month, day=t.day)
cal3.place(x=790,y=50)

```

```

l19=Label(top3,text='Ending Time for The Project',fg='blue')
l19.grid(row=3,column=3,pady=30,padx=70)

```

```

l20=Label(top3,text='HH      MM',fg='blue')
l20.place(x=800,y=300)

```

```

d7=ttk.Combobox(top3, width =
3,value=['00','01','02','03','04','05','06','07','08','09','10','11','12','13','14'
,'15','16','17','18','19','20','21','22','23','24'])
d7.grid(row=3,column=1,pady=100)
d7.place(x=800,y=320)
d7.current(0)

```

```

d8=ttk.Combobox(top3, width = 3,value=['00','15','30','45'])
d8.grid(row=3,column=1,padx=100)
d8.place(x=850,y=320)
d8.current(0)

```

```

b7=Button(top3,text='Submit',padx=35,pady=10,command=(st2))
b7.place(x=200,y=390)

```

```

def one():
    def st1():
        date=cal1.selection_get()
        nam=e5.get()
        stime=s1.get()+':'+s2.get()
        etime=s3.get()+':'+s4.get()
        if date<d:
            l47=Label(top3,text='Choose a future start date',fg='red')
            l47.place(x=200,y=450)
            l47.after(2000, lambda: (l47.destroy()))
        elif date == d and int(s1.get())<t.hour:
            l49=Label(top3,text='Choose a future start time',fg='red')
            l49.place(x=200,y=450)
            l49.after(2000, lambda: (l49.destroy()))
        elif date==d and int(s1.get())==t.hour and int(s2.get())<t.minute:
            l50=Label(top3,text='Choose a future start time',fg='red')
            l50.place(x=200,y=450)
            l50.after(2000, lambda: (l50.destroy()))
        elif etime<=stime:
            l51=Label(top3,text='Choose a future end time',fg='red')
            l51.place(x=200,y=450)
            l51.after(2000, lambda: (l51.destroy()))
        elif nam=="":
            l52=Label(top3,text='Do enter Name',fg='red')
            l52.place(x=200,y=450)
            l52.after(2000, lambda: (l52.destroy()))
        else:
            c.execute("select max(sno) from one ")
            f=c.fetchall()
            for i in f:
                for e in i:

                    if e==None:

```

```

        sno=1
    else:
        sno=int(e)+1

    c.execute("insert into one
values({}, '{}', '{}', '{}', '{}')".format(sno,nam,date,stime,etime))
    sql.commit()
    clear1()
    top3.geometry('500x500')
clear1()
t=datetime.now()
top3.geometry('500x500')

l9=Label(top3,text='Name of Project:',fg='blue')
l9.grid(row=0,column=0,pady=10)
e5=Entry(top3,width=50,borderwidth=5)
e5.grid(row=0,column=1,pady=10)

l10=Label(top3,text='Date for The Project',fg='blue')
l10.grid(row=1,column=0,pady=10)
cal1= Calendar(top3,
                font="Arial 14", selectmode='day',
                cursor="hand2", year=t.year, month=t.month, day=t.day)
cal1.grid(row=1,column=1,columnspan=2)

l12=Label(top3,text='The Starting Time for The Project',fg='blue')
l12.place(x=10,y=320)

l11=Label(top3,text='HH      MM',fg='blue')
l11.grid(row=2,column=1)
l11.place(x=200,y=300)

s1=StringVar()
d1=ttk.Combobox(top3, width =
3,value=['00','01','02','03','04','05','06','07','08','09','10','11','12','13','14'
,'15','16','17','18','19','20','21','22','23','24'],textvariable=s1)
d1.grid(row=3,column=1,pady=100)

```

```
d1.place(x=200,y=320)
d1.current(0)
```

```
s2=StringVar()
d2=ttk.Combobox(top3, width =
3,value=['00','15','30','45'],textvariable=s2)
d2.grid(row=3,column=1,padx=100)
d2.place(x=250,y=320)
d2.current(0)
```

```
l13=Label(top3,text='The Ending Time for The Project',fg='blue')
l13.place(x=10,y=365)
```

```
l14=Label(top3,text='HH      MM',fg='blue')
l14.grid(row=2,column=1)
l14.place(x=200,y=345)
```

```
s3=StringVar()
d3=ttk.Combobox(top3, width =
3,textvariable=s3,value=['00','01','02','03','04','05','06','07','08','09','10'
,'11','12','13','14','15','16','17','18','19','20','21','22','23','24'])
d3.grid(row=3,column=1,pady=100)
d3.place(x=200,y=365)
d3.current(0)
```

```
s4=StringVar()
d4=ttk.Combobox(top3, width =
3,textvariable=s4,value=['00','15','30','45'])
d4.grid(row=3,column=1,padx=100)
d4.place(x=250,y=365)
d4.current(0)
```

```
b6=Button(top3,text='Submit',padx=35,pady=10,command=(st1))
b6.place(x=200,y=390)
```

```
def n1():
    def n2():
```



```

s=e7.get()
e7.destroy()
l23.destroy()
b10.destroy()
c.execute("select * from one where name like
"%{}%".format(s))
f=c.fetchall()
lf11=LabelFrame(top3, text="",relief=RIDGE)
lf11.grid(row=1,column=0,columnspan=3)
lf7=LabelFrame(lf11, text="Name",relief=RIDGE)
lf7.grid(row=0,column=0)
lf8=LabelFrame(lf11, text="Start Date",relief=RIDGE)
lf8.grid(row=0,column=1)
lf9=LabelFrame(lf11, text="Start Time",relief=RIDGE)
lf9.grid(row=0,column=2)
lf10=LabelFrame(lf11, text="End Time",relief=RIDGE)
lf10.grid(row=0,column=3)
if f==[]:
    l24=Label(top3,text='None Found',fg='red')
    l24.grid(row=1,column=0,columnspan=2)
s=0
for i in f:

    l25=Label(lf7,text=i[1],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf8,text=i[2],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf9,text=i[3],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf10,text=i[4],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    s=s+1
b8['state']=NORMAL
b9['state']=NORMAL
b17['state']=NORMAL

b8['state']=DISABLED

```

```
b9['state']=DISABLED
b17['state']=DISABLED
```

```
l23=Label(top3,text='Name of Project',fg='blue')
l23.grid(row=1,column=0)
```

```
e7=Entry(top3,width=24,borderwidth=5)
e7.grid(row=1,column=1,pady=10)
```

```
b10=Button(top3,text='Search',padx=35,pady=10,command=n2)
b10.grid(column=0,row=2,columnspan=2)
```

```
def n3():
    def n4():
        s=e8.get()
        e8.destroy()
        l26.destroy()
        b11.destroy()
        try:
            c.execute("select * from one where date like
"%{}%"".format(s))
            f=c.fetchall()
            lf11=LabelFrame(top3, text="",relief=RIDGE)
            lf11.grid(row=1,column=0,columnspan=3)
            lf7=LabelFrame(lf11, text="Name",relief=RIDGE)
            lf7.grid(row=0,column=0)
            lf8=LabelFrame(lf11, text="Start Date",relief=RIDGE)
            lf8.grid(row=0,column=1)
            lf9=LabelFrame(lf11, text="Start Time",relief=RIDGE)
            lf9.grid(row=0,column=2)
            lf10=LabelFrame(lf11, text="End Time",relief=RIDGE)
            lf10.grid(row=0,column=3)
            if f==[]:
                l24=Label(top3,text='None Found',fg='red')
                l24.grid(row=1,column=0,columnspan=2)
            s=0
            for i in f:
```

```

l25=Label(lf7,text=i[1],fg='blue')
l25.grid(row=s,column=0,columnspan=3)
l25=Label(lf8,text=i[2],fg='blue')
l25.grid(row=s,column=0,columnspan=3)
l25=Label(lf9,text=i[3],fg='blue')
l25.grid(row=s,column=0,columnspan=3)
l25=Label(lf10,text=i[4],fg='blue')
l25.grid(row=s,column=0,columnspan=3)
s=s+1
except:
    l27=Label(top3,text='None Found',fg='red')
    l27.grid(row=1,column=0,columnspan=2)
    b8['state']=NORMAL
    b9['state']=NORMAL
    b17['state']=NORMAL

    b8['state']=DISABLED
    b9['state']=DISABLED
    b17['state']=DISABLED

    l26=Label(top3,text='Starting Date of the Project\n(YYYY-MM-
DD)',fg='blue')
    l26.grid(row=1,column=0)

    e8=Entry(top3,width=24,borderwidth=5)
    e8.grid(row=1,column=1,pady=10)

    b11=Button(top3,text='Search',padx=35,pady=10,command=n4)
    b11.grid(column=0,row=2,columnspan=2)

def n10():
    c.execute("select * from one")
    f=c.fetchall()
    lf11=LabelFrame(top3, text="",relief=RIDGE)
    lf11.grid(row=1,column=0,columnspan=3)
    lf7=LabelFrame(lf11, text="Name",relief=RIDGE)

```

```

lf7.grid(row=0,column=0)
lf8=LabelFrame(lf11, text="Start Date",relief=RIDGE)
lf8.grid(row=0,column=1)
lf9=LabelFrame(lf11, text="Start Time",relief=RIDGE)
lf9.grid(row=0,column=2)
lf10=LabelFrame(lf11, text="End Time",relief=RIDGE)
lf10.grid(row=0,column=3)
if f==[]:
    l24=Label(top3,text='None Found',fg='red')
    l24.grid(row=1,column=0,columnspan=2)
s=0
for i in f:

    l25=Label(lf7,text=i[1],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf8,text=i[2],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf9,text=i[3],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf10,text=i[4],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    s=s+1
def v1():
    clear1()
    top3.geometry('600x300')
    global b8
    global b9
    global b17
    b8=Button(top3,text='Search By
Name',padx=35,pady=10,command=lambda:(clear4(),n1()))
    b8.grid(column=0,row=0)

    b9=Button(top3,text='Search By
Date',padx=35,pady=10,command=lambda:(clear4(),n3()))
    b9.grid(column=1,row=0)

```

```

b17=Button(top3,text='Search
All',padx=35,pady=10,command=lambda:(clear4(),n10()))
b17.grid(column=2,row=0)

def n5():
    def n6():
        s=e9.get()
        e9.destroy()
        l29.destroy()
        b14.destroy()
        c.execute("select * from man")
        f=c.fetchall()
        lf11=LabelFrame(top3, text="",relief=RIDGE)
        lf11.grid(row=1,column=0,columnspan=3)
        lf7=LabelFrame(lf11, text="Name",relief=RIDGE)
        lf7.grid(row=0,column=0)
        lf8=LabelFrame(lf11, text="Start Date",relief=RIDGE)
        lf8.grid(row=0,column=1)
        lf9=LabelFrame(lf11, text="Start Time",relief=RIDGE)
        lf9.grid(row=0,column=2)
        lf10=LabelFrame(lf11, text="End Date",relief=RIDGE)
        lf10.grid(row=0,column=3)
        lf12=LabelFrame(lf11, text="End Time",relief=RIDGE)
        lf12.grid(row=0,column=4)
        if f==[]:
            l24=Label(top3,text='None Found',fg='red')
            l24.grid(row=1,column=0,columnspan=2)
        s=0
        for i in f:

            l25=Label(lf7,text=i[1],fg='blue')
            l25.grid(row=s,column=0,columnspan=3)
            l25=Label(lf8,text=i[2],fg='blue')
            l25.grid(row=s,column=0,columnspan=3)
            l25=Label(lf9,text=i[3],fg='blue')
            l25.grid(row=s,column=0,columnspan=3)
            l25=Label(lf10,text=i[4],fg='blue')

```

```

l25.grid(row=s,column=0,columnspan=3)
l25=Label(lf12,text=i[5],fg='blue')
l25.grid(row=s,column=0,columnspan=3)
s=s+1
b12['state']=NORMAL
b13['state']=NORMAL
b16['state']=NORMAL
b12['state']=DISABLED
b13['state']=DISABLED
b16['state']=DISABLED
l29=Label(top3,text='Name of Project',fg='blue')
l29.grid(row=1,column=0)

e9=Entry(top3,width=24,borderwidth=5)
e9.grid(row=1,column=1,pady=10)

b14=Button(top3,text='Search',padx=35,pady=10,command=n6)
b14.grid(column=0,row=2,columnspan=2)

```

```
def n7():
```

```
def n8():
```

```

s=e10.get()
e10.destroy()
l32.destroy()
b15.destroy()
c.execute("select * from man")
f=c.fetchall()
lf11=LabelFrame(top3, text="",relief=RIDGE)
lf11.grid(row=1,column=0,columnspan=3)
lf7=LabelFrame(lf11, text="Name",relief=RIDGE)
lf7.grid(row=0,column=0)
lf8=LabelFrame(lf11, text="Start Date",relief=RIDGE)
lf8.grid(row=0,column=1)
lf9=LabelFrame(lf11, text="Start Time",relief=RIDGE)
lf9.grid(row=0,column=2)
lf10=LabelFrame(lf11, text="End Date",relief=RIDGE)
lf10.grid(row=0,column=3)

```

```

lf12=LabelFrame(lf11, text="End Time",relief=RIDGE)
lf12.grid(row=0,column=4)
if f==[]:
    l24=Label(top3,text='None Found',fg='red')
    l24.grid(row=1,column=0,columnspan=2)
s=0
for i in f:

    l25=Label(lf7,text=i[1],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf8,text=i[2],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf9,text=i[3],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf10,text=i[4],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf12,text=i[5],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    s=s+1
b12['state']=NORMAL
b13['state']=NORMAL
b16['state']=NORMAL
b12['state']=DISABLED
b13['state']=DISABLED
b16['state']=DISABLED

l32=Label(top3,text='Start Date of Project\n(YYYY-MM-DD)',fg='blue')
l32.grid(row=1,column=0)

e10=Entry(top3,width=24,borderwidth=5)
e10.grid(row=1,column=1,pady=10)

b15=Button(top3,text='Search',padx=35,pady=10,command=n8)
b15.grid(column=0,row=2,columnspan=2)

def n9():

```

```

c.execute("select * from man")
f=c.fetchall()
lf11=LabelFrame(top3, text="",relief=RIDGE)
lf11.grid(row=1,column=0,columnspan=3)
lf7=LabelFrame(lf11, text="Name",relief=RIDGE)
lf7.grid(row=0,column=0)
lf8=LabelFrame(lf11, text="Start Date",relief=RIDGE)
lf8.grid(row=0,column=1)
lf9=LabelFrame(lf11, text="Start Time",relief=RIDGE)
lf9.grid(row=0,column=2)
lf10=LabelFrame(lf11, text="End Date",relief=RIDGE)
lf10.grid(row=0,column=3)
lf12=LabelFrame(lf11, text="End Time",relief=RIDGE)
lf12.grid(row=0,column=4)
if f==[]:
    l24=Label(top3,text='None Found',fg='red')
    l24.grid(row=1,column=0,columnspan=2)
s=0
for i in f:

    l25=Label(lf7,text=i[1],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf8,text=i[2],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf9,text=i[3],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf10,text=i[4],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    l25=Label(lf12,text=i[5],fg='blue')
    l25.grid(row=s,column=0,columnspan=3)
    s=s+1
def v2():
    clear1()
    top3.geometry('600x300')
    global b12
    global b13

```



```

global b16
b12=Button(top3,text='Search By
Name',padx=35,pady=10,command=lambda:(clear4(),n5()))
b12.grid(column=0,row=0)

b13=Button(top3,text='Search By
Date',padx=35,pady=10,command=lambda:(clear4(),n7()))
b13.grid(column=1,row=0)

b16=Button(top3,text='Search
All',padx=35,pady=10,command=lambda:(clear4(),n9()))
b16.grid(column=2,row=0)

def v3():
    def n11():
        nam=e11.get()
        date=cal4.selection_get()
        time=d9.get()+':'+d10.get()
        if nam=="":
            l60=Label(top3,text='Do Enter Reminders:',fg='red')
            l60.place(x=300,y=450)
            l60.after(12000,lambda: l60.destroy())
        elif date<d:
            l54=Label(top3,text='Choose a future Date',fg='red')
            l54.place(x=300,y=450)
            l54.after(10000,lambda: l54.destroy())
        elif date==d and int(d9.get())<t.hour:
            l55=Label(top3,text='Choose a future start time',fg='red')
            l55.place(x=300,y=450)
            l55.after(2000, lambda: (l55.destroy()))
        elif date==d and int(d9.get())==t.hour and
int(d10.get())<t.minute:
            l56=Label(top3,text='Choose a future start time',fg='red')
            l56.place(x=300,y=450)
            l56.after(2000, lambda: (l56.destroy()))
        else:

```

```

        c.execute("insert into rem
values('{}', '{}', '{}')".format(nam,date,time))
        sql.commit()
        clear1()
        top3.geometry('500x500')
clear1()
top3.geometry('500x500')
t=datetime.now()
l40=Label(top3,text='Reminders:',fg='blue')
l40.grid(row=0,column=0,sticky='W')

l41=Label(top3,text='Date',fg='blue')
l41.grid(row=1,column=0,pady=10)

l45=Label(top3,text='Time',fg='blue')
l45.grid(row=2,column=0,pady=35)

e11=Entry(top3,width=50,borderwidth=5)
e11.grid(row=0,column=1)

cal4= Calendar(top3,
                font="Arial 14", selectmode='day',
                cursor="hand2", year=t.year, month=t.month, day=t.day)
cal4.grid(row=1,column=1,columnspan=2,pady=10)

l46=Label(top3,text='HH      MM',fg='blue')
l46.place(x=100,y=300)

d9=ttk.Combobox(top3, width =
3,value=['00','01','02','03','04','05','06','07','08','09','10','11','12','13','14'
,'15','16','17','18','19','20','21','22','23','24'])
d9.place(x=100,y=320)
d9.current(0)

d10=ttk.Combobox(top3, width = 3,value=['00','15','30','45'])
d10.place(x=150,y=320)

```

```
d10.current(0)
```

```
b18=Button(top3,text='Submit',padx=35,pady=10,command=n11)  
b18.grid(column=0,row=3,columnspan=2)
```

```
def v4():  
    clear1()  
    ch=0  
    today=date.today()  
    c.execute("select * from rem order by date")  
    f=c.fetchall()  
    s=0  
    lf1=LabelFrame(top3, text="Name",relief=RIDGE)  
    lf1.grid(row=0,column=0)  
  
    lf2=LabelFrame(top3, text="Date",relief=RIDGE)  
    lf2.grid(row=0,column=1)  
  
    lf3=LabelFrame(top3, text="Time",relief=RIDGE)  
    lf3.grid(row=0,column=2)  
    for i in f:  
        if i[1]>=today:  
            ch=1  
            left = Label(lf1, text=i[0])  
            left.grid(row=s,column=0)  
  
            left = Label(lf2, text=i[1])  
            left.grid(row=s,column=0)  
  
            left = Label(lf3, text=i[2])  
            left.grid(row=s,column=0)  
  
            s=s+1  
    if ch==0:  
        left = Label(top3, text='None found',fg='blue')  
        left.grid(row=0,column=0)  
def f1():
```

```

def f2():

    c.execute("select * from rem where
name='{0}'.format(e12.get())")
    f=c.fetchall()

    if f==[]:

        l47=Label(top3,text='No such Name found',fg='blue')
        l47.grid(row=3,column=0,pady=60)
        l47.after(2000,l47.destroy)
    else:

        c.execute("delete from rem where name
='{0}'.format(e12.get())")
        sql.commit()
        clear1()
        f1()

    clear1()
    lf6=LabelFrame(top3, text="Reminders",relief=RIDGE)
    lf6.grid(row=0,column=0)

    lf4=LabelFrame(lf6, text="Names",relief=RIDGE)
    lf4.grid(row=0,column=0)

    lf5=LabelFrame(lf6, text="Dates",relief=RIDGE)
    lf5.grid(row=0,column=1)

    c.execute("select * from rem")
    f=c.fetchall()
    s=0
    for i in f:
        l44=Label(lf4,text=i[0],fg='blue')
        l44.grid(row=s,column=0)

        l45=Label(lf5,text=i[1],fg='blue')

```

```

l45.grid(row=s,column=0)

s=s+1

l46=Label(top3,text='Name of project to be deleted',fg='blue')
l46.grid(row=1,column=0,pady=30)

e12=Entry(top3,width=50,borderwidth=5)
e12.grid(row=1,column=1,pady=30)

b19=Button(top3,text='Submit',padx=35,pady=10,command=f2 )
b19.grid(row=2,column=0)

def f3():
    def f4():
        if var.get()==1:
            s='one'
        else:
            s='man'

    c.execute("select * from {} where
name='{}'.format(s,e12.get())")
    f=c.fetchall()

    if f==[]:

        l47=Label(top3,text='No such Name found',fg='blue')
        l47.grid(row=3,column=0,pady=60)
        l47.after(2000,l47.destroy)
    else:

        c.execute("delete from {} where name
='{}'.format(s,e12.get())")
        sql.commit()
        clear1()
        f3()

```

```

clear1()
lf12=LabelFrame(top3, text="One Day Pro",relief=RIDGE)
lf12.grid(row=0,column=0)

lf13=LabelFrame(lf12, text="Names",relief=RIDGE)
lf13.grid(row=0,column=0)

lf14=LabelFrame(lf12, text="Dates",relief=RIDGE)
lf14.grid(row=0,column=1)

c.execute("select * from one")
f=c.fetchall()
s=0
for i in f:
    l44=Label(lf13,text=i[1],fg='blue')
    l44.grid(row=s,column=0)

    l45=Label(lf14,text=i[2],fg='blue')
    l45.grid(row=s,column=0)

    s=s+1

lf15=LabelFrame(top3, text="Many Day Pro",relief=RIDGE)
lf15.grid(row=0,column=1)

lf16=LabelFrame(lf15, text="Names",relief=RIDGE)
lf16.grid(row=0,column=0)

lf17=LabelFrame(lf15, text="Start Dates",relief=RIDGE)
lf17.grid(row=0,column=1)

c.execute("select * from man")
f=c.fetchall()
s=0
for i in f:
    l46=Label(lf16,text=i[1],fg='blue')

```

```

l46.grid(row=s,column=0)

l47=Label(lf17,text=i[2],fg='blue')
l47.grid(row=s,column=0)

s=s+1

l46=Label(top3,text='Name of project to be deleted',fg='blue')
l46.grid(row=1,column=0,pady=30)

e12=Entry(top3,width=50,borderwidth=5)
e12.grid(row=1,column=1,pady=30)

var = IntVar()
r1 = Radiobutton(top3, text="One Day Project", variable=var,
value=1)
r1.grid(row=2,column=0)
r2 = Radiobutton(top3, text="Many Day Project", variable=var,
value=2)
r2.grid(row=3,column=0)

b19=Button(top3,text='Submit',padx=35,pady=10,command=f4 )
b19.grid(row=4,column=0)

def per():
    today=date.today()

    c.execute("select * from rem")
    f=c.fetchall()
    for i in f:
        if i[1]<today:
            c.execute("delete from rem where date='{ }'".format(i[1]))

    sql.commit()
    global top3
    top3=Toplevel()

```

```

top3.geometry('500x500')
my_menu=Menu(top3)

m1=Menu(my_menu,tearoff=0)
m1.add_command(label='One Day Project',command=(one))
m1.add_command(label='Multi Day Project',command=(man))
m1.add_command(label='Reminders',command=v3)
my_menu.add_cascade(label='Add Project/Reminders',
menu=m1)

m2=Menu(my_menu,tearoff=0)
m2.add_command(label='Viewing One Day
Project',command=v1)
m2.add_command(label='Viewing Many day
Project',command=v2)
m2.add_command(label='Viewing Reminders',command=v4)
my_menu.add_cascade(label='Viewing
Projects/Reminders',menu=m2)

m3=Menu(my_menu,tearoff=0)
m3.add_command(label='Deleting Project',command=f3)
m3.add_command(label="Deleting Reminders",command=f1)
my_menu.add_cascade(label='Deleting
Project/Reminders',menu=m3)
top3.config(menu=my_menu)

l43=Label(top3,text='Hello,how are you {}? Hope you are
havving a productive day'.format(name),fg='white',bg='purple')
l43.grid(row=2,column=0)"""
def cre():
    def sub():
        l2=Label(top1,text='Name:',fg='blue')
        l2.grid(row=0,column=0)
        l3=Label(top1,text='Password:',fg='blue')
        l3.grid(row=1,column=0)
        sign=-1
        global name

```



```

n=e1.get()
name=e1.get().replace(" ", "")
pas=e2.get()
file=open('g.txt','a+')
change=1
s=""
for i in range(0,len(pas)):
    k=chr(ord(pas[i])+i*change)
    change=change*-1
    s=s+k
pas=s
file.seek(0)
s=file.readlines()
file.close()
for i in s:
    i=i.strip('\n')
    i=i.split(',')
    if i[0]==name:
        l35=Label(top1,text='Use another User Name. User Name
Already exists.',fg='red')
        l35.grid(row=3,column=1)
        sign=2
    if name==" or pas==" :
        sign=2
        l36=Label(top1,text='Please enter name or
password',fg='red')
        l36.grid(row=3,column=1)
    if sign!=2:
        file=open('g.txt','a+')
        k='{},{}'.format(name,pas)
        file.write(k+'\n')
        file.close()

c.execute('create database {}'.format(name))
c.execute('use {}'.format(name))
c.execute('create table one (sno int(20),name char(20),date
date,stime char(5),etime char(5)) ')

```

```
c.execute('create table man (sno int(20),name char(20),sdate
date,stime char(5),edate date,etime char(5)) ')
```

```
c.execute('create table rem (name char(20),date date,time
char(5))')
```

```
sql.commit()
```

```
l7=Label(top1,text='account created',fg='blue')
```

```
l7.grid(row=3,column=1)
```

```
l7.after(500, lambda: (top1.destroy(),per()) )
```

```
global top1
```

```
top1=Toplevel()
```

```
l2=Label(top1,text='Name:',fg='blue')
```

```
l2.grid(row=0,column=0)
```

```
l3=Label(top1,text='Password:',fg='blue')
```

```
l3.grid(row=1,column=0)
```

```
e1=Entry(top1,width=50,borderwidth=5)
```

```
e1.grid(row=0,column=1)
```

```
e2=Entry(top1,width=50,borderwidth=5)
```

```
e2.grid(row=1,column=1)
```

```
b5=Button(top1,text='Submit',padx=35,pady=10,command=lambda:
(clear2(),sub()))
```

```
b5.grid(row=2,column=1)
```

```
def sig():
```

```
def ch():
```

```
sign=-1
```

```
l4=Label(top2,text='User Name:',fg='blue')
```

```
l4.grid(row=0,column=0)
```

```
l5=Label(top2,text='Password:',fg='blue')
```

```
l5.grid(row=1,column=0)
```

```
global name
```

```
name=e3.get()
```

```
pa=e4.get()
```

```
file=open('g.txt','a+')
```

```

file.seek(0)
d=file.readlines()

for i in d:
    i=i.strip("\n")
    i=i.split(',')
    change=-1
    s=""
    for k in range(0,len(i[1])):
        j=chr(ord(i[1][k])+k*change)
        change=change*-1
        s=s+j
    if i[0]==name and s==pa:
        sign=1
    if name==" or pa=="":
        l37=Label(top2,text='Please do enter name or
password',fg='red')
        l37.grid(row=3,column=0,columnspan=2)
    elif sign!=1:
        l6=Label(top2,text='Wrong User Name or Password',fg='red')
        l6.grid(row=3,column=0,columnspan=2)

    else:
        l8=Label(top2,text='Correct Information',fg='blue')
        l8.grid(row=3,column=1)
        l8.after(500, lambda: (top2.destroy(),per()) )
        c.execute('use {}'.format(name))

    file.close()
global top2
top2=Toplevel()
l4=Label(top2,text='User Name:',fg='blue')
l4.grid(row=0,column=0)
l5=Label(top2,text='Password:',fg='blue')
l5.grid(row=1,column=0)

e3=Entry(top2,width=50,borderwidth=5)

```

```
e3.grid(row=0,column=1)
e4=Entry(top2,width=50,borderwidth=5)
e4.grid(row=1,column=1)
```

```
b4=Button(top2,text='Submit',padx=35,pady=10,command=lambda:
(clear3(),ch()))
b4.grid(row=2,column=1)
```

```
from tkinter import *
from tkinter import ttk
from tkcalendar import *
from datetime import date
from datetime import datetime
```

```
t=datetime.now()
d=date.today()
root=Tk()
root.title('Personal Planner')
#root.iconbitmap('<path of the icon>') for changing Icon
```

```
import mysql.connector
sql=mysql.connector.connect(user='root',passwd='12345',host='localhost')
c=sql.cursor()
```

```
l1=Label(root,text='Welcome to your Personal Planner.\nChoose
from the below options.',fg='blue')
l1.grid(row=0,column=0,columnspan=3)
```

```
b1=Button(root,text='Exit',padx=35,pady=10,command=
lambda:(root.destroy()))
b1.grid(row=3,column=0)
b2=Button(root,text='Create
Account',padx=35,pady=10,command=cre)
b2.grid(row=3,column=1)
b3=Button(root,text='Log In',padx=35,pady=10,command=sig)
```

b3.grid(row=3,column=2)

root.mainloop()

#please make a line between different columns using widget

TESTING OF PROJECT

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

My humble effort to demonstrate the basic capabilities of Python also need to be tested. My little brother Peter Biju, was assigned the task of software tester. He did the following tasks : (The task below are self explanatory and can be used as User Guide/operation manual of the APP)

User Guide/operation Manual of the APP

Steps	Details of tasks	Remarks
1	Create an account	Account created with user name Peter with his secret password
2	Login	Login with User Peter was successful.
3	Add single day tasks	He created a task for a day named Game
4.	Add Multiple day tasks	Peter created a task stating from 01 December 2020 and ending at 10 December 2020 repeating everyday from 0700 hrs to 0800 hrs.
5.	Add reminders	He added reminder for his brother's

		birthday
6.	Search project for single day	He was able to search project by name or date. Also he could view list of his complete tasks.
7.	Search project for multiple day	He was able to search project by name or date. Also he could view list of his completed tasks.
8.	Delete single day or multiple day project	He deleted a project named game.
9	Delete Reminder	He deleted the reminder of a birthday
10.	Exit	Exit the code

CONCLUSION AND FUTURE ENHANCEMENTS

Python is free and simple to learn. Its primary features are that it is high-level, dynamically typed and interpreted. This makes debugging of errors easy and encourages the rapid development of application prototypes, marking itself as *the language to code with*. Python was developed in 1989 by Guido Van Rossum. Python supports cross-platform operating systems which makes building applications with it all the more convenient. Some of the globally known applications such as YouTube, BitTorrent, DropBox, etc. use Python to achieve their functionality. Thus, Python is widely used in real world applications. Some of the uses of Python are :

- Web Development
- Game Development
- Machine Learning and Artificial Intelligence
- Data Science and Data Visualization
- Desktop GUI
- Business Applications
- Audio and Video Applications
- CAD Applications
- Embedded Applications

The project primarily demonstrates basic capabilities of the powerful but simple use of Python coding techniques with GUI based on Tkinter. File handling and database management functions are used.

The APP is capable of managing and planning one's life. It can manage reminders, projects and tasks. However, always there is a scope for improvement like, we can use rich Graphical user interface to make it more attractive, can be made online to manage personal plans from anywhere, add more functionalities...

BIBLGRAPHY

- 1. Computer Science with Python by Sumita Arora**
- 2. Notes from Mrs R. Nagpal**
- 3. <https://www.geeksforgeeks.org/>**
- 4. <https://www.tutorialspoint.com/>**