Initial Interview

- What is your concern?

I am having difficulty of recording different types of expenditure on notes. So I want to have a digital household account.

- What is a category that you spend the most?

It is daily supplies since I am a housewife.

- How do you want to make and design it generally?

I want to have a space where I can write about my budget and income, multiple categories and items, calendar and notes that I can write for reminder.

- Is there any specific parts you want to have?

I wish there is a calculation of expenditure of items per category. Previous digital account I used only had summation of total costs, not per category. That is one of reasons why I started using notes.

- Are there more details you want to add?

I want graphs, that will help me analyzing easier.

- What type of graphs do you want?

Definitely a line graph and more types of graph if possible.

- It depends on how many categories or input values the program takes. Specifically in this case, 1 type of graph would be enough, but I will try to make more than 1.

Thank you.

- When do you want the project to be finished?

The end of January in 2019 please.

- Can the completion date can be modified later?

Yes, up to 3 weeks later would be ok.

- Thank you for having our company, ms. Kim.

Thank you too. +_+

I was asked by the client to take responsibility of the problem, and I will be using a GUI based software for handling her transactions digitally. I will be also developing more on computing, sorting and managing systems. My CS teacher, Mr. Livesey, agreed to be my supervisor for the project.

Client Feedback *(feedback given through a meeting with the client)*

Me: Hi, this meeting is for testing the program and getting feedback from you. I hope this satisfies and reaches your expectation as it has multiple functions as you asked, such as logging in or multiple windows divided. Please have a look.

Client: Ok. This program has login system and has exactly all functions I asked for. I see multiple windows of “Budgets”, “Notes”, “Calendar” and all functions work in a way I asked for. I like the background image and color too.

Me: Is there anything that you want to add or fix?

Client: Hmm, I wish there was a date added automatically to notes when I make notes and more graphs would be better for visual effects. Also these are subtle demands, but having notifications or amount of expenditure and income on specific date will remind me whenever I look at the calendar.

Me: Ok, is there more?

Client: Lastly, I wish I can save data of notes and budgets to help me to track my plans easier when I log-out. This is it, thank you for asking me.

Me: Thank you for providing feedbacks. I will try to satisfy your feedbacks as soon as possible.

```
from tkinter import *
from tkinter import messagebox as ms
import sqlite3
import os

# Connecting to database
with sqlite3.connect('login.db') as db:
    c = db.cursor()

c.execute('CREATE TABLE IF NOT EXISTS user (username TEXT NOT NULL, password
TEXT NOT NULL);')
db.commit()
db.close()


# The only and main Class
class main:
    def __init__(self, master):
        # Window
        self.master = master
        # Variables for username and password
        self.username = StringVar()
        self.password = StringVar()
        self.n_username = StringVar()
        self.n_password = StringVar()

        # Create Widgets
        self.widgets()
        root.title('Login window')

    # Login
    def login(self):
        # Have a connection
        with sqlite3.connect('login.db') as db:
            c = db.cursor()

        # Get data from database
        find_user = ('SELECT * FROM user WHERE username = ? and password = ?')
        c.execute(find_user, [(self.username.get()), (self.password.get())])
        result = c.fetchall()
        if result:
            os.system("python IA.py")
        else:
```

```
            ms.showerror('Oops!', 'Username Not Found.')

    def new_user(self):
        # Have a connection
        with sqlite3.connect('login.db') as db:
            c = db.cursor()

        # Find Existing username
        find_user = ('SELECT * FROM user WHERE username = ?')
        c.execute(find_user, [(self.username.get())])
        if c.fetchall():
            ms.showerror('Error!', 'Username Taken. Try a Different One.')
        else:
            ms.showinfo('Success!', 'Account Created!')
            self.log()

        # Create New Account
        insert = 'INSERT INTO user(username,password) VALUES(?,?)'
        c.execute(insert, [(self.n_username.get()), (self.n_password.get())])
        db.commit()

    # Frame Packing Methods

    def log(self):
        self.username.set('')
        self.password.set('')
        self.crf.pack_forget()
        self.head['text'] = 'LOGIN'
        self.logf.pack()

    def cr(self):
        self.n_username.set('')
        self.n_password.set('')
        self.logf.pack_forget()
        self.head['text'] = 'Create Account'
        self.crf.pack()

    # Display Widgets
    def widgets(self):
        self.head = Label(self.master, text='LOGIN', font=('', 35), pady=10)
        self.head.pack()
        self.logf = Frame(self.master, padx=10, pady=10)
        Label(self.logf, text='Username: ', font=('', 20), pady=5, padx=5).grid(sticky=W)
```

```
        Entry(self.logf, textvariable=self.username, bd=5, font=('', 15)).grid(row=0, column=1)
        Label(self.logf, text='Password: ', font=('', 20), pady=5, padx=5).grid(sticky=W)
        Entry(self.logf, textvariable=self.password, bd=5, font=('', 15), show='*').grid(row=1,
column=1)
        Button(self.logf, text=' Login ', bd=3, font=('', 15), width=8, padx=5, pady=5,
command=self.login).grid()
        Button(self.logf, text=' Create Account ', bd=3, font=('', 15), width=15,padx=5, pady=5,
command=self.cr).grid(row=2, column=1)
        self.logf.pack()

        self.crf = Frame(self.master, padx=10, pady=10)
        Label(self.crf, text='Username: ', font=('', 20), pady=5, padx=5).grid(sticky=W)
        Entry(self.crf, textvariable=self.n_username, bd=5, font=('', 15)).grid(row=0, column=1)
        Label(self.crf, text='Password: ', font=('', 20), pady=5, padx=5).grid(sticky=W)
        Entry(self.crf, textvariable=self.n_password, bd=5, font=('', 15), show='*').grid(row=1,
column=1)
        Button(self.crf, text='Create Account', bd=3, font=('', 15), width=15,padx=5, pady=5,
command=self.new_user).grid()
        Button(self.crf, text='Go to Login', bd=3, font=('', 15), width=13, padx=5, pady=5,
command=self.log).grid(row=2, column=1)


root = Tk()
main(root)
root.mainloop()
```

```
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
#NavigationToolbar2TkAgg
from matplotlib.figure import Figure
import tkinter as tk
import sys
import os
from PIL import Image, ImageTk
import calendar as cd
import time

# fonts
font = ("monaco", 12)
font_tk = ("monaco", 14)


class start(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)
        container = tk.Frame(self)

        container.pack(side="top", fill="both", expand=True)

        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}

        self.title('Jake\'s household account')
        self.geometry("1300x700")

        for Pages in (StartPage, QuickLook, Budgets, Calendar, Notes):
            frame = Pages(container, self)

            # Display Pages
            self.frames[Pages] = frame

            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame(StartPage)
```

```
    def show_frame(self, cont):
        frame = self.frames[cont]
        frame.tkraise()


# Intro
class StartPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.config(background='mediumslateblue')

        homelabel = tk.Label(self, text='Welcome To Jake\'s Household Account!')
        homelabel.pack(padx=5, pady=5)
        homelabel.config(bg='black', fg='white', font=('consolas', 50), height=2, width=50)

        self.homeimg = ImageTk.PhotoImage(Image.open('steak.png'))
        self.img = tk.Label(self, image=self.homeimg)
        self.homeimg.image = self.homeimg
        self.img.pack()

        # Restart Button
        buttonRe = tk.Button(self, text='Restart')

        def restart():
            python = sys.executable
            os.execl(python, python, *sys.argv)
        buttonRe.config(command=restart)
        buttonRe.pack(side=tk.BOTTOM, pady=3)

        # Quit Button
        buttonQuit = tk.Button(self, text='Quit')

        def quit():
            self.quit()
        buttonQuit.config(command=quit)
        buttonQuit.pack(side=tk.BOTTOM, pady=3)

        # Click Continue to continue
        buttonQuickLook = tk.Button(self, text="Continue", command=lambda:
controller.show_frame(QuickLook))
        buttonQuickLook.pack(side=tk.BOTTOM, pady=3)
```

```
class QuickLook(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.config(background='paleturquoise')
        # Main home_label
        homeL= tk.Label(self, text='Welcome To Jake\'s Household Account!?!')
        homeL.pack(padx=5, pady=5)
        homeL.config(bg='black', fg='white', font=('consolas', 50), height=1, width=50)

        titleL = tk.Label(self, text='This is QuickLook page.', bg='black', fg='white', font=font,
height=2, width=30)
        titleL.pack()

        # Image of Warren Buffett
        self.homeimg = ImageTk.PhotoImage(Image.open('warren_buffett.png'))
        self.img = tk.Label(self, image=self.homeimg)
        self.homeimg.image = self.homeimg
        self.img.pack(pady=15)

        pagesL = tk.Label(self, text='List of Windows', font=font, height=1)
        pagesL.place(x=10, y=100)

        # Buttons to direct to menu_windows
        # main_menus = ['Account', 'QuickLook', 'Budgets', 'Calendar', 'Charts&Report', 'Notes',
'Setting']

        quicklookB = tk.Button(self, text='QuickLook', width=10, command=lambda:
controller.show_frame(QuickLook))
        quicklookB.place(x=1, y=150)

        budgetB = tk.Button(self, text='Budgets', width=10, command=lambda:
controller.show_frame(Budgets))
        budgetB.place(x=1, y=300)

        calendarB = tk.Button(self, text="Calendar", width=10, command=lambda:
controller.show_frame(Calendar))
        calendarB.place(x=1, y=450)

        notesB = tk.Button(self, text='Notes', width=10, command=lambda:
controller.show_frame(Notes))
        notesB.place(x=1, y=600)
```

```
        # Quit Button
        def logout():
            self.quit()

        logoutB = tk.Button(self, text='Log-out', width=10, command=logout)
        logoutB.pack(side=tk.RIGHT)

        buttonRe = tk.Button(self, text='Restart')

        # Restart Button
        def restart():
            python = sys.executable
            os.execl(python, python, *sys.argv)

        buttonRe.config(command=restart, width=10)
        buttonRe.pack(side=tk.RIGHT)


class Budgets(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        homeL = tk.Label(self, text='Welcome To Jake\'s Household Account!?!')
        homeL.pack(padx=5, pady=5)
        homeL.config(bg='black', fg='white', font=('consolas', 50), height=1, width=50)

        titleL = tk.Label(self, text='This is Budgets page.', bg='black', fg='white', font=font, height=2,
width=30)
        titleL.pack()

        button1 = tk.Button(self, text="Back to Quicklook", command=lambda:
controller.show_frame(QuickLook))
        button1.pack()

        expense_recommended = tk.Label(self, text='Recommended Expense = Transportation,
Bills, Clothing, Food, Health Care, Housing, Leisure, Travel, Loans, Others', fg='deepskyblue',
font=('monaco', 13))
        expense_recommended.pack(pady=7)

        income_recommended = tk.Label(self, text='Recommended Income = Child Support,
Investments, Rental, Salary & Wages, Social Security, Others', fg='mediumvioletred',
font=('monaco', 13))
```

```
        income_recommended.pack()


        # Balance
        frame_bottom = tk.LabelFrame(self, text='Balance', relief=tk.GROOVE)

        option = tk.Label(frame_bottom, text="Click to display option", bg='black', fg='white',
width=20, height=1)
        option.pack(side=tk.BOTTOM, padx=10, pady=15)

        # Middle Frame
        middle_frame = tk.Frame(self)
        middle_frame.pack(side=tk.TOP)

        i = self

        # Expense
        def expense():
            category_name = tk.Label(middle_frame, text="What is the category's name?")
            category_name.pack()

            categoryE = tk.Entry(middle_frame)
            categoryE.pack()

            amount_income = tk.Label(middle_frame, text="What is amount of expense?")
            amount_income.pack()

            amountE = tk.Entry(middle_frame)
            amountE.pack()

            li = []
            name_list = []
            money_list = []

            def get_data():
                li.append(categoryE.get())
                li.append(amountE.get())

                for data in li:
                    if len(data) % 2 == 0:
                        string_categories = data
                        name_list.append(string_categories)
```

```
            if len(data) % 2 == 1:
                number_money = data
                money_list.append(number_money)

    get_data = tk.button = tk.Button(self, text="get data", command=get_data)
    get_data.pack(side=tk.TOP)

    # Graphs
    def graph_window():
        # count = 0
        top = tk.Toplevel(i)
        top.wm_title("Graph")  # % self.counter)
        label = tk.Label(top, text="This is Chart")  # #%s")# % self.counter)
        label.pack()  # side="top", fill="both", expand=True, padx=100, pady=100)

        figure = Figure(figsize=(5, 5), dpi=100)
        plots = figure.add_subplot(111)

        plots.plot(name_list, money_list)
        canvas = FigureCanvasTkAgg(figure, top)
        canvas.draw()
        canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

    graph = tk.button = tk.Button(self, text="Graph", command=graph_window)
    graph.pack(side=tk.TOP)

# Income
def income():
    category_name = tk.Label(middle_frame, text="What is the category's name?")
    category_name.pack()

    categoryE = tk.Entry(middle_frame)
    categoryE.pack()

    amount_income = tk.Label(middle_frame, text="What is amount of income?")
    amount_income.pack()

    amountE = tk.Entry(middle_frame)
    amountE.pack()

    li = []
    name_list = []
    money_list = []
```

```
    def get_data():
        li.append(categoryE.get())
        li.append(amountE.get())

        for data in li:
            if len(data) % 2 == 0:
                string_categories = data
                name_list.append(string_categories)

            if len(data) % 2 == 1:
                number_money = data
                money_list.append(number_money)

    graph = tk.button = tk.Button(self, text="get data", command=get_data)
    graph.pack(side=tk.TOP)

    # Graphs
    def graph_window():
        # count = 0
        top = tk.Toplevel(i)
        top.wm_title("Graph")
        label = tk.Label(top, text="This is Chart")
        label.pack()

        figure = Figure(figsize=(5, 5), dpi=100)
        plots = figure.add_subplot(111)

        plots.plot(name_list, money_list)
        canvas = FigureCanvasTkAgg(figure, top)
        canvas.draw()
        canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)
        # graph.destroy()

    graph = tk.button = tk.Button(self, text="Graph", command=graph_window)
    graph.pack(side=tk.TOP)

# create a menu
popup = tk.Menu(self, tearoff=0)
popup.add_command(label="Expense", command=expense)
popup.add_separator()
popup.add_command(label="Income", command=income)
```

```
        def do_popup(event):
            # try:
                popup.tk_popup(event.x_root, event.y_root, 0)
            # finally:
                popup.grab_release()

        option.bind("<Button-1>", do_popup)

        frame_bottom.place(x=200, y=574)


class Calendar(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        # Title of Calendar
        homeL = tk.Label(self, text='Welcome To Jake\'s Household Account!?!')
        homeL.pack(padx=5, pady=5)
        homeL.config(bg='black', fg='white', font=('consolas', 50), height=1, width=50)

        # Page Label
        titleL = tk.Label(self, text='This is Calendar page.', bg='black', fg='white', font=font,
height=2, width=30)
        titleL.pack()

        # QuickLook Button
        quicklookB = tk.Button(self, text="Back to QuickLook", command=lambda:
controller.show_frame(QuickLook))
        quicklookB.pack()

        # Today's calendar                                          # % H: % M
        calendar_today = tk.Label(self, text='Today\'s date is
{}'.format(time.strftime("%Y-%m-%d")), font=("monaco", 13, 'bold'))
        calendar_today.pack(padx=7, pady=7)

        group = tk.LabelFrame(self, text="Calendar", padx=5, pady=5)
        group.pack(padx=10, pady=10)

        # Year
        year_label = tk.Label(group, text='which year?')
        year_label.pack()
```

```
        yearE = tk.Entry(group)
        yearE.pack()

        # Month
        month_label = tk.Label(group, text='which month?')
        month_label.pack()

        monthE = tk.Entry(group)
        monthE.pack()

        # Calendar
        def calendar_button():
            calendar = cd.month(int(yearE.get()), int(monthE.get()))
            calendar_display = tk.Label(self, text=calendar, font=("monaco", 25), bg='lightskyblue')
            calendar_display.pack()

            # Remove calendar entries
            def calendar_destroy():
                calendar_display.destroy()
                calendar_remove.destroy()
                yearE.delete(0, tk.END)
                monthE.delete(0, tk.END)

            calendar_remove = tk.Button(self, text="Click to remove a calendar",
command=calendar_destroy)
            calendar_remove.pack()

        calendar_getB = tk.Button(self, text='Click to see a calendar', command=calendar_button)
        calendar_getB.pack()


class Notes(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        # Title and design of the page
        homeL = tk.Label(self, text='Welcome To Jake\'s Household Account!?!')
        homeL.pack(padx=5, pady=5)
        homeL.config(bg='black', fg='white', font=('consolas', 50), height=1, width=50)

        titleL = tk.Label(self, text='This is Notes page.', bg='black', fg='white', font=font, height=2,
width=30)
```

```
        titleL.pack()

        quicklookB = tk.Button(self, text="Back to QuickLook", command=lambda:
controller.show_frame(QuickLook))
        quicklookB.pack(padx=5, pady=5)

        # Click to make notes
        def make_notes():
            notesB = tk.Text(self, bg='lightsalmon', width=40, bd=5, height=5, font=font)
            notesB.pack()

            # Remove notes
            def remove_notes():
                notesB.destroy()
                remove_notesB.destroy()

            remove_notesB = tk.Button(self, text='Click to remove notes', command=remove_notes)
            remove_notesB.pack()

        # Button click to make notes
        make_notesB = tk.Button(self, text='Click to make notes', command=make_notes)
        make_notesB.pack(padx=5, pady=5)


app = start()
app.mainloop()
```