```python
# modules
import flet
from flet import *
import re

# set two lists where some controls will be stored
CONTROLS = []
STATUS = []

# set up a decorator function...
# we'll be using this to easily stored and subsequnetly retrieve the controls
we want to animate/change
def store_control(function):
    def wrapper(*args, **kwargs):
        reference =function(*args, **kwargs)
        # the control, which we set to 0, is part of the **kwargs parameters.
        # so if we call kwargs['control'], we'll get whatever we set that
control to.
        # here, to make the code concise, we will dividing the UI in two parts,
        # we will store the strength UI controls in one list, while we store
the status check boxes in another, If the control integer is 0, we place the
controls in the CONTROLS list.
        if kwargs["control"]== 0:
            CONTROLS.append(reference)
        else:
            STATUS.append(reference)

        return reference

    return wrapper


#
#

# create the logic
class PasswordStrengthChecker:
    def __init__(self, password):
        self.password = password
        # we'll be calling different parts of this class while checking the
password.

    # check password length
    def length_check(self):
        # when we call this function, we return integers depending on the
length of the input, based on these numbers, we will create a UI that shows the
strength of the password
        length = len(self.password)
        if length > 0 and length < 8:
             return 0
        elif length >= 8 and length < 12:
             return 1
```

```
        elif length >= 12 and length < 16:
             return 2
        elif length >= 16:
             return 3

    # check password character types
    def character_check(self):
        characters = set(self.password)
        lower_case = set("abcdefghijklmnopqrstuvwxyz")
        upper_case = set("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
        digits = set("0123456789")
        special_characters = set("!@#$%^&*)(_+-}{=][:;><,./?'|""\`~")
        # we'll be checking to see if the current password contians the above
criterira

        score = 0
        # here we're seeing if ANY lower case sets are in the password
        # if so, we increment the score by 1
        # note: characters in this loop is the password.split!
        if any(char in lower_case for char in characters):
            score +=1
        if any(char in upper_case for char in characters):
            score +=1
        if any(char in digits for char in characters):
            score +=1
        if any(char in special_characters for char in characters):
            score +=1

        if score == 1:
            return 0
        if score == 2:
            return 1
        if score == 3:
            return 2
        if score == 4:
            return 3

    #check for any repititions
    def repeat_check(self):
        if len(self.password) == 0:
            return (
                4 #
            )
        else:
            for i in range(len(self.password) - 1):
                # if self.password[i] == self.password[i+1] or self.password[i]
== self.password[i+2] or self.password[i] == self.password[i+3] or
self.password[i] == self.password[i+4] or self.password[i] ==
self.password[i+5]:
                if self.password[i] == self.password[i+1]:
                    return 0 # this means there is repeat of string in range of
2
            for i in range(len(self.password) - 2):
```

```
                if self.password[i] == self.password[i+1] or self.password[i]
== self.password[i+2]:
                    return 0 # this means there is repeat of string in range of
3
            for i in range(len(self.password) - 3):
                if self.password[i] == self.password[i+1] or self.password[i]
== self.password[i+2] or self.password[i] == self.password[i+3]:
                    return 1 # this means there is repeat of string in range of
4
            for i in range(len(self.password) - 4):
                if self.password[i] == self.password[i+1] or self.password[i]
== self.password[i+2] or self.password[i] == self.password[i+3] or
self.password[i] == self.password[i+4]:
                    return 2 # this means there is repeat of string in range of
5
            return 3 # if there is n repeat of string


    # check for any reoccuring sequences
    '''def sequential_check(self):
        if len(self.password) == 0:
            return 2
        else:
            for i in range(len(self.password) - 3):
                # here, we check the password within a range of 4 characters,
if for example, 4 characters are numbers, we return a weak status (in the UI),
if it's a squence of 4 lower case letters, we also return a weak status, and so
on and so forth
                if (
                    (self.password[i:i+3].isdigit())
                ):
                   return 0
                elif (
                   (self.password[i:i+3].islower())
                ):
                    return 0
                elif (
                   (self.password[i:i+3].isupper())
                ):
                    return 0
        return 1'''

    def sequential_check(self):
        if len(self.password) == 0:
            return 2
        else:
            pattern = re.compile(r'([a-z]{4})|([A-Z]{4})|(\d{4})')
        if pattern.search(self.password):
            return 0
        else:
            return 1


# start of UI part
```

```
class AppWindow(UserControl):
    def __init__(self):
        super().__init__()

    # here we need a function to call all the above class inner functions
    def check_password(self, e):
        password_strength_checker = PasswordStrengthChecker(e.data)
        #call the length checker
        password_length = password_strength_checker.length_check()
        self.password_length_status(password_length) # pass in the return here,
we'll use it for each criteria
        # call the character checker
        character_checker = password_strength_checker.character_check()
        self.character_check_status(character_checker)
        # call the repeat check
        repeat_check = password_strength_checker.repeat_check()
        self.repeat_check_status(repeat_check)
        # call the squential check
        sequential_check = password_strength_checker.sequential_check()
        self.sequential_check_status(sequential_check)

    # To show if criteria satisfied...
    def criteria_satisfied(self, index, status):
        # here, we only need to take in the required strenth, which is 3,
Anything else the criteria is NOT satisfied.
        if status == 3:
            STATUS[index].controls[0].offset = transform.Offset(0, 0)
            STATUS[index].controls[0].opacity = 1
            STATUS[index].controls[0].content.value = True
            STATUS[index].controls[0].update()
        else:
            STATUS[index].controls[0].content.value = False
            STATUS[index].controls[0].offset = transform.Offset(-0.5, 0)
            STATUS[index].controls[0].opacity = 0
            STATUS[index].controls[0].update()

    # let's start with showing the length strength
    def password_length_status(self, strength):
        if strength == 0:
            # now, we can use our list to access the controls
            CONTROLS[0].controls[1].controls[0].bgcolor = "red"
            CONTROLS[0].controls[1].controls[0].width = 40
        elif strength == 1:
            CONTROLS[0].controls[1].controls[0].bgcolor = "yellow"
            CONTROLS[0].controls[1].controls[0].width = 60
        elif strength == 2:
            CONTROLS[0].controls[1].controls[0].bgcolor = "green400"
            CONTROLS[0].controls[1].controls[0].width = 80
        elif strength == 3:
            CONTROLS[0].controls[1].controls[0].bgcolor = "green900"
            CONTROLS[0].controls[1].controls[0].width = 100
        else:
            CONTROLS[0].controls[1].controls[0].width = 0
```

```
        CONTROLS[0].controls[1].controls[0].opacity = 1
        CONTROLS[0].controls[1].controls[0].update()

        # here we'll show a more cleaner way of handling these data.
        # because we know where the controls are and their indexes, plus we
know when to show the OK checkbox, i.e. when the strength is 3,
        # we can create a single function that will handle this.
        # we can simply pass the index and the strength to the function, and it
will do the rest
        return self.criteria_satisfied(0, strength)

    def character_check_status(self, strength):
        if strength == 0:
            # now, we can use our list to access the controls
            CONTROLS[1].controls[1].controls[0].bgcolor = "red"
            CONTROLS[1].controls[1].controls[0].width = 40
        elif strength == 1:
            CONTROLS[1].controls[1].controls[0].bgcolor = "yellow"
            CONTROLS[1].controls[1].controls[0].width = 60
        elif strength == 2:
            CONTROLS[1].controls[1].controls[0].bgcolor = "green400"
            CONTROLS[1].controls[1].controls[0].width = 80
        elif strength == 3:
            CONTROLS[1].controls[1].controls[0].bgcolor = "green900"
            CONTROLS[1].controls[1].controls[0].width = 100
        else:
            CONTROLS[1].controls[1].controls[0].width = 0

        CONTROLS[1].controls[1].controls[0].opacity = 1
        CONTROLS[1].controls[1].controls[0].update()

        return self.criteria_satisfied(1, strength)

    def repeat_check_status(self, strength):
        if strength == 0:
            CONTROLS[2].controls[1].controls[0].bgcolor = "red"
            CONTROLS[2].controls[1].controls[0].width = 40
        elif strength == 1:
            CONTROLS[2].controls[1].controls[0].bgcolor = "yellow"
            CONTROLS[2].controls[1].controls[0].width = 60
        elif strength == 2:
            CONTROLS[2].controls[1].controls[0].bgcolor = "green400"
            CONTROLS[2].controls[1].controls[0].width = 80
        elif strength == 3:
            CONTROLS[2].controls[1].controls[0].bgcolor = "green900"
            CONTROLS[2].controls[1].controls[0].width = 100
        else:

            CONTROLS[2].controls[1].controls[0].width = 0
        CONTROLS[2].controls[1].controls[0].opacity = 1
        CONTROLS[2].controls[1].controls[0].update()

        return self.criteria_satisfied(2, strength)
```

```
    # sequential check
    def sequential_check_status(self, strength):
        if strength == 0:
            CONTROLS[3].controls[1].controls[0].bgcolor = "red"
            CONTROLS[3].controls[1].controls[0].width = 50
        elif strength == 1:
            CONTROLS[3].controls[1].controls[0].bgcolor = "green900"
            CONTROLS[3].controls[1].controls[0].width = 100
        else:
            CONTROLS[3].controls[1].controls[0].width = 0

        CONTROLS[3].controls[1].controls[0].opacity = 1
        CONTROLS[3].controls[1].controls[0].update()

        if strength == 1:
            strength = 3
            return self.criteria_satisfied(3, strength)
        else:
            return self.criteria_satisfied(3, strength)

    # we'll handle the strength UI/logic
    # now we start to outline the checking process for the Password
    #
    @store_control
    def check_criteria_display(self, criteria, description, control:int):
        return Row(
            alignment=MainAxisAlignment.SPACE_BETWEEN,
            vertical_alignment=CrossAxisAlignment.CENTER,
            spacing=5,
            controls=[
                Column(
                    spacing=2,
                    controls=[
                        Text(value=criteria, size=16, weight="bold"),
                        Text(value=description, size=12, weight="white54"),
                    ],
                ),
                # add the strength container
                Row(
                    spacing=0,
                    alignment=MainAxisAlignment.START,
                    controls=[
                        Container(
                            height=5,
                            opacity=0,
                            animate=350,
                            border_radius=10,
                            animate_opacity=animation.Animation(350,
"decelerate"),
                        )
                    ]
                )
            ],
        )
```

```
    # for the final UI, we can add a checkbox when the user has a strong
password for either of the criteria...
    #also don't forget to decorate the function
    @store_control
    def check_status_display(
        self, control: int
    ): #note; we need to pass the control int here to store these controls in a
sperate list
        return Row(
            alignment=MainAxisAlignment.END,
            controls=[
                Container(
                    opacity=0,
                    offset=transform.Offset(-0.5, 0),
                    animate_offset=animation.Animation(700, "decelerate"),
                    animate_opacity=animation.Animation(700, "decelerate"),
                    border_radius=50,
                    width=21,
                    height=21,
                    alignment=alignment.center,
                    content=Checkbox(
                        scale=Scale(0.7),
                        fill_color="#7df6dd",
                        check_color="black",
                        disabled=True,
                    ),
                )
            ],
        )

    # main display area where checks are done
    def password_strength_display(self):
        return Container(
            width=350,
            height=425,
            bgcolor="#1f262f", ##4C5159
            border_radius=10,
            padding=10,
            clip_behavior=ClipBehavior.HARD_EDGE,
            content=Column(
                horizontal_alignment=CrossAxisAlignment.CENTER,
                spacing=3,
                controls=[
                    Divider(height=5, color="transparent"),
                    Text("Password Strength Checker", size=25, weight='bold'),
                    Text(
                        "Type in a password and see how strong it is!",
                        size=15,
                        color="white54",
                        weight="w400"
                    ),
                    Divider(height=25, color="transparent"),
                    # place each criteria here
```

```
                    self.check_criteria_display(
                        "1. Length check",
                        "Strong password are 12 characters or above",
                        # this int here is going to be passed into the inner
function of the decorator above.
                        control=0,
                    ),
                    self.check_status_display(control=1),
                    Divider(height=10, color="transparent"),
                    self.check_criteria_display(
                        "2. Character Check",
                        "Upper, Lower and Special Characters",
                        control=0,
                    ),
                    self.check_status_display(control=1),
                    Divider(height=10, color="transparent"),
                    self.check_criteria_display(
                        "3. Repeat Check",
                        "Checking for any repetitions...",
                        control=0,
                    ),
                    self.check_status_display(control=1),
                    Divider(height=10, color="transparent"),
                    self.check_criteria_display(
                        "4. Sequential Check",
                        "Checking for sequences...",
                        control=0,
                    ),
                    self.check_status_display(control=1),
                ],
            ),
        )

    # User input field
    def password_text_field_display(self):
        return Row(
            spacing=20,
            vertical_alignment=CrossAxisAlignment.CENTER,
            controls=[
                Icon(name=icons.LOCK_OUTLINE_ROUNDED, size=14, opacity=0.85),
                #
                TextField(
                    border_color="transparent",
                    bgcolor="transparent",
                    height=20,
                    width=200,
                    text_size=12,
                    content_padding=3,
                    cursor_color="grey",
                    cursor_width=1,
                    color="grey",
                    hint_text="Start typing a password ...",
                    hint_style=TextStyle(
                        size=12,
```

```
                    ),
                    #
                    on_change=lambda e: self.check_password(e),
                    password=True,
                )
            ],
        )

    # user input field
    def password_input_display(self):
        return Card(
            width=275,
            height=55,
            elevation=12,
            offset=transform.Offset(0, -0.25),
            content=Container(
                padding=padding.only(left=15),
                content=Row(
                    alignment=MainAxisAlignment.SPACE_BETWEEN,
                    controls=[
                        # create a seperate function for the input field...
                        self.password_text_field_display(),
                        #IconButton(icon=icons.REMOVE_RED_EYE_OUTLINED,
icon_size=16),
                    ],
                ),
            ),
        )

    def build(self):
        return Card(
            elevation=20,
            content=Container(
                width=470,
                height=500,
                border_radius=10,
                bgcolor="#1f262f",
                content=Column(
                    spacing=0,
                    horizontal_alignment=CrossAxisAlignment.CENTER,
                    controls=[
                        # add main classes here...
                        self.password_strength_display(),
                        self.password_input_display(),
                    ],
                ),
            ),
        )


def main(page: Page):
    page.horizontal_alignment = "center"
    page.vertical_alignment = "center"
    #
```

```
    page.add(AppWindow())
    page.update()


if __name__ == "__main__":
    flet.app(target=main) #view=flet.WEB_BROWSER
```