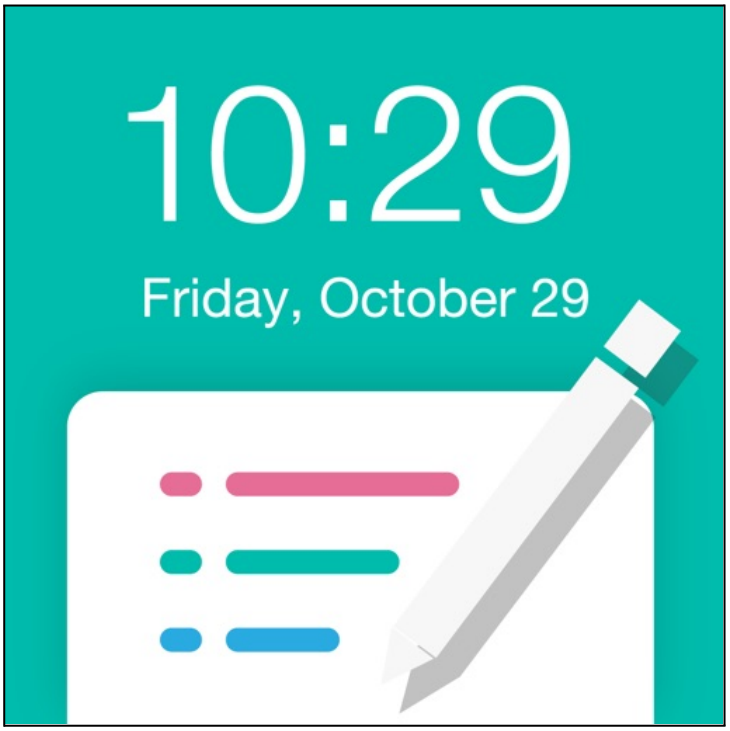


- MADE BY –
  IRITH CHATURVEDI
  A STUDENT OF STEP BY STEP SCHOOL - NOIDA

INDEX:

MODULES:

INBUILT FUNCTIONS:-

```
from kivymd.app import MDApp
from kivy.lang import Builder
from kivy.core.window import Window
from kivymd.uix.button import MDRectangleFlatButton, MDFlatButton
from kivymd.uix.button import MDFloatingActionButton
from kivymd.uix.button import MDRectangleFlatIconButton, MDFillRoundFlatIconButton
from kivy.uix.screenmanager import Screen, ScreenManager
from kivymd.uix.picker import MDThemePicker
from kivymd.uix.picker import MDTimePicker
from kivymd.uix.label import MDLabel
from kivymd.uix.button import MDRaisedButton
from kivy.uix.label import Label
from kivymd.uix.list import OneLineListItem
from kivy.properties import ListProperty
from kivy.properties import NumericProperty
from kivymd.uix.dialog import MDDialog
from kivy.uix.widget import Widget
from kivy.uix.button import Button
from kivy.graphics import Color, Ellipse, Line, Rectangle
from kivy.core.audio import SoundLoader
from kivymd.uix.picker import MDDatePicker
from kivymd.uix.datatables import MDDataTable
from kivy.metrics import dp
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.gridlayout import GridLayout
from kivymd.uix.behaviors import RectangularElevationBehavior, FocusBehavior
from kivymd.uix.textfield import MDTextField
from kivymd.uix.button import MDFloatingActionButtonSpeedDial
```

USER DEFINED FUNCTIONS:-

```
def on_touch_down(self, touch):
    with self.canvas:
        Color(255 / 255, 20 / 255, 147 / 255)
        d = 2
        Ellipse(pos=(touch.x - d / 2, touch.y - d / 2), size=(d, d))
        touch.ud['line'] = Line(points=(touch.x, touch.y))  # line is dictionary

def on_touch_move(self, touch):
    touch.ud['line'].points += [touch.x, touch.y]

def __init__(self, **kwargs):
    super().__init__(**kwargs)

def build(self):
    self.theme_cls.primary_palette = "Pink"
    self.theme_cls.accent_palette = "Orange"
    self.theme_cls.primary_hue = "A400"
    return Builder.load_string(screen_helper)

def on_start(self):
    self.painter = MyPaintWidget()
    clrbtn = MDFloatingActionButton(icon='delete-empty',
                                    pos_hint={'center_x': .1,
                                    'center_y': .075},
    padding='10 sp',ripple_color=self.theme_cls.primary_color)

    clrbtn.bind(on_release=self.paint_clear)
    self.emp = MDLabel(text="", size_hint_y=None, height=30)
    self.Num = MDLabel(text="No.", font_style="H6", size_hint=(None, None),
                       width=50, height=20)
    self.Item = MDLabel(text="    ITEM ", font_style="H6", size_hint_y=None,
                        height=20)
    self.Amount = MDLabel(text=" AMOUNT ", font_style="H6", size_hint_y=None,
                          height=20)
    self.s1 = MDLabel(text="", font_style="H6", size_hint=(None, None),
                      width=50,height=20)
```

```
        self.s2 = MDLabel(text="____", font_style="H6", size_hint_y=None, height=20)
        self.s3 = MDLabel(text="_______", font_style="H6", size_hint_y=None,
                          height=20)
        self.notebtn = Builder.load_string(note)
        self.itembtn = Builder.load_string(item)
        self.amountbtn = Builder.load_string(amount)
        self.addbtn = Builder.load_string(add_list)
        self.root.ids.paint.add_widget(self.painter)
        self.root.ids.paint.add_widget(clrbtn)
        self.root.ids.list.add_widget(self.itembtn)
        self.root.ids.list.add_widget(self.amountbtn)
        self.root.ids.list.add_widget(self.addbtn)
        self.root.ids.grid.add_widget(self.Num)
        self.root.ids.grid.add_widget(self.Item)
        self.root.ids.grid.add_widget(self.Amount)
        self.root.ids.grid.add_widget(self.s1)
        self.root.ids.grid.add_widget(self.s2)
        self.root.ids.grid.add_widget(self.s3)
        self.root.ids.note.add_widget(self.notebtn)
        self.root.ids.notes.add_widget(self.emp)
        self.note = Builder.load_string(new_note)
        self.root.ids.notes.add_widget(self.note)

    def show_alert_dialog(self):  # information
        if not self.dialog:
            self.dialog = MDDialog(
                title="INSTRUCTIONS:",
                text="NOTES:-  \nmultiline text notes:- tap on new note to add
                      note\nDRAW:-  \ndrawing notes:-
                      make drawing notes and then tap on clear button to clear
                      screen\nALARM:-  \nset an
                      alarm:- tap add alarm button to add alarm\nEVENTS:-
                      \nschedule events:- tap new event button to add alarm\nLIST:-
                      \ncreate shopping/grocery list (list is scrollable):- type
                      item and amount and then tap add item button\nCHANGING THEME
                      COLOR:- \ntap paint button on home screen:- Primary is for top
                      and bottom toolbar, Accent is for buttons",
                buttons=[
                    MDFlatButton(
                        text="CONTINUE", text_color=self.theme_cls.primary_color,
                        on_release=self.close_dialog
                    ),
                ], size_hint=(0.9, 1)
            )
        self.dialog.open()

    def show_data(self):  # for list
```

```
        self.item = MDRectangleFlatButton(text=self.itembtn.text,
                    text_color=self.theme_cls.accent_color,
                                          size_hint_y=None,
                                          height=40)
        self.amount = MDRectangleFlatButton(text=self.amountbtn.text,
                      text_color=self.theme_cls.accent_color,
                                            size_hint_y=None,
                                            height=40)
        self.no_text = MDLabel(text=str(self.t) + ".",
                       text_color=self.theme_cls.accent_color,
                               size_hint=(None, None),
                               width=50, height=40)
        self.root.ids.grid.add_widget(self.no_text)
        self.root.ids.grid.add_widget(self.item)
        self.root.ids.grid.add_widget(self.amount)
        self.itembtn.text = ""
        self.amountbtn.text = ""
        global t
        self.t += 1
        return

    def close_dialog(self, obj):
        self.dialog.dismiss()

    def paint_clear(self, obj):
        self.painter.canvas.clear()

    def show_theme_picker(self):
        theme_dialog = MDThemePicker()
        theme_dialog.open()

    def navigation_draw(self):
        print("Navigation")

    def show_time_picker(self):
        time_dialog = MDTimePicker()
        time_dialog.bind(time=self.get_time)
        time_dialog.open()

    def get_time(self, instance, time):
        print(time)
        l.append(time)

        if len(l) == 5:
            note = Builder.load_string(new_description)
            alarm = MDRectangleFlatButton(text=str(len(l)) + ') '+str(l[len(l)- 1]))
            self.label = MDLabel(text="SCROLL DOWN:", font_style='Caption',
```

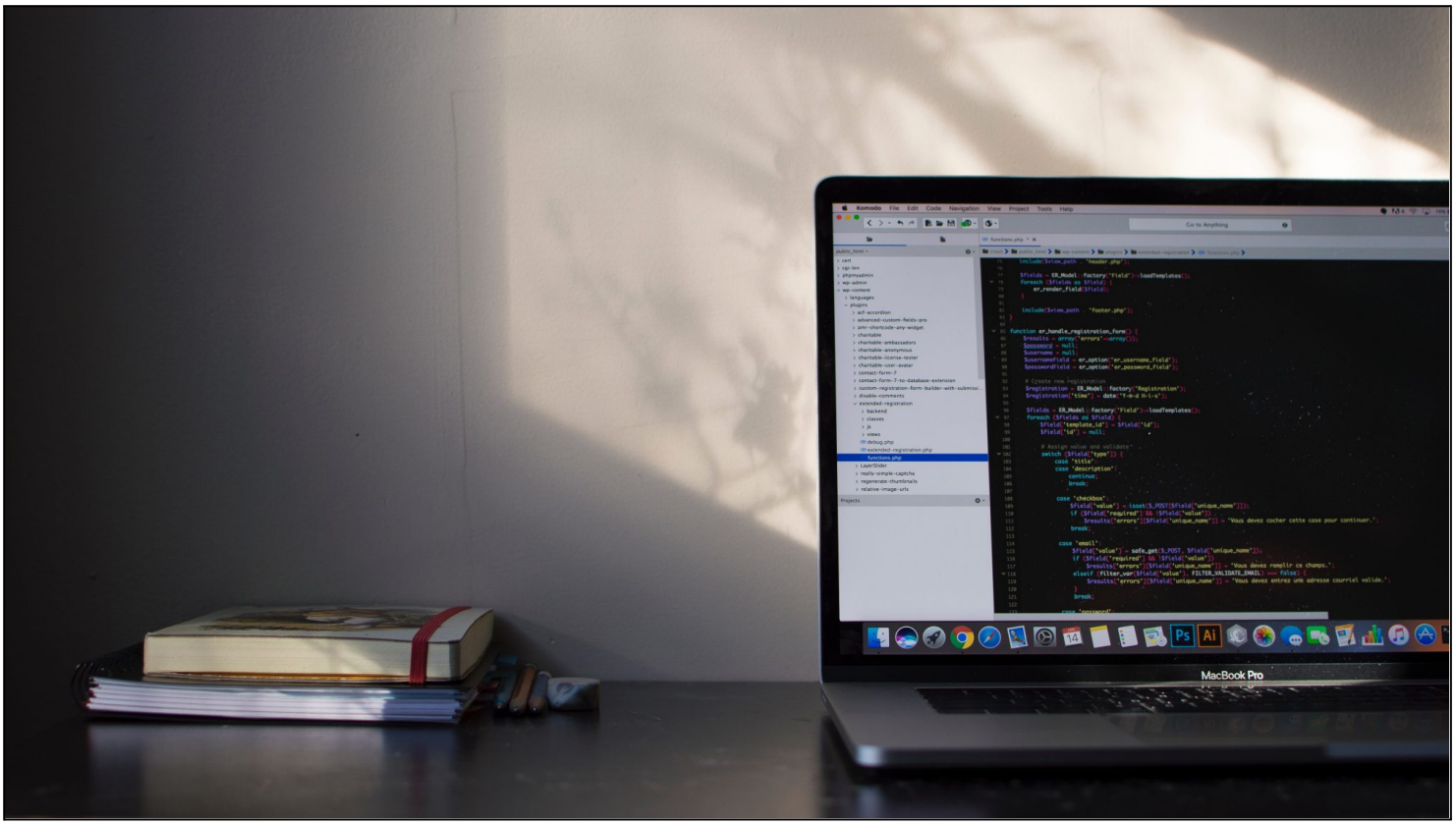
```
        size_hint_y=None, height=5)
        self.root.ids.time.add_widget(self.label)
        self.root.ids.time.add_widget(alarm)
        self.root.ids.time.add_widget(note)
    else:
        note = Builder.load_string(new_description)
        alarm = MDRectangleFlatButton(text=str(len(l)) + ') '+str(l[len(l)- 1]))
        self.root.ids.time.add_widget(alarm)
        self.root.ids.time.add_widget(note)


def get_date(self, date):
    '''
    :type date: <class 'datetime.date'>
    '''
    print(date)
    l1.append(date)  # use of list
    if len(l1) == 5:
        note = Builder.load_string(new_description)
        event = MDRectangleFlatButton(text=str(len(l1))+')'+str(l1[len(l1)- 1]))
        self.label = MDLabel(text="SCROLL DOWN:", font_style='Caption',
                             size_hint_y=None, height=5)
        self.root.ids.date.add_widget(self.label)
        self.root.ids.date.add_widget(event)
        self.root.ids.date.add_widget(note)

    else:
        note = Builder.load_string(new_description)
        event = MDRectangleFlatButton(text=str(len(l1))+')'+str(l1[len(l1)- 1]))
        self.root.ids.date.add_widget(event)
        self.root.ids.date.add_widget(note)

def show_date_picker(self):
    date_dialog = MDDatePicker(callback=self.get_date)
    date_dialog.open()

def new_note(self):
    if self.z == 6:
        self.label = MDLabel(text="SCROLL DOWN:", font_style='Caption',
                             size_hint_y=None, height=40)
        note = Builder.load_string(new_note)
        self.root.ids.notes.add_widget(self.label)
        self.root.ids.notes.add_widget(note)
        self.z += 1
    else:
        note = Builder.load_string(new_note)
        self.root.ids.notes.add_widget(note)
```

```
        self.z += 1

    def check_press(self, instance_table, current_row):
        print(instance_table, current_row)

    def row_press(self, instance_table, instance_row):
        print(instance_table, instance_row)

    def btn_pressed(self):
        self.sound1 = SoundLoader.load('12910_sweet_trip_mm_clap_mid.wav')
        if self.sound1:
            self.sound1.play()
        print("sound")

    def play_sound(self):
        self.sound = SoundLoader.load('12911_sweet_trip_mm_hat_cl.wav')
        self.sound.play()
        print("sound")
```

INSTALLED PACKAGES:

(PYTHON INTERPRETER VERSION- 3.7)

Package	Version	Latest version
Kivy	2.0.0	2.0.0
Kivy-Garden	0.1.4	0.1.4
KivyCalendar	0.1.3	0.1.3
Pillow	8.1.0	8.1.0
Pygments	2.7.3	▲ 2.7.4
certifi	2020.12.5	2020.12.5
chardet	4.0.0	4.0.0
docutils	0.16	0.16
idna	2.10	▲ 3.1
kivy-deps.angle	0.3.0	0.3.0
kivy-deps.glew	0.3.0	0.3.0
kivy-deps.gstreamer	0.3.1	0.3.1
kivy-deps.sdl2	0.3.1	0.3.1
kivymd	0.104.1	0.104.1
pip	20.3.3	20.3.3
pypiwin32	223	223
pywin32	300	300
requests	2.25.1	2.25.1
setuptools	51.1.2	51.1.2
urllib3	1.26.2	1.26.2



WORKING DESCRIPTION:

RemindME is an android based mobile application that accommodates all sorts of everyday requirements for which a person may need to install multiple applications. RemindME, here, plays the role of a versatile application, satisfying all of these requirements as a universal reminder app. The application is divided into 6 components; namely the Home, Notes, Draw, Alarm, Event, List screens.

Home Screen: The home serves as a welcome to all the users and is also equipped with a theme changer, allowing the users to better their experience by customizing the color notations in the app as per what they feel looks good (but we feel that the default color notations seem to merge best with the application layout). Along with the theme changer, all screens, including the home screen are equipped with an instructions dialog box, which can be accessed through the icon button in the top toolbar.

Notes Screen: The notes component of the application allows users to make multiline notes in whatever quantity they require, the screen is scrollable, so the number of notes created by the user won't cause any problem. The + NEW NOTE button adds a new note to the screen, each note (multiline) being typed on by tapping(clicking) on it.

Draw Screen: It's well known that pictographic memory serves as a much better tool to remember things as compared to its textual-based counterpart. RemindME is also equipped with a drawing tool to make drawing notes, which can be cleared every time they have been used and are no longer required. The users can draw whatever and wherever they want on the screen. When they no longer require the drawing note, they can simply press the clear button located at the bottom left to clear the screen.

Alarm Screen: The alarm page allows users to create alarms and stack them on a single page. This page again is scrollable, and the number of alarms created won't cause any problem. The + ADD ALARM button opens the alarm clock, on which we can input the time the alarm is to be scheduled, each alarm is accompanied by a note (multiline) which is typed on by tapping(clicking) on it.

Event Screen: Similar to the alarm page, the calendar page allows users to create and store their events. The + NEW EVENT button opens the alarm clock, on which we can input the date the event is to be scheduled, each event is accompanied by a note (multiline) which is typed on by tapping(clicking) on it.

List Screen: Lastly, the list page allows users to create and arrange items along with their quantities in a grid layout, making shopping much more convenient. The two text fields in the bottom of the screen are to input the name and quantity of the items respectively, and the + ADD ITEM button is to add the item to the list.

CODE:

```
from kivymd.app import MDApp
from kivy.lang import Builder
from kivy.core.window import Window
from kivymd.uix.button import MDRectangleFlatButton, MDFlatButton,
from kivymd.uix.button import MDFloatingActionButton
from kivymd.uix.button import MDRectangleFlatIconButton, MDFillRoundFlatIconButton
from kivy.uix.screenmanager import Screen, ScreenManager
from kivymd.uix.picker import MDThemePicker
from kivymd.uix.picker import MDTimePicker
from kivymd.uix.label import MDLabel
from kivymd.uix.button import MDRaisedButton
from kivy.uix.label import Label
from kivymd.uix.list import OneLineListItem
from kivy.properties import ListProperty
from kivy.properties import NumericProperty
from kivymd.uix.dialog import MDDialog
from kivy.uix.widget import Widget
from kivy.uix.button import Button
from kivy.graphics import Color, Ellipse, Line, Rectangle
from kivy.core.audio import SoundLoader
from kivymd.uix.picker import MDDatePicker
from kivymd.uix.datatables import MDDataTable
from kivy.metrics import dp
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.gridlayout import GridLayout
from kivymd.uix.behaviors import RectangularElevationBehavior, FocusBehavior
from kivymd.uix.textfield import MDTextField
from kivymd.uix.button import MDFloatingActionButtonSpeedDial
Window.size = (360, 600)
```

```
l = []
l1 = []
l2 = []
l3 = []
screen_helper = """
BoxLayout:

    orientation:'vertical'
    MDToolbar:
        title: 'RemindME'
        left_action_items: [["format-list-checkbox",lambda x: app.show_alert_dialog()]]
        font_size: "20dp"
        elevation: 10


    MDBottomNavigation:
        panel_color: 45/255, 57/255, 69/255, 1
        elevation: 10

        MDBottomNavigationItem:
            name: 'Home'
            text: 'Home'
            icon: 'home-variant'
            on_tab_release: app.navigation_draw()
            MDScreen:
                canvas.before:
                    Rectangle:
                        size: self.size
                        source: "background.png"

            MDProgressBar:
               size_hint_y: None
               height: 5
               value:16.6
               pos_hint: {'center_y': 0.008}

            MDFloatingActionButton:
                icon: "format-color-fill"
                pos_hint: {'center_x': .5, 'center_y': .1}
                on_release: app.show_theme_picker()
                md_bg_color: app.theme_cls.primary_color

            MDIconButton:
                icon: "alpha-a-circle"
                user_font_size: "120sp"
                pos_hint: {'center_x': .25, 'center_y': .5}
                on_press: app.play_sound()
```

```
    MDIconButton:
        icon: "alpha-i-circle"
        user_font_size: "120sp"
        pos_hint: {'center_x': .75, 'center_y': .5}
        on_press: app.play_sound()

    MDIconButton:
        icon: "bell-circle"
        user_font_size: "175sp"
        pos_hint: {'center_x': .5, 'center_y': .75}
        on_press: app.btn_pressed()





    MDLabel:
        text: "WELCOME! REMINDME IS THE ALL IN ONE SOLUTION FOR ALL REMINDERS'
        halign: 'center'
        pos_hint: {'center_x': .5, 'center_y': .3}
        font_size: 20
        font_style: 'Subtitle1'
        color: 0,0,0,1

MDBottomNavigationItem:
    name: 'Notes'
    text: 'Notes'
    icon: 'notebook'
    on_tab_release: app.navigation_draw()
    MDScreen:
        canvas.before:
            Rectangle:
                size: self.size
                source: "background.png"

    MDProgressBar:
       size_hint_y: None
       height: 5
       Value: 33.2
       pos_hint: {'center_y': 0.008}

    BoxLayout:
        MDScreen:
            radius: [25, 0, 0, 0]
            id: note
            ScrollView:
```

```
                GridLayout:
                    id: notes
                    padding: "10dp"
                    spacing: "10dp"
                    size_hint_y: None
                    height: self.minimum_height
                    size_hint_x: None
                    width: self.minimum_width
                    cols:1
                    rows_force_default: True
                    rows_default_height: 40
                    rows_force_default: True
                    rows_default_width: self.minimum_width


MDBottomNavigationItem:
    name: 'Draw'
    text: 'Draw'
    icon: 'pen'
    on_tab_press: app.navigation_draw()

    Screen:
        id: paint

    MDProgressBar:
       size_hint_y: None
       height: 5
       Value: 49.8
       pos_hint: {'center_y': 0.008

MDBottomNavigationItem:
    name: 'Alarm'
    text: 'Alarm'
    icon: 'alarm-bell'
    MDScreen:
        canvas.before:
            Rectangle:
                size: self.size
                source: "background.png"

    MDProgressBar:
       size_hint_y: None
       height: 5
       Value: 66.4
       pos_hint: {'center_y': 0.008}

    MDIconButton:
```

```
            icon: "clock"
            user_font_size: "150sp"
            pos_hint: {'center_x': .8, 'center_y': .5}
        Screen:
            ScrollView:
                MDList:
                    id: time
                    padding: "10dp"
                    spacing: "7dp"
                    MDRaisedButton:
                        text: "+  ADD ALARM"
                        text_color: 0,0,0,1
                        md_bg_color: app.theme_cls.accent_color
                        user_font_size: "140sp"
                        pos_hint: {'center_x': .1, 'center_y': .8}
                        on_release: app.show_time_picker()
                        elevation: 10
                        ripple_color: app.theme_cls.primary_color

    MDBottomNavigationItem:
        name: 'Calendar'
        text: 'Events'
        icon: 'calendar-text'
        MDScreen:
            canvas.before:
                Rectangle:
                    size: self.size
                    source: "background.png"

        MDProgressBar:
           size_hint_y: None
           height: 5
           Value: 83
           pos_hint: {'center_y': 0.008}

        MDIconButton:
            icon: "calendar-month"
            user_font_size: "150sp"
            pos_hint: {'center_x': .8, 'center_y': .5}

        Screen:
            ScrollView:
                MDList:
                    id: date
                    padding: "10dp"
                    spacing: "7dp"
                    MDRaisedButton:
```

```
                        text: "+  NEW EVENT"
                        text_color: 0,0,0,1
                        md_bg_color: app.theme_cls.accent_color
                        pos_hint: {'center_x': .1, 'center_y': .8}
                        on_release: app.show_date_picker()
                        elevation: 10
                        ripple_color: app.theme_cls.primary_color


        MDBottomNavigationItem:
            name: 'List'
            text: 'ITEM'
            icon: 'clipboard-list'
            on_tab_release: app.navigation_draw()
            MDScreen:
                canvas.before:
                    Rectangle:
                        size: self.size
                        source: "background.png"


            MDProgressBar:
               size_hint_y: None
               height: 5
               Value: 100
               pos_hint: {'center_y': 0.008}

            Screen:
                ScrollView:
                    GridLayout:
                        id: grid
                        padding: "10dp"
                        spacing: "10dp"
                        cols: 3
                        size_hint_y: None
                        height: self.minimum_height
                        rows_force_default: True
                        rows_default_height: 40

                MDScreen:
                    radius: [25, 0, 0, 0]
                    id: list
"""

note = """
MDRaisedButton:
    text: "+  NEW NOTE"
```

```
    text_color: 0,0,0,1
    md_bg_color: app.theme_cls.accent_color
    ripple_effect: False
    user_font_size: "140sp"
    pos_hint: {'center_x': .18, 'center_y': .945}
    on_release: app.new_note()
    elevation: 10
    ripple_color: app.theme_cls.primary_color
"""
new_note = """
MDTextField:
    hint_text: "Note"
    helper_text: "write something (then tap on screen)"
    helper_text_mode: "on_focus"
    line_color_normal: app.theme_cls.accent_color
    multiline: True
    size_hint_x: None
    width: 300
"""
new_description = """
MDTextField:
   hint_text: "Description"
   helper_text: "write something (then tap on screen)"
   helper_text_mode: "on_focus"
   line_color_normal: app.theme_cls.accent_color
   multiline: True
   size_hint_y: None
   height: 50
"""
item = """
MDTextField:
    hint_text: "Item"
    hint_text_color: 0,0,0,1
    line_color_normal: app.theme_cls.accent_color
    helper_text: "enter item"
    helper_text_mode: "on_focus"
    pos_hint: {'center_x':0.17,'center_y':0.1}
    size_hint_x: None
    width: 100
"""
amount = """
MDTextField:
    hint_text: "Amount"
    hint_text_color: 0,0,0,1
    line_color_normal: app.theme_cls.accent_color
    helper_text: "enter amount"
    helper_text_mode: "on_focus"
```

```
    pos_hint: {'center_x':0.83,'center_y':0.1}
    size_hint_x: None
    width: 100
"""
add_list = """
MDRaisedButton:
    text: "+  Add Item"
    text_color: 0,0,0,1
    md_bg_color: app.theme_cls.accent_color
    pos_hint: {'center_x': .5, 'center_y': .082}
    elevation: 10
    size_hint_x: None
    width: 100
    on_release: app.show_data()
    ripple_color: app.theme_cls.primary_color
"""

class HomeScreen(Screen):
    pass


class MyPaintWidget(Widget):

    def on_touch_down(self, touch):
        with self.canvas:
            Color(255 / 255, 20 / 255, 147 / 255)
            d = 2
            Ellipse(pos=(touch.x - d / 2, touch.y - d / 2), size=(d, d))
            touch.ud['line'] = Line(points=(touch.x, touch.y))  # line is dictionary

    def on_touch_move(self, touch):
        touch.ud['line'].points += [touch.x, touch.y]
class DraftApp(MDApp):
    dialog = None
    sound = None

    def __init__(self, **kwargs):
        super().__init__(**kwargs)

    def build(self):
        self.theme_cls.primary_palette = "Pink"
        self.theme_cls.accent_palette = "Orange"
        self.theme_cls.primary_hue = "A400"
        return Builder.load_string(screen_helper)

    def on_start(self):
        self.painter = MyPaintWidget()                                  # dictionary
```

```
        clrbtn = MDFloatingActionButton(icon='delete-empty', pos_hint={'center_x': .1,
                                        'center_y': .075},
        padding='10 sp',ripple_color=self.theme_cls.primary_color)

        clrbtn.bind(on_release=self.paint_clear)
        self.emp = MDLabel(text="", size_hint_y=None, height=30)
        self.Num = MDLabel(text="No.", font_style="H6", size_hint=(None, None),
                           width=50, height=20)
        self.Item = MDLabel(text="    ITEM ", font_style="H6", size_hint_y=None,
                            height=20)
        self.Amount = MDLabel(text=" AMOUNT ", font_style="H6", size_hint_y=None,
                              height=20)
        self.s1 = MDLabel(text="", font_style="H6", size_hint=(None, None), width=50,
                          height=20)
        self.s2 = MDLabel(text="____", font_style="H6", size_hint_y=None, height=20)
        self.s3 = MDLabel(text="_______", font_style="H6", size_hint_y=None, height=20)
        self.notebtn = Builder.load_string(note)
        self.itembtn = Builder.load_string(item)
        self.amountbtn = Builder.load_string(amount)
        self.addbtn = Builder.load_string(add_list)
        self.root.ids.paint.add_widget(self.painter)
        self.root.ids.paint.add_widget(clrbtn)
        self.root.ids.list.add_widget(self.itembtn)
        self.root.ids.list.add_widget(self.amountbtn)
        self.root.ids.list.add_widget(self.addbtn)
        self.root.ids.grid.add_widget(self.Num)
        self.root.ids.grid.add_widget(self.Item)
        self.root.ids.grid.add_widget(self.Amount)
        self.root.ids.grid.add_widget(self.s1)
        self.root.ids.grid.add_widget(self.s2)
        self.root.ids.grid.add_widget(self.s3)
        self.root.ids.note.add_widget(self.notebtn)
        self.root.ids.notes.add_widget(self.emp)
        self.note = Builder.load_string(new_note)
        self.root.ids.notes.add_widget(self.note)

    def show_alert_dialog(self):  # information
        if not self.dialog:
            self.dialog = MDDialog(
                title="INSTRUCTIONS:",
                text="NOTES:-  \nmultiline text notes:- tap on new note to add
                      note\nDRAW:-  \ndrawing notes:-
                      make drawing notes and then tap on clear button to clear
                      screen\nALARM:-  \nset an
                      alarm:- tap add alarm button to add alarm\nEVENTS:-  \nschedule
                      events:- tap new event
                      button to add alarm\nLIST:-  \ncreate shopping/grocery list (list
```

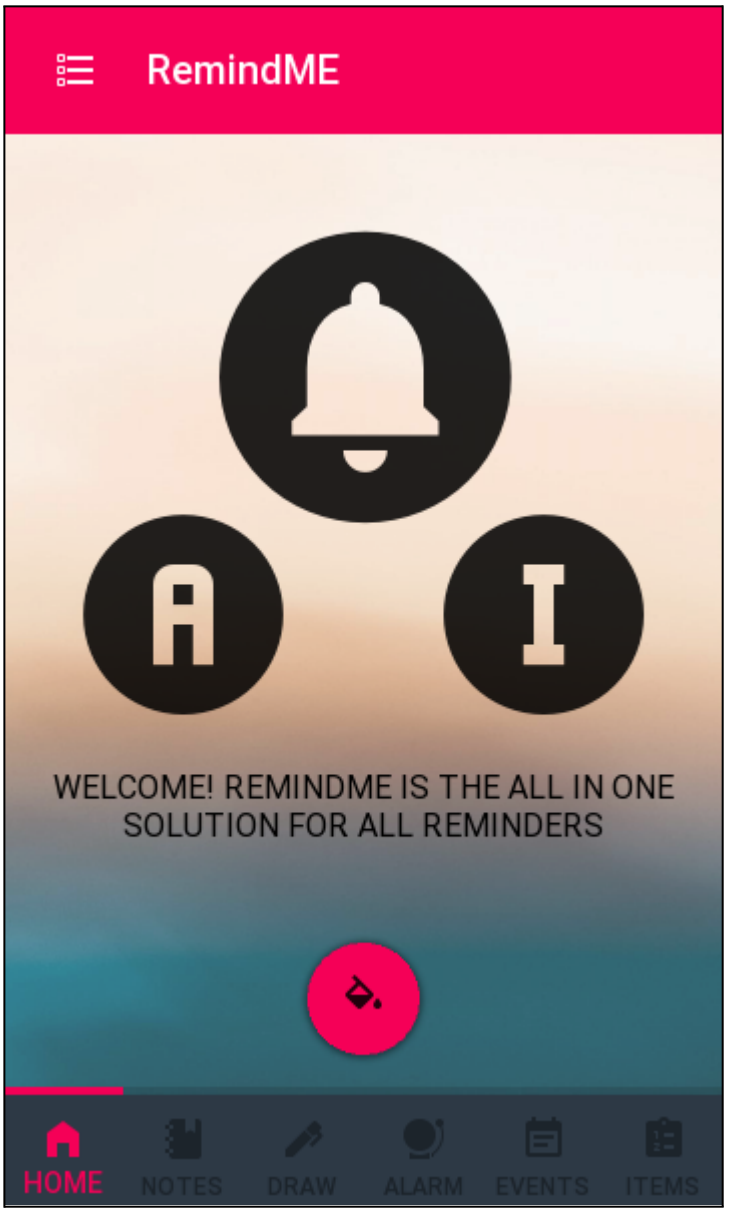
```
                    is scrollable):- type
                    item and amount and then tap add item button\nCHANGING THEME
                    COLOR:- \ntap paint button
                    on home screen:- Primary is for top and bottom toolbar, Accent is
                    for buttons",
                buttons=[
                    MDFlatButton(
                        text="CONTINUE", text_color=self.theme_cls.primary_color,
                        on_release=self.close_dialog
                    ),
                ], size_hint=(0.9, 1)
            )
        self.dialog.open()

    t = 1

    def show_data(self):  # for list
        self.item = MDRectangleFlatButton(text=self.itembtn.text,
                    text_color=self.theme_cls.accent_color,
                                          size_hint_y=None,
                                          height=40)
        self.amount = MDRectangleFlatButton(text=self.amountbtn.text,
                      text_color=self.theme_cls.accent_color,
                                            size_hint_y=None,
                                            height=40)
        self.no_text = MDLabel(text=str(self.t) + ".",
                       text_color=self.theme_cls.accent_color,
                               size_hint=(None, None),
                               width=50, height=40)
        self.root.ids.grid.add_widget(self.no_text)
        self.root.ids.grid.add_widget(self.item)
        self.root.ids.grid.add_widget(self.amount)
        self.itembtn.text = ""
        self.amountbtn.text = ""
        global t
        self.t += 1
        return

    def close_dialog(self, obj):
        self.dialog.dismiss()

    def paint_clear(self, obj):
        self.painter.canvas.clear()

    def show_theme_picker(self):
        theme_dialog = MDThemePicker()
        theme_dialog.open()
```

```
def navigation_draw(self):
    print("Navigation")

def show_time_picker(self):
    time_dialog = MDTimePicker()
    time_dialog.bind(time=self.get_time)
    time_dialog.open()

l = ListProperty([])

def get_time(self, instance, time):
    print(time)
    l.append(time)  # use of list

    if len(l) == 5:
        note = Builder.load_string(new_description)
        alarm = MDRectangleFlatButton(text=str(len(l)) + ') ' + str(l[len(l) - 1]))
        self.label = MDLabel(text="SCROLL DOWN:", font_style='Caption',
        size_hint_y=None, height=5)
        self.root.ids.time.add_widget(self.label)
        self.root.ids.time.add_widget(alarm)
        self.root.ids.time.add_widget(note)
    else:
        note = Builder.load_string(new_description)
        alarm = MDRectangleFlatButton(text=str(len(l)) + ') ' + str(l[len(l) - 1]))
        self.root.ids.time.add_widget(alarm)
        self.root.ids.time.add_widget(note)
l1 = ListProperty([])
def get_date(self, date):
    '''
    :type date: <class 'datetime.date'>
    '''
    print(date)
    l1.append(date)  # use of list
   if len(l1) == 5:
        note = Builder.load_string(new_description)
        event = MDRectangleFlatButton(text=str(len(l1)) + ') '+str(l1[len(l1) - 1]))
        self.label = MDLabel(text="SCROLL DOWN:", font_style='Caption',
                             size_hint_y=None, height=5)
        self.root.ids.date.add_widget(self.label)
        self.root.ids.date.add_widget(event)
        self.root.ids.date.add_widget(note)

    else:
        note = Builder.load_string(new_description)
        event = MDRectangleFlatButton(text=str(len(l1)) + ') '+str(l1[len(l1) - 1]))
```

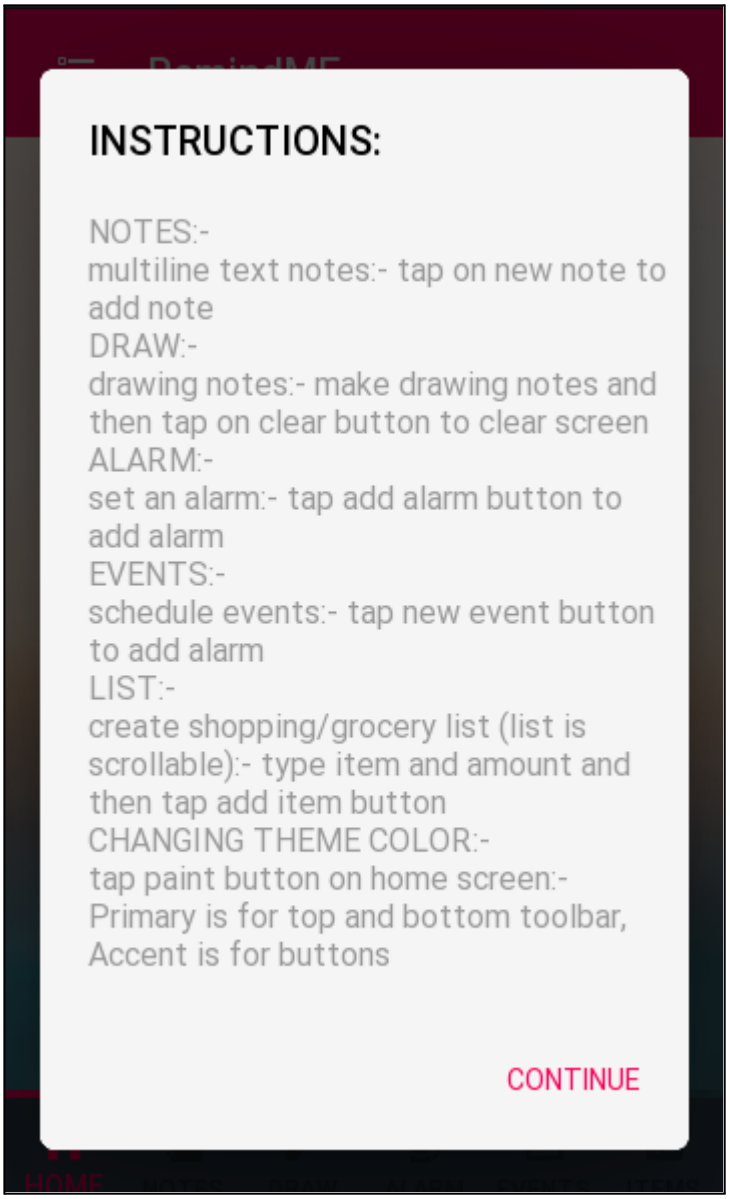
```
            self.root.ids.date.add_widget(event)
            self.root.ids.date.add_widget(note)
    def show_date_picker(self):
        date_dialog = MDDatePicker(callback=self.get_date)
        date_dialog.open()
    z = 1

    def new_note(self):
        if self.z == 6:
            self.label = MDLabel(text="SCROLL DOWN:", font_style='Caption',
                                 size_hint_y=None, height=40)
            note = Builder.load_string(new_note)
            self.root.ids.notes.add_widget(self.label)
            self.root.ids.notes.add_widget(note)
            self.z += 1
        else:
            note = Builder.load_string(new_note)
            self.root.ids.notes.add_widget(note)
            self.z += 1

    def check_press(self, instance_table, current_row):
        print(instance_table, current_row)

    def row_press(self, instance_table, instance_row):
        print(instance_table, instance_row)

    def btn_pressed(self):
        self.sound1 = SoundLoader.load('12910_sweet_trip_mm_clap_mid.wav')
        if self.sound1:
            self.sound1.play()
        print("sound")

    def play_sound(self):
        self.sound = SoundLoader.load('12911_sweet_trip_mm_hat_cl.wav')
        self.sound.play()
        print("sound")

DraftApp().run()
```

OUTPUT SCREENSHOTS:

HOME SCREEN:

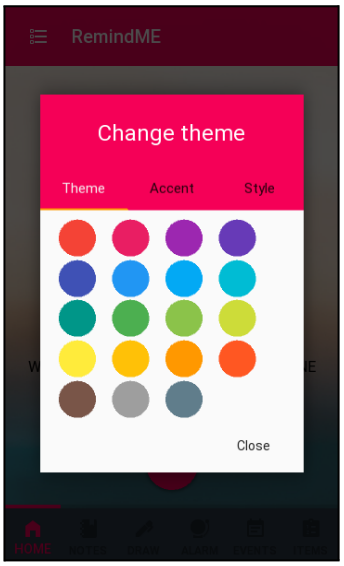


INSTRUCTIONS DIALOG BOX:

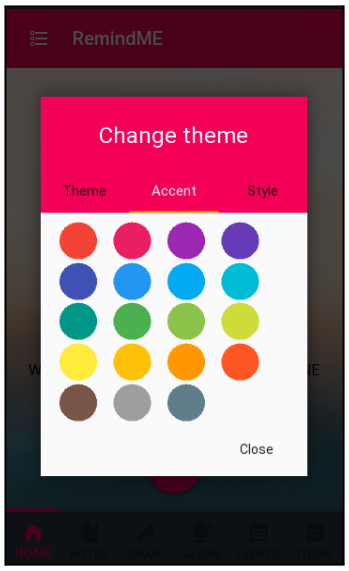


CHANGING THEME:

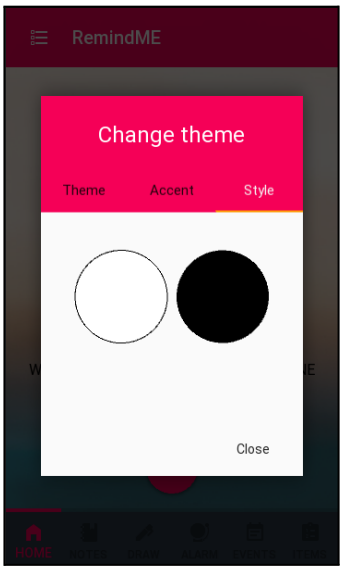
(TO CHANGE APP THEME COLOR)



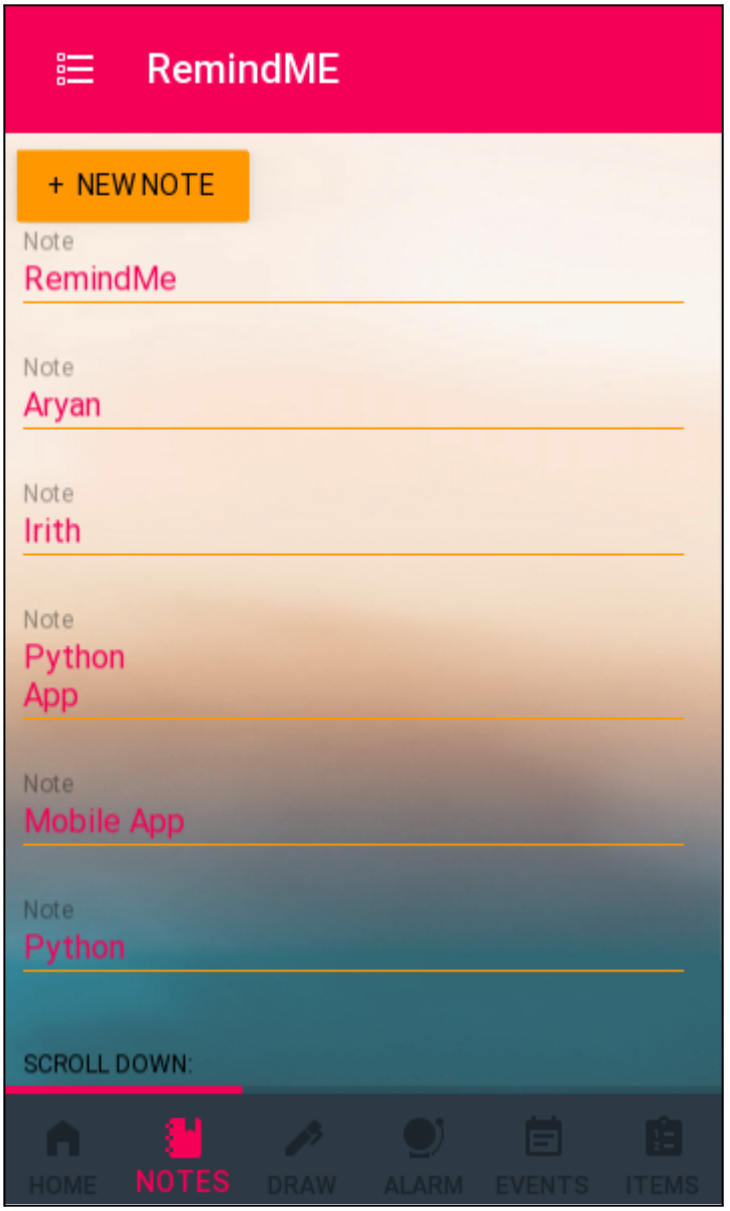
(TO CHANGE COLOR OF BUTTONS)



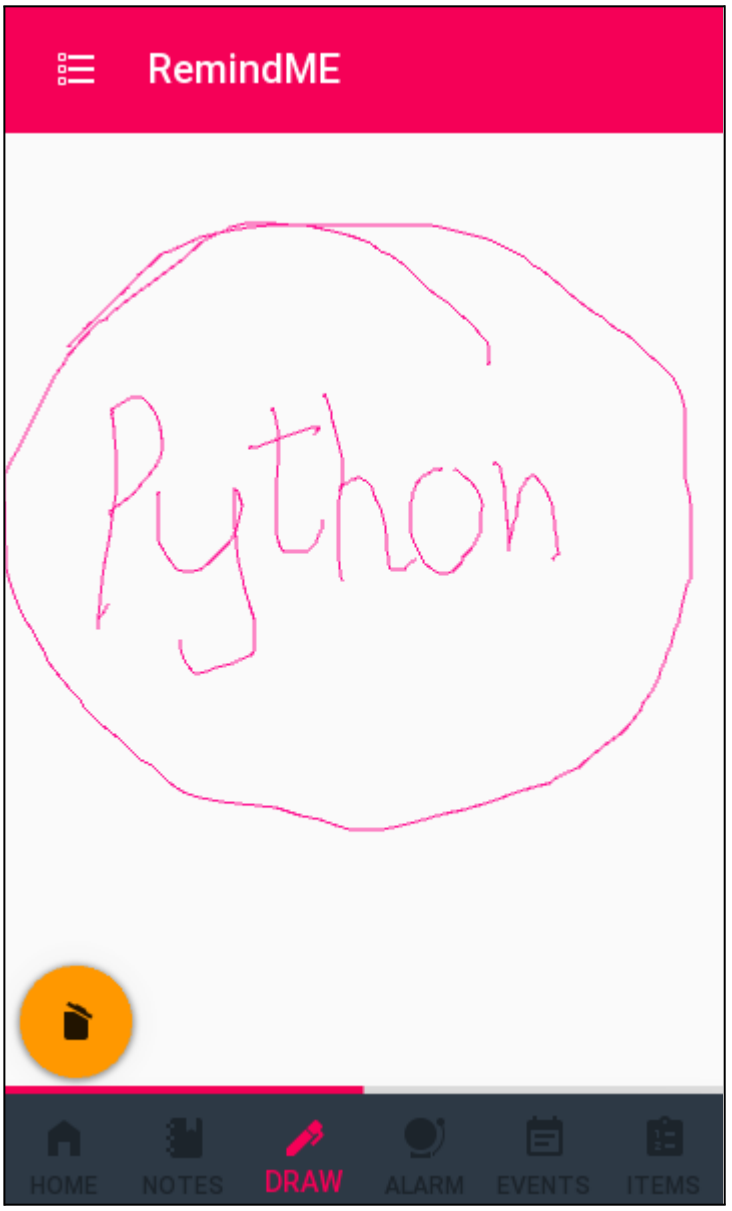
(TO SWITCH BETWEEN LIGHT AND DARK THEMES)



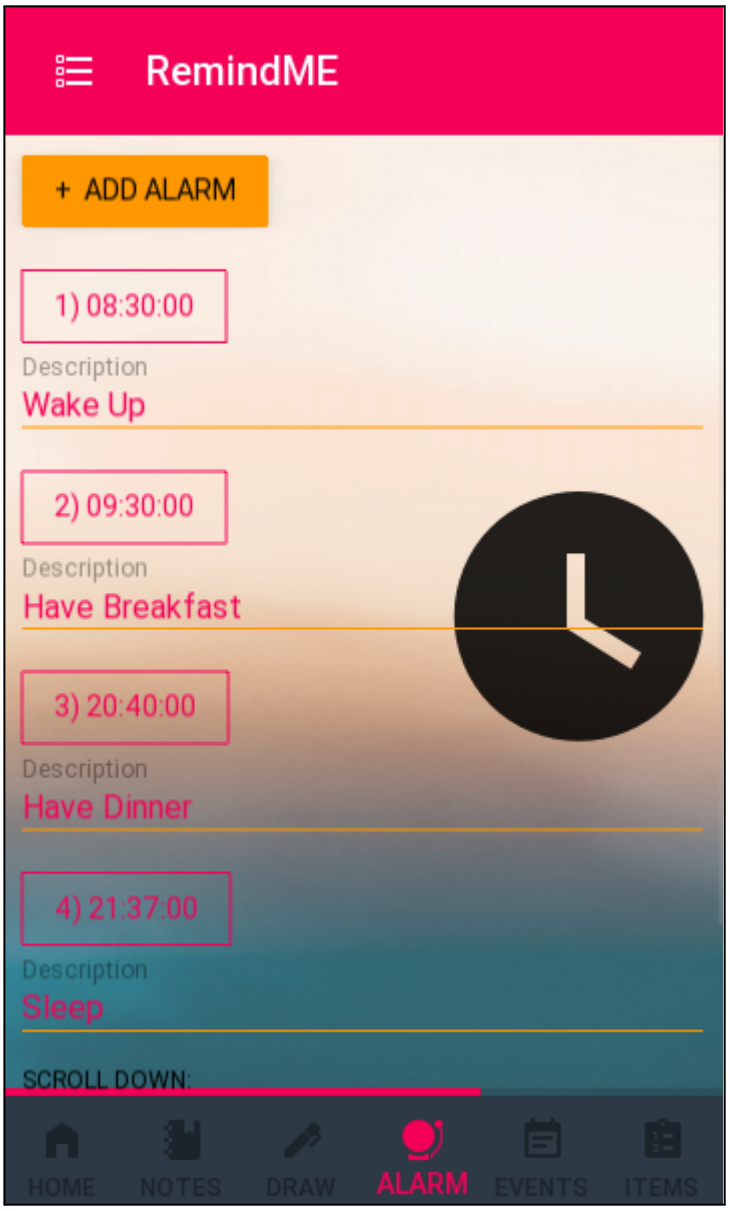
NOTES SCREEN:



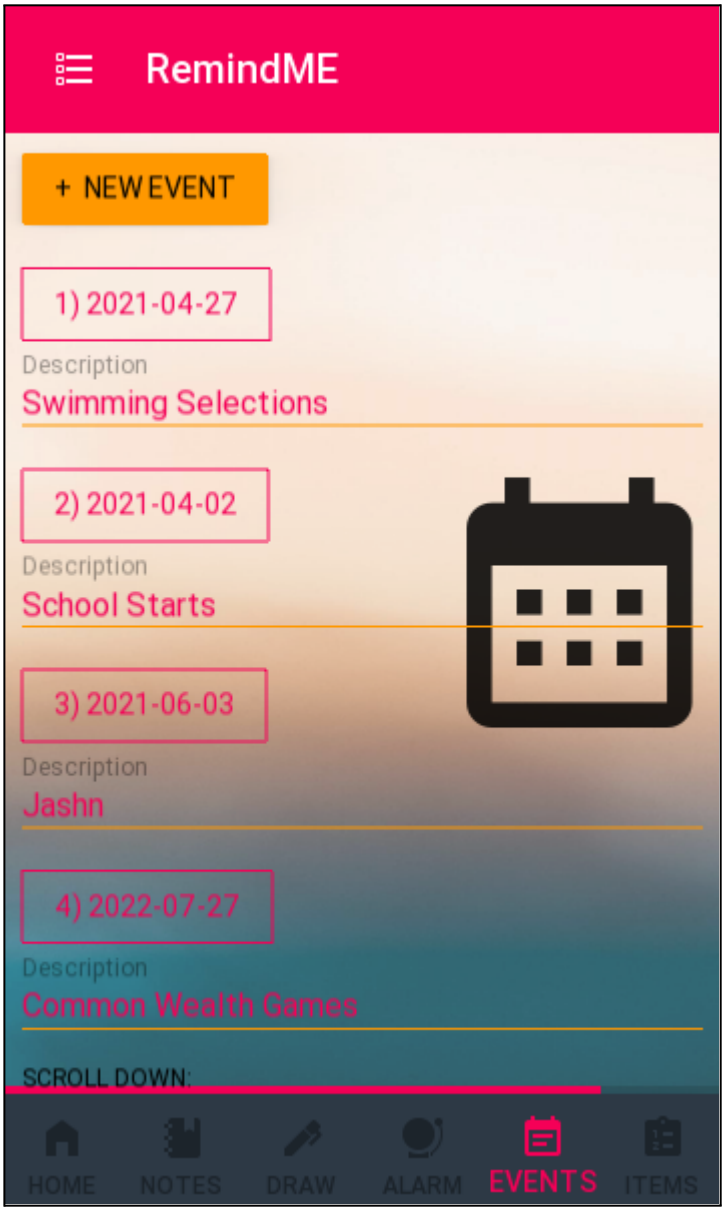
DRAWING SCREEN:



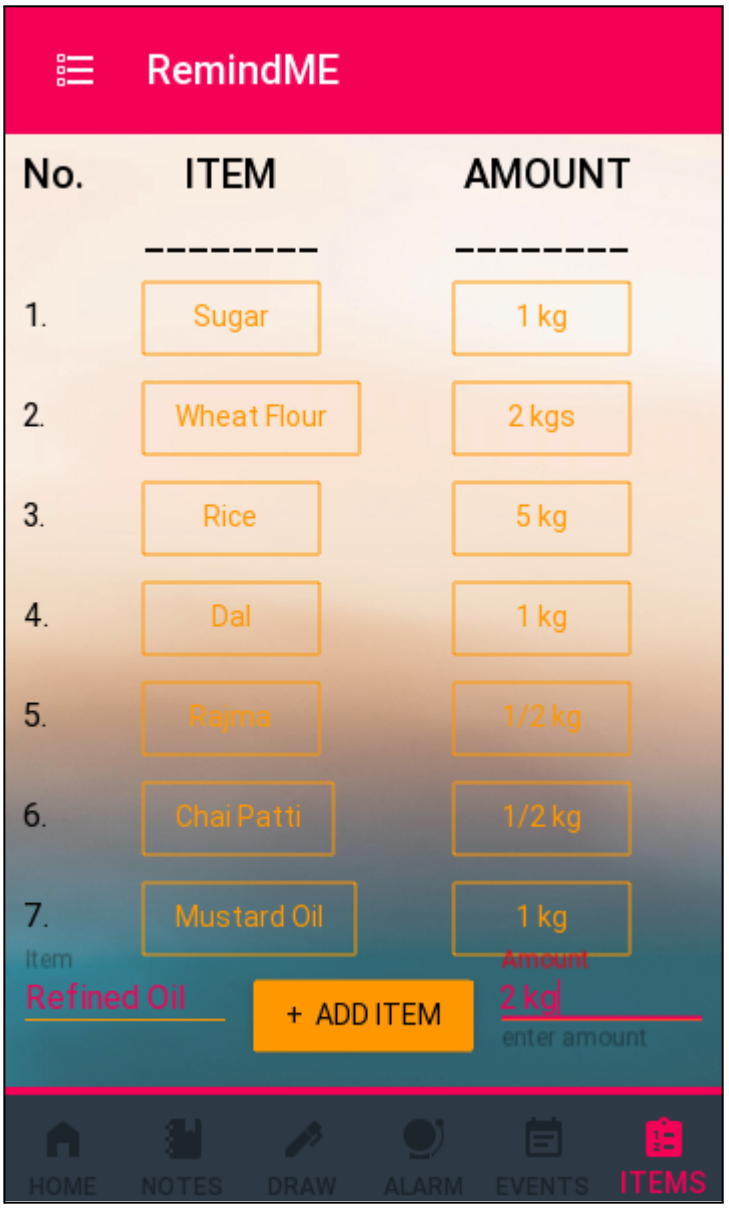
ALARM SCREEN:



EVENTS SCREEN:

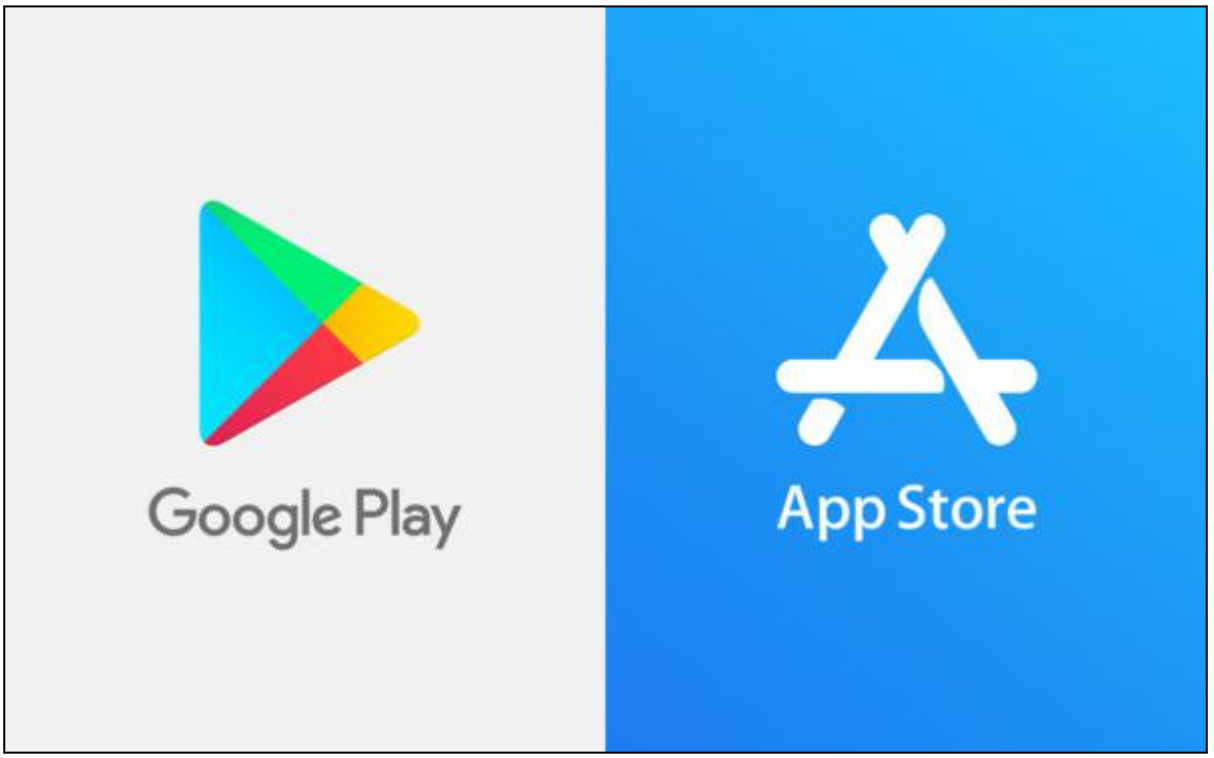


ITEM LIST SCREEN:



FUTURE PROSPECTS:

After the submission of the project I will be developing the app further, giving it the required final touches and then launching it on platforms such as the Google Play Store and the Apple App Store for the consumer market. I also believe that our work will help every user in making their lives easier as all the apps that are required for daily use have been combined into a single app - 'RemindME' , making things more organized for everyone.



BIBLIOGRAPHY:

Attreya Bhatt (Youtube Tutorials):
https://www.youtube.com/watch?v=LRXo0juuTrw&list=PLhTjy8cBISEoQQLZ9IBlVlr4WjVoStmy-

Kivymd Documentation:

[Welcome to KivyMD's documentation! — KivyMD 0.104.2.dev0 documentation]