DBMS Project

Focus Mode App

DBMS for a focus mode app

Naman Kashyap

PES1UG20CS260

V semester 'E' section

Roll no. 27

Format:

Mini Project:

1. A short description about the project and scope
2. ER Diagram
3. Relational Scheme
4. DDL statements to build the database
5. Different methods used to populate data. - Show statements used under different methods
6. JOIN queries
7. Aggregate Functions
8. SET Operators
9. Functions or Procedure
10. Triggers or cursors
11. Higher Level Programming - A simple frontend that talks to the backend database is required to be developed.

1. **A short description:**

Focus mode DBMS

A focus mode database, that keeps track of focus sessions, apps used, breaks taken, and app timers. This project uses MySQL and Python to simulate the features and functionalities of a focus mode assist app. The aim of this project is to enhance our understanding of MySQL and to create a minimalistic UI for the app using tkinter in Python.

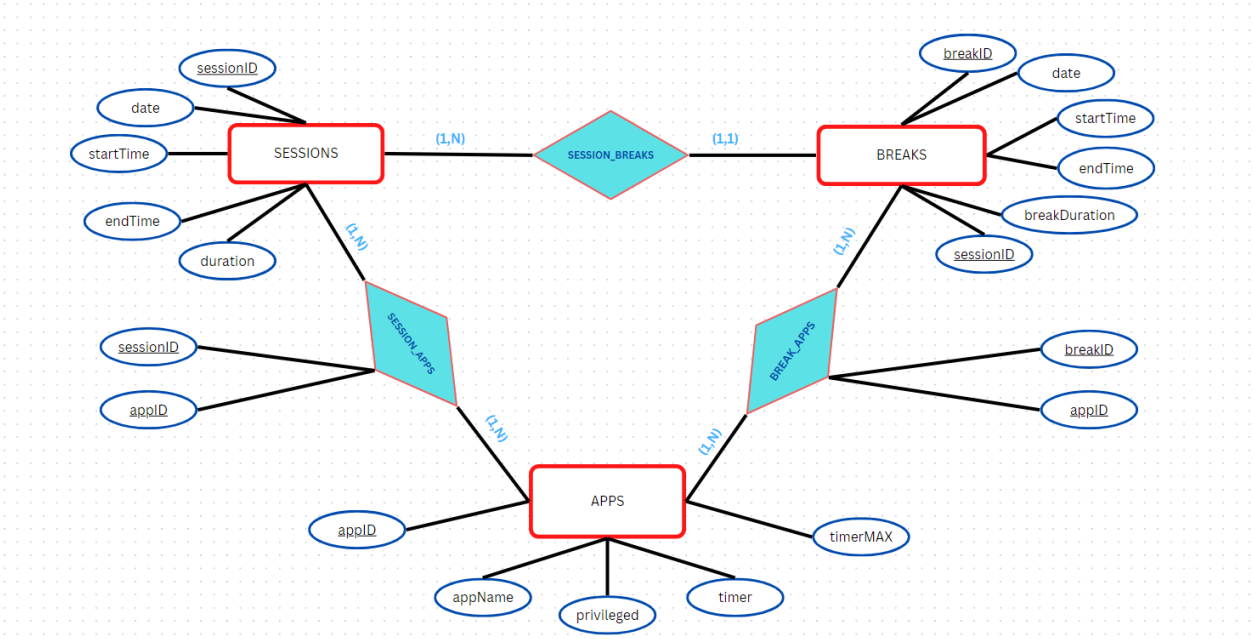
Tables:

- Sessions
- Breaks
- Apps

Relations:

- Session_breaks
- Session_apps
- Break_apps

2. ER diagram:


sessionID
date
startTime
SESSIONS
endTime
duration
(1,N)
SESSION_BREAKS
(1,1)
breakID
date
startTime
endTime
breakDuration
sessionID
BREAKS
(1,N)
(1,N)
SESSION_APPS
sessionID
appID
(1,N)
BREAK_APPS
breakID
appID
(1,N)
APPS
appID
appName
privileged
timer
timerMAX

3. **Relational schema:**

1. Sessions:

A history of all sessions

- sessionID (pk)
- date
- startTime
- endTime
- duration

2. Breaks:

A history of all breaks

- breakID (pk)
- date
- startTime
- endTime
- breakDuration
- sessionID (fk)

3. Apps:

A list of all apps on device

- appID (pk)
- appName
- privileged (yes/no)
- timer (if not privileged)
- timerMAX (if not privileged)

Relationship Descriptions:

4. Session_apps: (M:N)

Apps allowed during focus mode session. i.e the app(s) you had to focus on

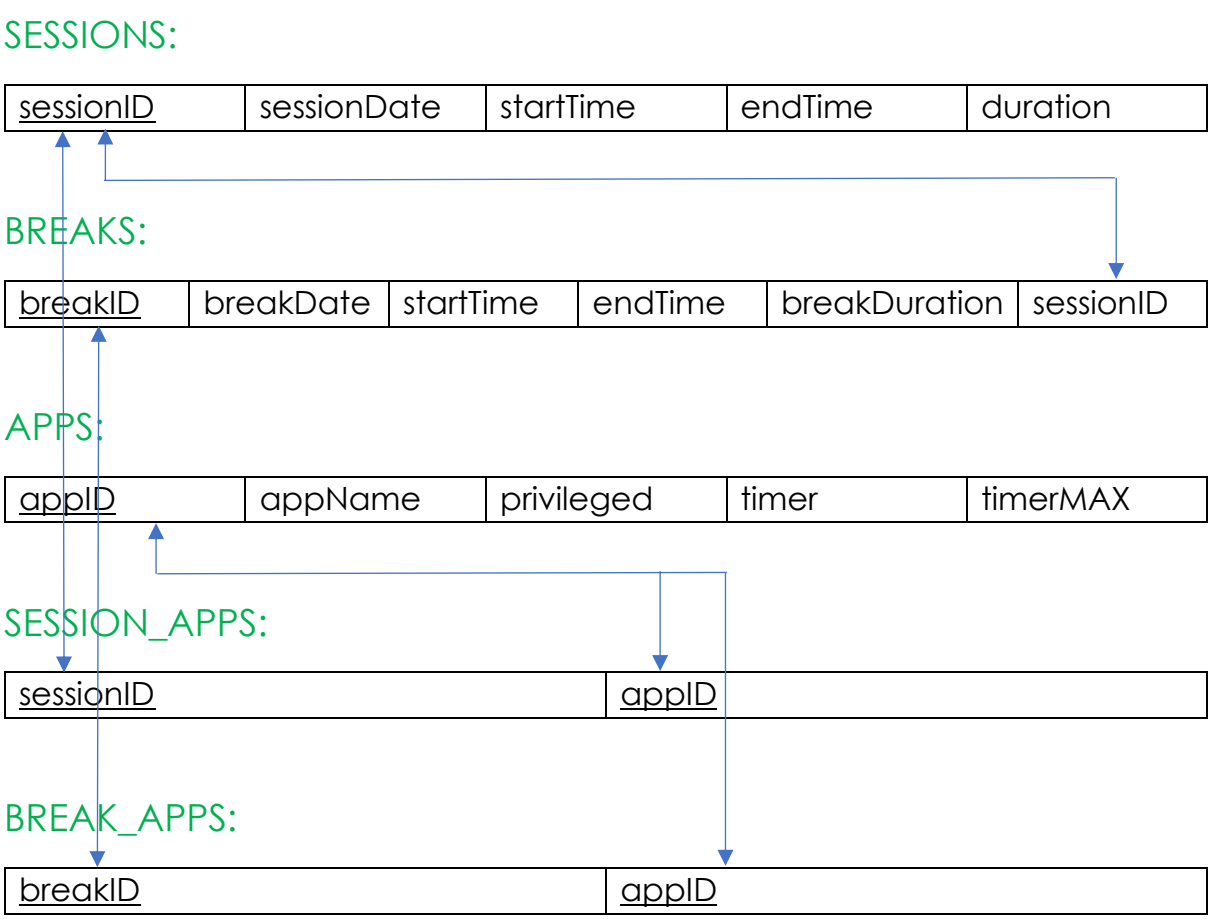
- sessionID
- appID

5. Break_apps: (M:N)

Apps used during breaks

- breakID
- appID

Relational Schema Diagram:



4. DDL statements:

create_tables.sql **file:**

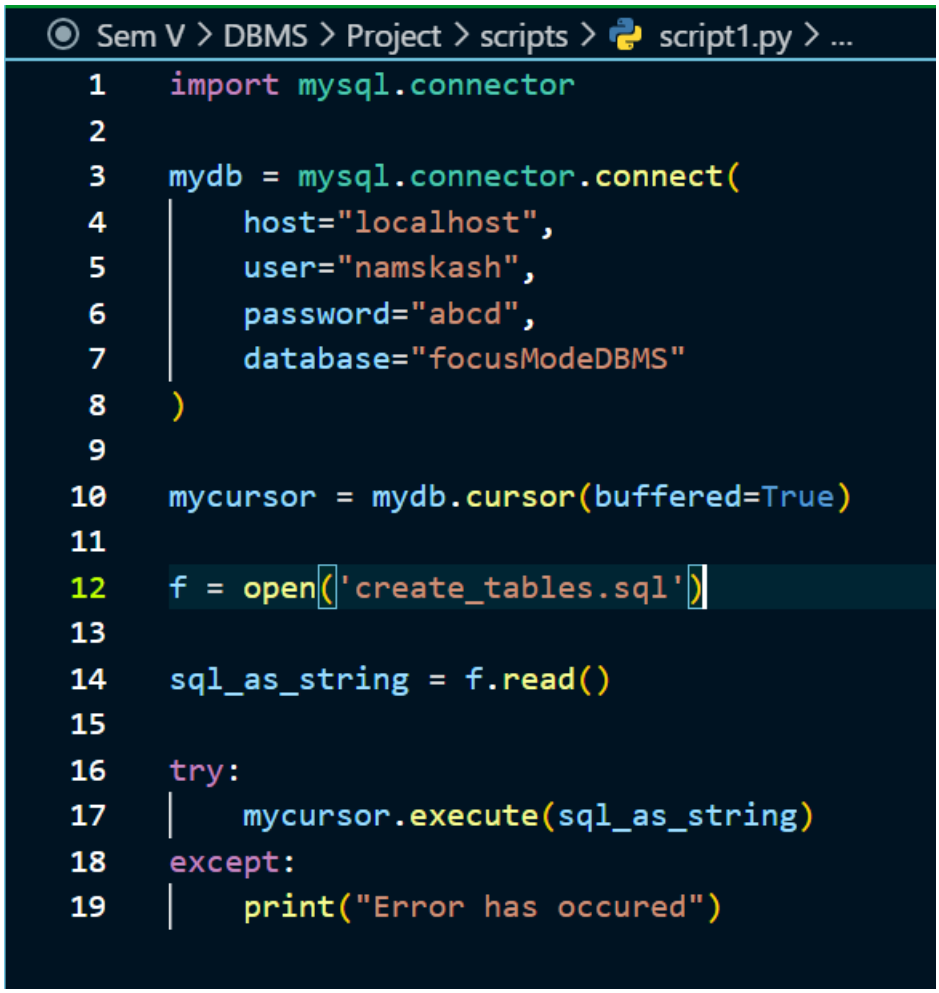
```
drop database focusModeDBMS;
create database focusModeDBMS;
use focusModeDBMS;

create table SESSIONS (
    sessionID varchar(10) not null,
    sessionDate date not null,
    startTime time not null,
    endTime time not null,
    duration varchar(20),
    primary key (sessionID)
);

create table BREAKS (
    breakID varchar(10) not null,
    breakDate date not null,
    startTime time not null,
    endTime time not null,
    breakDuration varchar(20) not null,
    sessionID varchar(10) not null,
    primary key (breakID),
    foreign key (sessionID) references SESSIONS(sessionID)
);

create table APPS (
    appID varchar(10) not null,
    appName varchar(50) not null,
    privileged bit not null,
    timer varchar(20),
    timerMAX varchar(20),
    primary key (appID)
);

create table SESSION_APPS (
    sessionID varchar(10) not null,
    appID varchar(10) not null,
    primary key (sessionID, appID),
    foreign key (sessionID) references SESSIONS (sessionID),
    foreign key (appID) references APPS (appID)
);

create table BREAK_APPS (
    breakID varchar(10) not null,
    appID varchar(10) not null,
    primary key (breakID, appID),
```

```
    foreign key (breakID) references BREAKS (breakID),
    foreign key (appID) references APPS (appID)
);

delimiter &&
create procedure getNEntries (in lim int)
begin
    select * from APPS limit lim;
end &&
delimiter;
```

5. **Populating data:**

Python scripts:

script1.py **file:**



script2.py **file:**

Sem V > DBMS > Project > scripts > script2.py > ...

```python
import mysql.connector
from datetime import date, timedelta
from random import randint,choices,choice,sample

def getTime(time):          # converts
    hours = int(time / 60)
    minutes = int(time % 60)
    seconds = int((time / 60) % 60)

    return str(hours) + ":" + str(minutes) + ":" + str(seconds)


mydb = mysql.connector.connect(
    host="localhost",
    user="namskash",
    password="abcd",
    database="focusModeDBMS"
)

mycursor = mydb.cursor(buffered=True)

# SESSIONS + BREAKS
date1, date2 = date(2021,11,4), date.today()
dates = [date1]
while(date1 <= date2):
    date1 += timedelta(days = randint(0,4))
    dates.append(date1)

dates = list(choices(dates,k=100))  # k=number of dates needed
#print(len(dates))

sessions = 1
breaks = 1
```

```python
# SESSIONS + BREAKS
date1, date2 = date(2021,11,4), date.today()
dates = [date1]
while(date1 <= date2):
    date1 += timedelta(days = randint(0,4))
    dates.append(date1)

dates = list(choices(dates,k=100))  # k=number of dates needed
#print(len(dates))

sessions = 1
breaks = 1

for i in range(100):    # entry into
    sessionID = "SES" + "0" * (3 - len(str(sessions))) + str(sessions)
    sessions += 1

    sessionDate = dates[i]
    sessionStartTime = randint(30,1410) # 1440 is midnight in minutes
    sessionEndTime = randint(sessionStartTime + 30,sessionStartTime + 180)  # half hour to 3 hours
    sessionDuration = sessionEndTime - sessionStartTime
    mycursor.execute("insert into SESSIONS values (%s,%s,%s,%s,%s)",(sessionID,sessionDate,getTime(sessionStartTime),getTime(sessionEndTime),sessionDuration

    breakID = "BRK" + "0" * (3 - len(str(breaks))) + str(breaks)
    breaks += 1

    breakDate = dates[i]    # same as session
    breakStartTime = randint(sessionStartTime + 20,sessionEndTime - 10)
    breakEndTime = randint(breakStartTime + 5,sessionEndTime)
    breakDuration = breakEndTime - breakStartTime
    mycursor.execute("insert into BREAKS values (%s,%s,%s,%s,%s,%s)",(breakID,breakDate,getTime(breakStartTime),getTime(breakEndTime),breakDuration,sessionI

    for j in range(randint(1,5)):
        try:
            breakID = "BRK" + "0" * (3 - len(str(breaks))) + str(breaks)
            breaks += 1
            breakStartTime = randint(breakEndTime + 5,sessionEndTime - 30)
            breakEndTime = randint(breakStartTime + 5,sessionEndTime)
            breakDuration = breakEndTime - breakStartTime
            mycursor.execute("insert into BREAKS values (%s,%s,%s,%s,%s,%s)",(breakID,breakDate,getTime(breakStartTime),getTime(breakEndTime),breakDuration,
        except:
            pass
```

```python
# APPS
sessionAppList = ['MS Word','MS PowerPoint','MS Excel','VS Code','Eclipse','Oracle VirtualBox','Ubuntu 22.04 VM']
breakAppList = ['Google Chrome','YouTube','Prime Video','Netflix','Solitaire','Hangman']
timers = ["30","45"]
timersMAX = ["45","60"]

for i in range(len(sessionAppList)):
    appID = "APP" + "0" * (3 - len(str(i+1))) + str(i+1)
    mycursor.execute("insert into APPS values (%s,%s,%s,%s,%s)",(appID,sessionAppList[i],1,None,None))

id = len(sessionAppList) + 1
for i in range(len(breakAppList)):
    appID = "APP" + "0" * (3 - len(str(i+id))) + str(i+id)
    mycursor.execute("insert into APPS values (%s,%s,%s,%s,%s)",(appID,breakAppList[i],0,choice(timers) + " mins",cho

mydb.commit()
```

```python
# SESSION_APPS
mycursor.execute("select sessionID from SESSIONS")
temp = mycursor.fetchall()
sessionIDs = []
for i in temp:
    sessionIDs.append(i[0]) #// to convert from tuple(tuple()) to a normal list

mycursor.execute("select appID from APPS where privileged = 0x01")
temp = mycursor.fetchall()
sessionAppIDs = []
for i in temp:
    sessionAppIDs.append(i[0])  #// to convert from tuple(tuple()) to a normal list

for i in range(len(sessionIDs)):
    apps_per_session = randint(1,4)
    temp = set(sample(sessionAppIDs,apps_per_session))  # n random apps, unique

    for j in temp:
        mycursor.execute("insert into SESSION_APPS values (%s,%s)",(sessionIDs[i],j))
```

```
104  # BREAK_APPS
105  mycursor.execute("select breakID from BREAKS")
106  temp = mycursor.fetchall()
107  breakIDs = []
108  for i in temp:
109      breakIDs.append(i[0])   #// to convert from tuple(tuple()) to a normal list
110
111  mycursor.execute("select appID from APPS where privileged = 0x00")
112  temp = mycursor.fetchall()
113  breakAppIDs = []
114  for i in temp:
115      breakAppIDs.append(i[0])    #// to convert from tuple(tuple()) to a normal list
116
117  for i in range(len(breakIDs)):
118      apps_per_break = randint(1,4)
119      temp = set(sample(breakAppIDs,apps_per_break))  # n random apps, unique
120
121      for j in temp:
122          mycursor.execute("insert into BREAK_APPS values (%s,%s)",(breakIDs[i],j))
123
124  mydb.commit()
125
```

6. JOIN queries:

Objective: Display the past sessions page.

```
145     # Fill into table
146     query = """
147         select s.*, round (duration / (duration + sum(breakDuration)), 3)
148         from SESSIONS s join BREAKS b
149         where s.sessionID = b.sessionID group by b.sessionID """
150     mycursor.execute(query)
151     temp = mycursor.fetchall()
152     for i in range(len(temp)):
153         session.insert('','end',values=temp[i])
```

Objective: Get sessions and all apps used in it.

```
mysql> select *
    -> from sessions natural join session_apps;
+-----------+-------------+-----------+----------+----------+---------+
| sessionID | sessionDate | startTime | endTime  | duration | appID   |
+-----------+-------------+-----------+----------+----------+---------+
| SES001    | 2022-09-07  | 03:44:03  | 06:31:06 | 167      | APP004  |
| SES002    | 2022-04-16  | 20:48:20  | 22:15:22 | 87       | APP001  |
| SES002    | 2022-04-16  | 20:48:20  | 22:15:22 | 87       | APP003  |
| SES002    | 2022-04-16  | 20:48:20  | 22:15:22 | 87       | APP005  |
| SES002    | 2022-04-16  | 20:48:20  | 22:15:22 | 87       | APP006  |
| SES003    | 2022-10-21  | 09:32:09  | 12:32:12 | 180      | APP004  |
| SES004    | 2022-10-22  | 13:46:13  | 16:05:16 | 139      | APP001  |
| SES004    | 2022-10-22  | 13:46:13  | 16:05:16 | 139      | APP002  |
| SES004    | 2022-10-22  | 13:46:13  | 16:05:16 | 139      | APP003  |
| SES005    | 2022-07-19  | 02:29:02  | 04:45:04 | 136      | APP003  |
| SES005    | 2022-07-19  | 02:29:02  | 04:45:04 | 136      | APP004  |
| SES005    | 2022-07-19  | 02:29:02  | 04:45:04 | 136      | APP005  |
| SES006    | 2022-06-02  | 15:57:15  | 17:33:17 | 96       | APP003  |
| SES006    | 2022-06-02  | 15:57:15  | 17:33:17 | 96       | APP004  |
| SES006    | 2022-06-02  | 15:57:15  | 17:33:17 | 96       | APP006  |
| SES006    | 2022-06-02  | 15:57:15  | 17:33:17 | 96       | APP007  |
| SES007    | 2022-09-19  | 12:11:12  | 13:46:13 | 95       | APP001  |
| SES007    | 2022-09-19  | 12:11:12  | 13:46:13 | 95       | APP004  |
| SES007    | 2022-09-19  | 12:11:12  | 13:46:13 | 95       | APP005  |
| SES007    | 2022-09-19  | 12:11:12  | 13:46:13 | 95       | APP006  |
| SES008    | 2022-05-06  | 12:59:12  | 15:50:15 | 171      | APP003  |
| SES008    | 2022-05-06  | 12:59:12  | 15:50:15 | 171      | APP005  |
| SES008    | 2022-05-06  | 12:59:12  | 15:50:15 | 171      | APP007  |
```

Objective: Get sessions and all apps used using inner join.

```
mysql> select *
    -> from sessions inner join session_apps
    -> where sessions.sessionID = session_apps.sessionID;
+-----------+-------------+-----------+----------+----------+-----------+--------+
| sessionID | sessionDate | startTime | endTime  | duration | sessionID | appID  |
+-----------+-------------+-----------+----------+----------+-----------+--------+
| SES001    | 2022-09-07  | 03:44:03  | 06:31:06 | 167      | SES001    | APP004 |
| SES002    | 2022-04-16  | 20:48:20  | 22:15:22 | 87       | SES002    | APP001 |
| SES002    | 2022-04-16  | 20:48:20  | 22:15:22 | 87       | SES002    | APP003 |
| SES002    | 2022-04-16  | 20:48:20  | 22:15:22 | 87       | SES002    | APP005 |
| SES002    | 2022-04-16  | 20:48:20  | 22:15:22 | 87       | SES002    | APP006 |
| SES003    | 2022-10-21  | 09:32:09  | 12:32:12 | 180      | SES003    | APP004 |
| SES004    | 2022-10-22  | 13:46:13  | 16:05:16 | 139      | SES004    | APP001 |
| SES004    | 2022-10-22  | 13:46:13  | 16:05:16 | 139      | SES004    | APP002 |
| SES004    | 2022-10-22  | 13:46:13  | 16:05:16 | 139      | SES004    | APP003 |
| SES005    | 2022-07-19  | 02:29:02  | 04:45:04 | 136      | SES005    | APP003 |
| SES005    | 2022-07-19  | 02:29:02  | 04:45:04 | 136      | SES005    | APP004 |
| SES005    | 2022-07-19  | 02:29:02  | 04:45:04 | 136      | SES005    | APP005 |
| SES006    | 2022-06-02  | 15:57:15  | 17:33:17 | 96       | SES006    | APP003 |
| SES006    | 2022-06-02  | 15:57:15  | 17:33:17 | 96       | SES006    | APP004 |
| SES006    | 2022-06-02  | 15:57:15  | 17:33:17 | 96       | SES006    | APP006 |
| SES006    | 2022-06-02  | 15:57:15  | 17:33:17 | 96       | SES006    | APP007 |
| SES007    | 2022-09-19  | 12:11:12  | 13:46:13 | 95       | SES007    | APP001 |
| SES007    | 2022-09-19  | 12:11:12  | 13:46:13 | 95       | SES007    | APP004 |
| SES007    | 2022-09-19  | 12:11:12  | 13:46:13 | 95       | SES007    | APP005 |
| SES007    | 2022-09-19  | 12:11:12  | 13:46:13 | 95       | SES007    | APP006 |
| SES008    | 2022-05-06  | 12:59:12  | 15:50:15 | 171      | SES008    | APP003 |
| SES008    | 2022-05-06  | 12:59:12  | 15:50:15 | 171      | SES008    | APP005 |
| SES008    | 2022-05-06  | 12:59:12  | 15:50:15 | 171      | SES008    | APP007 |
| SES009    | 2022-02-13  | 04:48:04  | 07:34:07 | 166      | SES009    | APP004 |
| SES009    | 2022-02-13  | 04:48:04  | 07:34:07 | 166      | SES009    | APP005 |
| SES009    | 2022-02-13  | 04:48:04  | 07:34:07 | 166      | SES009    | APP006 |
| SES009    | 2022-02-13  | 04:48:04  | 07:34:07 | 166      | SES009    | APP007 |
| SES010    | 2022-02-08  | 01:00:01  | 02:58:02 | 118      | SES010    | APP002 |
```

Objective: Get 10 breakIDs and all apps used in it.

```
mysql> select *
    -> from BREAKS b natural join break_apps a
    -> limit 10;
+---------+------------+-----------+----------+---------------+-----------+--------+
| breakID | breakDate  | startTime | endTime  | breakDuration | sessionID | appID  |
+---------+------------+-----------+----------+---------------+-----------+--------+
| BRK001  | 2022-09-07 | 04:16:04  | 04:54:04 | 38            | SES001    | APP009 |
| BRK001  | 2022-09-07 | 04:16:04  | 04:54:04 | 38            | SES001    | APP012 |
| BRK002  | 2022-09-07 | 05:57:05  | 06:07:06 | 10            | SES001    | APP009 |
| BRK002  | 2022-09-07 | 05:57:05  | 06:07:06 | 10            | SES001    | APP011 |
| BRK002  | 2022-09-07 | 05:57:05  | 06:07:06 | 10            | SES001    | APP013 |
| BRK004  | 2022-04-16 | 21:28:21  | 21:59:21 | 31            | SES002    | APP009 |
| BRK004  | 2022-04-16 | 21:28:21  | 21:59:21 | 31            | SES002    | APP010 |
| BRK004  | 2022-04-16 | 21:28:21  | 21:59:21 | 31            | SES002    | APP011 |
| BRK004  | 2022-04-16 | 21:28:21  | 21:59:21 | 31            | SES002    | APP012 |
| BRK008  | 2022-10-21 | 12:22:12  | 12:27:12 | 5             | SES003    | APP008 |
+---------+------------+-----------+----------+---------------+-----------+--------+
10 rows in set (0.00 sec)
```

1. AGGREGATE functions:

Objective: Get sum of all breakDurations.

```
144
145     # Fill into table
146     query = """
147         select s.*, round (duration / (duration + sum(breakDuration)), 3)
148         from SESSIONS s join BREAKS b
149         where s.sessionID = b.sessionID group by b.sessionID """
150     mycursor.execute(query)
151     temp = mycursor.fetchall()
152     for i in range(len(temp)):
153         session.insert('','end',values=temp[i])
```

Objective: Count all appIDs from session_apps

```
86      # Get appIDs and number of breaks the app was used in
87      mycursor.execute("select appID,count(appID) from SESSION_APPS group by appID order by appID")
88      temp = mycursor.fetchall()
89      appIDs = []
90      appBreakCount = []
91
92      for i in temp:
93          appIDs.append(i[0])
94          appBreakCount.append(i[1])
```

Objective: Count the number of apps in the top 5 sessions in terms of apps used.

```
mysql> select sessionID, count(*) from session_apps group by sessionID order by count(*) desc limit 5;
+-----------+----------+
| sessionID | count(*) |
+-----------+----------+
| SES009    |        4 |
| SES002    |        4 |
| SES007    |        4 |
| SES100    |        4 |
| SES010    |        4 |
+-----------+----------+
5 rows in set (0.00 sec)
```

Objective: Count the number of apps in the bottom 5 breaks in terms of apps used.

```
mysql> select breakID, count(*) from break_apps group by breakID order by count(*) limit 5;
+---------+----------+
| breakID | count(*) |
+---------+----------+
| BRK048  |        1 |
| BRK036  |        1 |
| BRK018  |        1 |
| BRK022  |        1 |
| BRK014  |        1 |
+---------+----------+
5 rows in set (0.00 sec)
```

2. SET operators:

Objective: Get a union of all sessionDurations and breakDurations.

```
158     query = """
159         select sum(breakDuration)
160         from BREAKS
161         union
162         select sum(duration)
163         from SESSIONS """
164     mycursor.execute(query)
165
```

Objective: Get all session IDs in both sessions and session_apps.

```
mysql> select sessionID from sessions union all select distinct appID from session_apps limit 5;
+-----------+
| sessionID |
+-----------+
| SES001    |
| SES002    |
| SES003    |
| SES004    |
| SES005    |
+-----------+
5 rows in set (0.00 sec)
```

Objective: Get a union of all session IDs and break IDs.

```
mysql> select sessionID as session_break
    -> from SESSIONS
    -> union
    -> select breakID
    -> from BREAKS;
+---------------+
| session_break |
+---------------+
| SES001        |
| SES002        |
| SES003        |
| SES004        |
| SES005        |
| SES006        |
| SES007        |
| SES008        |
| SES009        |
```

Objective: Get all app IDs from both APPS and break_apps.

```
mysql> select appID from APPS
    -> union all
    -> select appID from break_apps;
+--------+
| appID  |
+--------+
| APP001 |
| APP002 |
| APP003 |
| APP004 |
| APP005 |
| APP006 |
| APP007 |
| APP008 |
| APP009 |
| APP010 |
| APP011 |
| APP012 |
```

Update:

```
51      try:
52          mycursor.execute("update APPS set timer = %s, timerMAX = %s where appID = %s",(timer,timerMAX,appIDentry))
53          # SUCCESS!
54          messagebox.showinfo("Success!!","Timers updated for appID: %s"%appIDentry)
55          global appTimers
56          appTimers.delete(*appTimers.get_children())
57
58          mydb.commit()
59          #% Fill table again:
60          mycursor.execute("select appID,appName,timer,timerMAX from APPS where privileged = 0x00")
61          temp = mycursor.fetchall()
62          for i in temp:
63              appTimers.insert('','end',values=i)
```

3. Functions and procedures:

Function :

Objective: To display "privileged" or "not-privileged" instead 0x01 or 0x00.

```
delimiter $$
create function appType(privileged int)
returns varchar(20)
deterministic
begin
    declare typeOfApp varchar(20);
    if privileged > 0 then
        set typeOfApp = 'privileged';

    elseif privileged < 1 then
        set typeOfApp = 'non-privileged';

    end if;

    return (typeOfApp);
end $$
delimiter
```

Without function:

```
mysql> select appID,privileged from apps;
+--------+------------------------+
| appID  | privileged             |
+--------+------------------------+
| APP001 | 0x01                   |
| APP002 | 0x01                   |
| APP003 | 0x01                   |
| APP004 | 0x01                   |
| APP005 | 0x01                   |
| APP006 | 0x01                   |
| APP007 | 0x01                   |
| APP008 | 0x00                   |
| APP009 | 0x00                   |
| APP010 | 0x00                   |
| APP011 | 0x00                   |
| APP012 | 0x00                   |
| APP013 | 0x00                   |
+--------+------------------------+
13 rows in set (0.00 sec)
```

With function:

```
mysql> select appID,appType(privileged) from apps;
+--------+---------------------+
| appID  | appType(privileged) |
+--------+---------------------+
| APP001 | privileged          |
| APP002 | privileged          |
| APP003 | privileged          |
| APP004 | privileged          |
| APP005 | privileged          |
| APP006 | privileged          |
| APP007 | privileged          |
| APP008 | non-privileged      |
| APP009 | non-privileged      |
| APP010 | non-privileged      |
| APP011 | non-privileged      |
| APP012 | non-privileged      |
| APP013 | non-privileged      |
+--------+---------------------+
13 rows in set (0.00 sec)
```

Procedure:

Objective: Get N entries from APPS, n passed as argument.

```
61  delimiter &&
62  create procedure getNEntries (in lim int)
63  begin
64      select * from APPS limit lim;
65  end &&
66  delimiter;
```

```
mysql> call getNEntries(5);
+--------+--------------+-------------------------+-------+----------+
| appID  | appName      | privileged              | timer | timerMAX |
+--------+--------------+-------------------------+-------+----------+
| APP001 | MS Word      | 0x01                    | NULL  | NULL     |
| APP002 | MS PowerPoint | 0x01                   | NULL  | NULL     |
| APP003 | MS Excel     | 0x01                    | NULL  | NULL     |
| APP004 | VS Code      | 0x01                    | NULL  | NULL     |
| APP005 | Eclipse      | 0x01                    | NULL  | NULL     |
+--------+--------------+-------------------------+-------+----------+
5 rows in set (0.00 sec)

Query OK, 0 rows affected (0.02 sec)
```

4. Triggers and cursors:

Objective: To automatically update variable sum when a new row is inserted.

```
91    create trigger sumDuration before insert on BREAKS
92    |    for each row set @sum = @sum + NEW.breakDuration;
```

Objective: To get all the values of privileged bits.

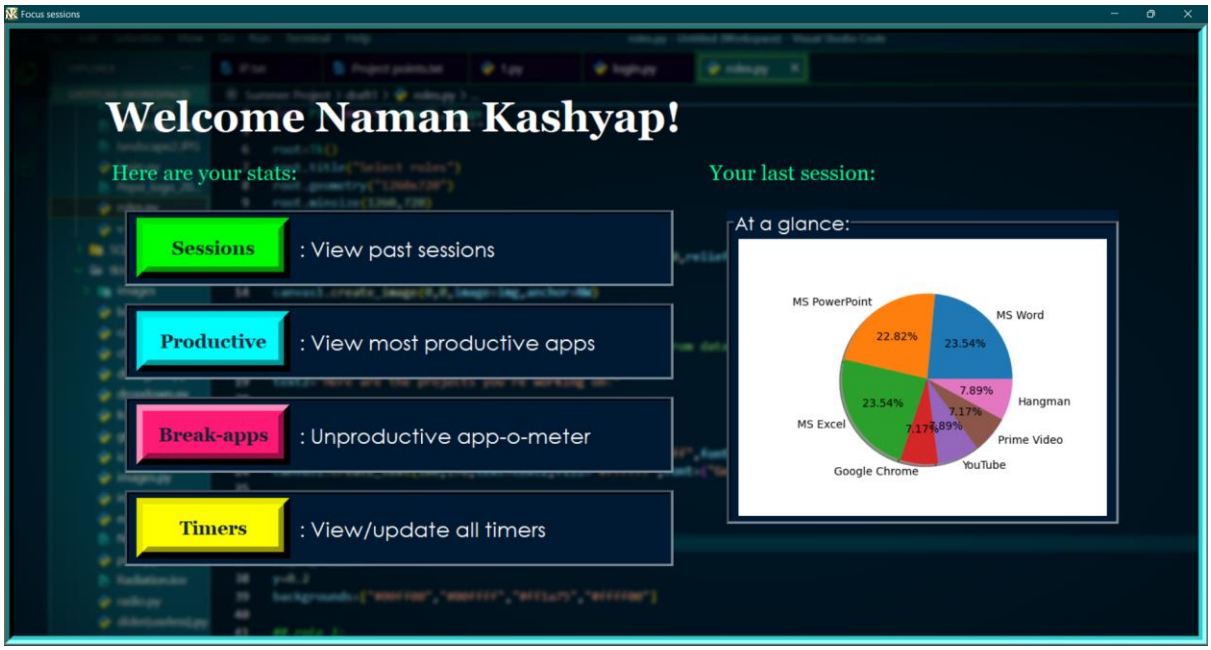
```
mysql> delimiter $$
mysql> create procedure list_apps (inout namelist varchar(100))
    -> begin
    -> declare finished integer default 0;
    -> declare a_name varchar(100) default "";
    ->
    -> /*declare cursor*/
    -> declare stud_cursor
    -> cursor for
    -> select privileged from APPS;
    ->
    -> /*declare not found handler*/
    -> declare continue handler
    -> for not found set finished = 1;
    ->
    -> /*open cursor*/
    -> open stud_cursor;
    ->
    -> /*iterate*/
    -> get_list: LOOP
    -> fetch stud_cursor into a_name;
    -> if finished = 1 then
    -> leave get_list;
    -> end if;
    ->
    -> /*build list of apps*/
    -> set namelist = concat(a_name,";",namelist);
    -> end loop get_list;
    -> close stud_cursor;
    -> end $$
Query OK, 0 rows affected (0.01 sec)

mysql> delimiter ;
mysql> SET @name_list ="";
Query OK, 0 rows affected (0.00 sec)

mysql> CALL list_apps(@name_list);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @name_list;
+---------------------------+
| @name_list                |
+---------------------------+
| 0;0;0;0;0;0;1;1;1;1;1;1;1; |
+---------------------------+
1 row in set (0.00 sec)
```

```
10    mydb = mysql.connector.connect(
11        host="localhost",
12        user="namskash",
13        password="abcd",
14        database="focusModeDBMS"
15    )
16
17    mycursor = mydb.cursor(buffered=True)
```

5. Higher Level Programming (GUI):

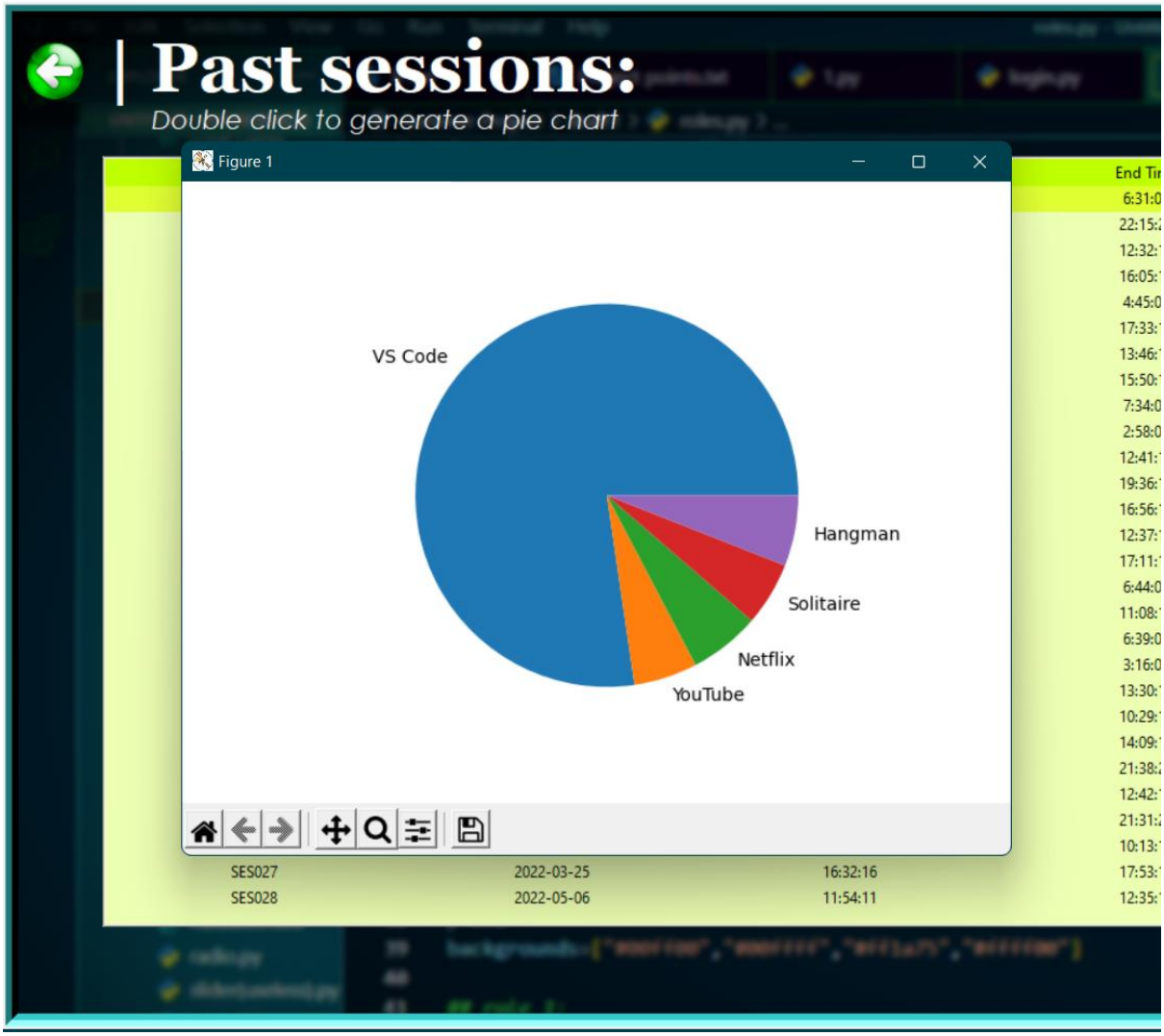
Front-end: Python tkinter

Back-end: MySQL

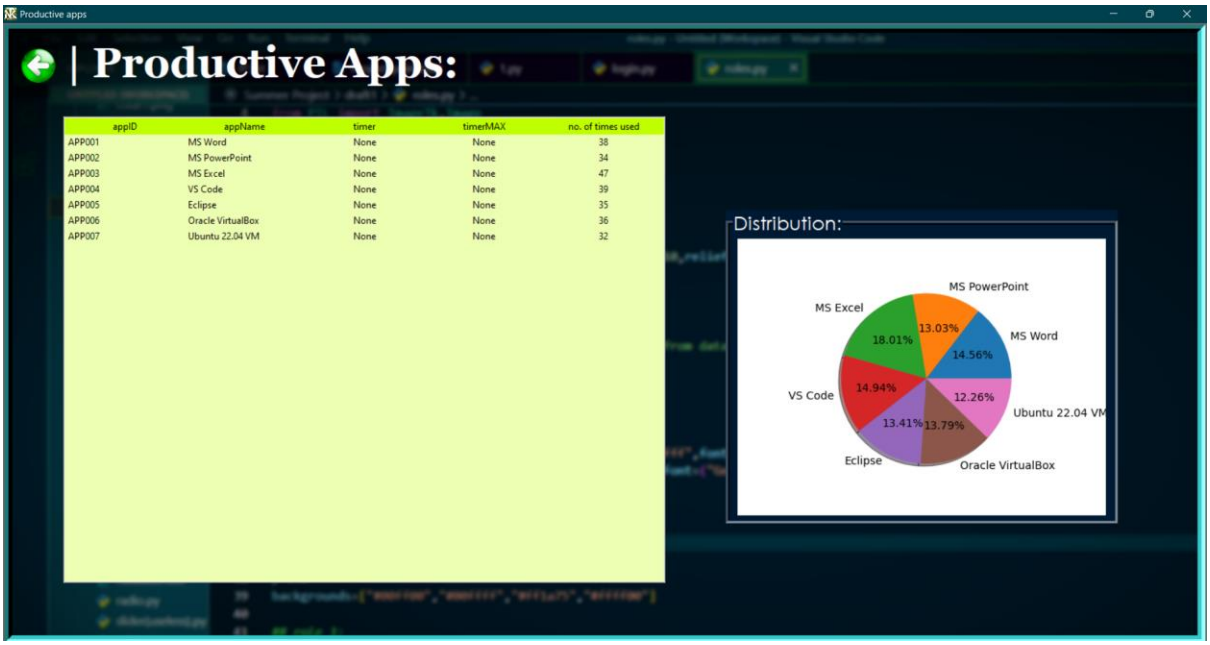


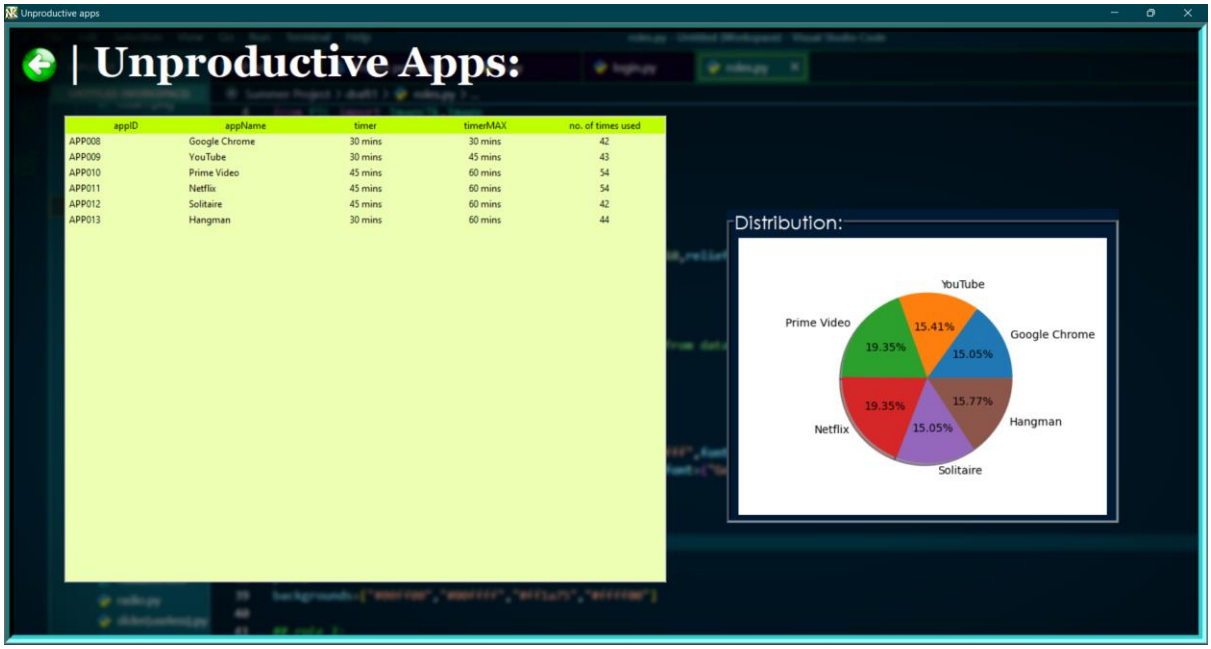
sessionID	Date	Start time	End Time	Duration	Efficiency
SES001	2022-09-07	3:44:03	6:31:06	167	0.777
SES002	2022-04-16	20:48:20	22:15:22	87	0.737
SES003	2022-10-21	9:32:09	12:32:12	180	0.973
SES004	2022-10-22	13:46:13	16:05:16	139	0.671
SES005	2022-07-19	2:29:02	4:45:04	136	0.642
SES006	2022-06-02	15:57:15	17:33:17	96	0.85
SES007	2022-09-19	12:11:12	13:46:13	95	0.748
SES008	2022-05-06	12:59:12	15:50:15	171	0.842
SES009	2022-02-13	4:48:04	7:34:07	166	0.79
SES010	2022-02-08	1:00:01	2:58:02	118	0.578
SES011	2022-03-30	10:41:10	12:41:12	120	0.583
SES012	2021-12-24	16:36:16	19:36:19	180	0.643
SES013	2021-11-11	16:02:16	16:56:16	54	0.794
SES014	2022-06-08	11:37:11	12:37:12	60	0.833
SES015	2022-07-04	14:54:14	17:11:17	137	0.628
SES016	2022-02-08	4:24:04	6:44:06	140	0.609
SES017	2021-12-02	9:36:09	11:08:11	92	0.893
SES018	2021-12-30	5:13:05	6:39:06	86	0.705
SES019	2021-12-17	2:41:02	3:16:03	35	0.7
SES020	2022-01-12	10:38:10	13:30:13	172	0.632
SES021	2022-07-31	9:25:09	10:29:10	64	0.842
SES022	2022-08-26	12:56:12	14:09:14	73	0.768
SES023	2022-03-07	19:04:19	21:38:21	154	0.922
SES024	2022-08-21	10:06:10	12:42:12	156	0.664
SES025	2022-10-04	19:27:19	21:31:21	124	0.747
SES026	2022-01-18	8:58:08	10:13:10	75	0.652
SES027	2022-03-25	16:32:16	17:53:17	81	0.871
SES028	2022-05-06	11:54:11	12:35:12	41	0.891

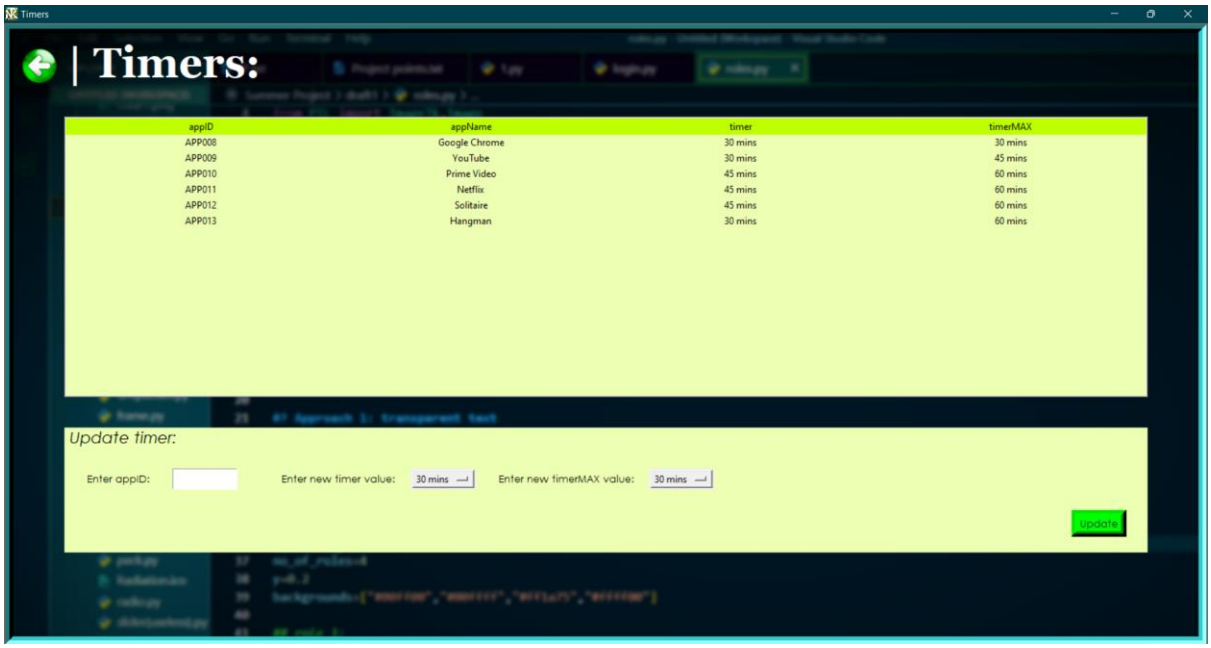
Past sessions

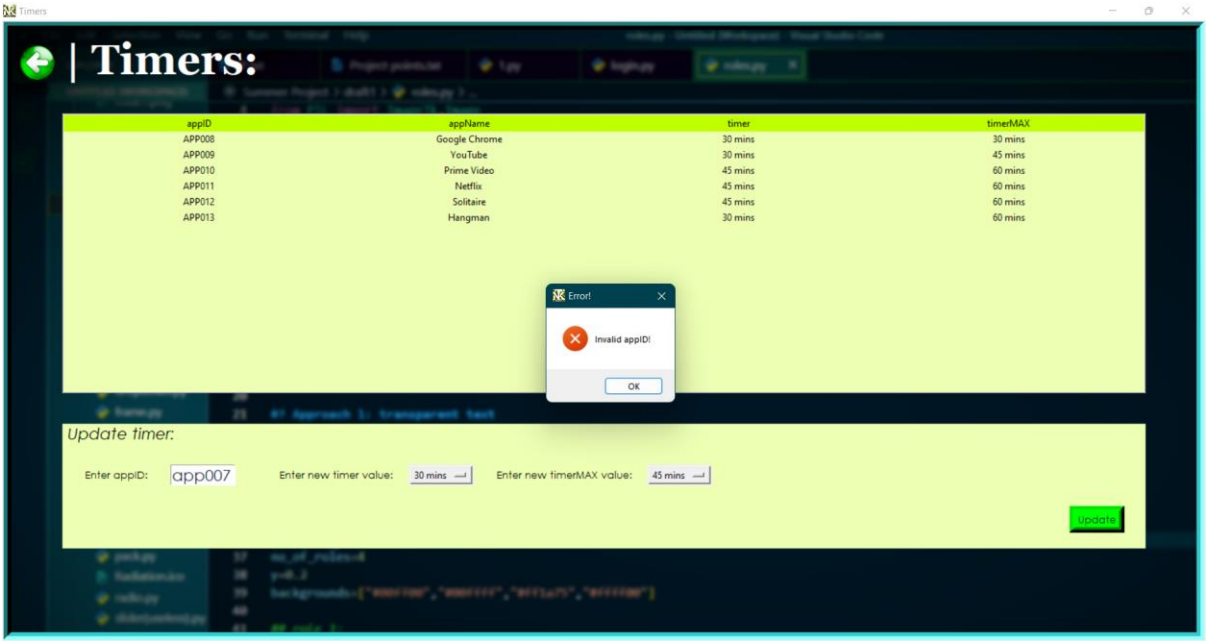


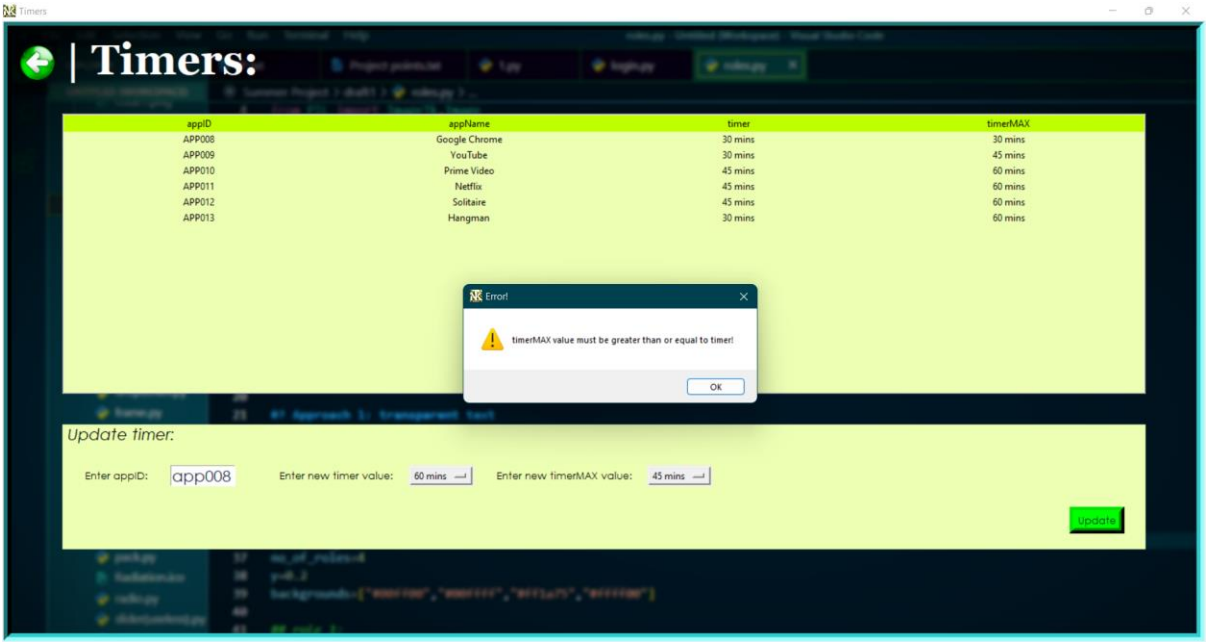
Productive apps

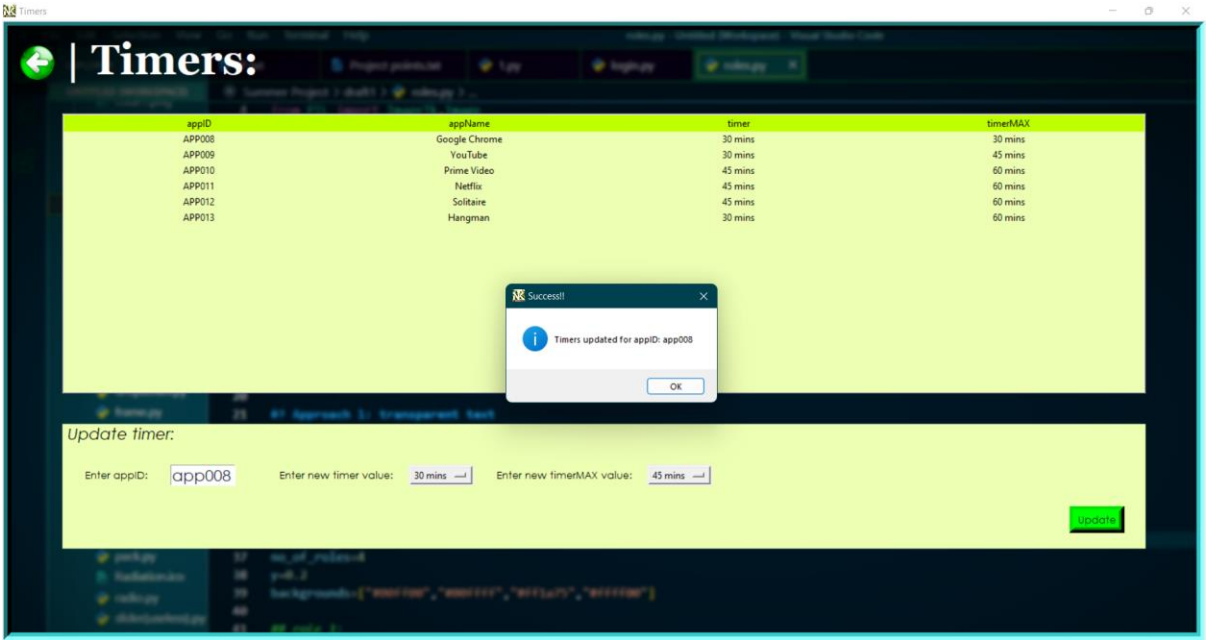


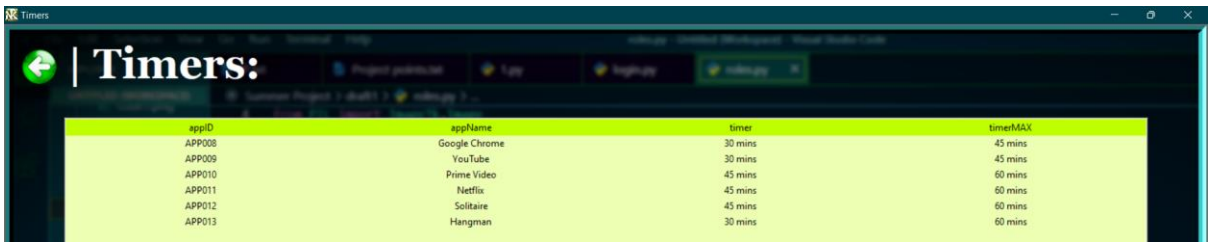

Unproductive apps
| Unproductive Apps:
appID appName timer timerMAX no. of times used
APP008 Google Chrome 30 mins 30 mins 42
APP009 YouTube 30 mins 45 mins 43
APP010 Prime Video 45 mins 60 mins 54
APP011 Netflix 45 mins 60 mins 54
APP012 Solitaire 45 mins 60 mins 42
APP013 Hangman 30 mins 60 mins 44
Distribution:
YouTube
Prime Video
Google Chrome
Netflix
Hangman
Solitaire
15.41%
15.05%
19.35%
19.35%
15.77%
15.05%


Timers
| Timers:
appID appName timer timerMAX
APP008 Google Chrome 30 mins 30 mins
APP009 YouTube 30 mins 45 mins
APP010 Prime Video 45 mins 60 mins
APP011 Netflix 45 mins 60 mins
APP012 Solitaire 45 mins 60 mins
APP013 Hangman 30 mins 60 mins
Update timer:
Enter appID:
Enter new timer value:
30 mins
Enter new timerMAX value:
30 mins
update


Timers
| Timers:
appID appName timer timerMAX
APP008 Google Chrome 30 mins 30 mins
APP009 YouTube 30 mins 45 mins
APP010 Prime Video 45 mins 60 mins
APP011 Netflix 45 mins 60 mins
APP012 Solitaire 45 mins 60 mins
APP013 Hangman 30 mins 60 mins
Error!
Invalid appID!
OK
Update timer:
Enter appID: app007
Enter new timer value: 30 mins
Enter new timerMAX value: 45 mins
update


Timers
| Timers:
appID appName timer timerMAX
APP008 Google Chrome 30 mins 30 mins
APP009 YouTube 30 mins 45 mins
APP010 Prime Video 45 mins 60 mins
APP011 Netflix 45 mins 60 mins
APP012 Solitaire 45 mins 60 mins
APP013 Hangman 30 mins 60 mins
Error!
timerMAX value must be greater than or equal to timer!
OK
Update timer:
Enter appID: app008
Enter new timer value: 60 mins
Enter new timerMAX value: 45 mins
update


Timers
| Timers:
appID appName timer timerMAX
APP008 Google Chrome 30 mins 30 mins
APP009 YouTube 30 mins 45 mins
APP010 Prime Video 45 mins 60 mins
APP011 Netflix 45 mins 60 mins
APP012 Solitaire 45 mins 60 mins
APP013 Hangman 30 mins 60 mins
Success!!
Timers updated for appID: app008
OK
Update timer:
Enter appID: app008
Enter new timer value: 30 mins
Enter new timerMAX value: 45 mins
update


Timers
| Timers:
appID appName timer timerMAX
APP008 Google Chrome 30 mins 45 mins
APP009 YouTube 30 mins 45 mins
APP010 Prime Video 45 mins 60 mins
APP011 Netflix 45 mins 60 mins
APP012 Solitaire 45 mins 60 mins
APP013 Hangman 30 mins 60 mins