

Worker App Tutorial Info

Warning

The Stuff you'd see will differ from the documentary since in this solo project, the documentary is written alongside designing and coding. The additional feature which includes the updated feature is located in the update segment

Alongside those warning, here's the complete Image of the app (final version):

Input Tab:

Worker App Design

Input

View

Edit

Worker Name: Doro Senior

Date: 23-05-2023

Delete

Instant Delete

Action

Select File: Browse

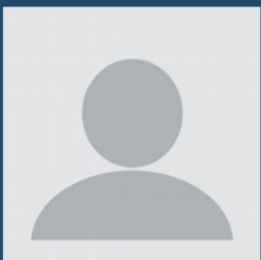
Export Data

Clear Window

Add

View

Input



Browse Image

Show

Add/Edit Image

Name / Title: Doro Senior

Edit

Date (dd-mm-yyyy): 23-05-2023

Date

Edit

Present Status: ☒ Present ☐ Absent

Edit

Payment (daily): 340800.0

Calc

Edit

Penalty (daily): 20000.0

Calc

Edit

Work Done:

Earning

Catching mouse = 50000*4.68

Guarding post = 30000*3.56


Cut

Littering = 10000*2

Edit

Filter View:

When filtered is off


Worker App Design

Input

View

Name:
Doro Senior
Month:
Filter
Show All
App Info

Name	Date	Present Status	Earning	Wage Cut
Garfield Junior	25-05-2023	1	Rp. 1.500.000	Rp. 20.000
Garfield Junior	18-05-2023	1	Rp. 1.500.000	Rp. 20.000
Garfield Junior	14-04-2023	1	Rp. 1.500.000	Rp. 20.000
Garfield Senior	14-04-2023	0	Rp. 48.000.000	Rp. 20.000
Garfield Junior	13-05-2023	1	Rp. 1.500.000	Rp. 20.000
Garfield Junior	05-05-2023	1	Rp. 1.500.000	Rp. 20.000
Garfield Junior	09-04-2023	1	Rp. 1.500.000	Rp. 20.000
Garfield Senior	11-04-2023	0	Rp. 1.500.000	Rp. 20.000
Garfield Senior	08-04-2023	0	Rp. 1.500.000	Rp. 20.000
Garfield Junior	01-03-2023	1	Rp. 1.500.000	Rp. 20.000

Filtered Result

Income Sum:
Reduction Sum:
Total Earn:
Total Presence:

When filtered button is clicked:

Worker App Design

Input

View

Name:

Doro Senior

Month:

Filter

Show All

App Info

Name	Date	Present Status	Earning	Wage Cut
Doro Senior	01-05-2023	1	Rp. 340.800	Rp. 20.000
Doro Senior	31-05-2023	1	Rp. 340.800	Rp. 20.000
Doro Senior	01-06-2023	1	Rp. 340.800	Rp. 20.000
Doro Senior	23-05-2023	1	Rp. 340.800	Rp. 20.000

Filtered Result

Income Sum:

Rp.1.363.200

Reduction Sum:

Rp.80.000

Total Earn:

Rp.1.283.200

Total Presence:

4/4

There would be 5 parts in this code:

1. Part 0: Program overview
2. Part 1: Adding Data
3. Part 2: Editing Data
4. Part 3: Filtering Data
5. Part 4: Exporting and Viewing Data as spreadsheet
6. Updated App Info and Guide

This tutorial would contain the procedure/steps to use this app. There would be only one block of code which is located in part 3. The rest would be a straightfoward tutorial.

Part 0: General Overview

So, this is the interface that you'd see when you ran the back end code of the app, there'd be a window consist of two tab. This is the first one

Worker App Design

Input

View

Edit

Worker Name: CAT

Date: 26-05-2023

Delete

Action

Select Dir:

Browse

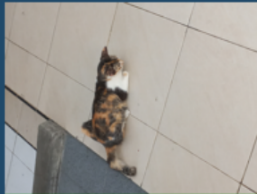
Export

Clear Window

Add

View

Input



Browse Image

Add/Edit Image

Name / Title: CAT

Edit

Date (dd-mm-yyyy): 26-05-2023

Date

Edit

Present Status: ☒ Present ☐ Absent

Edit

Payment (daily): 980000.0

Edit

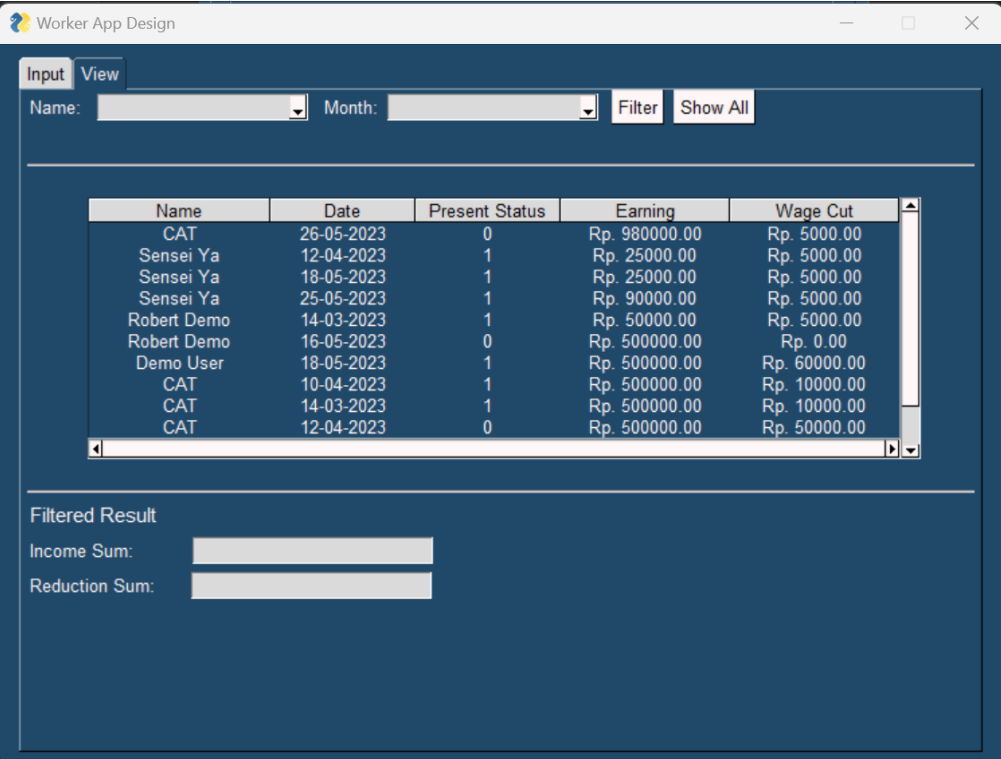
Penalty (daily): 5000.0

Edit

Work Done: Nothing in particular, Just another summer day

Edit

And this is the second tab:



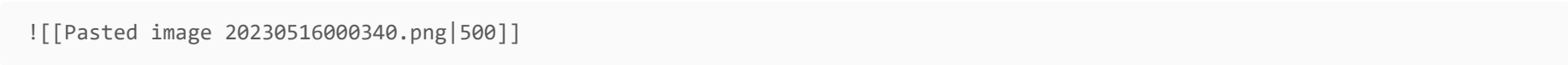
The first one would consists of input and edit elements while the second tab is consists of a two filters dropdown, a table and two entry elements.

Part 1: Adding Data

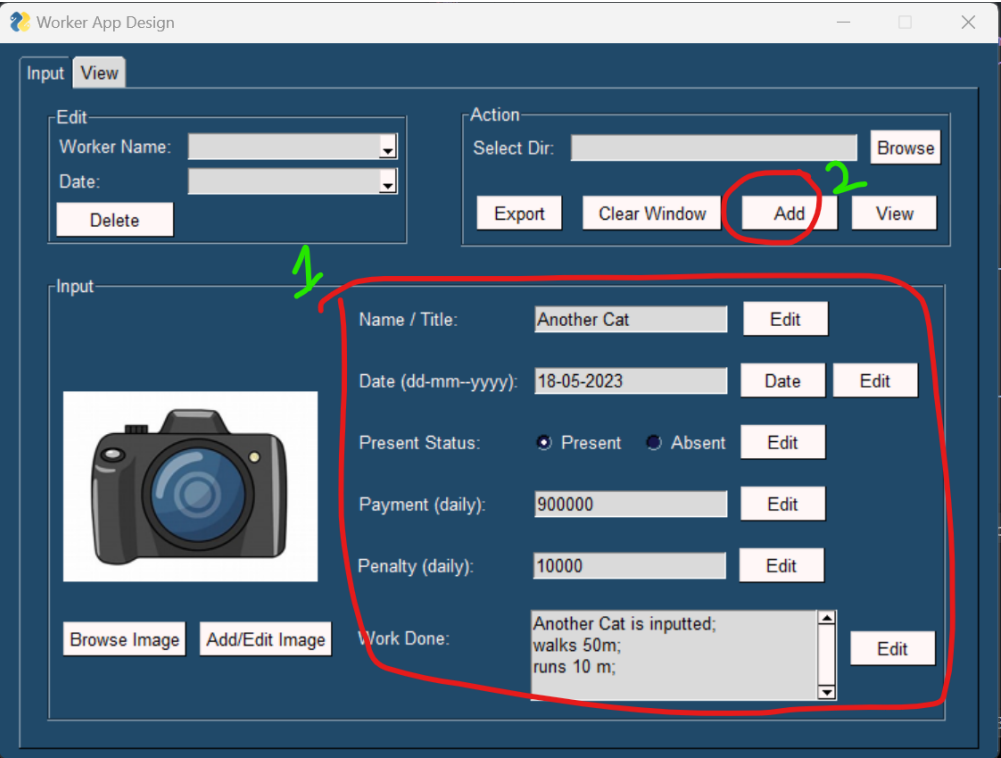
Here's the data that you could add alongst their respective input format:

- | Input | Format/How to fill |
- | ----- | ----- |
- | Name/Title | String, type a content |
- | Date | String, select the date using the Date button or input it to input box |
- | Present Status | Integer, choose between the 2 state |
- | Payment | Real, Enter a number (float is allowed) |
- | Penalty | Real, Enter a number (float is allowed) |
- | Work note | String, Enter some message or note |

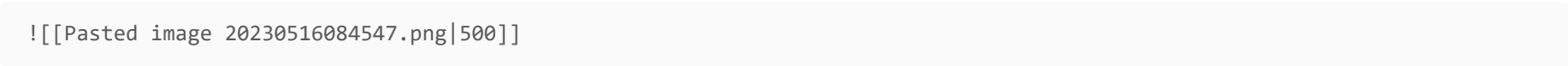
To add a data, simply clear all the window's content by hitting the **Clear Window** button on the action tab. This would clear all the content and it the app would look like Fig 3:



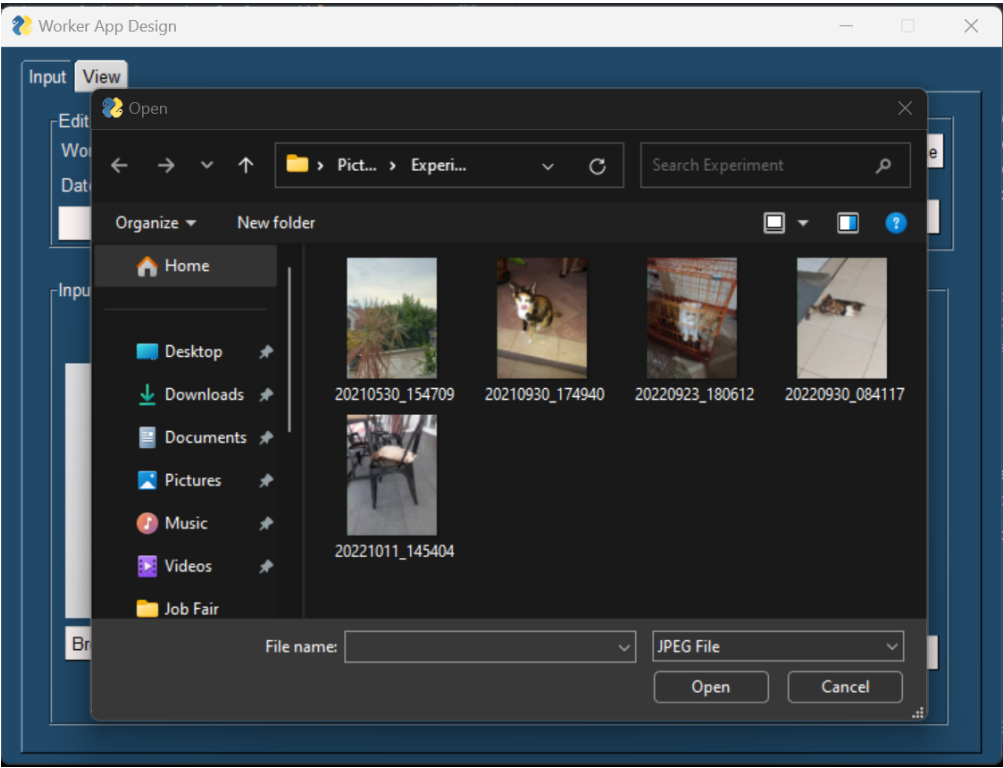
You would then entered the data and browse the image of (that particular) person. This feature is both comfirting and annoying. Once you browse the Image, no image update is shown (yet). You would need to add the data and the image in order to save both the data and the image. Here's the sample of a data input:



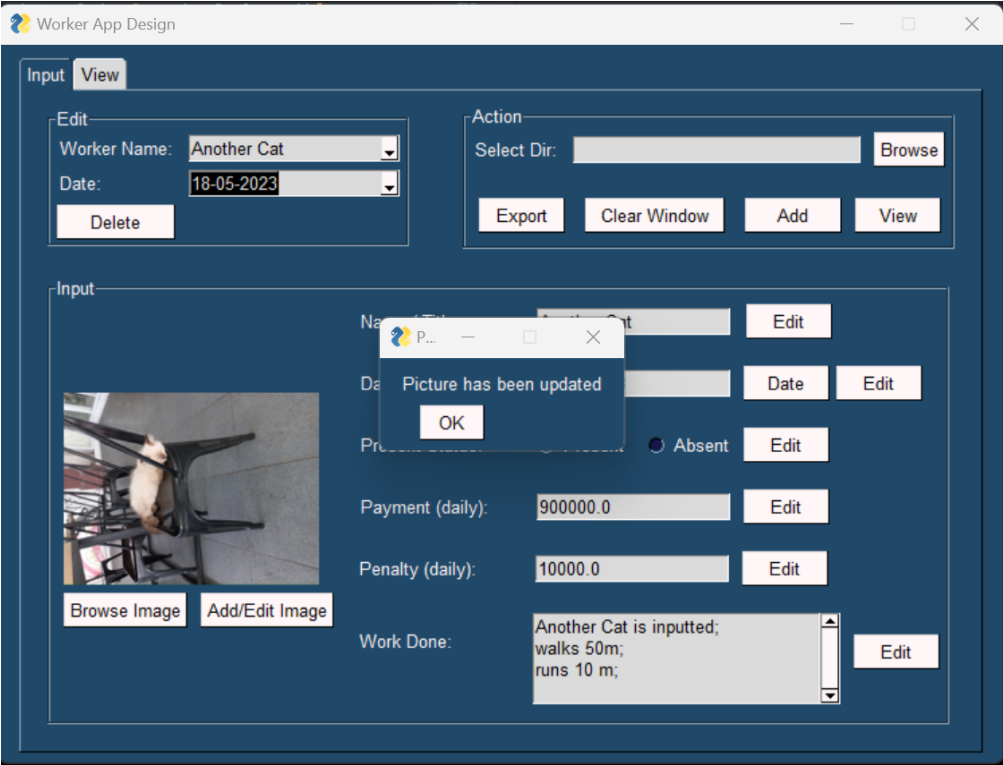
We would be adding the added people or object would be available on the drop down, select the worker and the date would pop up. Once a worker or object is selected from the combo box, it would display the inputted data alongst with a pop up warning that an image has not been inserted:



You would then hit the Browse Image, the folder manager containing a list of image option (JPEG, PNG, SVG, BP) would be opened. You could select one of the any image as long as it is available in the image option (File format).



After selecting an image, the folder dialogue would close. This time, you would see nothing yet. The changes can only be seen after you hit the **Add/Edit Image button**. So make sure that every time you want to change the image, choose/browse the image -> then commit the image by hitting the add/edit button. Once you add or edit the image, a popup message stating that the picture has been updated would be shown. Just hit the ok button and it would close.



Part 2: Editting Data

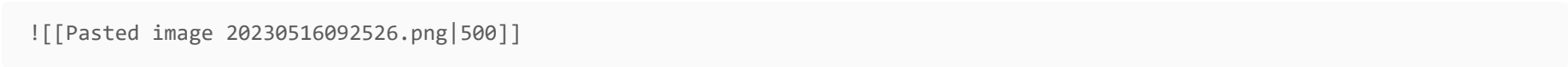
Here's the summary of different component's edit workflow:

- | Data | Edit Method |
- | ----- | ----- |
- | Name/Title | Popup appearance, change the popup content |
- | Date | Popup appearance, pick the date in the popup section |
- | Present status | Pick then click Edit |
- | Payment | Popup appearance |
- | Penalty | Popup appearance |
- | Work Done | Edit then click the Edit button |
- | Add/Edit Image | Click only after an image has been selected |

Here's the long explanation of the table:

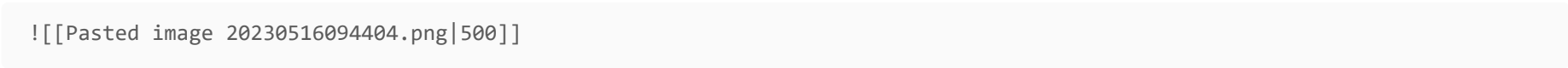
1. Name/Title Edit

The Edit button would pop up an input containing the current name as the default. Change the name and hit the ok button to edit the name/title. The change would clear out the Worker Name dropdown or combobox since it's set to ' ' by default. Here's a picture containing the edit popup window:



2. Date Edit

To Edit Date, hit the edit button besides add, then a popup would show up. Once it shows up, click the date and select the desired date from the available date popup. Once chosen, click the save button to save the date, a popup would appear to notify that you have change/edit the date.



Warning

One of the known issue I have is the cancel button. It is currently not working and you should just close the dialogue box using the right x close button.

3. Present/Absent Edit

You can choose between each state (there's only 2: Present or Absent). Since I used SQLite3 to construct the database, BOOL data type is not present. Therefor, I represent the present state as 1 and absent state as 0 (Much like true or false). To edit this value, choose the opposite state and click the edit button. It would immediately update the value without any popup message (I found this message to be annoying sometimes since I only need to toggle between two options).

![[Pasted image 20230516095210.png|500]]

4. Payment Edit

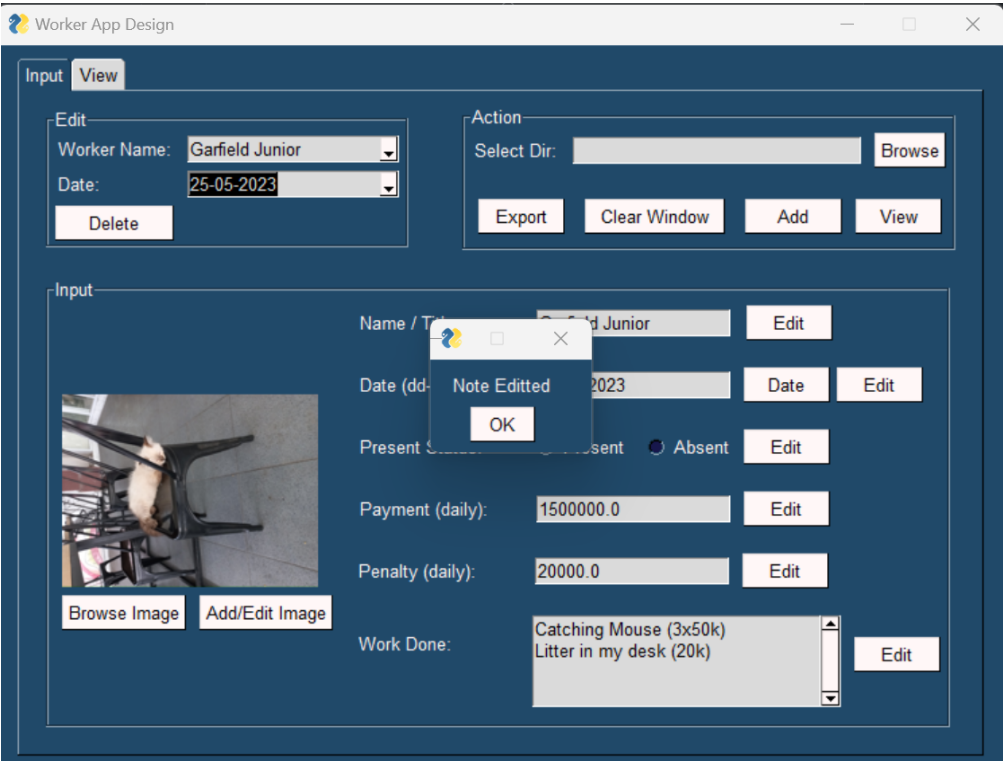
The payment Edit (both payment and penalty has the exact same backend code). One would enter the amount in the Input element when the popup message is shown.

![[Pasted image 20230516165446.png|500]]

Make sure to only enter number (without the , or . seperator). This is a little bit tricky since I am not yet able to safely modify the input format without ruining the general workflow. If one accidently enters a alphabet word or a whole sentence, the operation would immediately cancelled and a pop up message would be shown.

5. Note Edit

This multiline box is actually keeping strings as some sort of memo or note. You could add or any format and other stuffs. A best practice would be keeping the line short and extend your note in the next line by hitting enter instead of typing all sentence in one go. This is because this app supports excel export and view output and you'll see why later on at the export guide section (or you could read the source code). The note would hold a width of 40 xlwing's length dimation (I think it's in inch but you can search the xlwings documentation to verify this statement). Much like the present/absent radio button edit, this one would immediately be updated once you press the **Edit button**. Once edited, a popup message would be shown.



Deleting Data

Every deletion would only a name and date which was selected. Once the last remaining data is deleted, the image that correlates with the worker name would also be deleted. Therefor, it is necessary to choose both the **name** and the **date** of the object. Once a name and a date has been selected, all window would be restored to a null or None value. There would also be a notification that stated that you have delete a certain data.

![[Pasted image 20230516171900.png|500]]

If all the data; which hold the name and date of certain object has been deleted (entirely), the image would be deleted from the database.

Part 3: Filtering Data

To use this filter method, you would need to go to the second tab where the inputted value has been updated to a table:

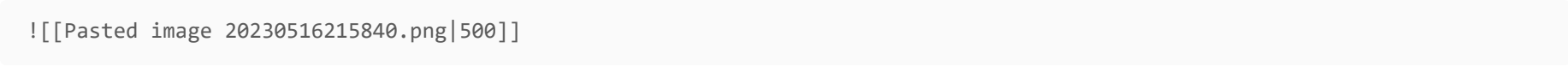
![[Pasted image 20230516173222.png|500]]

As you can see, the number is currently not formatted correctly since it's really hard to split and rejoin the number when a filter is applied. So I decided that it would be easier for me to operate the data in the income sum and reduction sum input. I am planning to improve my skills in the next project. So upon entering the filter field, you would find two dropdown menus: Name and Month. The name would contain list of inputted name which is inputted in the Input tab while the month's drop down would contain the month ranging from 1 (january) to 12 (december).

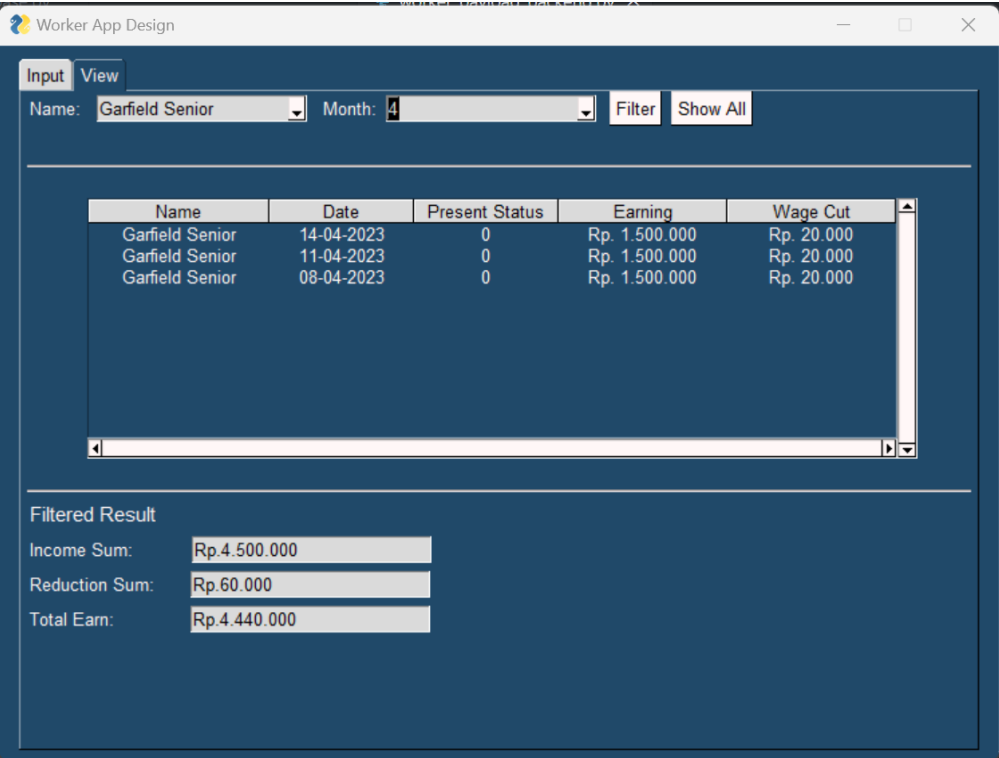
Filter could be done in one of three combinations: name, month, and both. The name would filter only the item that correlates with that name only, for example we'd filter the Garfield Junior using the name drop down. We'd select the dropdown and choose Garfield Junior. we would then hit the filter button. This is displayed format that you'd see in the screen.



You could see that it displays the "corect" formatting as one would expect from IDR. The filtered result is automatically summed upon clicking the filter button. The other method would be the month filter which would filter all the months.



The last option would be to filter both the month and the date. This quickly calculates the worker's salary for that particular month.



Those are the capabilities of this app. Having this kind of app would be really handy since I've spent so much time dealing with the spreadsheet method during my first internship.

Part 4: Exporting and Viewing Data

Not everyone is keen on having to work with python or this kind of thing. Spreadsheet is the gold standard of all industries and would likely remain so. That's why this program features two additional excel related stuff: the export button and the view button.

1. Export Button

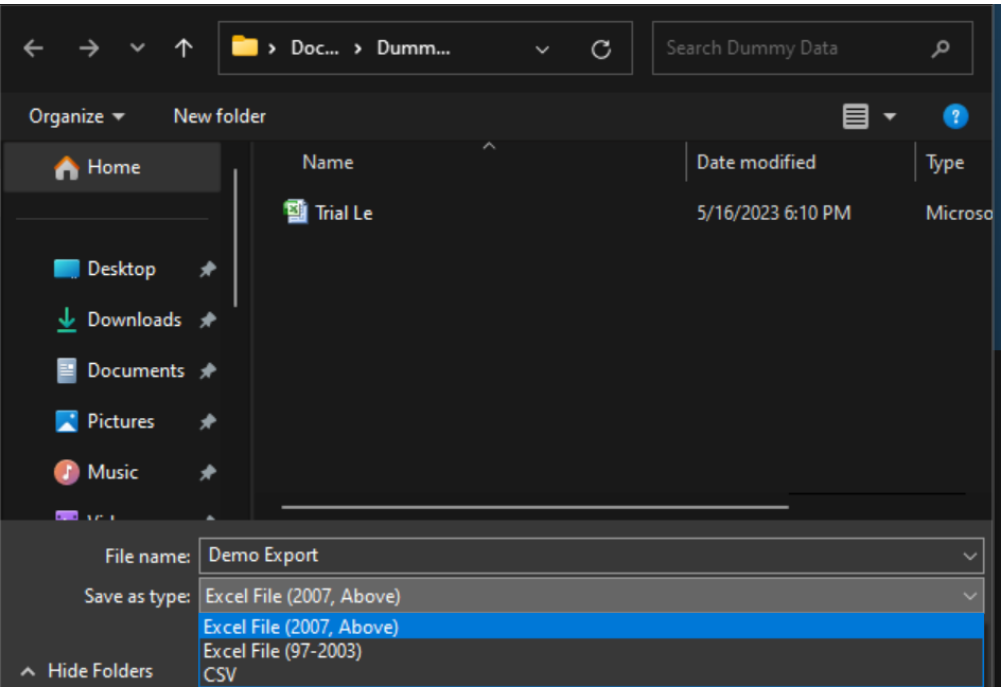
The export button would export the database content. The user could choose between three options:

- Excel 2003 and above (.xlsx)
- Excel 1997-2003 (.xls)
- CSV

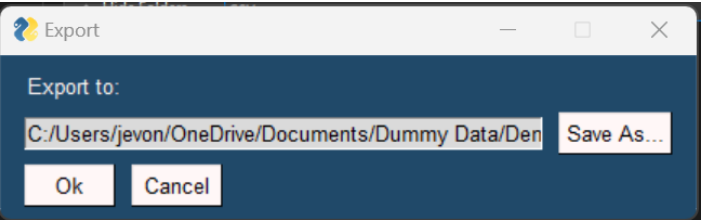
This option could be selected upon clicking the export button. A popup box would immediately upon. Click the save as button and the folder window would open up



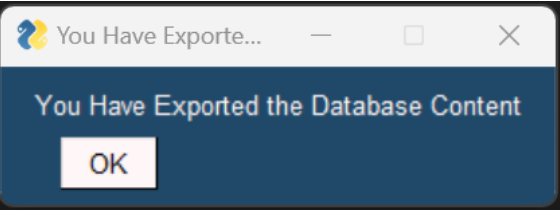
You could choose between three specified format.



Once you've entered the name and the selected format, the window would close up. You would see the popup window filled up with the full directory of the file. Click ok to save the spreadsheet form.

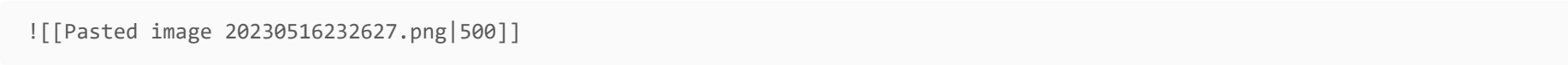


Once the ok button is clicked a popup message would notify the user that the file has been sucesfully exported.

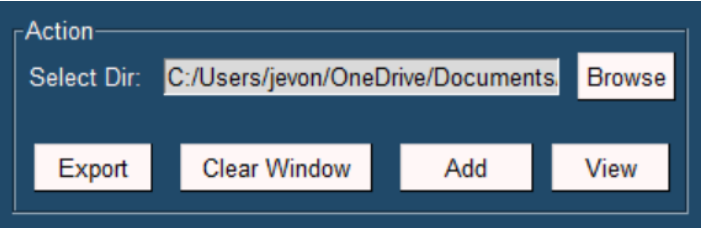


2. Viewing Excel File

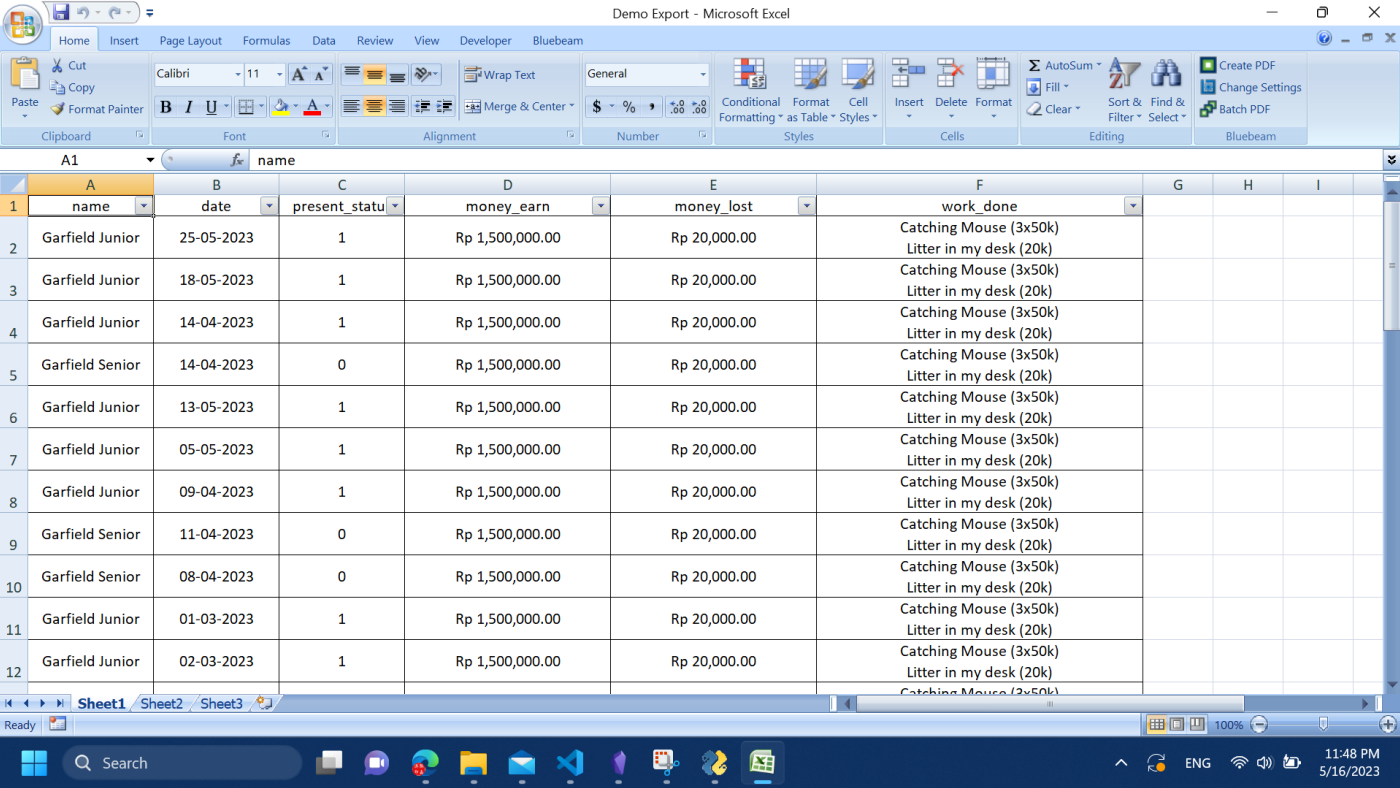
XLWings provide a rather straight foward way that enables a user to quickly view an excel file. To view the export (or any other excel file), click the **Browse** button besides the action input element. This would open a folder dialogue. You would need to select an excel file.



The window would return the full path of the excel file. You would then click the **View** button and excel would launch.



You would see that this actually works. You would also see that the header has the filter (more explanation in the app construction section). The formatting done here is actually done seperately in python and as the result, there have been several known issue on the incorrect formatting (especially the date). Therefor, it is often a best practice to check whatever data is outputted from the program.



Info

You could modify the spreadsheet's look by changing the method that creates the excel book , sheets, and other formats. It's this line of code:

```
def export_data(wb, filename):
    locale.setlocale(locale.LC_ALL, 'id_ID')
    data = wb_com.get_export_data()

    # #* Launch new excel app
    sheet = wb.sheets[0]

    #* Write data to the worksheet
    sheet.range('A1').value = data

    #* Set the number format for date
    sheet.range('B:B').number_format = 'mm-dd-yyyy'

    #* Set the currency number formatting
    sheet.range('D:E').number_format = f'Rp #,##0.00'

    #* Apply coloring to worksheet
    sheet.range('A1').expand('right').api.AutoFilter(1)

    #* Apply border to cells
    last_row = sheet.range('A' + str(sheet.cells.last_cell.row)).end('up').row
    last_column = sheet.range('A1').end('right').column
    sheet.range('A1', sheet.cells(last_row, last_column)).api.Borders.LineStyle = 1

    #* Centering the content
    range_to_center = sheet.range('A1', sheet.cells(last_row, last_column))
    range_to_center.api.HorizontalAlignment = xw.constants.HAlign.xlHAlignCenter
    range_to_center.api.VerticalAlignment = xw.constants.VAlign.xlVAlignCenter

    #* Fit cell's content
    sheet.range('A:A').column_width = 15
    sheet.range('B:B').column_width = 15
    sheet.range('C:C').column_width = 15
    sheet.range('D:D').column_width = 25
    sheet.range('E:E').column_width = 25
    sheet.range('F:F').column_width = 40

    #* Save the workbook
    wb.save(filename)

    #* Close and quit the app
    wb.close()
```

You could see from the snippet code that I adjust only a handful of parameters, which include:

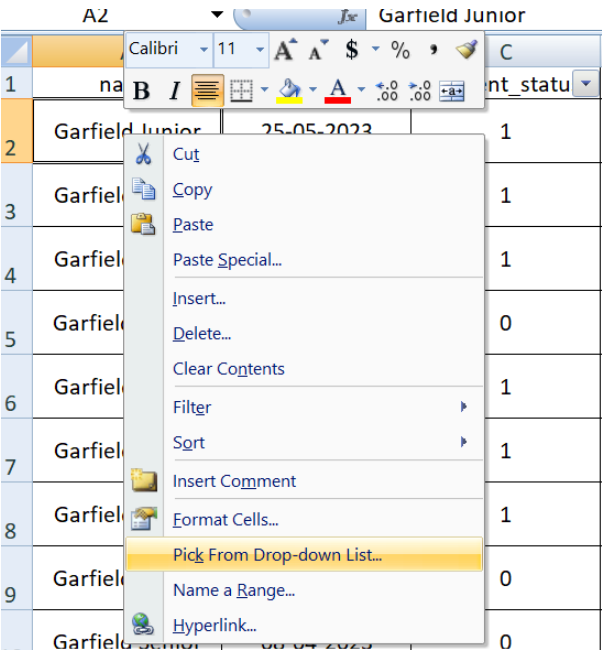
1. Date format
2. Currency format
3. Column width
4. Border width and style

You could customized more than this items by searching the desired effect or formatting in the XLWings documentation (it's in their website).

Note

You could clear the browsing Input box by pressing `ctrl+A` then `del` .

Another major thing that might be useful is that the both the date and name is a combobox element, thus the excel result would also inderectly inherit the object's property. What this means is that you could actually change both name and date from excel drop down option. To do that, select name or date cell element and rightclick the cell, you would then choose the pick from dropdown list option



After clicking the drop-down list, excel would popup its own version of dropdown or combobox which the user could choose

A2		Garfi
A	B	
1	name	date
2	Garfield Junior	25-05-2023
3	Garfield Junior Garfield Senior	18-05-2023
4	Garfield Junior	14-04-2023
5	Garfield Senior	14-04-2023

You could click the available options and it would change the cell value to the clicked option's value. Only the name and date option has this capability since both are extracted from the combobox value, which would hold the the unique name and date of the object.

That's it, the entirety of the program has been expalin. The more important thing is to check the source code of the application. This application has three components: the database ([Worker App Database Section](#)), the backend (), and the frontend().

Update and Patch: Calculation Update

Using python's build in eval library, I have added a functionality which is the calculation method where a symbolic operation such as `=50000*4+30000*8` is now available in the app. Users could now calculate the amount using a symbolic operation followed by pressing the calc button. This function is only available in the input window and not the edit window since a lot of issues has been found when this function is extended to the custom window.

A work around this feature would be to used the default input as the inherent default value in the popup window (you would calculate the edited value in the main window then apply it directly without any revision(s) when the popup shows up).

Overall all, this is the code that allows the calculation update:

```
def evaluate_expression(expression):
    try:
        expression = expression.strip()
        if expression.startswith('='):
            expression = expression[1:]
        node = ast.parse(expression, mode='eval')
        compiled_expr = compile(node, filename='<string>', mode='eval')
        evaluated = eval(compiled_expr, {}, {})
        return evaluated
    except Exception as e:
        print(f"Error evaluating expression: {e}")
        return None
```

Another thing which has been updated is the numbering format of the Table widget. Previously, I immediately replace or add rupiah in front of the integer. But I found a rather unique solution to make a method which would format both table and input widgets:

```
def format_currency(amount):
    formatted_amount = f"Rp. {amount:,.0f}".replace(',', ',.')
    return formatted_amount
```

You could modify this code to suit the another formatting. It is then applied to the table and filter:

```
def get_table_content():
    conn = sqlite3.connect(db_path)
    cursor = conn.cursor()
```

```

cursor.execute("SELECT name, date, present_status, money_earn, money_lost FROM workers")
rows = cursor.fetchall()

formatted_rows = []
for row in rows:
    formatted_row = list(row)
    # Format the currency using the method
    formatted_row[3] = format_currency(row[3])  #* Format money_earn
    formatted_row[4] = format_currency(row[4])  #* Format money_lost
    formatted_rows.append(tuple(formatted_row))

conn.close()
return formatted_rows

def get_table_filter(name=None, month=None):
    conn = sqlite3.connect(db_path)
    cursor = conn.cursor()
    month_str = ''
    if month:
        month_str = '-' + str(month).zfill(2) + '-'
    if name and month:
        cursor.execute("SELECT name, date, present_status, money_earn, money_lost FROM workers WHERE name = ? AND date LIKE ?",
                        (name, f'{month_str}%',))
    elif name:
        cursor.execute("SELECT name, date, present_status, money_earn, money_lost FROM workers WHERE name = ?",
                        (name,))
    elif month:
        cursor.execute("SELECT name, date, present_status, money_earn, money_lost FROM workers WHERE date LIKE ?",
                        (f'{month_str}%',))
    else:
        return []
    rows = cursor.fetchall()
    conn.close()

    formatted_rows = []
    for row in rows:
        formatted_row = list(row)
        # Format the currency using the method
        formatted_row[3] = format_currency(row[3])
        formatted_row[4] = format_currency(row[4])
        formatted_rows.append(formatted_row)

    return formatted_rows

```

Another important update is the Image element. By default both tkinter and PySimpleGUI sucks at image processing. Therefore, a method has been created which is used to convert images to PNG format. But this method would force PySimpleGUI to rotate the image if the size of the image is too large. This is a problem since many of the cats images is directly taken from a phone camera without any cropping or size editing. Thus, another library: `piexif` is used to rotate the position.

```

def load_img(img_path):
    try:
        exif_dict = piexif.load(img_path)
        if '0th' in exif_dict and piexif.ImageIFD.Orientation in exif_dict['0th']:
            orientation = exif_dict['0th'][piexif.ImageIFD.Orientation]
            if orientation == 3:
                image = Image.open(img_path).rotate(180, expand=True)
            elif orientation == 6:
                image = Image.open(img_path).rotate(270, expand=True)
            elif orientation == 8:
                image = Image.open(img_path).rotate(90, expand=True)
            else:
                image = Image.open(img_path)
        else:
            image = Image.open(img_path)

        image.thumbnail((190, 250))
        bio = io.BytesIO()
        image.save(bio, format="PNG")
        return bio.getvalue()
    except Exception as e:
        sg.popup(f'Error Converting Image: {str(e)}')

```

Thus, the cat or image would now be in the correct orientation instead of the one showed earlier in the documentation.

Worker App Design

Input

View

Edit

Worker Name: Garfield Senior

Date: 11-04-2023

Delete

Instant Delete

Action

Select File:

Browse

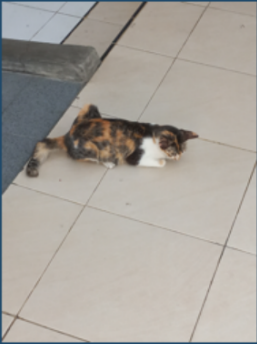
Export Data

Clear Window

Add

View

Input



Browse Image

Show

Add/Edit Image

Name / Title: Garfield Senior

Edit

Date (dd-mm-yyyy): 11-04-2023

Date

Edit

Present Status: Present Absent

Edit

Payment (daily): 1500000.0

Calc

Edit

Penalty (daily): 20000.0

Calc

Edit

Work Done: Catching Mouse (3x50k)
Litter in my desk (20k)

Edit

Some smaller update includes the use of ttk button, The addition of worker's pressence Input widget output, and some small layout padding adjustments.