```python
from kivy.core.window import Window
from kivy.properties import StringProperty
from kivy.app import App
from kivy.uix.image import Image
#from kivymd.uix.textfield import MDTextFieldRound
from kivy.uix.button import Button
from kivy.metrics import dp
from kivymd.app import MDApp
from kivymd.uix.card import MDCard
from kivymd.uix.label import MDLabel
from kivy.uix.widget import Widget
#from sql_functions import write_user,write_movement_data,get_user_data_with_email,update_user_data_with_email,get_movement_data,get_tasks,check_if_email_unique
from kivy.uix.screenmanager import Screen, NoTransition, CardTransition, ScreenManager
from kivy.lang import Builder
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.floatlayout import FloatLayout
from kivymd.uix.gridlayout import MDGridLayout
from kivymd.uix.floatlayout import MDFloatLayout
from kivymd.uix.behaviors import RoundedRectangularElevationBehavior
from kivymd.uix.button import MDRectangleFlatIconButton
#from kivymd.uix.textfield import MDTextFieldRect
from kivy.properties import ObjectProperty
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.input.providers.mouse import MouseMotionEvent
from kivymd.uix.menu import MDDropdownMenu
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFlatButton
from plyer import gravity,accelerometer,notification
from kivy.clock import Clock
from kivy.uix.switch import Switch
import random
# import ai_model
import pandas as pd
import numpy as np
import matplotlib as tf
# from ai_model import prediction
import keras
#from matplotlib import backend as K
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
from keras.layers import BatchNormalization
from keras.regularizers import L1L2
from keras import optimizers
from keras.models import load_model
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns
import os
import csv
from datetime import date, timedelta, datetime

#####################  SQL imports  ################################
import sqlite3
from sqlalchemy import create_engine, ForeignKey
```

```
57 from sqlalchemy import Column, Date, Integer, String
58 from sqlalchemy.ext.declarative import declarative_base
59 from sqlalchemy.orm import sessionmaker, relationship, backref
60
61
62 # os.remove('trainer.db')
63 engine = create_engine('sqlite:///trainer.db', echo=True)
64 Base = declarative_base()
65
66 Session = sessionmaker(bind=engine)
67 session = Session()
68 ################################  SQL Tables ###################################
69
70 class User(Base):
71     __tablename__ = "user"
72
73     email = Column(String, primary_key=True)
74     password = Column(String, nullable=False)
75     name = Column(String)
76     age = Column(Integer)
77     weight = Column(Integer)
78     uheight = Column(Integer)
79     gender = Column(String)
80     jp = Column(String)
81     tasks = relationship('Tasks', backref="user", uselist=False)
82
83     def __init__(self, email, password, name, age=0, weight=0,uheight=0,  gender=" ",
jp=" "):
84         self.email = email
85         self.password = password
86         self.name = name
87         self.age = age
88         self.weight = weight
89         self.uheight = uheight
90         self.gender = gender
91         self.jp = jp
92
93     def add_tasks(self, t1, t2, t3, t4, t5):
94         self.tasks = Tasks(t1, t2, t3, t4, t5)
95
96     def __repr__(self):
97         return "<User(email='%s', name='%s', age='%d', weight='%d',uheight=%d
gender='%d', task1='%s', task2='%s, task3='%s, task4='%s, task5='%s)>" % (self.email,
self.name, self.age, self.weight, self.uheight, self.gender, self.tasks.task1,
self.tasks.task2, self.tasks.task3, self.tasks.task4, self.tasks.task5)
98
99 class Tasks(Base):
100     __tablename__ = "tasks"
101
102     email = Column(String, ForeignKey(User.email), primary_key=True)
103     task1 = Column(String, nullable=False)
104     task2 = Column(String)
105     task3 = Column(String)
106     task4 = Column(String)
107     task5 = Column(String)
108
109     def __init__(self, task1, task2, task3, task4, task5):
110         self.task1 = task1
111         self.task2 = task2
112         self.task3 = task3
```

```
113         self.task4 = task4
114         self.task5 = task5
115 
116 class Motivation(Base):
117     __tablename__ = "motivationtasks"
118 
119     email = Column(String, ForeignKey(User.email), primary_key=True, nullable=False)
120     timestamp = Column(String, nullable=False, primary_key=True)
121     mvmntprofile = Column(Integer)
122 
123 
124     def __init__(self, email, timestamp, mvmntprofile):
125         self.email = email
126         self.timestamp = timestamp
127         self.mvmntprofile = mvmntprofile
128 
129 Base.metadata.create_all(engine)
130 ########################## Motivation
    Tasks#############################################################################
131 
132 taskm18=["Run 20 mins","Walk 5 mins", "Run 20 mins","Sit 10 mins","Run 6 mins"]
133 taskm24=["Run 15 mins","Walk 10 mins", "Run 15 mins","Sit 15 mins","Run 4 mins"]
134 taskm40=["Run 5 mins","Walk 15 mins", "Run 5 mins","Sit 20 mins","Run 2 mins"]
135 taskf18=["Run 20 mins","Walk 5 mins", "Run 19 mins","Sit 9 mins","Run 3 mins"]
136 taskf24=["Run 14 mins","Walk 5 mins", "Run 14 mins", "Sit 14 mins","Run 2 mins"]
137 taskf40=["Run 4 mins", "Walk 5 mins","Run 4 mins","Sit 19 mins","Run 1 mins"]
138 
139 ##########################################################################################
    ######################
140 
141 logged_in_user = None
142 logged_in_username = None
143 logged_in_user_age= 0
144 logged_in_weight = None
145 logged_in_user_gender=None
146 logged_in_height = None
147 logged_in_jp = None
148 
149 g_time_in_movement= 0
150 g_sensor_reachable = False
151 g_number_of_notifications_8h = 4
152 Sensor_values = []
153 dirname = os.path.dirname(__file__)
154 
155 ############################### AI MODEL
    ###################################################################
156 ACTIVITIES = {
157         0: 'sitting',
158         1: 'walking',
159         2: 'running',
160     }
161 
162 def loading_AI_model():
163     model = keras.models.load_model('C:/Users/Lenovo/Desktop/DIT AI Lectures/MSS-M-1
    Case Study Embedded Control Solutions (SS22)/FINAL Project/Models')
164     return model
165 
166 def predict(model,data):
167     predicted_model = model.predict([Sensor_values])
168     ai_values = ACTIVITIES[np.argmax(predicted_model)]
```

```
169     return ai_values
170 
171 m = loading_AI_model()
172   ##########################################################################################
173 
174 class LoginScreen(Screen):
175 
176     def loginBtn(self):
177         email = ObjectProperty(None)
178         password = ObjectProperty(None)
179         email_input = self.email.text
180         password_input = self.password.text
181         print(email_input)
182         print(password_input)
183         valid = session.query(User).filter(
184             User.email == email_input, User.password == password_input).count()
185         if valid == 0 or (email_input == '' or password_input == ''):
186             # if valid == 0:
187             print("Invalid email or password")
188         else:
189             result = session.query(User).filter(
190                 User.email == email_input, User.password == password_input)
191             for user in result:
192                 print(f"Welcome {user.name}!")
193                 global logged_in_user
194                 logged_in_user = user.email
195                 self.manager.current = "main"
196 
197 
198 class SignUpScreen(Screen):
199     def registerBtn(self):
200         email_input = self.email.text
201         password_input = self.password.text
202         password_confirmation_input = self.password_confirmation.text
203         username_input = self.username.text
204         age_input = self.age.text
205         weight_input = self.weight.text
206         height_input = self.uheight.text
207         gender_input = self.gender.text
208         jp_input = self.jp.text
209         age_input=int(age_input)
210         weight_input=int(weight_input)
211 
212         if gender_input != "male" or gender_input != "female":
213             print("Password unmatch!!")
214         if password_confirmation_input != password_input:
215             print("Password unmatch!!")
216 
217         else:
218             data = User(email_input, password_input, username_input, age_input,
    weight_input,height_input, gender_input, jp_input)
219 
220             if jp_input== "Software Engineer":
221                 if age_input<18 and  gender_input== "female":
222                     data.tasks = Tasks("Run 20 mins","Walk 5 mins","Sit 9 mins","Walk
    5 mins","Run 20 mins")
223                 elif age_input>=18 and  age_input<40 and  gender_input== "female":
224                     data.tasks = Tasks("Run 20 mins","Walk 5 mins","Sit 9 mins","Walk
    5 mins","Run 20 mins")
```

```
                elif age_input>40 and  gender_input== "female":
                    data.tasks = Tasks("Run 20 mins", "Walk 5 mins","Sit 9 mins","Walk 5 mins","Run 20 mins")
                elif age_input<18 and  gender_input== "male":
                    data.tasks = Tasks("Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5 mins","Run 20 mins")
                elif age_input>=18 and  age_input<40 and  gender_input== "male":
                    data.tasks = Tasks("Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5 mins","Run 20 mins")
                elif age_input>40 and  gender_input== "male":
                    data.tasks = Tasks("Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5 mins","Run 20 mins")

            else:
                if age_input<18 and  gender_input== "female":
                    data.tasks = Tasks(taskf18[0],taskf18[1],taskf18[2],taskf18[3],taskf18[4])
                elif age_input>=18 and  age_input<40 and  gender_input== "female":
                    data.tasks = Tasks(taskf24[0],taskf24[1],taskf24[2],taskf24[3],taskf24[4])
                elif age_input>40 and  gender_input== "female":
                    data.tasks = Tasks(taskf40[0],taskf40[1],taskf40[2],taskf40[3],taskf40[4])
                elif age_input<18 and  gender_input== "male":
                    data.tasks = Tasks(taskm18[0],taskm18[1],taskm18[2],taskm18[3],taskm18[4])
                elif age_input>=18 and  age_input<40 and  gender_input== "male":
                    data.tasks = Tasks(taskm24[0],taskm24[1],taskm24[2],taskm24[3],taskm24[4])
                elif age_input>40 and  gender_input== "male":
                    data.tasks = Tasks(taskm40[0],taskm40[1],taskm40[2],taskm40[3],taskm40[4])

            #data.tasks = Tasks('a','b','c','d','e')
            session.add(data)
            session.commit()

class TermsandCondition(Screen):
    pass

class MainScreen(Screen):

    event = ObjectProperty(None)
    event_notification = ObjectProperty(None)
    notification_interval = ObjectProperty(None)
    movement_profile = ObjectProperty(None)

    def on_pre_enter(self):
        self.callback()
        # self.get_info()

    def callback(self):
        global logged_in_user
        global logged_in_user_age
        global logged_in_user_gender
        global logged_in_jp

        result_user = session.query(User).filter(User.email == logged_in_user)

        for user in result_user:
```

```
            logged_in_user_age= user.age
            logged_in_user_gender= user.gender
            logged_in_username= user.name
            logged_in_jp= user.jp
        u_age=0
        u_gender=""
        u_jp=""
        mtasks=[]
        u_age=logged_in_user_age
        u_gender=logged_in_user_gender
        u_jp= logged_in_jp


        if u_jp== "Software Engineer":
            if u_gender == "male":
                if u_age <18:
                    mtask= ["Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5 mins","Run 20 mins"]
                if (u_age) > 18 and (u_age) < 40:
                    mtasks = ["Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5 mins","Run 20 mins"]
                else:
                    mtasks = ["Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5 mins","Run 20 mins"]

            elif u_gender == "female":
                if u_age <18:
                    mtask= ["Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5 mins","Run 20 mins"]
                if (u_age) > 18 and (u_age) < 40 :
                    mtasks = ["Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5 mins","Run 20 mins"]
                else:
                    mtasks = ["Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5 mins","Run 20 mins"]
        else:
            if u_gender == "male":
                if u_age <18:
                    mtask= ["Run 20 mins","Walk 5 mins", "Run 20 mins","Sit 10 mins","Run 6 mins"]
                if (u_age) > 18 and (u_age) < 40:
                    mtasks = ["Run 15 mins","Walk 10 mins", "Run 15 mins","Sit 15 mins","Run 4 mins"]
                else:
                    mtasks = ["Run 5 mins","Walk 15 mins", "Run 5 mins","Sit 20 mins","Run 2 mins"]

            elif u_gender == "female":
                if u_age <18:
                    mtask= ["Run 5 mins","Walk 15 mins", "Run 5 mins","Sit 20 mins","Run 2 mins"]
                if (u_age) > 18 and (u_age) < 40 :
                    mtasks = ["Run 14 mins","Walk 5 mins", "Run 14 mins", "Sit 14 mins","Run 2 mins"]
                else:
                    mtasks = ["Run 4 mins", "Walk 5 mins","Run 4 mins","Sit 19 mins","Run 1 mins"]

        self.m_task_one.text = mtasks[0]
        self.m_task_two.text = mtasks[1]
```

```
323         self.m_task_three.text = mtasks[2]
324         self.m_task_four.text = mtasks[3]
325         self.m_task_five.text = mtasks[4]
326 
327 
328         session.commit()
329 
330     ############################   CHECK IN/OUT ##########################################################################
331 
332 
333     def AI_callback_get_movement_data(self,dt):
334 
335         global g_time_in_movement
336 
337         if g_sensor_reachable:
338 
 Sensor_values.append([accelerometer.accelerometer[0],accelerometer.accelerometer[1],accelerometer.accelerometer[2]])
339         else:
340             Sensor_values.append([random.random(),random.random(),random.random()])
341         print(self.movement_profile.text)
342         if self.movement_profile.text not in ACTIVITIES.values():
343             self.movement_profile.text = "Reading Sensor Values!"
344 
345         if (len(Sensor_values) == 12):
346             last_movement_profile = self.movement_profile.text
347             mp = predict(m,Sensor_values)
348             self.movement_profile.text = mp
349             somethign  = Motivation(logged_in_user, datetime.now(), list(ACTIVITIES.values()).index(mp))
350             print(list(ACTIVITIES.values()).index(mp))
351             session.add(somethign)
352             session.commit()
353 
354             #write_movement_data(get_user_data_with_email(self.email)["user_id"],datetime.now().replace(microsecond=0),ACTIVITIES[mp])
355             Sensor_values.clear()
356 
357             if last_movement_profile == self.movement_profile.text:
358                 g_time_in_movement += 1
359             else:
360                 g_time_in_movement = 0
361 
362             if g_time_in_movement == 3:
363                 if self.notification:
364                     notification.notify(title='test', message=f'Hello {logged_in_username}, you are now {self.movement_profile.text} for 2h. let'+"'"+'s change it')
365                     g_time_in_movement = 0
366 
367     def check_in_deactivate(self, value):
368         if value == False:
369             self.ids.check_in_button.disabled = True
370             self.ids.check_out_button.disabled = False
371             self.check_in_start_movement_analysis()
372 
373     def check_out_deactivate(self, value):
374         if value == False:
375             self.ids.check_in_button.disabled = False
```

```
376             self.ids.check_out_button.disabled = True
377             self.check_out_stop_movement_analysis()
378 
379     def check_in_start_movement_analysis(self):
380 
381 
382         self.event = Clock.schedule_interval(self.AI_callback_get_movement_data,
    1/12.)#8.)
383         # age = date.today().year - datetime.strptime(g_user_birthdate,
    '%d.%m.%Y').year
384 
385         # if self.movement_profile not in ['sitting','walking','runing']:
386         #     sql_movement_profile = 0
387         # else:
388         #     sql_movement_profile = ACTIVITIES[self.movement_profile]
389 
390 
391 
392     def check_out_stop_movement_analysis(self):
393         self.event.cancel()
394         self.movement_profile.text = "Inactive!"
395 
396 
397 
398     def reminder(self):
399 
400         self.event_notification = Clock.schedule_once(self.callback_reminder, 2.)#8.)
401 
402 
     ##########################################################################################
    ######################
403 
404     # def callback(self):
405     #     global logged_in_user
406     #     result = session.query(Tasks).filter(User.email == logged_in_user)
407     #     for tasks in result:
408     #         self.m_task_one.text=tasks.task1
409     #         self.m_task_two.text = tasks.task2
410     #         self.m_task_three.text = tasks.task3
411     #         self.m_task_four.text = tasks.task4
412     #         self.m_task_five.text = tasks.task5
413     ##############            NOTIFICATION
    ###################################################################
414 
415     show_notif = True
416     def notif_switch_click(self, switchObject, switchValue):
417         # show_notif = True
418         if(switchValue):
419             self.ids.notification_label.text = "Notification ON"
420             self.show_notif = True
421             self.ids.checkbox_one.disabled = False
422             self.ids.checkbox_two.disabled = False
423             self.ids.checkbox_three.disabled = False
424             self.ids.checkbox_four.disabled = False
425             self.ids.checkbox_five.disabled = False
426             # self.notification_ON_time()
427         else:
428             self.ids.notification_label.text = "Notification OFF"
429             self.show_notif = False
430             self.motiv_task_list = []
```

```
            self.ids.checkbox_one.active = False
            self.ids.checkbox_two.active = False
            self.ids.checkbox_three.active = False
            self.ids.checkbox_four.active = False
            self.ids.checkbox_five.active = False
            self.ids.checkbox_one.disabled = True
            self.ids.checkbox_two.disabled = True
            self.ids.checkbox_three.disabled = True
            self.ids.checkbox_four.disabled = True
            self.ids.checkbox_five.disabled = True
            #self.notification_OFF_time()

    notification_interval= ObjectProperty(None)

    def notification_ON_time(self):
        self.notification_interval = Clock.schedule_interval(self, notification_popupreminder, 10)

    def notification_popupreminder(self):
        notification.notify(title= "Notification is Turned ON!", message= "Start work out!")

    def notification_OFF_time(self):
        self.pop_up_notification()
        self.notification_interval.cancel()


    # self.trigger1()
    # self.trigger1 = Clock.create_trigger(self.callback_reminder_task1, 10.)

    # def push_noticificaton(self,*args):
    #     plyer.notification.notify(title='GetFit!',
    #     message= '2 hour of working! Great! now Lets GetFit!',
    #     app_name='GetFit')

    # def stop_noticification(self):
    #     #print('noticifation stopped')
    #     self.notif_interval.cancel()

    def pop_up_notification(self):
        try:
            notification.notify(title='LETS GO!!!!', message="Start You Workout Now! ", timeout=10)
        except:
            self.show_notification_alert_dialog()


    def remind_me_notification(self):
        if checkbox_one.value == True:
            print("active baby")
        elif checkbox_one.value== False:
            print("you have not selected any options")



    list_of_mtasks = []
        # collect_the_text=[]

    def checkbox_click(self,instance, value, mtask):
        if value== True:
```

```
488             self.list_of_mtasks.append(mtask)
489             # txt= ''
490             # for x in self.list_of_mtasks:
491             #     txt=f'{txt} {x}'
492             # self.ids.remind_me_tasks.text= f'You Selected: {txt}'
493 
494         else:
495             self.list_of_mtasks.remove(mtask)
496             # txt= ''
497             # for x in self.list_of_mtasks:
498             #     txt=f'{txt} {x},'
499             # self.ids.remind_me_tasks.text = f'You Selected: {txt}'
500             # print("none")
501 
502     #  def notification_ON_time(self):
503     #     self.notification_interval = Clock.schedule_interval(self,
    notification_popupreminder, 10)
504 
505     # def notification_popupreminder(self):
506     #     plyer.notification.notify(title= " Notification is Turned ON!", message=
    "Start Work out!")
507 
508     def reminder_remindme(self):
509         txt= ''
510         for x in self.list_of_mtasks:
511             txt=f'{txt} {x},'
512         notification.notify(title= " LETS GO!!", message= f"dont forget to do your
    tasks! {txt} ")
513         # Clock.schedule_interval(self, notification_popupreminder, 10)
514 
515 
516 
517 
518 class SettingsScreen(Screen):
519     pass
520 
521 class GraphScreen(Screen):
522  #   graph_label
523     piechartimage = ObjectProperty(None)
524     piechartlabel =ObjectProperty(None)
525 
526 
527     def on_enter(self):
528         self.DPiechart()
529 
530 
531     def DPiechart(self):
532 
533 
534         self.piechartlabel.text = "Daily"
535 
536         piechartdata = []
537         #comare timestamp, to be filtered
538         result = session.query(Motivation).filter(Motivation.email == logged_in_user,
    Motivation.timestamp)
539         #datetime.now()
540         today = date.today()
541         for res in result:
542             graph_movementprofile= res.mvmntprofile
543             graph_timestamp= res.timestamp
```

```
544             List_day_of_record = graph_timestamp.split(' ')[0].split('-')
545             day_of_record = date(int(List_day_of_record[0]),
int(List_day_of_record[1]),int(List_day_of_record[2]))
546             if today == day_of_record:
547                 piechartdata.append(str(graph_movementprofile))
548 
549         piechartdata_yaxis=[0,0,0]
550         piechartdata_yaxis[0] = piechartdata.count('0')
551         piechartdata_yaxis[1] = piechartdata.count('1')
552         piechartdata_yaxis[2] = piechartdata.count('2')
553         piechartdata_xaxis = ['Sitting', 'Walking', 'Running', ]
554         plt.pie(piechartdata_yaxis, labels=piechartdata_xaxis,startangle= 90)
555         plt.legend(title = "Daily Tasks:")
556         # plt.show()
557         plt.savefig('piechartresults.png')
558         self.piechartimage.source = os.path.join(dirname, 'piechartresults.png')
559         self.piechartimage.reload()
560 
561     # fix, ax = plt.subplots()
562     # ax.bar(values_xaxis, values_yaxis)
563     # ax.set_ylabel('minutes')
564     # figure = plt.gcf()
565     # figure.set_size_inches(3, 3)
566     # plt.savefig('plot.png')
567     # self.graph.source = os.path.join(dirname, 'plot.png')
568     # self.graph.reload()
569 
570     # y = np.array([35, 25, 25, 15])
571     # mylabels = [ graph_movementprofile[0], graph_movementprofile[1],
graph_movementprofile[2]]
572     # plt.pie(y, labels = mylabels)
573     # plt.show()
574 
575     def WPiechart(self):
576 
577 
578         self.piechartlabel.text = "Weekly"
579         today = date.today()
580         piechartdata = []
581         #comare timestamp, to be filtered
582         result = session.query(Motivation).filter(Motivation.email == logged_in_user,
Motivation.timestamp)
583         #datetime.now()
584         startofweek= today - timedelta(days=today.weekday())
585         endofweek = startofweek + timedelta(days=6)
586 
587         self.piechartlabel.text = f'{str(startofweek)} - {str(endofweek)}'
588 
589         for res in result:
590             graph_movementprofile= res.mvmntprofile
591             graph_timestamp= res.timestamp
592             List_day_of_record = graph_timestamp.split(' ')[0].split('-')
593             day_of_record = date(int(List_day_of_record[0]),
int(List_day_of_record[1]),int(List_day_of_record[2]))
594             if day_of_record == startofweek:
595                 print(today,day_of_record)
596                 piechartdata.append(str(graph_movementprofile))
597 
598 
599         piechartdata_yaxis=[0,0,0]
```

```
600         piechartdata_yaxis[0] = piechartdata.count('0')
601         piechartdata_yaxis[1] = piechartdata.count('1')
602         piechartdata_yaxis[2] = piechartdata.count('2')
603         piechartdata_xaxis = ['Sitting', 'Walking', 'Running', ]
604         plt.pie(piechartdata_yaxis, labels=piechartdata_xaxis,startangle= 90)
605         plt.legend(title = "Weekly Tasks:")
606         # plt.show()
607         plt.savefig('piechartresults.png')
608         self.piechartimage.source = os.path.join(dirname, 'piechartresults.png')
609         self.piechartimage.reload()
610 
611     def MPiechart(self):
612 
613 
614         self.piechartlabel.text = "Monthly"
615         today = date.today()
616         piechartdata = []
617         #comare timestamp, to be filtered
618         result = session.query(Motivation).filter(Motivation.email == logged_in_user,
    Motivation.timestamp)
619         #datetime.now()
620 
621         for res in result:
622             graph_movementprofile= res.mvmntprofile
623             graph_timestamp= res.timestamp
624             List_day_of_record = graph_timestamp.split(' ')[0].split('-')
625             day_of_record = date(int(List_day_of_record[0]),
    int(List_day_of_record[1]),int(List_day_of_record[2]))
626             if today.month == day_of_record.month and today.year==day_of_record.year:
627                 piechartdata.append(str(graph_movementprofile))
628 
629         piechartdata_yaxis=[0,0,0]
630         piechartdata_yaxis[0] = piechartdata.count('0')
631         piechartdata_yaxis[1] = piechartdata.count('1')
632         piechartdata_yaxis[2] = piechartdata.count('2')
633         piechartdata_xaxis = ['Sitting', 'Walking', 'Running', ]
634         plt.pie(piechartdata_yaxis, labels=piechartdata_xaxis,startangle= 90)
635         plt.legend(title = "Monthly Tasks:")
636         # plt.show()
637         plt.savefig('piechartresults.png')
638         self.piechartimage.source = os.path.join(dirname, 'piechartresults.png')
639         self.piechartimage.reload()
640 
641 
642 class NotificationScreen(Screen):
643     pass
644 
645 class UserProfileScreen(Screen):
646 
647     # we need to get the user value from the database
648     # current_user= session.query(User).filter()
649 
650     def on_pre_enter(self):
651         self.callback()
652 
653     def callback(self):
654         global logged_in_user
655         result = session.query(User).filter(User.email == logged_in_user)
656         for user in result:
657             self.email_label.text=user.email
```

```
            self.username_label.text = user.name
            self.age_label.text = str(user.age)
            self.weight_label.text = str(user.weight)
            self.height_label.text = str(user.uheight)
            self.gender_label.text = user.gender
            self.jp_label.text = user.jp
            global logged_in_user_age
            logged_in_user= user.age
            global logged_in_user_gender
            logged_in_user_gender= user.gender

class EditUserProfileScreen(Screen):

    def on_pre_enter(self):
        self.callback()

    def callback(self):
        global logged_in_user
        result = session.query(User).filter(User.email == logged_in_user)
        for user in result:
            self.chg_username.text = user.name
            self.chg_age.text = str(user.age)
            self.chg_weight.text = str(user.weight)
            self.chg_height.text = str (user.uheight)
            self.chg_gender.text = user.gender
            self.chg_jp.text = user.jp

    def savechanges(self):
        username_input = self.chg_username.text
        age_input = self.chg_age.text
        age_input = int(age_input)
        weight_input = self.chg_weight.text
        height_input = self.chg_height.text
        weight_input = int(weight_input)
        height_input = int(height_input)
        gender_input = self.chg_gender.text
        jp_input = self.chg_jp.text
        session.query(User).filter(User.email == logged_in_user).update(
            {'name': username_input, 'age': age_input, 'weight': weight_input,
"uheight": height_input,
             'gender': gender_input, 'jp': jp_input})


        if jp_input== "Software Engineer":
            if age_input<18 and  gender_input== "female":
                data.tasks = Tasks("Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5
mins","Run 20 mins")
            elif age_input>=18 and  age_input<40 and  gender_input== "female":
                data.tasks = Tasks("Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5
mins","Run 20 mins")
            elif age_input>40 and  gender_input== "female":
                data.tasks = Tasks("Run 20 mins", "Walk 5 mins","Sit 9 mins","Walk 5
mins","Run 20 mins")
            elif age_input<18 and  gender_input== "male":
                data.tasks = Tasks("Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5
mins","Run 20 mins")
            elif age_input>=18 and  age_input<40 and  gender_input== "male":
                data.tasks = Tasks("Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5
mins","Run 20 mins")
            elif age_input>40 and  gender_input== "male":
```

```
                data.tasks = Tasks("Run 20 mins","Walk 5 mins","Sit 9 mins","Walk 5 mins","Run 20 mins")

        else:
            if age_input<18 and  gender_input== "female" :
                User.tasks = Tasks(taskf18[0],taskf18[1],taskf18[2],taskf18[3],taskf18[4])
            elif age_input>=18 and  age_input<40 and  gender_input== "female":
                User.tasks = Tasks(taskf24[0],taskf24[1],taskf24[2],taskf24[3],taskf24[4])
            elif age_input>40 and  gender_input== "female":
                User.tasks = Tasks(taskf40[0],taskf40[1],taskf40[2],taskf40[3],taskf40[4])
            elif age_input<18 and  gender_input== "male":
                User.tasks = Tasks(taskm18[0],taskm18[1],taskm18[2],taskm18[3],taskm18[4])
            elif age_input>=18 and  age_input<40 and  gender_input== "male":
                User.tasks = Tasks(taskm24[0],taskm24[1],taskm24[2],taskm24[3],taskm24[4])
            elif age_input>40 and  gender_input== "male":
                User.tasks = Tasks(taskm40[0],taskm40[1],taskm40[2],taskm40[3],taskm40[4])

        session.commit()
        self.manager.current = "settings"

class SystemSettingsScreen(Screen):
    sensorschecks = []
    def sensorcheckbox_click(self,instance, value, sensors): #value will pass true r pass
        if value== True:
            SystemSettingsScreen.sensorschecks.append(sensors)
            sen = ''
            for x in self.sensorchecks:
                sen = f'{sen}{x}'
            self.ids.sensorselectiontext.text=f'{sen} has been Activated!'
        else:
            SystemSettingsScreen.sensorschecks.remove(sensors)
            sen = ''
            for x in self.sensorchecks:
                sen = f'{sen}{x}'
                self.ids.sensorselectiontext.text=f'{sen} has been Activated!'



sm = ScreenManager()
# sm.add_widget(MDScreen(name="md"))
sm.add_widget(LoginScreen(name="login"))
sm.add_widget(TermsandCondition(name="termsandcondition"))
sm.add_widget(SignUpScreen(name="signup"))
sm.add_widget(MainScreen(name="main"))
sm.add_widget(SettingsScreen(name="settings"))
sm.add_widget(GraphScreen(name="graph"))
sm.add_widget(NotificationScreen(name="notify"))
sm.add_widget(UserProfileScreen(name="userprofile"))
sm.add_widget(EditUserProfileScreen(name="edituserprofile"))
sm.add_widget(SystemSettingsScreen(name="systemsettings"))
```

```
764 Window.size = (320, 550)
765
766
767
768 # GUI= Builder.load_file("main.kv") #Make Sure this is after all  definitions!
769 class MainApp(MDApp):
770     def __init__(self, **kwargs):
771         super(MainApp, self).__init__(**kwargs)
772         try:
773             accelerometer.enable() # enable the accelerometer
774             global g_sensor_reachable
775             g_sensor_reachable = True
776         except:
777             print ("No Accelerometer Detected!")
778
779     def build(self):
780         #builder = Bzuilder.load_file("main.kv")
781         # return #builder
782         pass
783
784
785 if __name__ == '__main__':
786     MainApp().run()
787
```