

# ***A Project Report on “Cue the Schedule Manager”***

Submitted by:

19BCA41001 [Sylvester Correya]  
19BCA41002 [Joel Liao Liang Chyang]  
19BCA41024 [Pooja K J]  
19BCA41065 [Thrupthi R]

Under the Guidance of:  
Professor. Dr. B. G. Prashanthi

Department of Computer Science

Date/ December /2021



Department of Computer Science  
St. Joseph's College (Autonomous)  
36, Lalbagh Road, Bengaluru – 560027.

*[Individual Copy]*

# *Certificate*



This is to certify that Joel Liao Liang Chyang (Reg no: 19BCA41002) have successfully completed the project titled “Cue the Schedule Manager” at St. Joseph’s College (Autonomous) under the supervision and guidance in the fulfillment of requirements of the fifth semester, Bachelor of Science (Computer Science) of Bengaluru City University, Bengaluru.

*Prof: Dr. B. G. Prashanthi*

*Prof: Sandhya*  
*Head of the Department*  
*Computer Science*

*[Group Copy]*

## *Certificate*



This is to certify that Sylvester Correya (Reg no: 19BCA41001), Joel Liao Liang Chyang (Reg no: 19BCA41002), Pooja K J (Reg no: 19BCA41024), Thrupthi R (Reg no: 19BCA41065), have successfully completed the project titled “Cue the Schedule Manager” at St. Joseph’s College (Autonomous) under the supervision and guidance in the fulfillment of requirements of the fifth semester, Bachelor of Science (Computer Science) of Bengaluru City University, Bengaluru.

*Prof: Dr. B. G. Prashanthi*

*Prof: Sandhya*  
*Head of the Department*  
*Computer Science*

# *Acknowledgment*

We thank God almighty for giving us strength and guidance in working towards the completion of this project.

We would like to thank the Department of Computer Science and our guide Dr. B. G. Prashanthi for giving us the opportunity to learn and work on a project as a team.

We are grateful for all the support from our family and friends during the time in doing our project, we are also grateful to Joel Asher who as a friend gave us lots of advice and input towards the development of our software.

We as a team have learned a lot more after doing this project., such as we got more familiar with designing a front end using a python framework (PyQt5), we learned more about how the backend can insert, update, get and delete data in the database.

This project was successfully completed with the joint efforts of everyone in the team, The documentation of this project was done by Joel Liao Liang Chyang. The front-end designs were done by Pooja J K and Thrupthi R, they were always ready to constantly work on re-designing to make the home screen and other forms more user interactive and appealing. PyQt5 a python framework was used for designing the front end of Cue.

The coding aspects of this project were done by Sylvester Correya, Joel Liao Liang Chyang, they have done the backend code of this software using python and sqllite3.

# *Cue*

## *Software Requirements Specification*

### *Introduction to Project Cue*

#### *The Purpose of Cue*

- Cue is a desktop software to help create, view, edit and delete reminders for various events as desired by the user.
- These reminders can give out notifications to the user to remind them of any event in advance.
- It also has functions to create, view, edit and delete notes according to the users.

#### *Intended audience and Reading suggestions*

This Software Requirement Specification Document is for the software developers to refer to it as a blueprint or roadmap of our Software Cue.

This is also for all those third-party people who wish to know all the various functions that exist in this software.

#### *SRS Document Conventions*

This SRS document will have a separate set of content for the users and software developers each will be highlighted by a specific color.

- The content for the **users** will have the headings marked as **Blue**.
- The content for the **software developers** and technical users will have the headings marked in **Red**.
- The content for **both users** and the developers will have the headings colored **Green**.
- The **Highlighted** Content is for the **software developers**, these are points to remember.
- All Algorithms and code will be highlighted as silver

#### *Project Scope*

This project is for us students to get some experience in the development of software and for the Minor Project requirement for our Bachelor's Degree by our College.

Cue is personal software and is to not be used for any commercial purposes.

### *Overall Software Description*

#### *Features in the Software*

##### *Schedules*

1. Allows users to **create, view, edit and delete Schedules** or also called Reminders to remind them later in the future.
2. When deleting schedules users are prompted for confirmation for deletion.
3. The **Notifications for these schedules/ reminders can be toggled either** on or off.

4. When viewing the schedules, it is possible to search/ filter them based on the user's date/Month/Year input.
5. When the user creates or edits a schedule, they have an option to save and close the window and if they close the window by mistake without saving then they are given a pop-up where they are asked if they would like to save the Schedule or discard it.
6. When the user first opens the software, the Schedules for the present day are listed on the home screen.
7. These reminders can be set to remind users earlier before the schedule is to take place in advance.

### Notes

1. Also allows users to create simple notes where an individual note will consist of a Name/ Heading for the note and then the context of the note or description.
2. There is a view option wherein these notes can be viewed, edited, and deleted.
3. When the user creates or edits a note, they have an option to save and close the window and if they close the window by mistake without saving then they are given a pop-up where they are asked if they would like to save the note or discard it.
4. When the User wishes to view the notes in the view notes screen all the notes will be listed at the left wherein the user can select and view each of them one at a time.
5. When notes are deleted, the users are prompted for confirmation for the deletion of the notes.

### Others

1. There is also an option for the users to open the file location of where the Schedules, Notes, and program files are located in their system.
2. There is an option where users can access and read documents such as SRS documentation, Step-by-step Guide for using Cue.

### Minimum System Requirements for Cue

Operating System: Windows 7 or above / MacOS 10.11 or above/ Linux

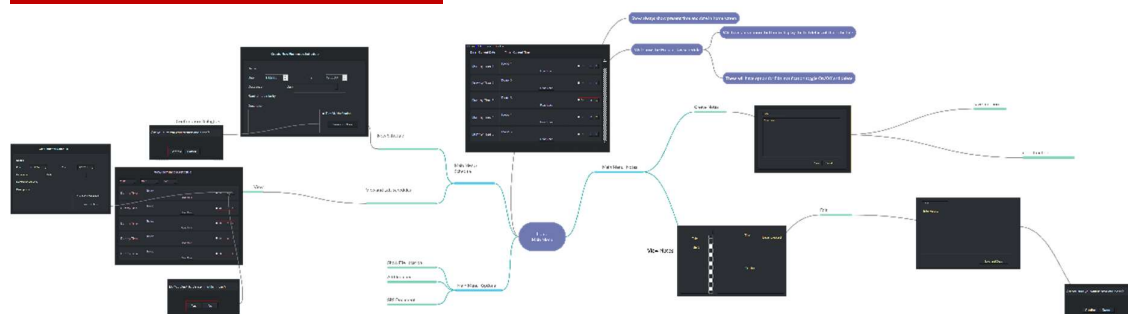
Intel HD Integrated graphics

Processor: Intel Core2Duo Equivalent or Higher

RAM: 2GB

Disk Space: 1GB

### Mind map of the Software



### Distribution of tasks among the Software Developing team

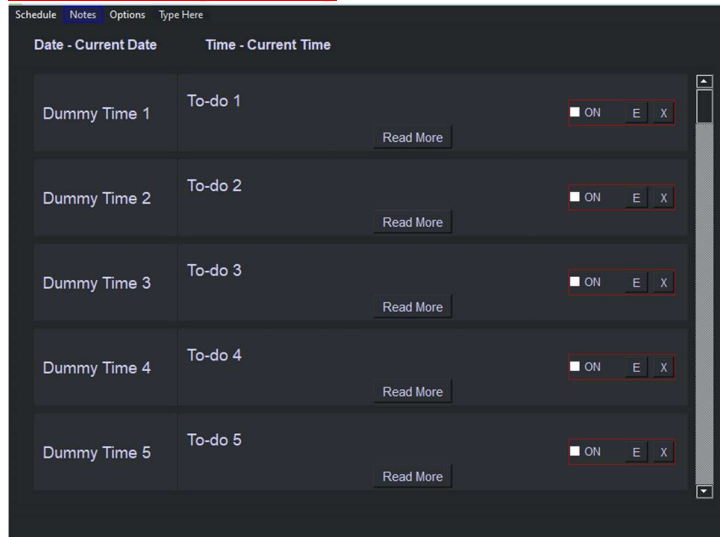
#### Team Members

- Sylvester Correya [19BCA41001]
- Joel Liao Liang Chyang [19BCA41002]

- Pooja K J [19BCA41024]
- Thrupthi R [19BCA41065]

## **Functions of the Software**

### **Home Window [Joel Liao]**



#### **1. current\_date\_time()**

```
date_str=now.toString(Qt.DefaultLocaleLongDate)
```

This function will always run while the Home Window is open and will update the time every minute.

#### **2. today\_schedule\_display(self, date)**

the function will display today's schedule on the Home Window and within each of these schedules there are buttons with various functions as shown below:

**Variables:** `date=now.toString(Qt.DefaultLocaleLongDate)`

`schedule_query=step1`

**Step1: Get the Schedule from the database**

```
get_schedule(self, date)
```

**Step1- get the data from the Database based on today's date**

**Step2- Return the pulled data from database**

```
return schedule_query()
```

**Step2: Display the Schedule retrieved from the database**

```
class schedule_frame:
```

```
def __init__(self,)
```

**The generate function will create the buttons and labels**

```
def generate(self, schedule_title, schedule_date, schedule_readmore,
            schedule_edit, schedule_delete, schedule_notifytoggle):
```

```
self.schedule_title=
```

```
self.schedule_date=
```

```
self.schedule_readmore=
```

```
self.schedule_edit=
```

```
self.schedule_delete=
```

```
self.schedule_notifytoggle=
```

The `gen_val` function will assign values to the buttons and labels using the `translate` function

```
def gen_val(self, schedule_title, schedule_date, schedule_readmore,
            schedule_edit, schedule_delete, schedule_notifytoggle):
    self.schedule_title.setText(self.translate(""))
    self.schedule_date.setText(self.translate(""))
    self.schedule_readmore.setText(self.translate(""))
    self.schedule_edit.setText(self.translate(""))
    self.schedule_delete.setText(self.translate(""))
    self.schedule_notifytoggle.setValue(self.translate(""))
```

**Step3: create an object for the class `schedule_frame()`**

**Step4: display the frame using the object in a loop**

**Step3: End of this Function**

**today\_schedule\_display()**

**a) button\_schedule\_readmore()**

This function will give a popup window wherein the data relating to the schedule is pulled from the database and displayed as follow

Lib.py

```
class Ui_MainWindow(object):
    def clickedButtonReadMore(self, *args, **kwargs):
        #put in code to call the window wherein the full details can be displayed
        self.pushButton_ReadMore.setText(self._translate("MainWindow", "Done"))

    def setupUi(self, MainWindow):
        self.pushButton_ReadMore.clicked.connect(self.clickedButtonReadMore)
```

**[form to be created]**

**b) schedule\_notify\_switch()**

This is a Toggle button or can be a normal button where it calls in a function to turn the flag for notification of this schedule OFF/ False.

```
class Ui_MainWindow(object):
    def clicked_notify_toggle (self, *args, **kwargs):
        """
        put in function to call the window wherein the toggle button value can be changed on interacting with it
        and return the value
        True/or False by checking the current status of the toggle button
        """
        self.pushButton_ReadMore.setValue(self._translate("MainWindow", "True/or False by checking the
        current status of the toggle button"))

    def setupUi(self, MainWindow):
        self.notify_toggle.clicked.connect(self.clicked_notify_toggle)
```

**c) button\_schedule\_edit()**



This function will give a popup window wherein the data relating to the schedule is pulled from the database and displayed with the option to change the values and save them as shown below:

**Call the pop-up window (QWidget) and pass a parameter to retrieve the schedule for displaying the schedule**

<https://www.pyblog.in/python-gui/pyqt5/pyqt5-window-widget/>

**Step1: Get the Schedule from the database**

```
get_schedule(self, date)
```

```
    Step1- get the data from the Database based on today's date
```

```
    Step2- Return the pulled data from database
```

```
    return schedule_query()
```

**Step2: Display the Schedule retrieved from the database**

```
class schedule_frame:
```

```
    def __init__(self,)
```

**The generate function will create the buttons and labels (we have to change labels to textbox or textfields for user input/editing)**

```
    def generate(self, schedule_title, schedule_date, schedule_readmore,  
                schedule_edit, schedule_delete, schedule_notifytoggle):
```

```
        self.schedule_title=
```

```
        self.schedule_date=
```

```
        self.schedule_time=
```

```
        self.schedule_occur=
```

```
        self.schedule_remind_early=
```

```
        self.schedule_description=
```

```
        self.schedule_save_close_button=
```

```
        self.schedule_notifytoggle=
```

**The gen\_val function will assign values to the buttons (must specify the values from the database) and labels using the \_translate function**

```
    def gen_val(self, schedule_title, schedule_date, schedule_readmore,  
               sched, schedule_delete, schedule_notifytoggle):
```

```
        self.schedule_title.setText(self.translate(""))
```

```
        self.schedule_date.setText(self.translate(""))
```

```
        self.schedule_time.setText(self.translate(""))
```

```
        self.schedule_occur.setText(self.translate(""))
```

```
        self.schedule_remind_early.setText(self.translate(""))
```

```
        self.schedule_description.setText(self.translate(""))
```

```
        self.schedule_notifytoggle.setValue(self.translate(""))
```

```
        self.schedule_save_and_close.setText(self.translate(""))
```

**Step3: create an object for the class schedule\_frame()**

**Step4: display the schedule\_frame in the pop-up window called at the beginning of this step.**

**Edit Reminder/Schedule**

Name :

Date :  Time :

Occurrence :

Remind me earlier by -

Description -

☐ Turn ON Notification

#### d) `button_schedule_del()`

This function will delete the schedule from the database after the confirmation prompt and on successfully deleting it a nothing popup message is displayed with the message “Deleted Schedule Successfully”

#### Confirmation Prompt

Do You Want To Delete This Reminder ?

#### Step1: Call in the pop-up message box for confirmation

<https://www.techwithtim.net/tutorials/pyqt5-tutorial/messageboxes/>

```
class Ui_MainWindow(object):
    def clicked_button_delete (self, *args, **kwargs):
        #put in code to call the window wherein the message box appears for user confirmation

    def setupUi(self, MainWindow):
        self.button_schedule_delete.clicked.connect(self.clicked_button_delete)
```

#### Step2: In Message Box

```
class Ui_Form (object):
    def clicked_button_delete_yes (self, *args, **kwargs):
        #delete the schedule from the database.
        Display message box deleted successfully.

    def clicked_button_delete_no (self, *args, **kwargs):
        #close the delete message pop up.

    def setupUi(self, MainWindow):
        self.button_schedule_delete_yes.clicked.connect(self.clicked_button_delete_yes)

        self.button_schedule_delete_no.clicked.connect(self.clicked_button_delete_no)
```

**[deleted successfully message box to be created]**

#### 3. `menu_schedule_create()`

This function will display a popup window wherein the user can enter all the details of the reminder and save it as shown below:

Same as Edit Schedule just with no value in the various fields.

**Step1: Display the create form**

The generate function will create the buttons and labels (we have to change labels to textbox or textfields for user input/editing)

```
def generate(self, schedule_title, schedule_date, schedule_readmore,
              schedule_edit, schedule_delete, schedule_notifytoggle):
    self.schedule_title=
    self.schedule_date=
    self.schedule_time=
    self.schedule_occur=
    self.schedule_remind_early=
    self.schedule_description=
    self.schedule_save_close_button=
    self.schedule_notifytoggle=
```

The gen\_val function will assign values to the buttons (must specify the values from the database) and labels using the \_translate function

```
def gen_val(self, schedule_title, schedule_date, schedule_readmore,
            sched, schedule_delete, schedule_notifytoggle):
    self.schedule_title.setText(self.translate(""))
    self.schedule_date.setText(self.translate(""))
    self.schedule_time.setText(self.translate(""))
    self.schedule_occur.setText(self.translate(""))
    self.schedule_remind_early.setText(self.translate(""))
    self.schedule_description.setText(self.translate(""))
    self.schedule_notifytoggle.setValue(self.translate(""))
    self.schedule_save_and_close.setText(self.translate(""))
```

**Step 2: Take in inputs by the user**

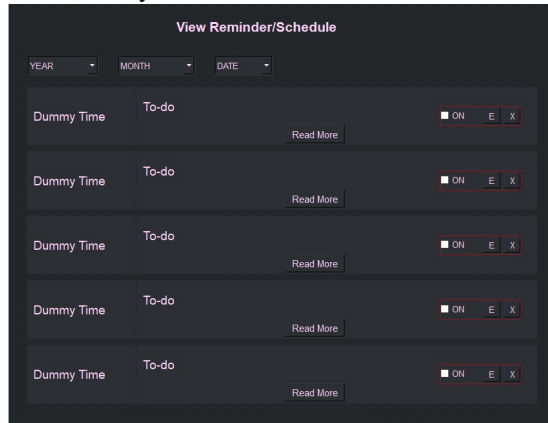
**Step 3: On button\_save\_close click save the schedule in the database.**

```
class Ui_MainWindow(object):
    def clicked_button_save_close(self, *args, **kwargs):
        #put in code to save schedule in database and display message box to show user it is saved
        successfully.
    def setupUi(self, MainWindow):
```

```
self.button_save.clicked.connect(self.clicked_button_save_close)
```

#### 4. menu\_schedule\_edit\_view()

This function will display a popup window where there are three combo boxes displayed which on inputting the date, month or year will display the schedules as per the filtered date, month, and year.



**Step1: Display ttoday's**

**Schedule as the in-home screen does by retrieving and generating in a QWidget Window**

**Step2: the Year Month and date will have today's date by default while showing today's schedule**

**Loop{**

**Step3: Let the user input the year month and date input**

**Step4: on input, The following schedule will be updated as per user input in the same window.**

**}**

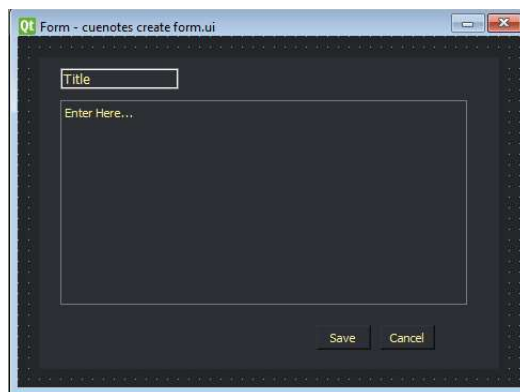
**This will go on forever until the user closes the QWidget**

Each schedule will have the same function calls as that of the ones in the home window

- a. button\_schedule\_readmore()
- b. schedule\_notify\_switch()
- c. button\_schedule\_edit()
- d. button\_schedule\_del()

#### 5. menu\_notes\_create()

This function will display a popup window wherein user can enter all the details of the notes and save it as shown below:



**Step1: Display the Qwidget Note Create Window**

## Step2: Let user input data on saving

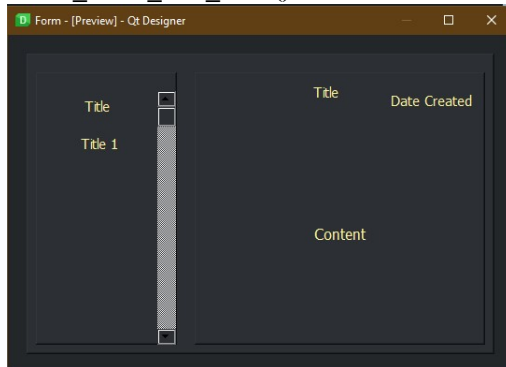
```
class Ui_Form(object):
    def setupUi(self, Form):
        self.button_save.clicked.connect(self.clicked_button_save)

        self.button_close.clicked.connect(self.clicked_button_close)

        def clicked_button_save(self, *args, **kwargs):
            #put in code to save schedule in the database and display a message box to show user it is saved
            successfully.

        def clicked_button_close(self, *args, **kwargs):
            #put in code to close the create note window
```

### 6. menu\_notes\_edit\_view()



#### STEP1: Display the QWidget Window with these Contents

The contents are to be retrieved and generated for display

### 7. menu\_options\_showfilelocation()

This function will open up a Window Explorer window with the set path to the program files.

### 8. menu\_options\_guide()

This function will open a window that displays the content to show the step-by-step guide to the user.

### 9. menu\_options\_srs\_doc()

This function will open a window that displays the content to show the SRS document to the user.

---

## Schedule Window Functions [Pooja]

### 1. menu\_schedule\_create()

#### **clicked\_button\_save\_close()**

This function will make sure all the details are entered by the user and when it is done checking and if all the details are entered it will save the reminder into the database. If there are details that have not been entered let the user know to please fill all the details through a pop-up dialog box.

**Same as Edit Schedule just with no value in the various fields.**

**Step1: Display the create form**

**The generate function will create the buttons and labels (we have to change labels to textbox or textfields for user input/editing)**

```
def generate(self, schedule_title, schedule_date, schedule_readmore,
              schedule_edit, schedule_delete, schedule_notifytoggle):
    self.schedule_title=
    self.schedule_date=
    self.schedule_time=
    self.schedule_occur=
    self.schedule_remind_early=
    self.schedule_description=
    self.schedule_save_close_button=
    self.schedule_notifytoggle=
```

**The gen\_val function will assign values to the buttons (must specify the values from the database) and labels using the \_translate function**

```
def gen_val(self, schedule_title, schedule_date, schedule_readmore,
            sched, schedule_delete, schedule_notifytoggle):
    self.schedule_title.setText(self.translate(""))
    self.schedule_date.setText(self.translate(""))
    self.schedule_time.setText(self.translate(""))
    self.schedule_occur.setText(self.translate(""))
    self.schedule_remind_early.setText(self.translate(""))
    self.schedule_description.setText(self.translate(""))
    self.schedule_notifytoggle.setValue(self.translate(""))
    self.schedule_save_and_close.setText(self.translate(""))
```

**Step 2: Take in inputs by the user**

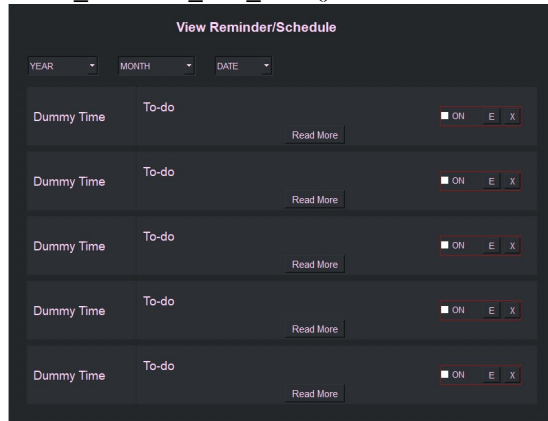
**Step 3: On button\_save\_close click save the schedule in the database.**

```
class Ui_MainWindow(object):
    def clicked_button_save_close(self, *args, **kwargs):
```

#put in code to save schedule in database and display message box to show user it is saved successfully.

```
def setupUi(self, MainWindow):  
    self.button_save_close.clicked.connect(self.clicked_button_save_close)
```

## 2. menu\_schedule\_edit\_view()



### display\_schedule()

This function will display a popup window where there are three combo boxes displayed which on inputting the date, month or year will display the schedules as per the filtered date, month, and year.

**Step1: Display today's Schedule as in-home screen does by retrieving and generating in a QWidget Window**

**Step2: the Year Month and date will have today's date by default while showing today's schedule**

**Loop{**

**Step3: Let the user input the year month and date input**

**Step4: on input, The following schedule will be updated as per user input in the same window.**



**}**

**This will go on forever until the user closes the QWidget**

This function will display the schedules by default (today's Schedule) and then display the schedules once the filter date, month and year are entered.

**Scroll bar not there need research to see if its needed**

**Filter Functions [needs research]**

-  **Either update Immediately as the filter data are being updated**
-  **Or ask for the user to input all the date, month and year and then click on a search button**

Each schedule will have the same function calls as that of the ones in the home window

#### a. button\_schedule\_readmore()

This function will give a popup window wherein the data relating to the schedule is pulled from the database and displayed as follow

**[form to be created]**

#### b. schedule\_notify\_switch()

This is a Toggle button or can be a normal button where in calls in a function to turn the flag for notification of this schedule OFF/ False.

#### c. button\_schedule\_edit()

This function will give a popup window wherein the data relating to the schedule is pulled from the database and displayed with the option to change the values and save them as shown below:

**a) button\_schedule\_readmore()**

This function will give a popup window wherein the data relating to the schedule is pulled from the database and displayed as follow

Lib.py

```
class Ui_MainWindow(object):
    def clickedButtonReadMore(self, *args, **kwargs):
        #put in code to call the window wherein the full details can be displayed
        self.pushButton_ReadMore.setText(self._translate("MainWindow", "Done"))

    def setupUi(self, MainWindow):
        self.pushButton_ReadMore.clicked.connect(self.clickedButtonReadMore)
```

[form to be created]

**b) schedule\_notify\_switch()**

This is a Toggle button or can be a normal button where in calls in a function to turn the flag for notification of this schedule OFF/ False.

```
class Ui_MainWindow(object):
    def clicked_notify_toggle (self, *args, **kwargs):
        """
        put in function to call the window wherein the toggle button value can be changed on interacting with it
        and return the value

        True/or False by checking the current status of the toggle button

        """
        self.pushButton_ReadMore.setValue(self._translate("MainWindow", "True/or False by checking the
        current status of the toggle button"))

    def setupUi(self, MainWindow):
        self.notify_toggle.clicked.connect(self.clicked_notify_toggle)
```

**c) button\_schedule\_edit()**

This function will give a popup window wherein the data relating to the schedule is pulled from the database and displayed with the option to change the values and save them as shown below:

**Call the pop up window (QWidget) and pass parameter to retrieve schedule for displaying the schedule**

<https://www.pyblog.in/python-gui/pyqt5/pyqt5-window-widget/>

**Step1: Get the Schedule from database**

**get\_schedule(self, date)**

**Step1- get the data from the Database based on today's date**

**Step2- Return the pulled data from database**

**return schedule\_query()**

**Step2: Display the Schedule retrieved from database**



```
class schedule_frame:
    def __init__(self,)
```

The generate function will create the buttons and labels (we have to change labels to textbox or textfields for user input/editing)

```
    def generate(self, schedule_title, schedule_date, schedule_readmore,
                schedule_edit, schedule_delete, schedule_notifytoggle):
        self.schedule_title=
        self.schedule_date=
        self.schedule_time=
        self.schedule_occur=
        self.schedule_remind_early=
        self.schedule_description=
        self.schedule_save_close_button=
        self.schedule_notifytoggle=
```

The gen\_val function will assign values to the buttons (must specify the values from the database) and labels using the \_translate function

```
    def gen_val(self, schedule_title, schedule_date, schedule_readmore,
                sched, schedule_delete, schedule_notifytoggle):
        self.schedule_title.setText(self.translate(""))
        self.schedule_date.setText(self.translate(""))
        self.schedule_time.setText(self.translate(""))
        self.schedule_occur.setText(self.translate(""))
        self.schedule_remind_early.setText(self.translate(""))
        self.schedule_description.setText(self.translate(""))
        self.schedule_notifytoggle.setValue(self.translate(""))
        self.schedule_save_and_close.setText(self.translate(""))
```

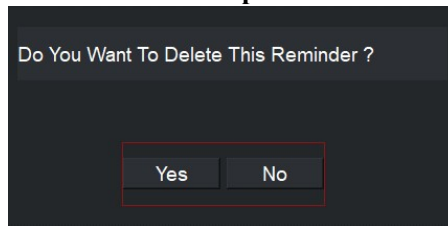
Step3: create an object for the class schedule\_frame()

Step4: display the schedule\_frame in the pop up window called in the beginning of this step.

d) button\_schedule\_del()

This function will delete the schedule from the database after the confirmation prompt and on successfully deleting it a nothing popup message is displayed with the message “Deleted Schedule Successfully”

#### Confirmation Prompt



#### Step1: Call in the pop up message box for confirmation

<https://www.techwithtim.net/tutorials/pyqt5-tutorial/messageboxes/>

```
class Ui_MainWindow(object):
    def clicked_button_delete (self, *args, **kwargs):
        #put in code to call the window wherein the message box appears for user confirmation

    def setupUi(self, MainWindow):
        self.button_schedule_delete.clicked.connect(self.clicked_button_delete)
```

#### Step2: In Message Box

```
class Ui_Form (object):
    def clicked_button_delete_yes (self, *args, **kwargs):
        #delete the schedule from the database.
        Display message box deleted successfully.

    def clicked_button_delete_no (self, *args, **kwargs):
        #close the delete message pop up.

    def setupUi(self, MainWindow):
        self.button_schedule_delete_yes.clicked.connect(self.clicked_button_delete_yes)

        self.button_schedule_delete_no.clicked.connect(self.clicked_button_delete_no)
```

 A screenshot of a Qt widget titled "Edit Reminder/Schedule". It contains several input fields: "Name:" with a text box, "Date:" with a date picker showing "1/1/2000", "Time:" with a time picker showing "12:00 AM", "Occurrence:" with a dropdown menu showing "Daily", and "Remind me earlier by -" with a text box. There is also a "Description -" label above a large text area. At the bottom right, there is a checkbox labeled "Turn ON Notification" and a "Save and Close" button.

#### d. button\_save\_close()

This Function will save the changes and close the edit window.

#### Step1: Display the Qwidget Note Create Window

**Step2: Let user input data on save**

```
class Ui_Form(object):
    def setupUi(self, Form):
        self.button_save.clicked.connect(self.clicked_button_save)

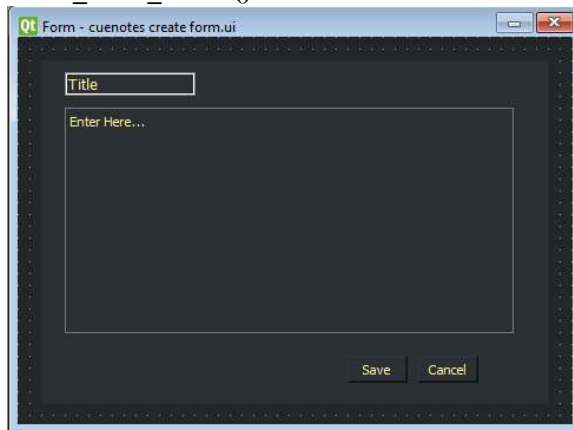
        self.button_close.clicked.connect(self.clicked_button_close)

        def clicked_button_save(self, *args, **kwargs):
            #put in code to save schedule in database and display message box to show user it is saved
            successfully.

        def clicked_button_close(self, *args, **kwargs):
            #put in code to close the create note window
```

### **Notes Window Functions [Thrupthi]**

#### **1. menu\_notes\_create()**



**Step1: Display the create notes form**

**The generate function will create the buttons and labels (we have to change labels to textbox or textfields for user input/editing)**

```
def generate(self, notes_title, notes_description,
button_notes_save_close):
    self.notes_title=
    self.notes_description=
    self.button_notes_save_close=
```

**The gen\_val function will assign values to the buttons (must specify the values from the database) and labels using the \_translate function**

```
def gen_val(self, notes_title, notes_description,
button_notes_save_close):
    self.notes_title.setText(self.translate(""))
    self.description_date.setText(self.translate(""))
    self.button_notes_save_close.setText(self.translate(""))
```

**Step 2: Take in inputs by the user**

**Step 3: On button\_save\_close click save the schedule in the database and close the pop up.**

```
class Ui_MainWindow(object):
```

```
def clicked_button_notes_save_close(self, *args, **kwargs):
    #put in code to save notes in database and display message box to show user it is saved successfully.
def setupUi(self, MainWindow):
    self.button_notes_save_close.clicked.connect(self.clicked_button_notes_save_close)
```

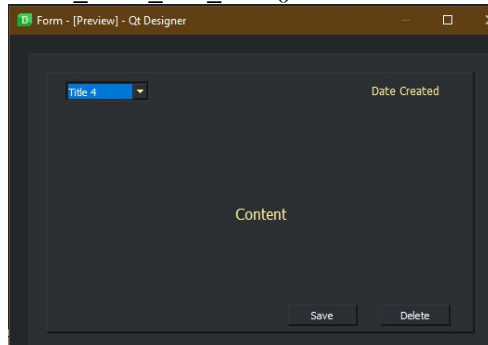
a. **button\_notes\_save()**

This function will save the note to the database and then close the create window

b. **Button\_notes\_cancel()**

This Function will just cancel/ close the pop-up notes create window.

2. **menu\_notes\_edit\_view()**



**View Window**

All the details from the database will have to be displayed pertaining to notes.

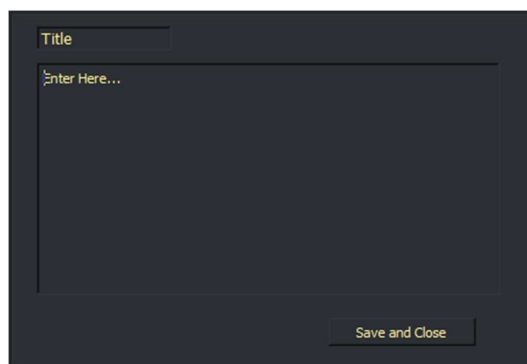
a. **display\_selected\_note()**

This function will display the content and title of the note selected on the right-side frame.

From the View Screen I think an **Edit button** for the note must be there to go to the Editing of a note as shown below. Can look at the Mindmap image above and see that there is no way to get to this edit option. [Thrupthi]

There is also no option to delete notes from the view Window [Thrupthi]

b. **Edit Window of the selected Note**



The saved details will be displayed on this screen wherein the user can do the desired changes and then click on the save and close button

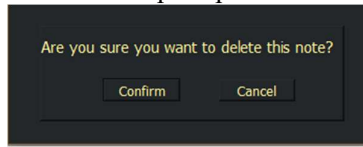
c. **button\_save\_close()**

This Function will save the changes and close the edit window.

Successfully saved dialog box optional

d. **button\_notes\_del()**

This function will delete the selected note from the database after a deletion confirmation prompt as shown below:



e. ....

### **Something for [Sylvester]**

For now, Sylvester will be given tasks to help Pooja and Trupthi as they too will require some help.

## **Guidelines to be followed by the Software Developers**

### **Python Programming Conventions**

#### **Rules in general for python naming conventions**

- **Rule-1:** Always give a meaningful full name for any of the python objects. Don't give a name like x, y, or z, etc.
- **Rule-2:** Don't give space in between object names, instead use **underscore (\_)** when there are more than one-word exists.
- **Rule-3:** Better to use camelCase only when it makes sense or else try avoiding more use of camelCase.

#### **Naming Functions**

- **Rule-1:** We should write the [Python function](#) name with all **lower case characters**.
- **Rule-2:** **Do not use uppercase character** while naming a function in python.
- **Rule-3:** Use **underscore (\_)** in between the words instead of space while naming a function.

#### **Naming Variables**

- **Rule-1:** You should start variable name with an alphabet or **underscore (\_)** character.
- **Rule-2:** A variable name can only contain **A-Z, a-z, 0-9** and **underscore (\_)** .
- **Rule-3:** You cannot start the variable name with a **number**.
- **Rule-4:** You cannot use special characters with the variable name such as such as \$, %, #, &, @, ., -, ^ etc.
- **Rule-5:** Variable names are **case sensitive**. For example, str and Str are two different variables.
- **Rule-6:** Do not use reserve keyword as a variable name for example keywords like **class, for, def, del, is, else, try, from**, etc.

#### **Naming Classes**

- **Rule-1:** We need to follow the **camelCase** convention
- **Rule-2:** When you are writing any classes for an exception then that name should end with **"Error"**.
- **Rule-3:** If you are calling the class from somewhere or **callable** then, in that case, you can give a class name like a **function**.
- **Rule-4:** The built-in classes in python are in **lowercase**.

**Example:** class MyClass, class Hello, class InputError

#### **Naming Objects**

- **Rule-1:** You should use all **lowercase** while deciding a name for an Object.
- **Rule-2:** Choose a very **short name**.
- **Rule-3:** If there are multiple words in your object name then they should be separated by an **underscore (\_)** .

#### **Naming Files in Python**

- **Rule-1:** We should choose a **short name** as a file name.
- **Rule-2:** You should use all **lowercase** while deciding a name for a file.
- **Rule-3:** We can also use **underscore (\_)** with the file name.

### **Naming Packages and Modules**

- **Rule-1:** You should use all **lowercase** while deciding a name for a package or module.
- **Rule-2:** If there are multiple words in your method name then they should be separated by an **underscore (\_)**.
- **Rule-3:** It is always better to use a **single word** for a **package or module name**.

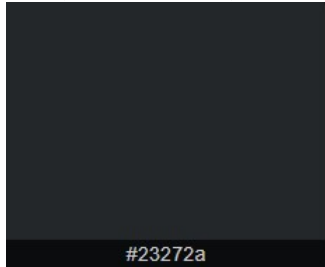
### **Reference**

<https://pythonguides.com/python-naming-conventions/>

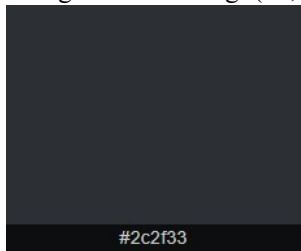
### **Cue Designer Points to remember**

1. Name the objects in PyQt Designer appropriately as per the python convention and make the name as short and simple as possible.
2. Try to add the type of object it is in the name, for example: for the Read more button, the name would be button\_read\_more.
3. **The Color palette** decided and used in this software are as follows:

- a. **Background color for the Main Windows** which will be the darkest  
background-color: rgb(35, 39, 42); #23272a



- b. **The Background color of the containers in the Main Windows will have a lighter color**  
background-color: rgb(44, 47, 51); #2c2f33



4. **All Forms (Widget and Main Menu Window) are in 850x650**
5. **The Read More Form in 550x390**
6. **All Dialog Boxes in 340x130**