اونيۏرسيتي مليسيا ڤهڠ السلطان عبد الله

UNIVERSITI MALAYSIA PAHANG

AL-SULTAN ABDULLAH

UNIVERSITI MALAYSIA PAHANG AL-SULTAN ABDULLAH

CENTRE OF MATHEMATICS SCIENCES

BSD2213 DATA SCIENCE PROGRAMMING 1

GROUP PROJECT TITLE : EQUIDEBT

LECTURER'S NAME : DR. NORAZIAH BINTI ADZHAR

SEMESTER 1 SESSION 2023/2024

GROUP MEMBERS

NO	NAME	MATRIC NO
1.	NUR SABIHAH BINTI ANUAR	SD22019
2.	PUTRI BALQIS BATRISYIA BINTI MOHD RIZAL	SD22012
3.	NURUL NAJWA BINTI NORHISHAM	SD22035
4.	MUHAMMAD DANISH AIMAN HARISS BIN ROSLI	SD22008

TABLE OF CONTENT

1.0 INTRODUCTION

Equidebt is an expense sharing app specially designed to streamline the often difficult process of managing shared expenses among friends, family or colleagues. With the increasing trend of collaborative living arrangements and group activities, this app provides a digital solution to track and settle shared bills and also making financial collaboration hassle-free.

For example, this situation illustrates that there is someone covering the Uber ride, another person takes care of the food and drinks, and yet another handles the hotel costs. Then, they need to keep track of all these expenses and divide the costs among the participants without any disputes arising. So, by using EquiDebt, the app can step in to efficiently track and reconcile all these expenses, ensuring a fair and straightforward way to split the overall cost among the participants without any confusion and chaos.

With features like real-time expense tracking, seamless expense sharing, and access from anywhere either through the website, android app or iPhone app. The user can efficiently manage all costs per individual, track 'who paid how much' via their mobile devices or from a desktop browser. EquiDebt ensures fair and transparent financial collaboration. Plus, it can work offline, providing flexibility even in low-connectivity areas.

Therefore, apps like Equidebt play a crucial role in simplifying group finances, promoting transparency, and fostering better financial collaboration. As financial dynamics continue to evolve in our interconnected world, these applications provide a modern solution for managing shared expenses in an effective and user-friendly manner. The application's user-friendly design suits people of all ages, making sure that both young and old users can use it easily without any problem. You can say goodbye to confusion and welcome an efficient way to manage group expenses.

2.0 WHY IS THIS PROJECT?

Equidebt is a platform that was created to make the process of managing shared spending and finances among groups of people easier. It allows the users to enter their shared expenses, such as bills, rent, food, or group activities with their friends, roommates, family members, or any group of individuals who share the expenses.

This app calculates how much each individual owes based on their shared expenses. It allows users to split the cost equally or unequally, so that they can customise the distribution based on specific proportion or price. It is fair for everyone involved. They can use this app to track and pay their financial transactions in a fair and transparent manner. It's a practical solution for anyone who wants to avoid misunderstandings and conflict related to shared expenses.

Our application offers a user-friendly interface to ensure a smooth experience for the users. The design of our application is thoughtfully designed to be easily navigable, with a clean and organised layout that allows the users to access the features effortlessly. Whether the seasoned user or the newcomer, the users will find that this application interface is easy to understand. This aims to ensure that users of various ages use this application more easily and smoothly.

3.0 HOW THIS PROJECT CAN BE EXTENDED?

The expense tracker feature in this application serves as the backbone of efficient financial management within group settings. This essential tool allows the users to monitor every expense and ensure a comprehensive overview of shared financial activities. Expense trackers allow the users to enter the exact details such as who paid, what the expense was for and the amount. It produces a transparent and accurate ledger. By providing real-time updates and calculations, this feature simplifies the process of sharing and dividing costs for everyone in the group. This feature not only simplifies the tracking procedure, but it also promotes a collaborative and harmonious financial environment among the users. This feature not only simplifies the tracking procedure, but it also promotes collaborative and harmonious financial conditions among users.

One notable and user-friendly aspect of the EquiDebt project is its commitment to user satisfaction and financial transparency, our platform will not charge them with any fees. We believe in providing a seamless and fair experience for our users, it allows them to track and split their expenses without any burden of additional charges. At the same time, the user can use this application without worrying about the hidden costs. This strategy not only sets the project differently in a market characterised by transaction costs, but it also reflects a user-centric attitude that prioritises its users financial well-being and satisfaction.

The availability of real-time date and time capability in these applications is critical for presenting the users with the correct and up-to-current information. The date and time feature ensures that the users may accurately manage the transactions and shared expenses by displaying the current time and date. At the same time, it can also contribute to the transparency and reliability of the financial data presented. Whether the user is entering a new expense or settling a debt, the real-time date functionality provides a clear and immediate understanding of when each financial transaction occurred.

4.0 SOURCE CODE

```
import tkinter as tk
from tkinter import ttk, messagebox
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from datetime import datetime
from PIL import ImageTk, Image

class User:
    def __init__(self, username, password):
        self.username = username
        self.password = password
        self.group = Group()

    def __str__(self):
        return f"{self.description} - RM{self.amount:.2f}"


class Group:
    def __init__(self):
        self.expenses = []

    def add_expense(self, expense):
        self.expenses.append(expense)

    def delete_expense_by_description(self, description):
        self.expenses = [expense for expense in self.expenses if str(expense)
!= description]

    def calculate_balances(self):
        balances = {}
        for expense in self.expenses:
            total_people = len(expense.split_details) + 1
            amount_per_person = expense.amount / total_people

            for person in expense.split_details:
                if person not in balances:
                    balances[person] = 0

                if person == expense.paid_by:
                    balances[person] -= expense.amount
                else:
                    balances[person] += amount_per_person

        return balances


class PersonList:
    def __init__(self, parent_frame):
        self.people_frame = ttk.Frame(parent_frame, style="Custom.TFrame")
        self.people_frame.grid(row=0, column=0, padx=10, pady=10)
```

```
        # Set column weights to center the elements
        self.people_frame.columnconfigure(0, weight=3, uniform='a')
        self.people_frame.columnconfigure(1, weight=3, uniform='a')
        self.people_frame.columnconfigure(2, weight=3, uniform='a')

        ttk.Label(self.people_frame, font=("Comic Sans MS", 15),
text="People:", style="Custom.TLabel").grid(row=0, column=2, columnspan=3,
pady=10)

        # Adjust the column for Listbox, Entry, and Button
        self.people_listbox = tk.Listbox(self.people_frame,
selectmode=tk.SINGLE, font=("Comic Sans MS", 10))
        self.people_listbox.grid(row=1, column=2, pady=5, padx=10,
sticky="nsew")

        self.people_entry = ttk.Entry(self.people_frame, font=("Comic Sans
MS", 10))
        self.people_entry.grid(row=2, column=2, pady=5, padx=5, sticky="nsew")

        ttk.Button(self.people_frame, text="Add Person",
command=self.add_person, style="TButton").grid(row=3, column=2,

pady=10,

padx=10,

sticky="nsew")


        self.people = []

    def add_person(self):
        person = self.people_entry.get()
        if person:
            self.people_listbox.insert(tk.END, person)
            self.people_entry.delete(0, tk.END)

    def get_people(self):
        return list(self.people_listbox.get(0, tk.END))


class Authentication:
    def __init__(self):
        self.users = {}

    def register_user(self, username, password):
        if username not in self.users:
            self.users[username] = User(username, password)
            return True
        return False
```

```
  def authenticate_user(self, username, password):
        user = self.users.get(username)
        if user and user.password == password:
            return user
        return None


class LoginRegisterWindow:
   def __init__(self, root, authentication, on_successful_login):
        self.root = root
        self.authentication = authentication
        self.on_successful_login = on_successful_login

        ttk.Label(root, text="WELCOME TO EQUIDEBT", style="Custom.TLabel",
font=("Comic Sans MS", 20)).grid(row=0, column=1, padx=10, pady=10,
sticky="nsew")


        self.login_register_frame = ttk.Frame(root, style="Custom.TFrame",
width=700, height=500)
        self.login_register_frame.grid(row=3, column=2, pady=60, padx=10)

        ttk.Label(self.login_register_frame, text="Username:", font=("Comic
Sans MS", 10)).grid(row=0, column=0, pady=20, padx=5, sticky="e")
        self.username_entry = ttk.Entry(self.login_register_frame)
        self.username_entry.grid(row=0, column=1, pady=5)

        ttk.Label(self.login_register_frame, text="Password:", font=("Comic
Sans MS", 10)).grid(row=1, column=0,  pady=25, padx=5, sticky="e")
        self.password_entry = ttk.Entry(self.login_register_frame, show="*")
        self.password_entry.grid(row=1, column=1, pady=5)

        ttk.Button(self.login_register_frame, text="Login",
command=self.login).grid(row=3, column=1, pady=10)
        ttk.Button(self.login_register_frame, text="Register",
command=self.register).grid(row=2, column=1, pady=10)

        # Center the frame
        self.center_frame()

   def center_frame(self):
        screen_width = self.root.winfo_screenwidth()
        screen_height = self.root.winfo_screenheight()

        frame_width = 1000  # Adjust this value based on your frame's width
        frame_height = 600  # Adjust this value based on your frame's height

        x = (screen_width - frame_width) // 2
        y = (screen_height - frame_height) // 2

        self.root.geometry(f"{frame_width}x{frame_height}+{x}+{y}")

   def login(self):
```

```
        username = self.username_entry.get()
        password = self.password_entry.get()
        user = self.authentication.authenticate_user(username, password)

        if user:
            self.on_successful_login(user)
        else:
            messagebox.showerror("Error", "Invalid username or password")

    def register(self):
        username = self.username_entry.get()
        password = self.password_entry.get()

        if self.authentication.register_user(username, password):
            messagebox.showinfo("Success", "Registration successful. You can
now log in.")
        else:
            messagebox.showerror("Error", "Username already exists. Please
choose a different username.")

class ExpenseApp:
    def __init__(self, parent_frame, person_listbox, current_user):
        self.expenses_list = []
        self.parent_frame = parent_frame
        self.person_listbox = person_listbox
        self.current_user = current_user

        notebook = ttk.Notebook(parent_frame, style="Custom.TFrame")
        notebook.grid(row=1, column=0, columnspan=2, pady=10)

        # Person List Tab
        person_list_tab = ttk.Frame(notebook, style="Custom.TFrame")
        notebook.add(person_list_tab, text="Person List")
        self.person_list = PersonList(person_list_tab)

        # Expense Details Tab
        expense_details_tab = ttk.Frame(notebook, style="Custom.TFrame")
        notebook.add(expense_details_tab, text="Expense Details")
        self.setup_expense_details_tab(expense_details_tab)

        # View Balances Tab
        view_balances_tab = ttk.Frame(notebook, style="Custom.TFrame")
        notebook.add(view_balances_tab, text="View Balances")
        self.setup_view_balances_tab(view_balances_tab)

        self.group = Group()

        # Expense tracker tab
        expense_tracker_tab = ttk.Frame(notebook, style="Custom.TFrame")
        notebook.add(expense_tracker_tab, text="Expense tracker")
        self.setup_expense_tracker_tab(expense_tracker_tab)

    def setup_expense_details_tab(self, frame):
```

```
        # Configure columns to have equal weight
        frame.columnconfigure(0, weight=1, uniform='a')
        frame.columnconfigure(1, weight=1, uniform='a')
        frame.columnconfigure(2, weight=1, uniform='a')

        ttk.Label(frame, font=("Comic Sans MS", 15), text="Enter expense
details:", style="Custom.TLabel").grid(row=0,

column=0,

columnspan=3,

pady=10)

        ttk.Label(frame, text="Description:",
style="Custom.TLabel").grid(row=1, column=0, columnspan=2, padx=5)
        self.description_entry = ttk.Entry(frame, font=("Comic Sans MS", 10))
        self.description_entry.grid(row=1, column=1, pady=5)

        ttk.Label(frame, text="Amount (RM):",
style="Custom.TLabel").grid(row=2, column=0, columnspan=2, padx=5)
        self.amount_entry = ttk.Entry(frame, font=("Comic Sans MS", 10))
        self.amount_entry.grid(row=2, column=1, pady=5)

        ttk.Label(frame, text="Paid by:", style="Custom.TLabel").grid(row=3,
column=0, columnspan=2, padx=5)
        self.paid_by_var = tk.StringVar()
        self.paid_by = ttk.Combobox(frame, textvariable=self.paid_by_var,
values=self.person_list.get_people(),
                                    font=("Comic Sans MS", 10))
        self.paid_by.grid(row=3, column=1, pady=5)

        self.paid_by.bind("<<ComboboxSelected>>", lambda event:
self.update_split_options(event))

        ttk.Label(frame, text="Split by:", style="Custom.TLabel").grid(row=4,
column=0, columnspan=2, padx=5)

        self.split_options = ["Equally", "Unequally"]
        self.split_var = tk.StringVar()
        self.split_var.set(self.split_options[0])
        self.split_dropdown = ttk.Combobox(frame, textvariable=self.split_var,
values=self.split_options,
                                           font=("Comic Sans MS", 10))
        self.split_dropdown.grid(row=4, column=1, pady=5)

        self.split_dropdown.bind("<<ComboboxSelected>>", lambda event:
self.update_split_options(event))

        self.split_options_frame = ttk.Frame(frame, style="Custom.TFrame")
        self.split_options_frame.grid(row=5, column=0, columnspan=3)
```

```
        self.expenses_text = tk.Text(frame, height=10, width=30, font=("Comic
Sans MS", 10))
        self.expenses_text.grid(row=6, column=0, columnspan=3, pady=10)

        self.buttons_frame = ttk.Frame(frame, style="Custom.TFrame")
        self.buttons_frame.grid(row=7, column=0, columnspan=3, pady=10)

        ttk.Button(self.buttons_frame, text="Add Expense",
command=self.add_expense, style="TButton").grid(row=0,

column=0,

padx=5)
        ttk.Button(self.buttons_frame, text="Reset Details",
command=self.reset_details, style="TButton").grid(row=0,

column=1,

padx=5)

    def setup_view_balances_tab(self, frame):
        frame.columnconfigure(0, weight=1, uniform='a')
        ttk.Label(frame, font=("Comic Sans MS", 15), text="Balances:",
style="Custom.TLabel").grid(row=0, column=0,

pady=10)

        self.balances_text = tk.Text(frame, height=20, width=30, font=("Comic
Sans MS", 10))
        self.balances_text.grid(row=1, column=0, pady=5)

        ttk.Button(frame, text="Calculate Balances",
command=self.calculate_and_display_balances, style="TButton").grid(
            row=2, column=0, pady=10)

    def update_split_options(self, event):
        selected_option = self.split_var.get()

        # Destroy only Entry widgets, not other widgets
        for widget in self.split_options_frame.winfo_children():
            if isinstance(widget, tk.Entry):
                widget.destroy()

        ttk.Label(self.split_options_frame, text="Split by:",
background="#81DAF5", font=("Comic Sans MS", 10)).grid(
            row=0, column=0, sticky="E", pady=5)

        # Add new label
        ttk.Label(self.split_options_frame, text="Split by:",
background="#81DAF5", font=("Comic Sans MS", 10)).grid(
            row=0, column=0, sticky="E", pady=5)

        if selected_option == "Equally":
```

```
            # Initialize checkbox_vars
            self.checkbox_vars = [tk.BooleanVar() for _ in
range(len(self.person_list.get_people()))]

            # Create Checkbuttons for each person
            for i, person in enumerate(self.person_list.get_people()):
                ttk.Checkbutton(self.split_options_frame, text=person,
variable=self.checkbox_vars[i],
                                style="TCheckbutton").grid(row=i + 1,
column=0, pady=5)

        elif selected_option == "Unequally":
            # Initialize amount_entries
            self.amount_entries = []

            # Create Labels and Entry widgets for each person
            for i, person in enumerate(self.person_list.get_people()):
                ttk.Label(self.split_options_frame, text=person,
background="#FFFFFF", font=("Comic Sans MS", 10)).grid(
                    row=i + 1, column=0, sticky="E", pady=5)
                entry_var = tk.DoubleVar()
                ttk.Entry(self.split_options_frame, textvariable=entry_var,
font=("Comic Sans MS", 10)).grid(row=i + 1,

column=1,

pady=5)
                self.amount_entries.append(entry_var)

    def add_expense(self):
        selected_option = self.split_var.get()

        if selected_option == "Equally":
            selected_people = [person for person, var in
zip(self.person_list.get_people(), self.checkbox_vars) if
                               var.get()]
        elif selected_option == "Unequally":
            amounts = [entry_var.get() for entry_var in self.amount_entries if
hasattr(entry_var, 'get')]
            selected_people = list(zip(self.person_list.get_people(),
amounts))

        description = self.description_entry.get()
        amount = self.amount_entry.get()
        paid_by = self.paid_by_var.get()

        # Get current date and time in the desired format
        date_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        expense_info_str = self.get_expense_info_str(description, amount,
paid_by, selected_option, selected_people)
```

```
        expense_info = {"description": description, "amount": amount,
"paid_by": paid_by,
                        "split_method": selected_option, "split_details":
selected_people, "date_time": date_time}

        self.expenses_list.append(expense_info)
        self.expenses_text.insert(tk.END,
self.get_expense_info_str(description, amount, paid_by, selected_option,

selected_people) + "\n")

        # Update the Treeview in the Expense Tracker tab
        self.update_expense_tracker_treeview()

    def update_expense_tracker_treeview(self):
        # Assuming self.expense_treeview is the reference to your Treeview
widget
        # Clear existing items in the Treeview
        for item in self.expense_treeview.get_children():
            self.expense_treeview.delete(item)

        # Iterate through the expenses_list and insert each expense into the
Treeview
        for expense_info in self.expenses_list:
            self.expense_treeview.insert("", tk.END, values=(
                expense_info.get("description", ""),
                expense_info.get("amount", ""),
                expense_info.get("paid_by", ""),
                expense_info.get("split_method", ""),
                expense_info.get("date_time", "")  # Use "date_time" from
expense_info
            ))

    def reset_details(self):
        self.description_entry.delete(0, tk.END)
        self.amount_entry.delete(0, tk.END)
        self.paid_by_var.set('')
        self.split_var.set(self.split_options[0])

        for widget in self.split_options_frame.winfo_children():
            widget.destroy()

        # Reset the checkbox_vars and amount_entries if they exist
        if hasattr(self, 'checkbox_vars'):
            del self.checkbox_vars
        if hasattr(self, 'amount_entries'):
            del self.amount_entries

        ttk.Label(self.split_options_frame, text="Split by:",
background="#81DAF5", font=("Comic Sans MS", 10)).grid(
            row=0, column=0, sticky="E", pady=5)

    def calculate_and_display_balances(self):
```

```
        # Assuming you have a tkinter GUI and a text widget named
self.balances_text
        self.balances_text.delete(1.0, tk.END)

        # Calculate individual expenses
        individual_expenses = {}
        for expense in self.expenses_list:
            if expense['split_method'] == 'Equally':
                amount_per_person = float(expense['amount']) /
len(expense['split_details'])
                for person in expense['split_details']:
                    if person not in individual_expenses:
                        individual_expenses[person] = 0.0
                    individual_expenses[person] += amount_per_person
            elif expense['split_method'] == 'Unequally':
                for person, amount in expense['split_details']:
                    if person not in individual_expenses:
                        individual_expenses[person] = 0.0
                    individual_expenses[person] += amount

        # Display individual expenses
        self.balances_text.insert(tk.END, "The calculated individual expenses
are:\n")
        for person, expense in individual_expenses.items():
            self.balances_text.insert(tk.END, f"{person}: RM {expense:.2f}\n")

        # Calculate and display balances
        self.balances_text.insert(tk.END, "\nNow, let's look at the
balances:\n")
        owes_dict = {person: 0.0 for person in individual_expenses.keys()}

        for person1, expense1 in individual_expenses.items():
            for person2, expense2 in individual_expenses.items():
                if person1 != person2:
                    debt = expense2 - expense1
                    if debt > 0:
                        owes_dict[person2] += debt

        # Display owes for each person
        for person, amount in owes_dict.items():
            self.balances_text.insert(tk.END, f"{person} owes: RM
{amount:.2f}\n")



    def get_expense_info_str(self, description, amount, paid_by,
selected_option, selected_people):
        if selected_option == "Equally":
            return f"{description}: RM {amount}, Equally Split Among: {',
'.join(selected_people)}"
        elif selected_option == "Unequally":
            return f"{description}: RM {amount}, Unequally Split Among: {',
'.join([f'{person} ({amt})' for person, amt in selected_people])}"
```

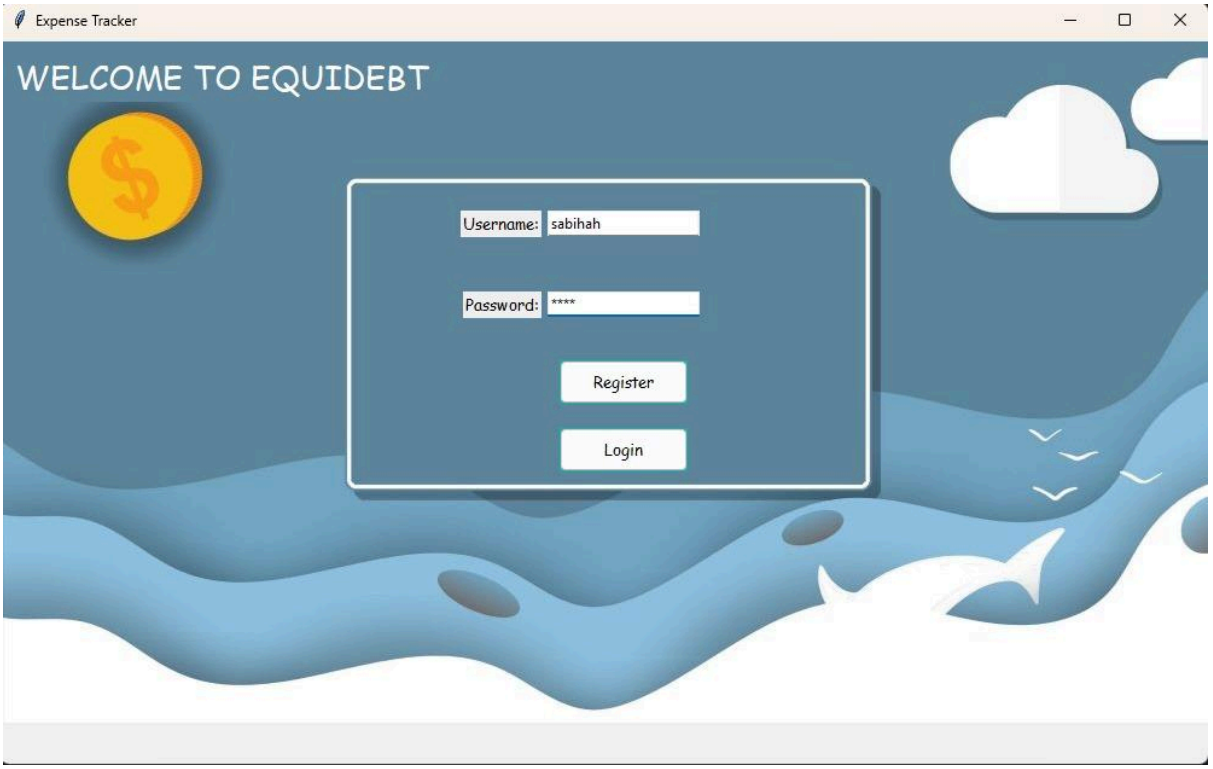
```
    def get_expense_info_str(self, description, amount, paid_by,
selected_option, selected_people):
        if selected_option == "Equally":
            return f"{description}: RM {amount}, Equally Split Among: {',
'.join(selected_people)}"
        elif selected_option == "Unequally":
            return f"{description}: RM {amount}, Unequally Split Among: {',
'.join([f'{person} ({amt})' for person, amt in selected_people])}"

    def setup_expense_tracker_tab(self, frame):
        ttk.Label(frame, text="Expense History:",
style="Custom.TLabel").grid(row=0, column=0, pady=10)

        # Expense History Treeview
        columns = ("Description", "Amount", "Paid By", "Split Method", "Date
and Time")
        self.expense_treeview = ttk.Treeview(frame, columns=columns,
show="headings", height=15)
        for col in columns:
            self.expense_treeview.heading(col, text=col)
        self.expense_treeview.grid(row=2, column=0, pady=5)

        ttk.Button(frame, text="Add Expense",
command=self.add_expense_to_tracker, style="TButton").grid(row=3,

column=0,

pady=10)

    def add_expense_to_tracker(self):
        # Get information from the Expense Details Tab
        selected_option = self.split_var.get()

        if selected_option == "Equally":
            selected_people = [person for person, var in
zip(self.person_list.get_people(), self.checkbox_vars) if
                               var.get()]
        elif selected_option == "Unequally":
            amounts = [entry_var.get() for entry_var in self.amount_entries if
hasattr(entry_var, 'get')]
            selected_people = list(zip(self.person_list.get_people(),
amounts))

        description = self.description_entry.get()
        amount = self.amount_entry.get()
        paid_by = self.paid_by_var.get()

        # Get current date and time in the desired format
        date_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

```
        expense_info = {"description": description, "amount": amount,
"paid_by": paid_by,
                        "split_method": selected_option, "date_time":
date_time}

        # Append the expense information to the expenses list
        self.expenses_list.append(expense_info)

        # Add the expense to the Treeview
        self.expense_treeview.insert("", "end", values=(
            expense_info.get("description", ""),
            expense_info.get("amount", ""),
            expense_info.get("paid_by", ""),
            expense_info.get("split_method", ""),
            expense_info.get("date_time")  # Use "date_time" from expense_info
        ))
        # Clear the details in the Expense Details Tab
        self.reset_details()

    def add_expense_to_treeview(self):
        selected_option = self.split_var.get()

        if selected_option == "Equally":
            selected_people = [person for person, var in
zip(self.person_list.get_people(), self.checkbox_vars) if
                               var.get()]
        elif selected_option == "Unequally":
            amounts = [entry_var.get() for entry_var in self.amount_entries if
hasattr(entry_var, 'get')]
            selected_people = list(zip(self.person_list.get_people(),
amounts))

        description = self.description_entry.get()
        amount = self.amount_entry.get()
        paid_by = self.paid_by_var.get()

        # Get current date and time in the desired format
        date_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        expense_info = {"description": description, "amount": amount,
"paid_by": paid_by,
                        "split_method": selected_option, "date_time":
date_time}

        self.expenses_list.append(expense_info)
        self.expenses_text.insert(tk.END,
self.get_expense_info_str(description, amount, paid_by, selected_option,
                                                                    date_time)
+ "\n")

        # Add the expense to the Treeview
        self.expense_treeview.insert("", "end", values=(
            expense_info.get("description", ""),
```

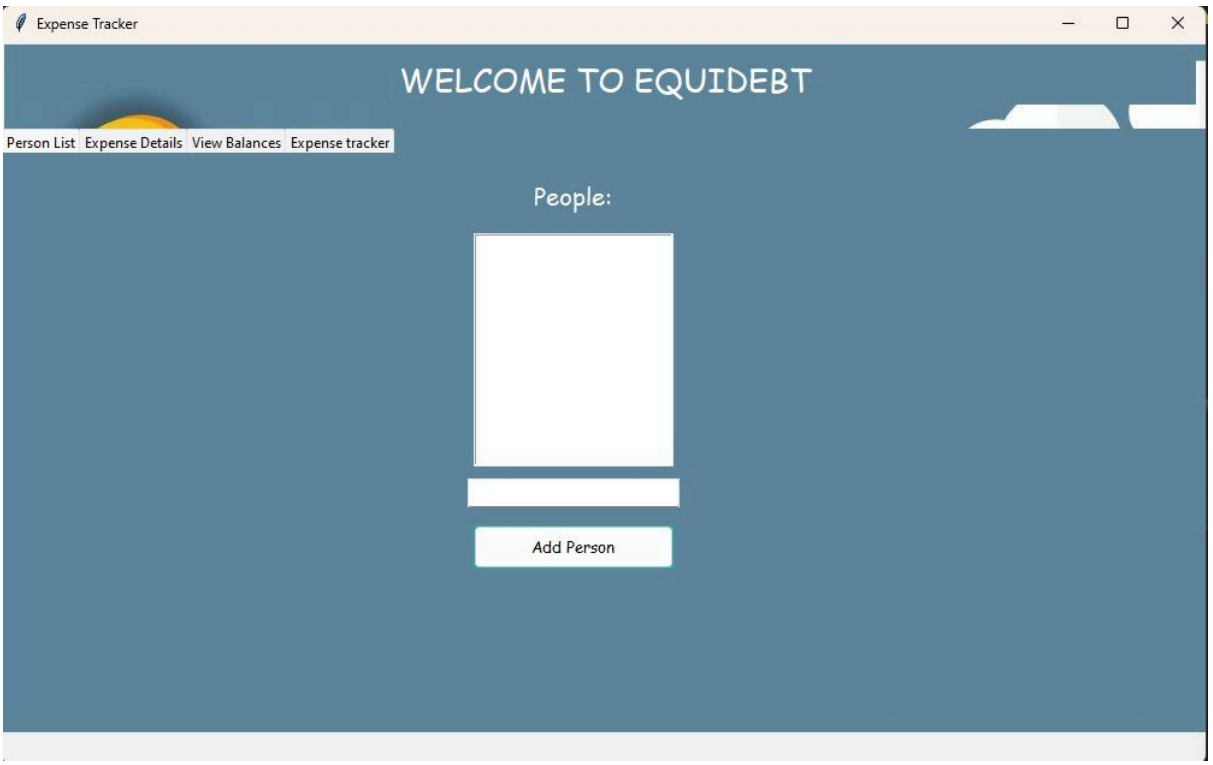
```
            expense_info.get("amount", ""),
            expense_info.get("paid_by", ""),
            expense_info.get("date_time", "")  # Use "date_time" from
expense_info
        ))



    def display_expense_history(self):
        # Clear existing content in the Text widget
        self.expense_history_text.delete(1.0, tk.END)

        # Display expense history
        for expense_info in self.expenses_list:
            description = expense_info.get('description', '')
            amount = expense_info.get('amount', '')
            paid_by = expense_info.get('paid_by', '')
            split_method = expense_info.get('split_method', '')
            split_details = expense_info.get('split_details', '')

            expense_info_str = self.get_expense_info_str(description, amount,
paid_by, split_method, split_details)
            self.expense_history_text.insert(tk.END, expense_info_str + "\n")

if __name__ == "__main__":
    root = tk.Tk()
    root.title("Expense Tracker")
    root.geometry("1003x564")

    image_0 = Image.open('C:\\Users\\User\\Pictures\\Blue Cream Aesthetic
Illustration Group Project Presentation.jpg')
    bck_end = ImageTk.PhotoImage(image_0)
    lbl = tk.Label(root, image=bck_end)
    lbl.place(x=0, y=0)

    ttk.Label(root, text="WELCOME TO EQUIDEBT", style="Custom.TLabel",
font=("Comic Sans MS", 20)).grid(row=0, column=1,

padx=10,

pady=10,

sticky="w")
    style = ttk.Style()
    style.configure("Custom.TFrame", background="#5D869D")
    style.configure("Custom.TLabel", background="#5D869D", font=("Comic Sans
MS", 10), foreground="#FFFFFF")
    style.configure("TButton", background="#20B2AA", font=("Comic Sans MS",
10), padding=5)

    authentication = Authentication()

    person_list_tab = ttk.Frame(root, style="Custom.TFrame")
```

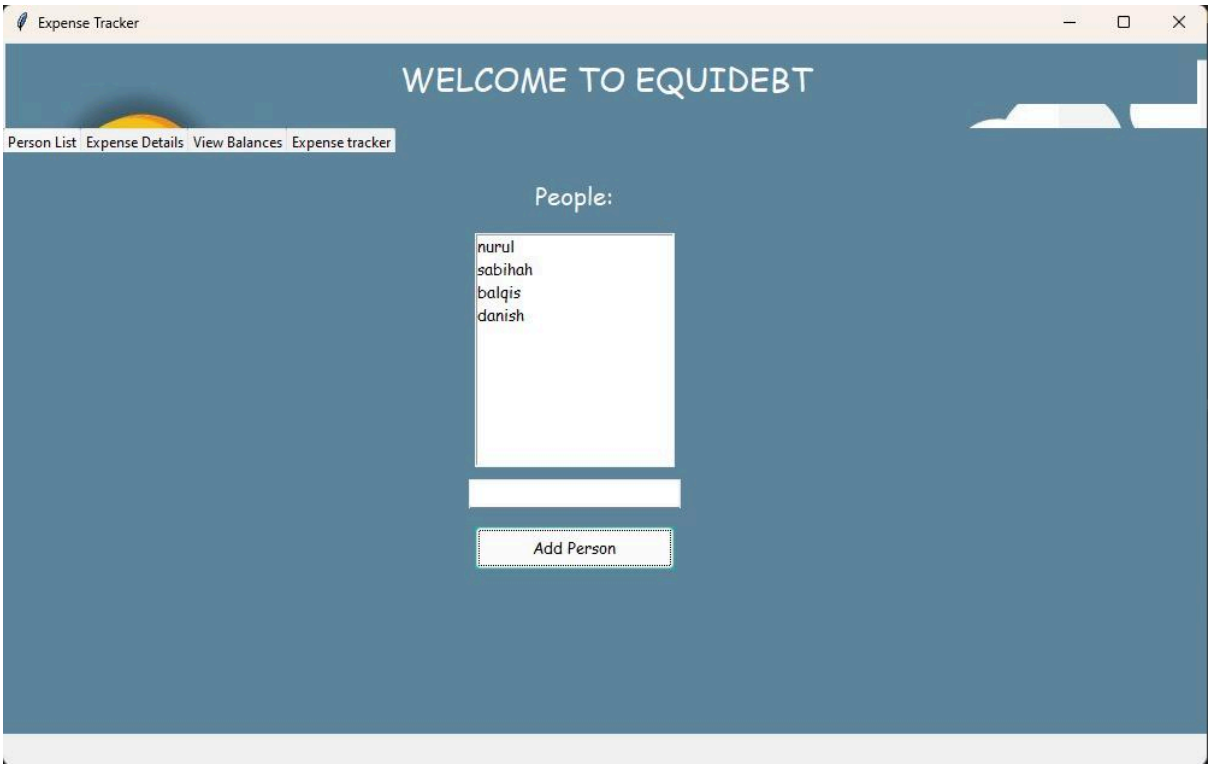
```
    expense_app_tab = ttk.Frame(root, style="Custom.TFrame")

    # Create the PersonList instance here
    def on_successful_login(user):
        login_register_window.login_register_frame.destroy()
        person_list = PersonList(person_list_tab)  # Use the same instance
created earlier
        expense_app = ExpenseApp(root, person_list, user)

    login_register_window = LoginRegisterWindow(root, authentication,
on_successful_login)

    person_list = PersonList(person_list_tab)

    root.mainloop()
```

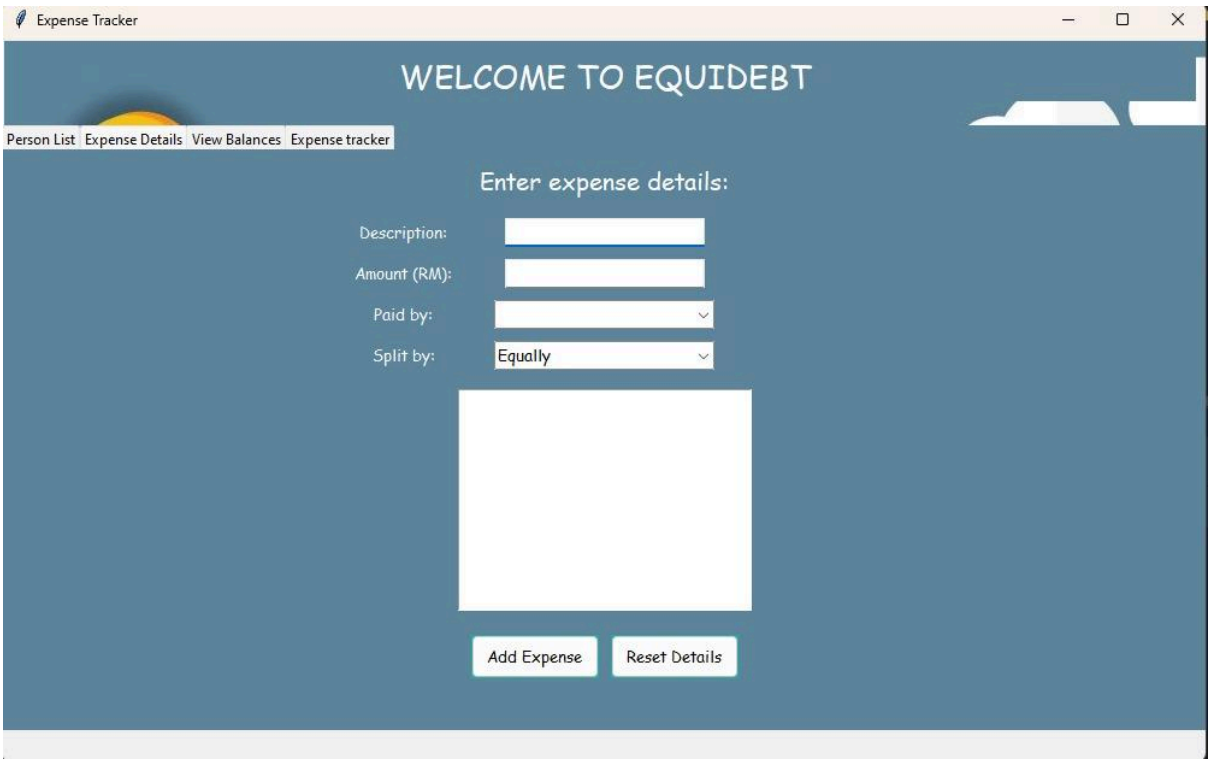
5.0 COMPLETE GUI SCREENSHOT


Expense Tracker
WELCOME TO EQUIDEBT
Username:
sabihah
Password:
****
Register
Login


Expense Tracker
WELCOME TO EQUIDEBT
Username:
sabihah
Success
Registration successful. You can now log in.
OK

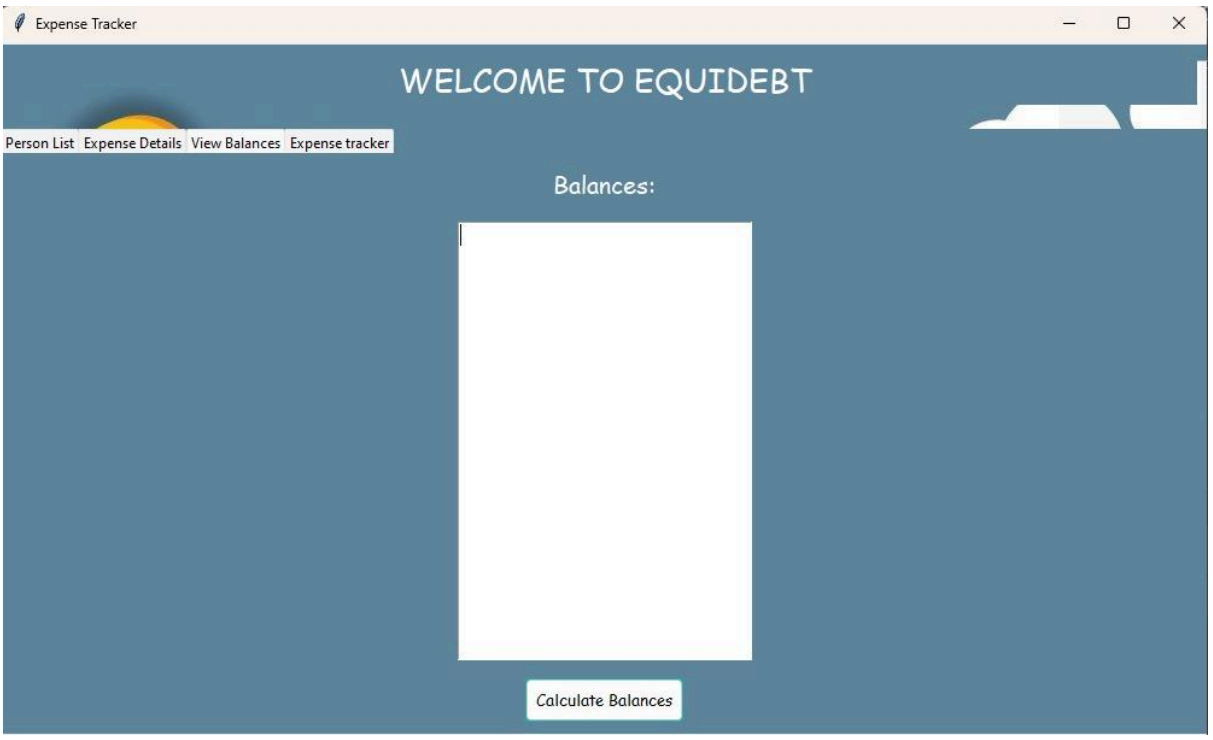

Expense Tracker
WELCOME TO EQUIDEBT
Person List
Expense Details
View Balances
Expense tracker
People:
Add Person

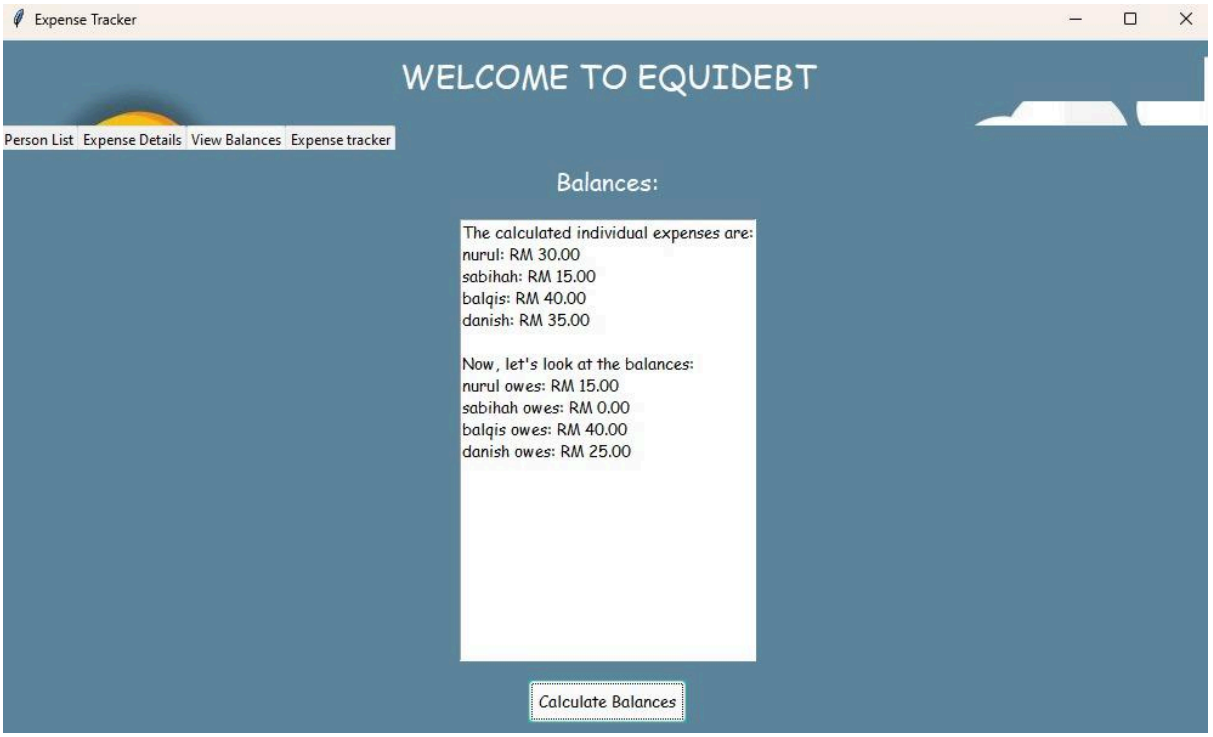

Expense Tracker
WELCOME TO EQUIDEBT
Person List
Expense Details
View Balances
Expense tracker
People:
nurul
sabihah
balqis
danish
Add Person

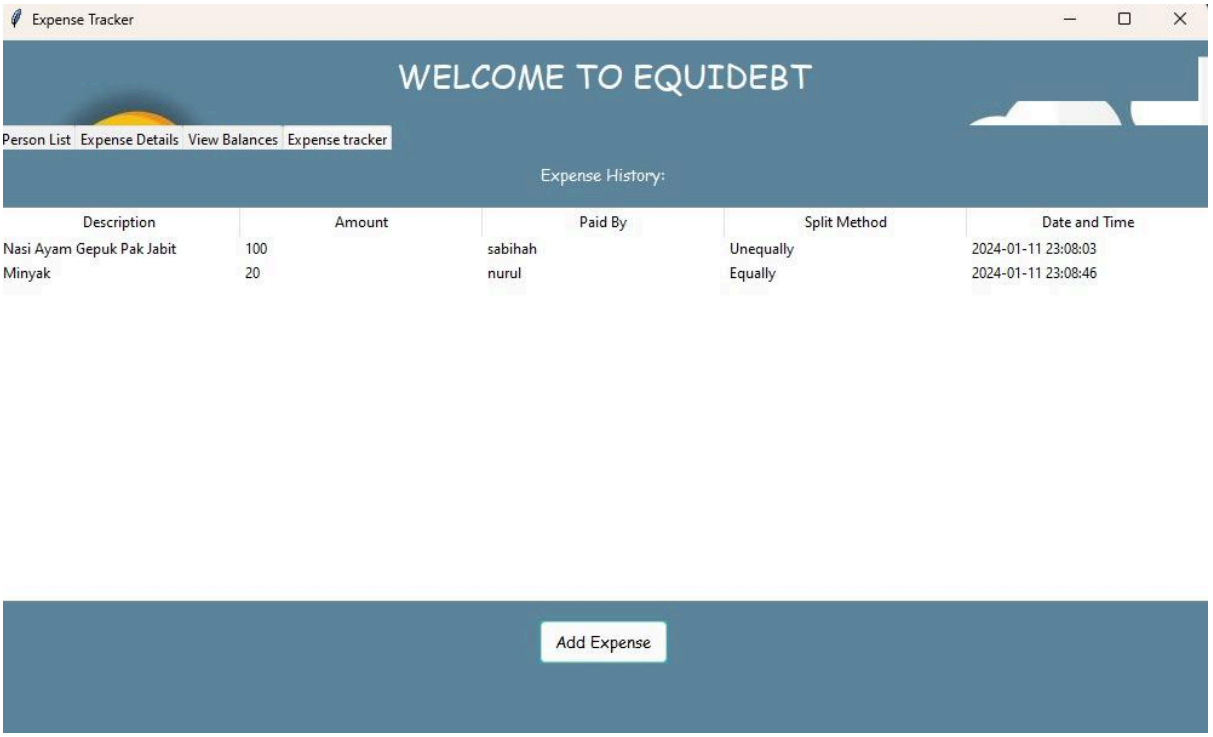

Expense Tracker
WELCOME TO EQUIDEBT
Person List
Expense Details
View Balances
Expense tracker
Enter expense details:
Description:
Amount (RM):
Paid by:
Split by:
Equally
Add Expense
Reset Details


Expense Tracker
WELCOME TO EQUIDEBT
Person List
Expense Details
View Balances
Expense tracker
Enter expense details:
Description:
asi Ayam Gepuk Pak Jabit
Amount (RM):
100
Paid by:
sabihah
Split by:
Unequally
Split by:
nurul 25
sabihah 10
balqis 35
danish 30
Nasi Ayam Gepuk Pak Jabit: RM 100, Unequally Split Among: nurul (25.0), sabihah (10.0), balqis (35.0), danish (30.0)
Add Expense
Reset Details


Expense Tracker
WELCOME TO EQUIDEBT
Person List
Expense Details
View Balances
Expense tracker
Enter expense details:
Description:
Minyak
Amount (RM):
20
Paid by:
nurul
Split by:
Equally
Split by:
nurul
sabihah
balqis
danish
Nasi Ayam Gepuk Pak Jabit: RM 100, U
nequally Split Among: nurul (25.0), sabi
hah (10.0), balqis (35.0), danish (30.0)
Minyak: RM 20, Equally Split Among: nu
rul, sabihah, balqis, danish
Add Expense
Reset Details


Expense Tracker
WELCOME TO EQUIDEBT
Person List
Expense Details
View Balances
Expense tracker
Balances:
Calculate Balances


Expense Tracker
WELCOME TO EQUIDEBT
Person List
Expense Details
View Balances
Expense tracker
Balances:
The calculated individual expenses are:
nurul: RM 30.00
sabihah: RM 15.00
balqis: RM 40.00
danish: RM 35.00
Now, let's look at the balances:
nurul owes: RM 15.00
sabihah owes: RM 0.00
balqis owes: RM 40.00
danish owes: RM 25.00
Calculate Balances


Expense Tracker
WELCOME TO EQUIDEBT
Person List
Expense Details
View Balances
Expense tracker
Expense History:
Description
Amount
Paid By
Split Method
Date and Time
Nasi Ayam Gepuk Pak Jabit
100
sabihah
Unequally
2024-01-11 23:08:03
Minyak
20
nurul
Equally
2024-01-11 23:08:46
Add Expense

6.0 REFERENCES

Chaves, Z. (2014). *News articles – The Splitwise Blog*. The Splitwise Blog. Retrieved January 11, 2024, from https://blog.splitwise.com/category/news-articles/

Knights, I. W. (2022, June 19). *The Splitwise app is excellent for divvying up the bill, but it can't fix human nature | Imogen West-Knights*. The Guardian. Retrieved January 11, 2024, from https://www.theguardian.com/commentisfree/2022/jun/19/splitwise-app-bill-split-fairly-transfer

Reiff, N., & Khartit, K. (2023). *How Splitwise Makes Money*. Investopedia. Retrieved January 11, 2024, from https://www.investopedia.com/articles/company-insights/090816/how-splitwise-works-and-makes-money.asp