COLLEGE OF COMPUTING, INFORMATICS, AND MATHEMATICS

UNIVERSITI TEKNOLOGI MARA (UiTM) CAWANGAN KEDAH

DIPLOMA IN LIBRARY INFORMATICS (CDIM144)

IML208 PROGRAMMING FOR LIBRARIES

**GROUP ASSIGNMENT:**

# ‘BABELPAWP’ ENTERTAINMENT PURCHASING SYSTEM

**PREPARED BY:**


| NAME | MATRIC NUMBER |
|---|---|
| ALIA SOFEA BINTI AHMAD FADZLI | 2022834288 |
| AMEERA NAFEESA BINTI AZHARI | 2022629292 |
| HANI RASYIQAH IRDINA BINTI HASSANAL ZAIRI | 2022863864 |
| NURUL SYAKIRAH BINTI ASRIL | 2022618038 |

**CLASS:** KCDIM1443F


**PREPARED FOR:**

SIR AIRUL SHAZWAN BIN NORSHAHIMI

**SUBMISSION DATE:**

17TH JANUARY 2024

# ‘BABELPAWP’ ENTERTAINMENT PURCHASING SYSTEM


| | |
|---|---|
| ALIA SOFEA BINTI AHMAD FADZLI | (2022834288) |
| AMEERA NAFEESA BINTI AZHARI | (2022629292) |
| HANI RASYIQAH IRDINA BINTI HASSANAL ZAIRI | (2022863864) |
| NURUL SYAKIRAH BINTI ASRIL | (2022618038) |


17TH JANUARY 2024


COLLEGE OF COMPUTING, INFORMATICS, AND MATHEMATICS

UNIVERSITI TEKNOLOGI MARA (UiTM) CAWANGAN KEDAH

DIPLOMA IN LIBRARY INFORMATICS (CDIM144)

**ACKNOWLEDGEMENT**

First and foremost, we would like to hand out a heartfelt appreciation to our parents for their unwavering support, understanding, and encouragement throughout this academic endeavor. Their belief in our abilities has been a driving force behind our success.

We are also grateful to our lecturer, Sir Airul Shazwan for his guidance, valuable feedback, and imparting knowledge that has significantly contributed to the development of this assignment. His expertises and enthusiasm for the subject matter have been inspiring and helping.

We also extend our appreciation to our group mates for each other's toleration and collaboration. Our combined efforts have improved this assignment's quality. Your collective contributions have played a crucial role in the completion of this assignment. Thank you all for being an integral part of this academic journey.

**TABLE OF CONTENTS**

1.0 INTRODUCTION.....1
2.0 PROBLEM STATEMENT.....2
2.1 Restricted platform for ticket purchases.....2
2.2 Scalping and reselling issues.....2
2.3 Performance issues.....2
3.0 OBJECTIVES.....3
4.0 FLOWCHART.....4
5.0 SNAPSHOT.....9
5.1 CODING.....9
5.2 GUI.....25
5.3 DATABASE.....26
5.3.1 Database structure.....27
5.3.2 Table Structure.....27
6.0 CONCLUSION.....30

## 1.0 INTRODUCTION

For our final project, we have decided to work on a purchasing system which is also called ‘Babelpawp Entertainment Purchasing System’. At Babelbawp, we are powered by cutting-edge technologies, such as XAMPP, Python, Visual Studio Code and GUI using Tkinter.

Python is an interpreted, dynamically semantic high-level object-oriented programming language. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming while XAMPP is a cross-platform web server which helps developers to create and test their programs. Next, Visual Studio code is a free, open-source code editor developed by Microsoft, made for developers for their various programming languages. Lastly, to create a Graphical User Interface (GUI) with Tkinter, utilize the Tkinter library, which is Python's default GUI toolkit. Tkinter also provides a set of tools and widgets for building desktop applications with graphical interfaces.

For Babelpawp, we have created three submodules which are the membership registration submodule, ticketing submodule and fan meeting submodule. Each submodule consists of its own attributes in which the user will input their information into the GUI. We also included Create Read Update and Delete (CRUD) operations in our system and we also have calculations for each submodule. Furthermore, we named our database as ‘kpop_site’ and our python file as ‘babelpawp.py’.

Basically, Babelpawp is targeted towards Malaysian Korean-pop (K-Pop) fans, offering them a secure and enjoyable platform to connect with their favorite artists for their convenience.

## 2.0 PROBLEM STATEMENT

A number of issues impede the effectiveness of the current ticketing purchase system, leading to a less than ideal user experience and operational setbacks. Users have reported running into issues that range from making mistakes during the transaction process to having trouble navigating the platform. These problems lead to higher support requests, a drop in customer satisfaction, and possible revenue loss. These are some of the key issues of an online ticketing purchase system:

### 2.1 Restricted platform for ticket purchases

The majority of K-pop fans are facing considerable challenges in securing tickets for their favorite events due to the limited availability of accurate and trustworthy ticketing platforms. The dedicated and passionate fanbase often encounters difficulties in navigating through the existing platforms, resulting in a frustrating and uncertain ticket purchasing experience.

### 2.2 Scalping and reselling issues

For fans hoping to get into the highly sought-after K-pop events, the secondary ticket market's prevalence of scalping and reselling presents serious obstacles. The situation is made worse by the lack of tickets available on official platforms, which forces fans to turn to unofficial sources. There, they face a variety of problems, such as exorbitant costs and a higher chance of becoming victims of fraud.

### 2.3 Performance issues

There is more traffic and congestion on the system during peak hours when there is a high demand for tickets. The performance problems are made worse by this increased demand, which delays the processing of user requests and transactions. Not to forget the lag in transaction processing, adds to the delay in the ticket buying process, along with payment authorization and ticket confirmation. Users' satisfaction is delayed as a result, which could result in an unsatisfactory experience all around.

## 3.0 OBJECTIVES

1. To gracefully make the ticketing purchase process quick, efficient, unbiased and easy. This involves cutting down on the number of steps customers must take in order to choose, purchase and receive their tickets.
2. To assure that the data entered during the ticketing purchase process is accurate and that the system creates tickets with the correct information, including the date, time, and location.
3. To enhance the customer's overall experience by providing users a user-friendly interface, lucid guidelines and prompt customer service. A satisfying encounter can increase client loyalty and satisfaction.

## 4.0 FLOWCHART

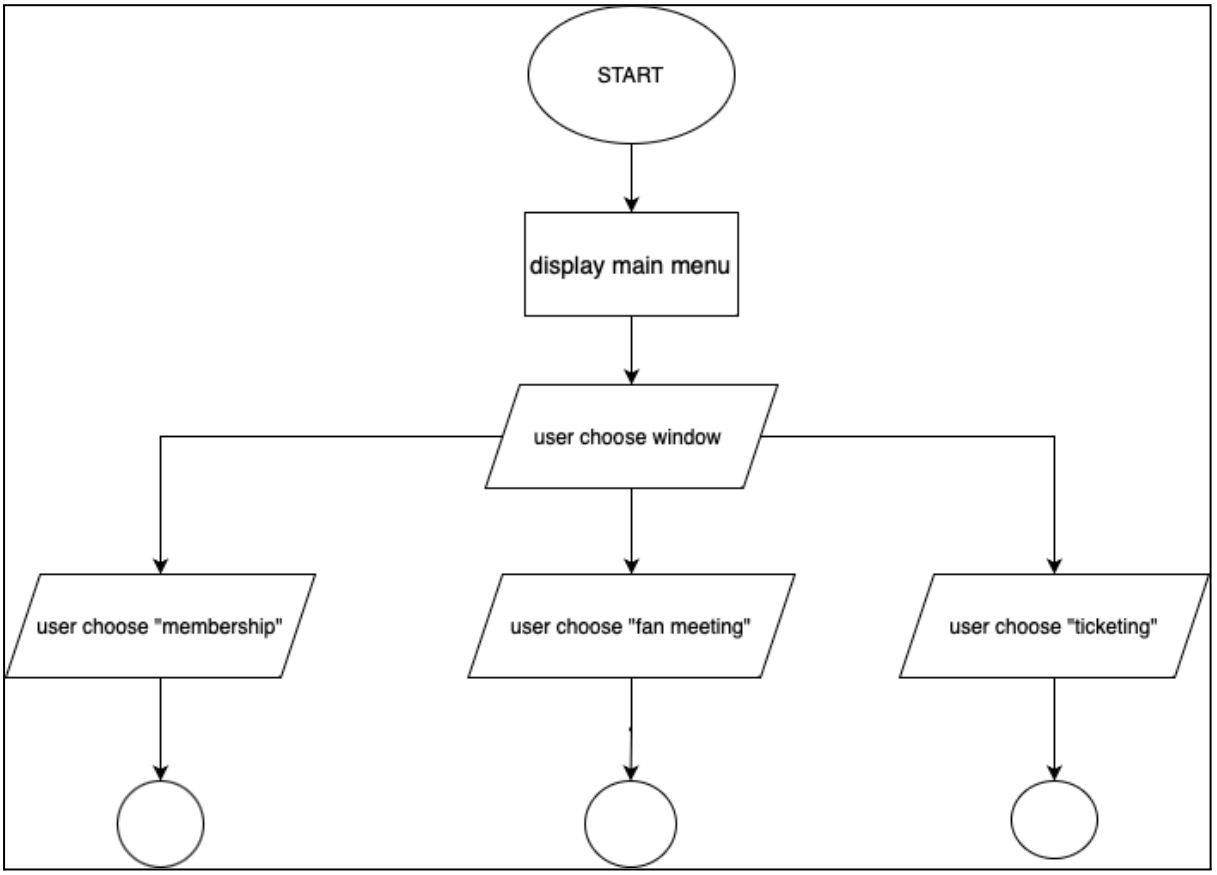


Figure 1.0 Flowchart of Babelpawp main menu

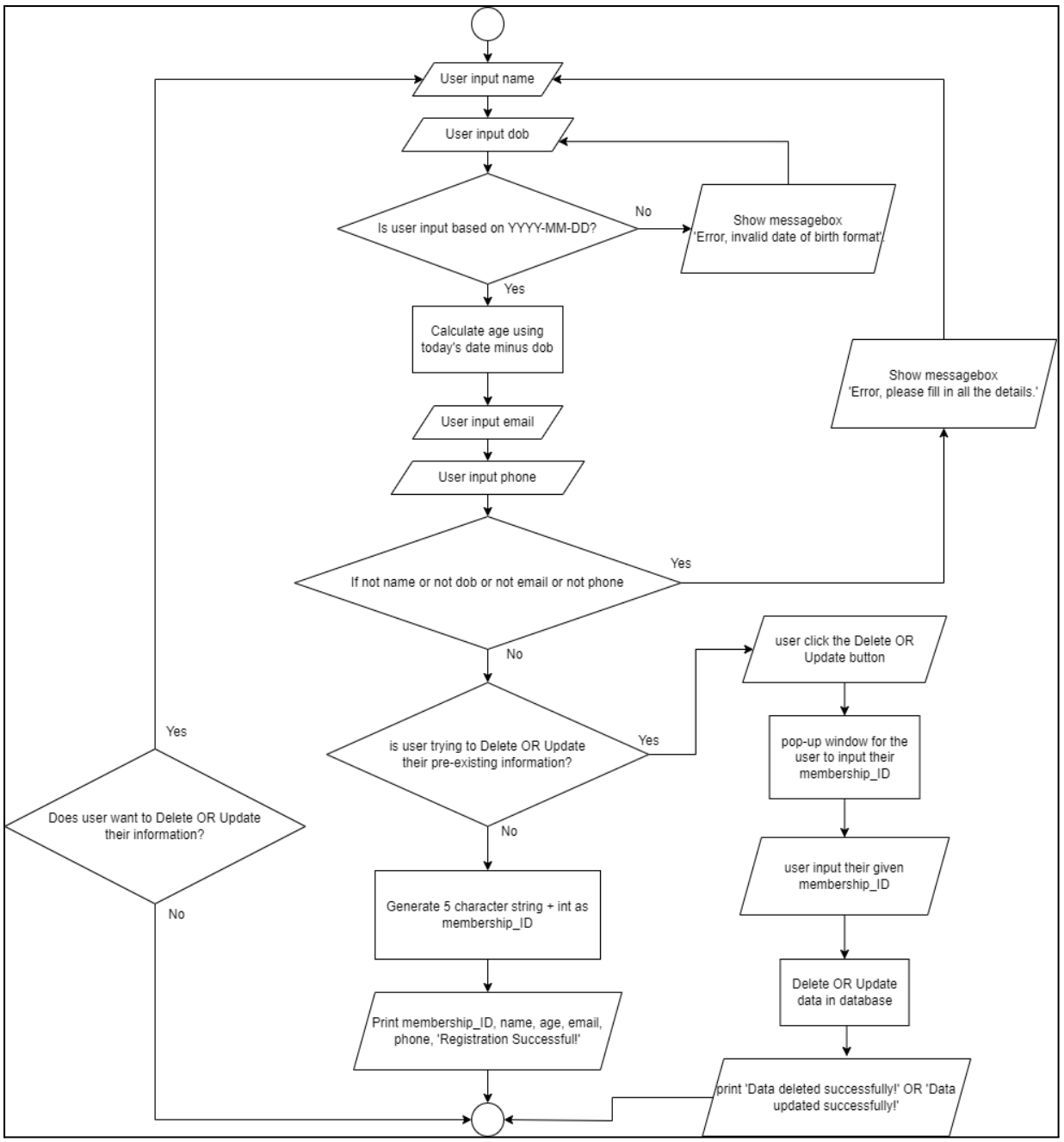


Figure 1.1 Flowchart of the membership window of the GUI

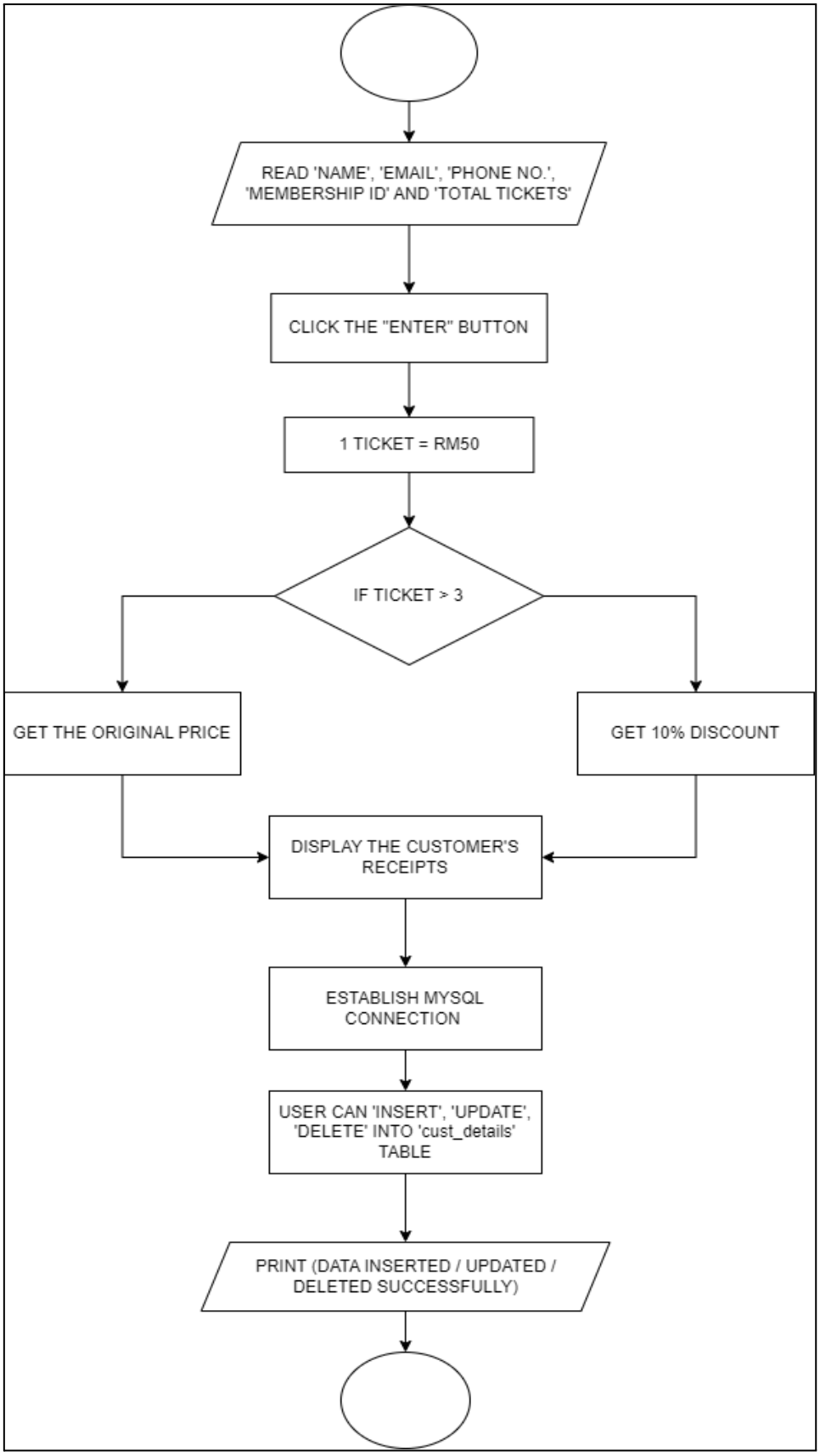


Figure 1.2 Flowchart of the ticketing window of the GUI

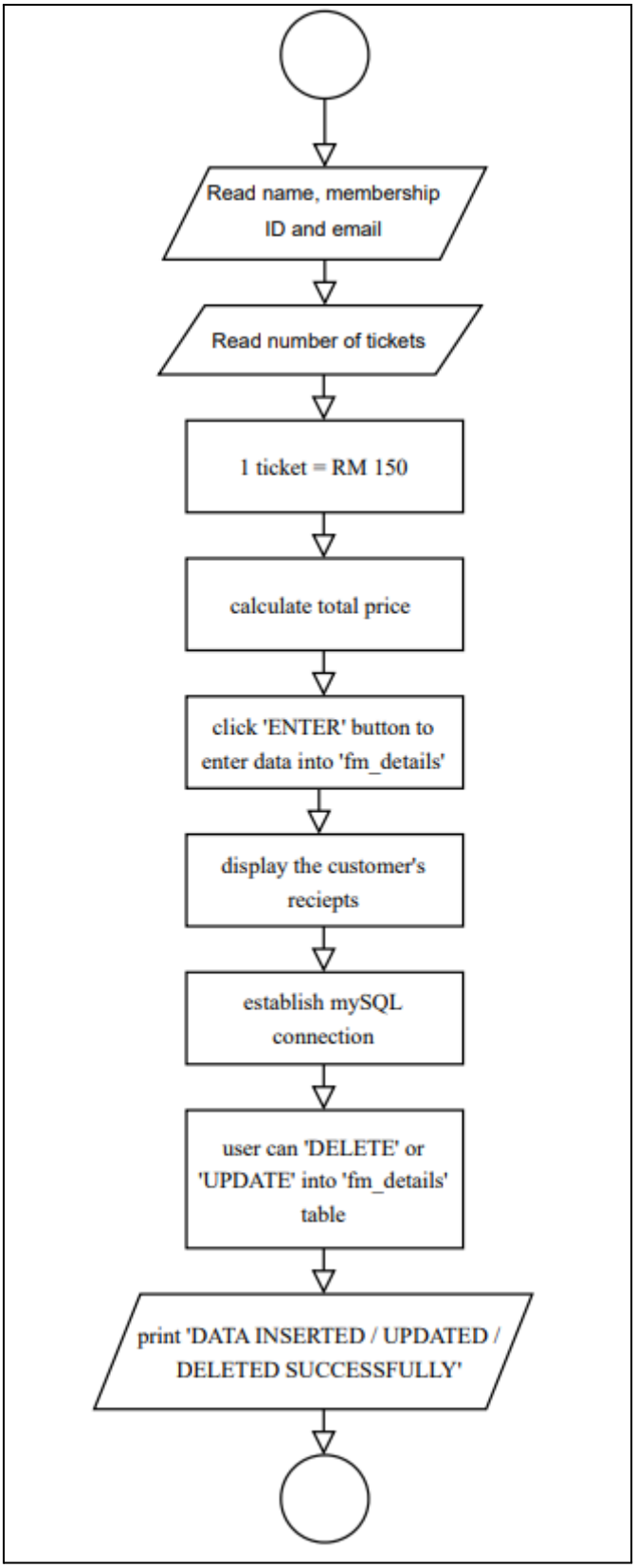


Figure 1.3 Flowchart of the fanmeeting window of the GUI

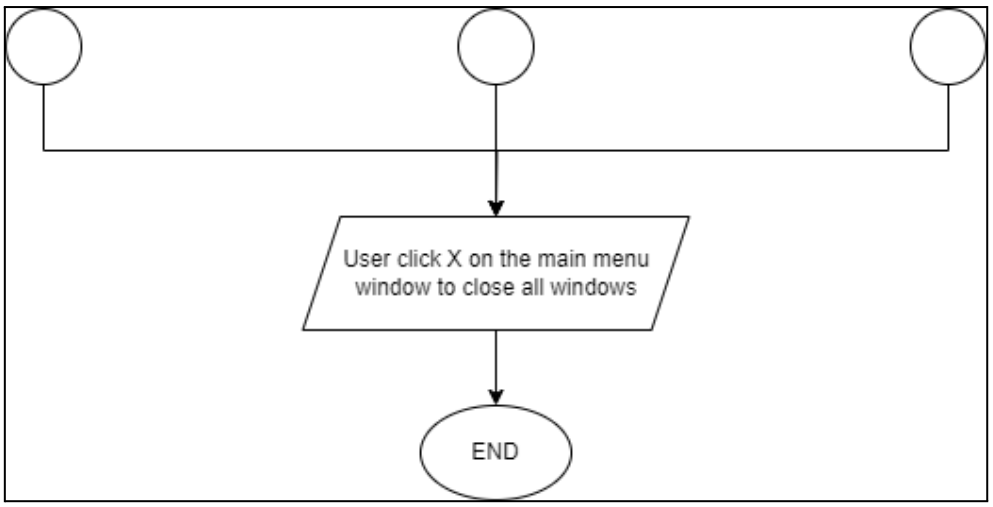


Figure 1.4 End part of Babelpawp flowchart

## 5.0 SNAPSHOT

## 5.1 CODING

babelpawp.py > KpopPurchasingSystem > __init__

```python
import tkinter as tk
from tkinter import messagebox, ttk, simpledialog
import mysql.connector
from datetime import datetime, date
import random, string, re

class KpopPurchasingSystem:
    def __init__(self, master):
        self.master = master
        self.master.title("K-pop Purchasing System")
        self.master.geometry("760x305")
        self.master.configure(bg='mistyrose')

        # GUI for main menu
        self.master_frame = tk.Frame(master,bd=5,relief='groove')
        self.master_frame.pack()

        self.master_label = tk.Label(self.master_frame, text="BABELPAWP ENTERTAINMENT", font=('Terminal', 20, 'bold'),bg='indian red2', fg= 'lemon chiffon')
        self.master_label.pack(ipadx=760, ipady=15)

        #notes area
        notes_text = tk.Text(self.master, height=7, width=800, font=('terminal', 11),bg='mistyrose',fg= 'palevioletred3')
        notes_text.pack()
        notes_text.insert(tk.END, "\nWelcome to Babelpawp Entertainment page!\n\nThere are few features presented in this page\n\n")
        notes_text.insert(tk.END, "If you need any help, you can press the button 'HELP' at the bottom of each window :)")
        notes_text.configure(state='disabled')
        notes_text.tag_configure('center', justify='center')
        notes_text.tag_add('center', 1.0, tk.END)

        # Create buttons for each GUI
        self.masterframe = tk.Frame(master,bd=5,relief='groove',bg='mistyrose')
        self.masterframe.pack(ipadx=760,pady=2)

        # Membershipbutton
        self.registration_btn = tk.Button(self.masterframe, text="Membership", font=('terminal', 15), bg='powderblue', fg='steelblue',bd=10,relief= 'groove', command=self.open_registration_gui)
        self.registration_btn.grid(ipadx=30,pady=5,padx=30,row=0,column=0)
        notestext=tk.Text(self.masterframe,width=25,height=5,font=('terminal', 9),bg='mistyrose',fg= 'palevioletred3')
        notestext.grid(row=1,column=0,pady=5)
        notestext.insert(tk.END, "\nThis button will take\nyou to the membership\nregistration window")
        notestext.configure(state='disabled')
        notestext.tag_configure('center', justify='center')
        notestext.tag_add('center', 1.0, tk.END)

        # Ticketingbutton
        self.ticketing_btn = tk.Button(self.masterframe, text="Ticketing", font=('terminal', 15), bg='pink', fg='palevioletred3', bd=10, relief='groove', command=self.open_ticketing_gui)
        self.ticketing_btn.grid(ipadx=30,pady=5,padx=30,row=0,column=1)
        notestext2=tk.Text(self.masterframe,width=25,height=5,font=('terminal', 9),bg='mistyrose',fg= 'palevioletred3')
        notestext2.grid(row=1,column=1,pady=5)
        notestext2.insert(tk.END, "\nThis button will take \nyou to the ticket \npurchasing window")
        notestext2.configure(state='disabled')
        notestext2.tag_configure('center', justify='center')
        notestext2.tag_add('center', 1.0, tk.END)

        # Fanmeeting
        self.schedule_btn = tk.Button(self.masterframe, text="Fanmeeting", font=('terminal', 15),bg='plum1', fg='orchid3',bd=10, relief='groove', command=self.open_fanmeeting_gui)
        self.schedule_btn.grid(ipadx=30,pady=5,padx=30,row=0,column=2)
        notestext3=tk.Text(self.masterframe,width=25,height=5,font=('terminal', 9),bg='mistyrose',fg= 'palevioletred3')
        notestext3.grid(row=1,column=2,pady=5)
        notestext3.insert(tk.END, "\nThis button will take \nyou to the fanmeeting\npurchasing window")
        notestext3.configure(state='disabled')
        notestext3.tag_configure('center', justify='center')
        notestext3.tag_add('center', 1.0, tk.END)
```

Figure 2.0 Line 1 to Line 52 of babelpawp.py coding in Visual Studio code

```python
53
54    def show_help_mem(self):
55        # Help text button for membership registration window
56        help_text_mem = """
57        How to register for BABELPAWP's membership?
58
59        Fill in the following information:
60        - Name: Your full name.
61        - Date of Birth (YYYY-MM-DD): Your date of birth
62          in the specified format.
63        - Email (xxx@xxx.xxx): Your email address.
64        - Phone number (+60 only): Your phone number, which
65          must start with '60', as this membership is only valid for
66          Malaysians.
67
68        After filling in the details, click the 'Register' button to
69        complete the registration.
70
71        If you encounter any issues, reach out to our support
72        team on either of these contacts below.
73
74        Phone number: +601133228274
75        e-mail: babelpawp@gmail.com
76
77        Thank you for supporting BABELPAWP!
78        """
79
80        messagebox.showinfo("Help", help_text_mem)
81
82    def open_registration_gui(self):
83        registration_window = tk.Toplevel(self.master)
84        registration_window.title("Membership Registration")
85        registration_window.geometry("400x600")
86
87        # Heading label
88        self.label_reg = tk.Label(registration_window, text='BABELPAWP MEMBERSHIP', font=('Georgia Font', 20, 'italic'), bg='light sky blue', fg='dark blue', bd=10, relief='groove')
89        self.label_reg.pack(ipadx=470)
90
91        # Name frama and entry grid
92        name_frame=tk.LabelFrame(registration_window,text="Name:", font=("Arial", 15), bg='#e6ffff', fg= 'dark blue',bd=5,relief= 'groove')
93        name_frame.pack(ipadx=470)
94        self.name_entry = tk.Entry(name_frame, font=("Arial", 12))
95        self.name_entry.grid(ipadx=101)
96
97        # Email frame and entry grid
98        email_frame=tk.LabelFrame(registration_window, text="Email (xxx@xxx.xxx):", font=("Arial", 15), bg='#e6ffff', fg= 'dark blue',bd=5,relief= 'groove')
99        email_frame.pack(ipadx=470)
100       self.email_entry = tk.Entry(email_frame, font=("Arial", 12))
101       self.email_entry.grid(ipadx=101)
102
103       # DOB frame and entry grid
104       dob_frame=tk.LabelFrame(registration_window, text="Date of Birth (YYYY-MM-DD):", font=("Arial", 15), bg='#e6ffff', fg= 'dark blue',bd=5,relief= 'groove')
105       dob_frame.pack(ipadx=470)
106       self.dob_entry = tk.Entry(dob_frame, font=("Arial", 12))
107       self.dob_entry.grid(ipadx=101)
108
109       # Phone frame and entry grid
110       phone_frame=tk.LabelFrame(registration_window, text="Phone number (+60 only):", font=("Arial", 15), bg='#e6ffff', fg= 'dark blue',bd=5,relief= 'groove')
111       phone_frame.pack(ipadx=470)
112       self.phone_entry = tk.Entry(phone_frame, font=("Arial", 12))
113       self.phone_entry.grid(ipadx=101)
114
115       # Register, Update and Delete frame
116       RUD_frame=tk.LabelFrame(registration_window, bg='#e6ffff', bd=5, relief='groove')
117       RUD_frame.pack(ipadx=470)
118
```

Figure 2.1 Line 53 to Line 128 of babelpawp.py coding in Visual Studio code

```
129         # Register, Update and Deletebutton and grid
130         registermem_button = tk.Button(RUD_frame, text="Register", command=self.register_member, font=("Times New Roman", 14), bg="#84e0b3", fg="black")
131         registermem_button.grid(row=5, column=1, pady=10, padx=30, sticky="ew")
132         updmem_button = tk.Button(RUD_frame, text="Update", command=self.update_member_gui, font=("Times New Roman", 14), bg="#5bc0de", fg="black")
133         updmem_button.grid(row=5, column=2, pady=10, padx=30, sticky="ew")
134         deletemem_button = tk.Button(RUD_frame, text="Delete", command=self.delete_member_gui, font=("Times New Roman", 14), bg="#d9534f", fg="white")
135         deletemem_button.grid(row=5, column=3, pady=10, padx=30, sticky="ew")
136
137         # Data display frame
138         memdisplay_frame = tk.LabelFrame(registration_window, bg='#e6ffff', bd=5, relief='groove')
139         memdisplay_frame.pack(ipadx=470)
140
141         self.data_display_box = tk.Text(memdisplay_frame, height=10, width=50)
142         self.data_display_box.pack(pady=10)
143
144         # Disable the data display box
145         self.data_display_box.config(state='disabled')
146
147         # Help, register, and quit button frame
148         button_frame=tk.LabelFrame(registration_window, bg='#e6ffff', bd=5, relief='groove')
149         button_frame.pack(ipadx=470)
150
151         # Help and quit button with grid
152         help_button = tk.Button(button_frame, text="Help", command=self.show_help_mem, font=("Times New Roman", 14), bg="#5bc0de", fg="black")
153         help_button.grid(row=5, column=0, pady=10, padx=100, sticky="e")
154         quit_button = tk.Button(button_frame, text="Back", command=registration_window.destroy, font=("Times New Roman", 14), bg="#d9534f", fg="white")
155         quit_button.grid(row=5, column=1, pady=10, padx=1, sticky="w")
156
157     def register_member(self):
158         # Get user information
159         name_mem = self.name_entry.get()
160         email_mem = self.email_entry.get()
161         dob_mem = self.dob_entry.get()
162         phone_mem = self.phone_entry.get()
163
164         # Validate input
165         if not name_mem or not dob_mem or not email_mem or not phone_mem:
166             messagebox.showerror("Error", "Please fill in all fields.")
167             return
168
169         # Check if name contains only alphabetic characters
170         if not name_mem.replace(" ", "").isalpha():
171             messagebox.showerror("Invalid", "Name must contain only alphabetic characters.")
172             return
173
174         elif not phone_mem.startswith("60") or not phone_mem[3:].isdigit():
175             messagebox.showerror("Invalid", "Phone number must start with '+60' and contain only numeric values.")
176             return
177
178         # Check email format using a regular expression
179         email_pattern = re.compile(r'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z0-9]+$')
180         if not email_pattern.match(email_mem):
181             messagebox.showerror("Invalid", "Invalid email format. Please use the format xxx@xxx.xxx.")
182             return
183
184         # Convert dob to date and calculate age
185         try:
186             dob_date = datetime.strptime(dob_mem, "%Y-%m-%d").date()
187             age = (date.today() - dob_date).days // 365
188         except ValueError:
189             messagebox.showerror("Error", "Invalid Date of Birth format.")
190             return
191
192         # Generate a random alphanumeric Member ID
193         member_ID = ''.join(random.choices(string.ascii_uppercase + string.digits, k=5))
194
195         # Display registration information in a message box
196         registration_success_message = f"Registration Successful!\n\nMember ID: {member_ID}\nName: {name_mem}\nAge: {age}\nEmail: {email_mem}\nPhone: {phone_mem}"
197         messagebox.showinfo("Registration Successful", registration_success_message)
```

Figure 2.2 Line 129 to Line 197 of babelpawp.py coding in Visual Studio code

```
198
199         # Update the data display box in the GUI
200         self.data_display_box.config(state='normal')
201         self.data_display_box.insert(tk.END, registration_success_message + '\n\n')
202         self.data_display_box.config(state='disabled')
203
204         # Insert the data into the database
205         self.insert_into_database_mem(member_ID, name_mem, dob_mem, age, email_mem, phone_mem)
206
207     def insert_into_database_mem(self, member_ID, name_mem, dob_mem, age, email_mem, phone_mem):
208         try:
209             # Establish a connection to the MySQL server
210             mydb = mysql.connector.connect(
211                 host="localhost",
212                 user="root",
213                 password="",
214                 database="kpop_site"
215             )
216
217             # Create a cursor object to interact with the database
218             cursor = mydb.cursor()
219
220             # Inserting data into a table
221             sql = "INSERT INTO `memship_details` (member_ID, user_name, user_dob, user_age, user_email, user_phone) VALUES (%s, %s, %s, %s, %s, %s)"
222             val = (member_ID, name_mem, dob_mem, age, email_mem, phone_mem)
223
224             cursor.execute(sql, val)
225             mydb.commit()
226             print("Data inserted successfully!")
227
228         except mysql.connector.Error as err:
229             print(f"Error: {err}")
230             mydb.rollback()
231
232         finally:
233             cursor.close()
234             mydb.close()
235
236     def update_member(self, member_ID):
237         new_name = self.name_entry.get()
238         new_email = self.email_entry.get()
239         new_dob = self.dob_entry.get()
240         new_phone = self.phone_entry.get()
241
242         # Validate input
243         if not member_ID or not new_name or not new_dob or not new_email or not new_phone:
244             messagebox.showerror("Error", "Please fill in all fields.")
245             return
246
247         # Check if name contains only alphabetic characters
248         if not new_name.replace(" ", "").isalpha():
249             messagebox.showerror("Invalid", "Name must contain only alphabetic characters.")
250             return
251
252         elif not new_phone.startswith("60") or not new_phone[3:].isdigit():
253             messagebox.showerror("Invalid", "Phone number must start with '+60' and contain only numeric values.")
254             return
255
256         # Check email format using a regular expression
257         email_pattern = re.compile(r'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z0-9]+$')
258         if not email_pattern.match(new_email):
259             messagebox.showerror("Invalid", "Invalid email format. Please use the format xxx@xxx.xxx.")
260             return
261
262         # Convert dob to date and calculate age
263         try:
264             new_dob_date = datetime.strptime(new_dob, "%Y-%m-%d").date()
265             new_age = (date.today() - new_dob_date).days // 365
266         except ValueError:
267             messagebox.showerror("Error", "Invalid Date of Birth format.")
```

Figure 2.3 Line 198 to Line 267 of babelpawp.py coding in Visual Studio code

```
268                 return
269
270         # Establish a connection to the MySQL server
271         mydb = mysql.connector.connect(
272             host="localhost",
273             user="root",
274             password="",
275             database="kpop_site"
276         )
277
278         # Create a cursor object to interact with the database
279         cursor = mydb.cursor()
280
281         # Fetch the updated data from the database
282         cursor.execute("SELECT * FROM `memship_details` WHERE member_id=%s", (member_ID,))
283         updated_member_info = cursor.fetchone()
284
285         # Updating data in the table
286         sql = "UPDATE `memship_details` SET user_name=%s, user_dob=%s, user_age=%s, user_email=%s, user_phone=%s WHERE member_id=%s"
287         val = (new_name, new_dob, new_age, new_email, new_phone, member_ID)
288
289         try:
290             cursor.execute(sql, val)
291             mydb.commit()
292             print("Data updated successfully!")
293
294             # Fetch the updated data from the database
295             cursor.execute("SELECT * FROM `memship_details` WHERE member_id=%s", (member_ID,))
296             updated_member_info = cursor.fetchone()
297
298         except mysql.connector.Error as err:
299             print(f"Error: {err}")
300             mydb.rollback()
301
302         finally:
303             # Close cursor and database connection
304             cursor.close()
305             mydb.close()
306
307         messagebox.showinfo("Update Successful", "Member information updated successfully.")
308
309         # Return the updated member information
310         return updated_member_info
311
312     def update_member_gui(self):
313         # Prompt the user for Member ID
314         member_id_to_update = simpledialog.askstring("Update Member", "Enter Member ID to update:")
315
316         if member_id_to_update:
317             # Call the update_member method
318             updated_member_info = self.update_member(member_id_to_update)
319
320             if updated_member_info:
321                 # Clear existing content in data display box
322                 self.data_display_box.config(state='normal')
323                 self.data_display_box.delete(1.0, tk.END)
324
325                 # Display updated member information in the data display box
326                 updated_info_mem_str = (
327                     f"Updated Member Information:\n\n"
328                     f"Member ID: {updated_member_info[0]}\n"
329                     f"Name: {updated_member_info[1]}\n"
330                     f"Age: {updated_member_info[3]}\n"
331                     f"Email: {updated_member_info[4]}\n"
332                     f"Phone: {updated_member_info[5]}"
333                 )
334                 self.data_display_box.insert(tk.END, updated_info_mem_str)
335                 self.data_display_box.config(state='disabled')
336
```

Figure 2.4 Line 268 to Line 336 of babelpawp.py coding in Visual Studio code

```
337     def delete_member(self, member_ID):
338         # Validate input
339         if not member_ID:
340             messagebox.showerror("Error", "Please fill in the Member ID.")
341             return
342 
343         # Establish a connection to the MySQL server
344         mydb = mysql.connector.connect(
345             host="localhost",
346             user="root",
347             password="",
348             database="kpop_site"
349         )
350 
351         # Create a cursor object to interact with the database
352         cursor = mydb.cursor()
353 
354         # Deleting data from the table
355         sql = "DELETE FROM `memship_details` WHERE member_ID=%s"
356         val = (member_ID,)
357 
358         try:
359             cursor.execute(sql, val)
360             mydb.commit()
361             print("Data deleted successfully!")
362 
363         except mysql.connector.Error as err:
364             print(f"Error: {err}")
365             mydb.rollback()
366 
367         cursor.close()
368         mydb.close()
369 
370         messagebox.showinfo("Delete Successful", "Member information deleted successfully.")
371 
372     def delete_member_gui(self):
373         # Prompt the user for Member ID
374         member_id_to_delete = simpledialog.askstring("Delete Member", "Enter Member ID to delete:")
375 
376         if member_id_to_delete:
377             # Call the delete_member method
378             self.delete_member(member_id_to_delete)
379 
380             # Clear existing content in data display box
381             self.data_display_box.config(state='normal')
382             self.data_display_box.delete(1.0, tk.END)
383             self.data_display_box.config(state='disabled')
384 
385     def quit_application(self):
386         self.master.quit()
```

Figure 2.5 Line 337 to Line 386 of babelpawp.py coding in Visual Studio code

```python
387
388     def show_help_ticket(self):
389         # Help text button for ticketing window
390         help_text_tic = """
391         How to register for BABELPAWP's ticketing?
392
393         Fill in the following information:
394         - Name: Your full name.
395         - Email (xxx@xxx.xxx): Your email address.
396         - Phone number (+60 only): Your phone number, which
397           must start with '60', as this membership is only valid for
398           Malaysians.
399         - Membership ID : Only insert the membership ID that
400           has been registered in the Membership Registration
401           section.
402         - RM 50 per ticket
403         - DISCOUNT 10% IS APPLIED WHEN CUSTOMER
404           PURCHASED MORE THAN 3 TICKETS
405
406         After filling in the details, click the 'Enter' button to
407         complete the registration.
408
409         If you encounter any issues, reach out to our support
410         team on either of these contacts below.
411
412         Phone number: +601133228274
413         e-mail: babelpawp@gmail.com
414
415         Thank you for supporting BABELPAWP!
416         """
417
418         messagebox.showinfo("Help", help_text_tic)
419
420     def open_ticketing_gui(self):
421         ticket_window = tk.Toplevel(self.master)
422         ticket_window.title("Ticket purchase")
423         ticket_window.geometry("680x420")
424
425         # Create a title label
426         title_label = tk.Label(ticket_window, text='TICKETING SYSTEM', font=("constantia", 16, "bold"), fg='black', bg='lightcoral')
427         title_label.grid(row=0, column=0,columnspan=5, ipadx=240)
428
429         self.ticket_price = 50.0
430         self.total_cost = 0.0
431         self.discount_percentage = 0.0
432
433         # Create labels and entry widgets for personal details
434         labels = ["NAME", "EMAIL", "PHONE NO", "MEMBERSHIP ID", "TOTAL TICKETS"]
435         for i, label_text in enumerate(labels):
436             tk.Label(ticket_window, text=f"{label_text}:", font=("Times", 10, "bold"), fg='gray9', bg='mistyrose').grid(row=i+2, column=0, padx=5, pady=5,sticky="e")
437
438         self.entry_name = tk.Entry(ticket_window)
439         self.entry_email = tk.Entry(ticket_window)
440         self.entry_phone = tk.Entry(ticket_window)
441         self.entry_membershipid = tk.Entry(ticket_window)
442         self.entry_num_tickets = tk.Entry(ticket_window)
443
444         entries = [self.entry_name, self.entry_email, self.entry_phone, self.entry_membershipid, self.entry_num_tickets]
445         for i, entry_widget in enumerate(entries):
446             entry_widget.grid(row=i+2, column=1, padx=10, pady=5, sticky="ew")
447
448         # Create button to calculate total cost
449         calculate_button = tk.Button(ticket_window, text="ENTER", font=("Times", 10, "bold"), fg='black', bg='lightpink1', padx=40, pady=20, command=self.calculate_total_ticket)
450         calculate_button.grid(row=8, column=0, rowspan=2, pady=5,sticky='e')
451
452         # Create button to update data in the table
453         update_button = tk.Button(ticket_window, text="UPDATE", font=("Times", 10, "bold"), fg='black', bg='lightpink2', padx=40, pady=20, command=self.update_ticket)
454         update_button.grid(row=8, column=1,rowspan=2, pady=5,padx=10,sticky="sw")
```

Figure 2.6 Line 387 to Line 454 of babelpawp.py coding in Visual Studio code

```
455
456         # Create button to delete data from the table
457         delete_button = tk.Button(ticket_window, text="DELETE", font=("Times", 10, "bold"), fg='black', bg='lightpink3', padx=40, pady=20, command=self.delete_ticket)
458         delete_button.grid(row=10, column=0,columnspan=2,rowspan=2, pady=5)
459
460         # Help and quit button with grid
461         help_button = tk.Button(ticket_window, text="Help", command=self.show_help_ticket, font=("Times New Roman", 10), bg="lightpink4", fg="white",padx=11,pady=2)
462         help_button.grid(row=10,column=2,pady=2)
463         quit_button = tk.Button(ticket_window, text="Back", command=ticket_window.destroy, font=("Times New Roman", 10), bg="lightpink4", fg="white",padx=10,pady=2)
464         quit_button.grid(row=11, column=2, pady=2)
465
466         # Create billing area
467         bill_frame = tk.Frame(ticket_window, bd=8, relief='groove')
468         bill_frame.grid(row=1, column=2, ipadx=8, ipady=14, padx=15, pady=10, rowspan=9)
469         bill_area_label = tk.Label(bill_frame, text='RECEIPTS', font=('Times New Roman', 13, 'bold'), fg='brown')
470         bill_area_label.grid(row=0, column=0, pady=(0, 5))
471         self.text_area = tk.Text(bill_frame, width=24, height=13)
472         self.text_area.grid(row=1, column=0, sticky='ew')
473
474     def calculate_total_ticket(self):
475         mydb = mysql.connector.connect(
476         host="localhost",
477         user="root",
478         password="",
479         database="kpop_site"
480         )
481
482         # Create a cursor object to interact with the database
483         cursor = mydb.cursor()
484
485         # User input
486         name_ticket = self.entry_name.get()
487         email_ticket = self.entry_email.get()
488         phone_ticket = self.entry_phone.get()
489         memid_ticket = self.entry_membershipid.get()
490         numtic_ticket = self.entry_num_tickets.get()
491
492         # Validate input
493         if not name_ticket or not email_ticket or not phone_ticket or not memid_ticket or not numtic_ticket:
494             messagebox.showerror("Error", "Please fill in all fields.")
495             return
496
497         # Check if name contains only alphabetic characters
498         if not name_ticket.replace(" ", "").isalpha():
499             messagebox.showerror("Invalid", "Name must contain only alphabetic characters.")
500             return
501
502         elif not phone_ticket.startswith("60") or not phone_ticket[3:].isdigit():
503             messagebox.showerror("Invalid", "Phone number must start with '+60' and contain only numeric values.")
504             return
505
506         # Check email format using a regular expression
507         email_pattern = re.compile(r'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z0-9]+$')
508         if not email_pattern.match(email_ticket):
509             messagebox.showerror("Invalid", "Invalid email format. Please use the format xxx@xxx.xxx.")
510             return
511
512         try:
513             num_tickets_str = self.entry_num_tickets.get()
514
515             if not num_tickets_str.isdigit():
516                 messagebox.showerror("Error", "Please enter a valid number of tickets.")
517                 return
518
519             num_tickets = int(num_tickets_str)
520
```

Figure 2.7 Line 455 to Line 520 of babelpawp.py coding in Visual Studio code

```
521             if num_tickets > 0 and num_tickets <= 10:
522                 self.total_cost = self.ticket_price * num_tickets
523                 self.discount_percentage = self.calculate_discount_ticket(num_tickets)
524                 discounted_amount = (self.discount_percentage / 100) * self.total_cost
525                 discounted_cost = self.total_cost - discounted_amount
526
527                 details_text = f"Name: {self.entry_name.get()}\nEmail: {self.entry_email.get()}\n"
528                 details_text += f"Phone: {self.entry_phone.get()}\nMembership ID: {self.entry_membershipid.get()}\n"
529                 details_text += f"Total Tickets: {num_tickets}\n"
530                 details_text += f"Cost: RM{self.total_cost:.2f}\nDiscount: {self.discount_percentage}%\n"
531                 details_text += f"TOTAL: RM{discounted_cost:.2f}"
532                 bill_text = details_text
533                 self.text_area.insert(tk.END, bill_text)
534                 self.text_area.configure(state='disabled')
535
536                 # Inserting data into a table
537                 sql = "INSERT INTO `cust_details` (cust_name, cust_email, cust_phone, membership_id, total_ticket, TOTAL) VALUES (%s, %s, %s, %s, %s, %s)"
538                 val = (self.entry_name.get(), self.entry_email.get(), self.entry_phone.get(), self.entry_membershipid.get(), self.entry_num_tickets.get(), discounted_cost)
539
540                 try:
541                     cursor.execute(sql, val)
542                     mydb.commit()
543                     print("Data inserted successfully!")
544                 except mysql.connector.Error as err:
545                     print(f"Error: {err}")
546                     mydb.rollback()
547             else:
548                 messagebox.showerror("Error", "Please enter a valid number of tickets (greater than 0 and less than or equal to 10).")
549
550         except ValueError:
551             messagebox.showerror("Error", "Please enter valid numeric values.")
552
553         finally:
554         # Close the cursor and the connection after finishing all database operations
555             cursor.close()
556             mydb.close()
557
558     def calculate_discount_ticket(self, num_tickets):
559         if num_tickets >= 3:
560             return 10  # 10% discount for 3 or more tickets
561         else:
562             return 0   # No discount for fewer than 3 tickets
563
564     def update_ticket(self):
565         mydb = mysql.connector.connect(
566             host="localhost",
567             user="root",
568             password="",
569             database="kpop_site"
570         )
571
572         # Create a cursor object to interact with the database
573         cursor = mydb.cursor()
574
575         try:
576             new_name = self.entry_name.get()
577             new_email = self.entry_email.get()
578             new_phone = self.entry_phone.get()
579
580             # Updating data in the table
581             sql = "UPDATE `cust_details` SET cust_name=%s, cust_email=%s WHERE cust_phone=%s"
582             val = (new_name, new_email, new_phone)
583
584             cursor.execute(sql, val)
585             mydb.commit()
586             print("Data updated successfully!")
587
```

Figure 2.8 Line 521 to Line 587 of babelpawp.py coding in Visual Studio code

```
588             # Fetch the updated data from the database
589             select_sql = "SELECT * FROM `cust_details` WHERE cust_phone=%s"
590             select_val = (new_phone,)
591
592             cursor.execute(select_sql, select_val)
593             updated_data = cursor.fetchone()
594
595             if updated_data:
596                 # Update the billing area with the updated data
597                 details_text = f"Name: {updated_data[1]}\nEmail: {updated_data[2]}\n"
598                 details_text += f"Phone: {updated_data[3]}\nMembership ID: {updated_data[4]}\n"
599                 details_text += f"Total Tickets: {updated_data[5]}\n"
600                 details_text += f"Cost: RM{updated_data[6]:.2f}\nDiscount: {self.discount_percentage}%\n"
601                 details_text += f"TOTAL: RM{updated_data[7]:.2f}"
602
603                 # Clear the existing content in the billing area and insert the updated data
604                 self.text_area.configure(state='normal')
605                 self.text_area.delete(1.0, tk.END)
606                 self.text_area.insert(tk.END, details_text)
607                 self.text_area.configure(state='disabled')
608
609                 # Optional: Display a message to the user indicating a successful update
610                 messagebox.showinfo("Success", "Data updated successfully!")
611
612         except mysql.connector.Error as err:
613             print(f"Error: {err}")
614             mydb.rollback()
615             # Optional: Display an error message to the user
616             messagebox.showerror("Error", f"Error updating data: {err}")
617
618         finally:
619             # Close the cursor and the connection when the application is closed
620             cursor.close()
621             mydb.close()
622
623     def delete_ticket(self):
624         mydb = mysql.connector.connect(
625         host="localhost",
626         user="root",
627         password="",
628         database="kpop_site"
629         )
630
631         # Create a cursor object to interact with the database
632         cursor = mydb.cursor()
633
634         try:
635
636             # Deleting data from the table
637             sql = "DELETE FROM `cust_details` WHERE cust_name=%s"
638             val = (self.entry_name.get(),)
639
640             cursor.execute(sql, val)
641             mydb.commit()
642             print("Data deleted successfully!")
643
644             # Optional: Display a message to the user indicating a successful delete
645             messagebox.showinfo("Success", "Data deleted successfully!")
646
647         except mysql.connector.Error as err:
648             print(f"Error: {err}")
649             mydb.rollback()
650             # Optional: Display an error message to the user
651             messagebox.showerror("Error", f"Error deleting data: {err}")
652
653         # Close the cursor and the connection when the application is closed
654         cursor.close()
655         mydb.close()
```

Figure 2.9 Line 588 to Line 655 of babelpawp.py coding in Visual Studio code

```
656
657     def show_help_fan(self):
658         # Help text button for fanmeeting registration window
659         help_text_fan = """
660         How to register for PAWPAWP FANMEETING?
661
662         Fill in the following information:
663         - Name: Your full name.
664         - ID number: Your identification number.
665         - Email (xxx@xxx.xxx): Your email address.
666         - Membership ID: Your Babelpawp Membership ID.
667         - Fanmeetings are available for members only.
668           If you don't have a Membership ID, please press
669           the back button and proceed to register as a
670           member first.
671         - Choose the number of ticket you want to buy
672           (limited to 5 ticket per purchase)
673         - 1 ticket = RM150
674
675         After filling in the details, click the 'Enter' button to
676         complete the purchasing.
677
678         If you encounter any issues, reach out to our support
679         team on either of these contacts below.
680
681         Phone number: +601133228274
682         e-mail: babelpawp@gmail.com
683
684         Thank you for supporting BABELPAWP!
685         """
686
687         messagebox.showinfo("Help", help_text_fan)
688
689     def open_fanmeeting_gui(self):
690         # GUI
691         fanmeeting_window = tk.Toplevel(self.master)
692         fanmeeting_window.title("Fanmeet booking")
693         fanmeeting_window.geometry("485x520")
694
695         # GUI label
696         label= tk.Label(fanmeeting_window, text='PAWPAWP FANMEETING',font=('Georgia Font', 20, 'italic'),bg='magenta4',fg='mistyrose',bd=10, relief='groove')
697         label.pack(ipadx=480)
698
699         # Customer details frame
700         customer_details_frame=tk.LabelFrame(fanmeeting_window,text='Customer Details',font=('Times New Roman',10,'bold'),bg='thistle2',fg= 'purple4',bd=5,relief= 'groove')
701         customer_details_frame.pack(ipadx=470)
702
703         # Name
704         self.nameLabel=tk.Label(customer_details_frame,text='Name',font=('Times New Roman',10,'bold'),bg='thistle2',fg= 'magenta4')
705         self.nameLabel.grid (row=0,column=0,ipadx=5,ipady=5)
706         self.nameEntry=tk.Entry(customer_details_frame,font=('Times New Roman', 10),bd=3)
707         self.nameEntry.grid (row=0,column=1,ipadx=17)
708
709         # Email
710         self.emailLabel=tk.Label(customer_details_frame,text='Email',font=('Times New Roman',10,'bold'),bg='thistle2',fg= 'magenta4')
711         self.emailLabel.grid(row=1,column=0,ipadx=5,ipady=5)
712         self.emailEntry=tk.Entry(customer_details_frame,font=('Times New Roman', 10),bd=3)
713         self.emailEntry.grid (row=1,column=1,ipadx=17)
714
715         # Membership ID
716         self.membershipLabel=tk.Label(customer_details_frame,text='Member ID',font=('Times New Roman',10,'bold'),bg='thistle2',fg= 'magenta4')
717         self.membershipLabel.grid(row=0,column=2,ipadx=3,ipady=5)
718         self.membershipEntry=tk.Entry(customer_details_frame,font=('Times New Roman', 10),bd=3)
719         self.membershipEntry.grid (row=0,column=3,ipadx=20)
```

Figure 2.10 Line 656 to Line 719 of babelpawp.py coding in Visual Studio code

```
720
721         # Ticket number
722         self.ticketLabel=tk.Label(customer_details_frame,text='Ticket amount',font=('Times New Roman',10,'bold'),bg='thistle2',fg= 'magenta4')
723         self.ticketLabel.grid(row=1,column=2,ipadx=3,ipady=5)
724         self.ticket_spinbox = ttk.Spinbox(customer_details_frame, values=["1", "2", "3","4","5"])
725         self.ticket_spinbox.grid(row=1,column=3,ipadx=15)
726
727         # Ticket bill frame
728         self.ticketframe=tk.LabelFrame(fanmeeting_window,text='Ticket bill',font=('Times New Roman',10,'bold'),bg='thistle2',fg= 'purple4',bd=5,relief= 'groove')
729         self.ticketframe.pack(ipadx=470)
730
731         # Enter frame
732         self.enterbuttonFrame=tk.Frame(self.ticketframe,bd=8,relief='groove')
733         self.enterbuttonFrame.grid(row=2,column=0,padx=5,pady=20)
734         self.enterButton=tk.Button(self.ticketframe,text='ENTER',font=('Times New Roman',10,'bold'),bd=5,bg='thistle2',fg= 'magenta4',padx=50,pady=10,command=self.register_fan)
735         self.enterButton.grid(row=2,column=0,pady=20, padx=5, columnspan=2)
736
737         # Update frame
738         self.updatebuttonFrame=tk.Frame(self.ticketframe,bd=8,relief='groove')
739         self.updatebuttonFrame.grid(row=3,column=0,padx=5,pady=20)
740         self.updatebutton=tk.Button(self.ticketframe, text="UPDATE", font=('Times New Roman',10,'bold'),bd=5,bg='thistle2',fg= 'magenta4',padx=50,pady=10, command=self.update_fan_gui)
741         self.updatebutton.grid(row=3,column=0,pady=20,padx=5,columnspan=2)
742
743         # Delete Frame
744         self.deletebuttonFrame=tk.Frame(self.ticketframe,bd=8,relief='groove')
745         self.deletebuttonFrame.grid(row=4,column=0,padx=5,pady=20)
746         self.deleteButton=tk.Button(self.ticketframe,text='DELETE',font=('Times New Roman',10,'bold'),bd=5,bg='thistle2',fg= 'magenta4',padx=50,pady=10,command=self.delete_fan_gui)
747         self.deleteButton.grid(row=4,column=0,pady=20, padx=5, columnspan=2)
748
749         # Bill frame
750         self.billframe=tk.Frame(self.ticketframe,bd=8,bg='thistle1',relief='groove')
751         self.billframe.grid(row=0,column=2,ipadx=5,pady=5,padx=10,rowspan=20)
752         self.billareaLabel=tk.Label(self.billframe,text='TOTAL',font=('Times New Roman',10,'bold'),bg='thistle1',fg= 'magenta4')
753         self.billareaLabel.pack()
754         self.scrollbar=tk.Scrollbar(self.billframe,orient='vertical')
755         self.scrollbar.pack(side='right', fill='y')
756         self.textareafan=tk.Text(self.billframe,width=30,height=15)
757         self.textareafan.pack(ipady=5)
758         self.scrollbar.config(command=self.textareafan.yview)
759
760         # Disable the text area
761         self.textareafan.config(state='disabled')
762
763         # Help, register, and quit button frame
764         button_frame=tk.LabelFrame(fanmeeting_window, bg='thistle1', bd=8, relief='groove')
765         button_frame.pack(ipadx=470)
766
767         # Help and quit button with grid
768         help_button = tk.Button(button_frame, text="Help", command=self.show_help_fan, font=("Times New Roman", 12), bg="magenta4", fg="white")
769         help_button.grid(row=0, column=0, pady=5, padx=80, ipadx=30, sticky="ew")
770         quit_button = tk.Button(button_frame, text="Back", command=fanmeeting_window.destroy, font=("Times New Roman", 12), bg="magenta4", fg="white")
771         quit_button.grid(row=0, column=1, pady=5, padx=1, ipadx=30, sticky="ew")
772
773     # Calculation
774     def register_fan(self):
775         name_fan = self.nameEntry.get()
776         email_fan = self.emailEntry.get()
777         numtic_fan = self.ticket_spinbox.get()
778         memid_fan = self.membershipEntry.get()
779
780         if not name_fan or not email_fan or not numtic_fan or not memid_fan:
781             messagebox.showerror("Error", "Please fill in all fields.")
782             return
783
784         # Check if name contains only alphabetic characters
785         if not name_fan.replace(" ", "").isalpha():
786             messagebox.showerror("Invalid", "Name must contain only alphabetic characters.")
787             return
788
```

Figure 2.11 Line 720 to Line 788 of babelpawp.py coding in Visual Studio code

```
789         # Check email format using a regular expression
790         email_pattern = re.compile(r'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z0-9]+$')
791         if not email_pattern.match(email_fan):
792             messagebox.showerror("Invalid", "Invalid email format. Please use the format xxx@xxx.xxx.")
793             return
794 
795         try:
796             ticket_price = 150
797             num_tickets = int(self.ticket_spinbox.get())
798             total_cost = num_tickets * ticket_price
799             bill_text = f"Total ticket: {num_tickets}\nTotal Cost:RM {total_cost:}\n"
800 
801         except ValueError:
802             messagebox.showerror("Error", "Please enter a valid number of tickets.")
803             return
804 
805         # Generate a random alphanumeric Ticket ID
806         fantic_id = ''.join(random.choices(string.ascii_uppercase + string.digits, k=6))
807 
808         # Display registration information in a message box
809         fan_success_message = f"Purchase Successful!\n\nTicket ID: {fantic_id}\nName: {name_fan}\nEmail: {email_fan}\nMembership ID: {memid_fan}\nNumber of tickets: {numtic_fan}\nTotal cost: {total_cost}"
810         messagebox.showinfo("Purchase Successful", fan_success_message)
811 
812         self.textareafan.config(state='normal')
813         self.textareafan.insert(tk.END, fan_success_message + '\n\n' + bill_text)
814         self.textareafan.config(state='disabled')
815 
816         self.insert_into_database_fan(fantic_id, name_fan, email_fan, memid_fan, numtic_fan, total_cost)
817 
818     def insert_into_database_fan(self, fantic_id, name_fan, email_fan, memid_fan, numtic_fan, total_cost):
819     # Establish a connection to the MySQL server
820         try:
821             mydb = mysql.connector.connect(
822                 host="localhost",
823                 user="root",
824                 password="",
825                 database="kpop_site"
826             )
827 
828             # Create a cursor object to interact with the database
829             cursor = mydb.cursor()
830 
831             # Inserting data into a table
832             sql = "INSERT INTO `fm_details` (Ticket_ID, Name, Email, Membership_ID, Number_of_Tickets, Total_Cost) VALUES (%s, %s, %s, %s, %s, %s)"
833             val = (fantic_id, name_fan, email_fan, memid_fan, numtic_fan, total_cost)
834 
835             cursor.execute(sql, val)
836             mydb.commit()
837             print("Data inserted successfully!")
838 
839         except mysql.connector.Error as err:
840             print(f"Error: {err}")
841             mydb.rollback()
842             cursor.close()
843             mydb.close()
844             return
845 
846         finally:
847             cursor.close()
848             mydb.close()
849 
850     def update_fan(self, fantic_id):
851         # User input new entry
852         newname_fan = self.nameEntry.get()
853         newmemid_fan = self.membershipEntry.get()
854         newemail_fan = self.emailEntry.get()
855         newnumtic_fan = self.ticket_spinbox.get()
856 
```

Figure 2.12 Line 789 to Line 856 of babelpawp.py coding in Visual Studio code

```
857         if not fantic_id or not newname_fan or not newmemid_fan or not newemail_fan or not newnumtic_fan:
858             messagebox.showerror("Error", "Please fill in all fields.")
859             return
860
861         # Check if name contains only alphabetic characters
862         if not newname_fan.replace(" ", "").isalpha():
863             messagebox.showerror("Invalid", "Name must contain only alphabetic characters.")
864             return
865
866         # Check email format using a regular expression
867         email_pattern = re.compile(r'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z0-9]+$')
868         if not email_pattern.match(newemail_fan):
869             messagebox.showerror("Invalid", "Invalid email format. Please use the format xxx@xxx.xxx.")
870             return
871
872         try:
873             ticket_price = 150
874             newnumtic_fan = int(self.ticket_spinbox.get())
875             total_cost = newnumtic_fan * ticket_price
876
877         except ValueError:
878             messagebox.showerror("Error", "Please enter a valid number of tickets.")
879             return
880
881         # Establish a connection to the MySQL server
882         mydb = mysql.connector.connect(
883             host="localhost",
884             user="root",
885             password="",
886             database="kpop_site"
887         )
888
889         # Create a cursor object to interact with the database
890         cursor = mydb.cursor()
891
892         # Fetch the updated data from the database
893         cursor.execute("SELECT * FROM `fm_details` WHERE Ticket_ID=%s", (fantic_id,))
894         updated_fan_info = cursor.fetchone()
895
896         # UPDATE SQL statement
897         sql = "UPDATE `fm_details` SET `Name`= %s, `Email`= %s, `Membership_ID`= %s, `Number_of_Tickets`= %s, `Total_Cost`= %s WHERE Ticket_ID=%s"
898         val = (newname_fan, newemail_fan, newmemid_fan, newnumtic_fan, total_cost, fantic_id)
899
900         try:
901             cursor.execute(sql, val)
902             mydb.commit()
903             print("Data updated successfully!")
904
905             # Fetch the updated data from the database
906             cursor.execute("SELECT * FROM `fm_details` WHERE Ticket_ID=%s", (fantic_id,))
907             updated_fan_info = cursor.fetchone()
908
909         except mysql.connector.Error as err:
910             print(f"Error: {err}")
911             mydb.rollback()
912
913         finally:
914             # Close the cursor and the connection when the application is closed
915             cursor.close()
916             mydb.close()
917
918         # Display a message to the user indicating a successful update
919         messagebox.showinfo("Update Successful", "Data updated successfully!")
920
921         # Return the updated member information
922         return updated_fan_info
923
```

Figure 2.13 Line 857 to Line 923 of babelpawp.py coding in Visual Studio code

```
924     def update_fan_gui(self):
925         # Prompt the user for Ticket ID to update
926         fantic_id_to_update = simpledialog.askstring("Update Fan", "Enter Ticket ID to update:")
927 
928         if fantic_id_to_update is not None:
929             # Call the update_fan method
930             updated_fan_info = self.update_fan(fantic_id_to_update)
931 
932             if updated_fan_info:
933                 # Clear existing content in data display box
934                 self.textareafan.config(state='normal')
935                 self.textareafan.delete(1.0, tk.END)
936 
937                 # Display updated member information in the bill area
938                 updated_fan_info_str = (
939                     f"Updated Member Information:\n\n"
940                     f"Ticket ID: {updated_fan_info[0]}\n"
941                     f"Name: {updated_fan_info[1]}\n"
942                     f"Email: {updated_fan_info[2]}\n"
943                     f"Member ID: {updated_fan_info[3]}\n"
944                     f"Number of Tickets: {updated_fan_info[4]}\n"
945                     f"Total Cost: {updated_fan_info[5]}"
946                 )
947 
948                 self.textareafan.insert(tk.END, updated_fan_info_str)
949                 self.textareafan.config(state='disabled')
950 
951     def delete_fan(self, fantic_id):
952         # Validate input
953         if not fantic_id:
954             messagebox.showerror("Error", "Please fill in the Ticket ID.")
955             return
956 
957 
958         mydb = mysql.connector.connect(
959             host="localhost",
960             user="root",
961             password="",
962             database="kpop_site"
963         )
964 
965         # Create a cursor object to interact with the database
966         cursor = mydb.cursor()
967 
968 
969         try:
970             # Deleting data from the table based on Ticket ID
971             sql = "DELETE FROM fm_details WHERE Ticket_ID=%s"
972             val = (fantic_id,)
973 
974             cursor.execute(sql, val)
975             mydb.commit()
976             print("Data deleted successfully!")
977 
978             # Display a message to the user indicating a successful delete
979             messagebox.showinfo("Success", "Data deleted successfully!")
980 
981         except mysql.connector.Error as err:
982             print(f"Error: {err}")
983             mydb.rollback()
984 
985             # Optional: Display an error message to the user
986             messagebox.showerror("Error", f"Error deleting data: {err}")
987 
988         finally:
989             # Close the cursor and the connection when the application is closed
990             cursor.close()
991             mydb.close()
992 
```

Figure 2.14 Line 924 to Line 992 of babelpawp.py coding in Visual Studio code

```python
992
993
994     def delete_fan_gui(self):
995         # Prompt the user for Membership ID to delete
996         fantic_id_to_delete = simpledialog.askstring("Delete Fan", "Enter Ticket ID to delete:")
997
998         if fantic_id_to_delete:
999             # Call the delete_fan method
1000            self.delete_fan(fantic_id_to_delete)
1001
1002            # Clear existing content in data display box
1003            self.textareafan.config(state='normal')
1004            self.textareafan.delete(1.0, tk.END)
1005            self.textareafan.config(state='disabled')
1006
1007    def quit_application(self):
1008        self.master.quit()
1009
1010
1011 def main():
1012     root = tk.Tk()
1013     app: KpopPurchasingSystem = KpopPurchasingSystem(root)
1014     app.master.mainloop()
1015
1016 if __name__ == "__main__":
1017     main()
1018
```

Figure 2.15 Line 993 to Line 1018 of babelpawp.py coding in Visual Studio code

## 5.2 GUI


Figure 3.0 Main menu of Babelpawp

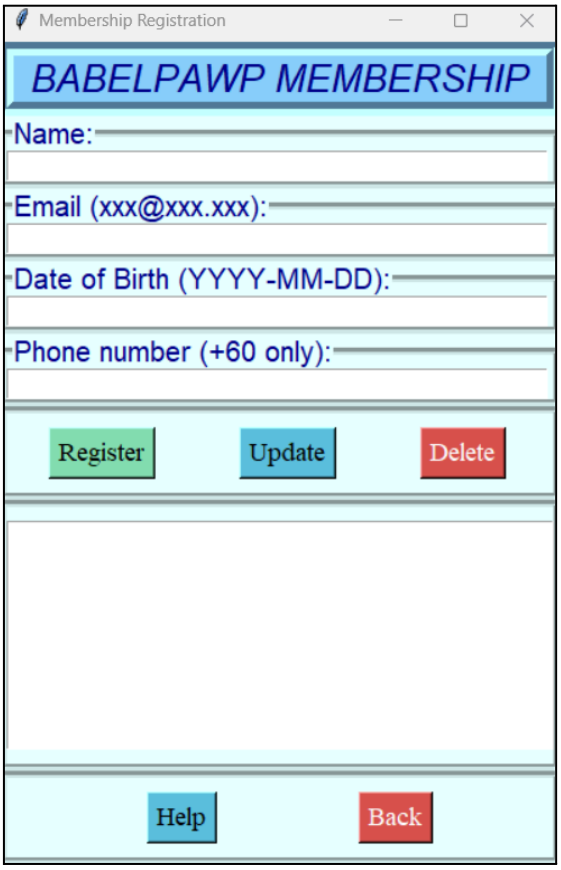


Figure 3.1 Membership registration window

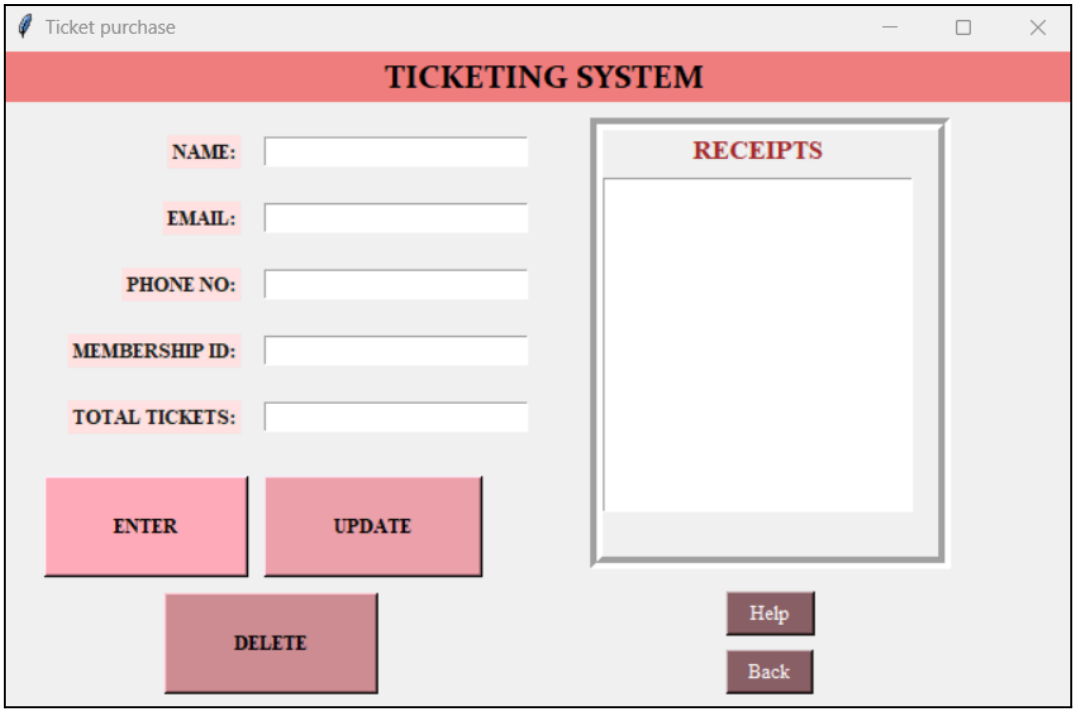


Figure 3.2 Ticketing system window

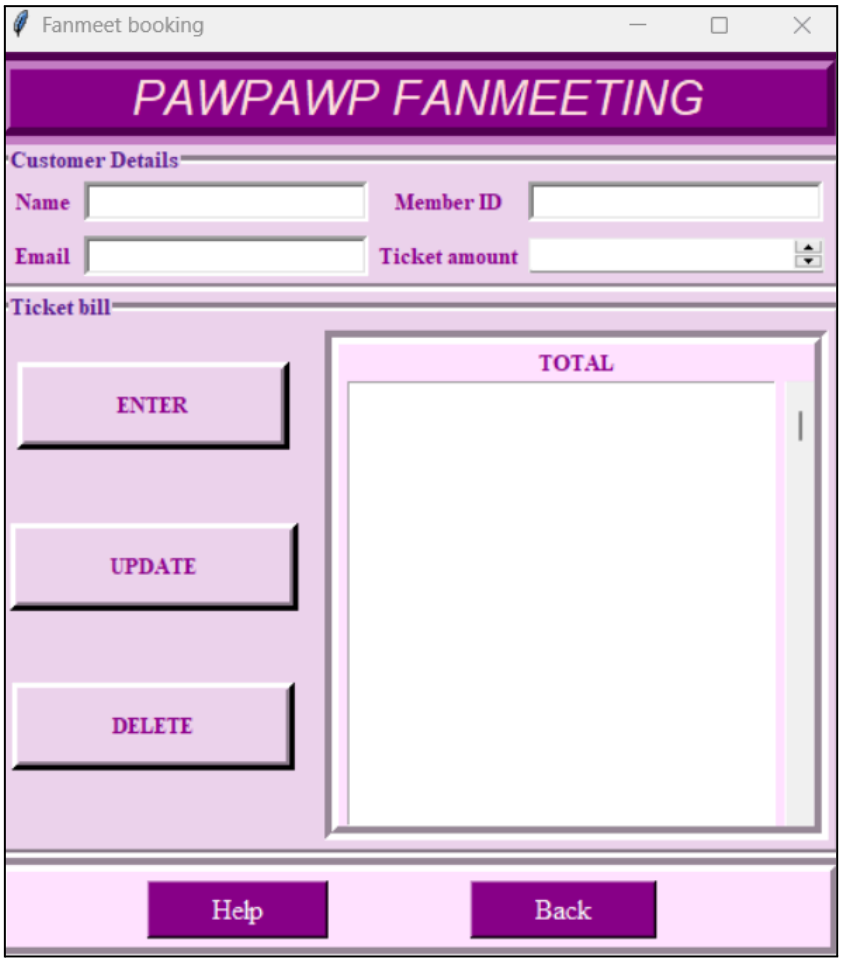


Figure 3.3 Fan Meeting ticket system window

## 5.3 DATABASE

### 5.3.1 Database structure

Server: 127.0.0.1 » Database: kpop_site

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Routines | Events | Triggers | Designer

Filters

Containing the word:

| Table | Action | Rows | Type | Collation | Size | Overhead |
|---|---|---|---|---|---|---|
| cust_details | Browse Structure Search Insert Empty Drop | 4 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| fm_details | Browse Structure Search Insert Empty Drop | 4 | InnoDB | utf8mb4_general_ci | 32.0 KiB | - |
| memship_details | Browse Structure Search Insert Empty Drop | 4 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| 3 tables | Sum | 12 | InnoDB | utf8mb4_general_ci | 64.0 KiB | 0 B |

Check all With selected:

Figure 5.0 'kpop_site' database structure

### 5.3.2 Table Structure

Server: 127.0.0.1 » Database: kpop_site » Table: memship_details

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)

```
SELECT * FROM `memship_details`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

| member_ID | user_name | user_dob | user_age | user_email | user_phone |
|---|---|---|---|---|---|
| R1G9K | Nurul Syakirah | 2004-12-10 | 19 | ira@gmail.com | 6011332273847 |
| 2HHKH | Alia Sofea | 2004-06-10 | 19 | aliasofea@gmail.com | 60112385745 |
| QCOPB | Ameera Nafeesa | 2004-03-20 | 19 | nana@gmail.com | 60142345726 |
| KQM4U | Hani Rasyiqah Irdina | 2004-01-23 | 19 | hani@gmail.com | 60115726348 |

Show all | Number of rows: 25 Filter rows: Search this table

Figure 5.0 'memship_details' browse page

Server: 127.0.0.1 » Database: kpop_site » Table: memship_details

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

Table structure | Relation view

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | member_ID | varchar(6) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| 2 | user_name | text | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| 3 | user_dob | date | | | No | *None* | | | Change Drop More |
| 4 | user_age | int(3) | | | No | *None* | | | Change Drop More |
| 5 | user_email | varchar(100) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| 6 | user_phone | bigint(12) | | | No | *None* | | | Change Drop More |

Figure 5.1 'memship_details' table structure

Server: 127.0.0.1 » Database: kpop_site » Table: cust_details

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)

```
SELECT * FROM `cust_details`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

| cust_name | cust_email | cust_phone | membership_id | total_ticket | TOTAL |
|---|---|---|---|---|---|
| Nurul Syakirah | ira@gmail.com | 601123454837 | R1G9K | 2 | 100 |
| Alia Sofea | alia@gmail.com | 60112385745 | 2HHKH | 1 | 50 |
| Ameera Nafeesa | nana@gmail.com | 60142345726 | QCOPB | 3 | 135 |
| Hani Rasyiqah Irdina | hani@gmail.com | 60115726348 | KQM4U | 5 | 225 |

Show all | Number of rows: 25 Filter rows: Search this table

Figure 6.0 ‘cust_details’ browse

Server: 127.0.0.1 » Database: kpop_site » Table: cust_details

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

Table structure | Relation view

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | cust_name | varchar(30) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 2 | cust_email | varchar(30) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 3 | cust_phone | bigint(12) | | | No | None | | | Change Drop More |
| 4 | membership_id | varchar(6) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 5 | total_ticket | int(3) | | | No | None | | | Change Drop More |
| 6 | TOTAL | int(10) | | | No | None | | | Change Drop More |

Figure 6.1 ‘cust_details’ table structure

Server: 127.0.0.1 » Database: kpop_site » Table: fm_details

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 3 (4 total, Query took 0.0004 seconds.)

```
SELECT * FROM `fm_details`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

| Ticket_ID | Name | Email | Membership_ID | Number_of_Tickets | Total_Cost |
|---|---|---|---|---|---|
| O8KRIG | Nurul Syakirah | ira@gmail.com | R1G9K | 1 | 150 |
| W0I9I3 | Alia Sofea | alia@gmail.com | 2HHKH | 4 | 600 |
| DXDBOJ | Ameera Nafeesa | nana@gmail.com | QCOPB | 2 | 300 |
| MIR1LL | Hani Rasyiqah Irdina | hani@gmail.com | KQM4U | 3 | 450 |

Show all | Number of rows: 25 Filter rows: Search this table

Figure 7.0 'fm_details' browse structure

Server: 127.0.0.1 » Database: kpop_site » Table: fm_details

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

Table structure | Relation view

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | **Ticket_ID** | varchar(8) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| 2 | **Name** | char(100) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| 3 | **Email** | varchar(30) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| 4 | **Membership_ID** | varchar(5) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| 5 | **Number_of_Tickets** | int(1) | | | No | *None* | | | Change Drop More |
| 6 | **Total_Cost** | int(10) | | | No | *None* | | | Change Drop More |

Figure 7.1 'fm_details' table structure

## 6.0 CONCLUSION

In summary, Babelpawp is a revolutionary solution for ticketing purchase systems, meeting a wide range of user needs with its cutting-edge features and intuitive interface. The system utilizes Tkinter GUI and phpMyAdmin sharing the same goal which is to provide users with a visual and user-friendly experience.

We created this system in hopes to satisfy users' changing needs in the ever-changing event ticketing market. In addition to streamlining the ticket buying process, the platform has the potential to grow into a reliable resource for fans of all kinds of events as it develops further and takes user feedback into consideration.

Reflecting on this project, we have learned various new things as in how to provide a systematic GUI including how to design it. In addition, we also gained knowledge on how to troubleshoot our coding and acknowledged the technicalities of Create Read Update and Delete operations.

**STUDENT PLEDGE OF ACADEMIC INTEGRITY**

As a student of Universiti Teknologi MARA (UiTM), it is my responsibility to act in accordance with UiTM's academic assessment and evaluation policy. I hereby pledge to act and uphold academic integrity and pursue scholarly activities in UiTM with honesty and responsible manner. I will not engage or tolerate acts of academic dishonesty, academic misconduct, or academic fraud including but not limited to:

a. **Cheating:** Using or attempt to use any unauthorized device, assistance, sources, practice or materials while completing academic assessments. This include but not limited to copying from another, allowing others to copy, unauthorized collaboration on an assignment or open book tests, or engaging in any act or conduct that can be construed as cheating.
b. **Plagiarism:** Using or attempts to use the work of others (ideas, design, words, art, music, etc.) without acknowledging the source; using or purchasing materials prepared by another person or agency or engaging in other behavior that a reasonable person would consider as plagiarism.
c. **Fabrication:** Falsifying data, information, or citations in any academic assessment and evaluation.
d. **Deception:** Providing false information with intend to deceive an instructor concerning any academic assessment and evaluation.
e. **Furnishing false information:** Providing false information or false representation to any UiTM official, instructor, or office.

With this pledge, I am fully aware that I am obliged to conduct myself with utmost honesty and integrity. I fully understand that a disciplinary action can be taken against me if I, in any manner, violate this pledge.

**Name : ALIA SOFEA BINTI AHMAD FADZLI**
**Matric Number : 2022834288**
**Course Code : IML208**
**Programme Code :-**
**Faculty / Campus : UiTM Kampus Sungai Petani**

**STUDENT PLEDGE OF ACADEMIC INTEGRITY**

As a student of Universiti Teknologi MARA (UiTM), it is my responsibility to act in accordance with UiTM's academic assessment and evaluation policy. I hereby pledge to act and uphold academic integrity and pursue scholarly activities in UiTM with honesty and responsible manner. I will not engage or tolerate acts of academic dishonesty, academic misconduct, or academic fraud including but not limited to:

a. **Cheating:** Using or attempt to use any unauthorized device, assistance, sources, practice or materials while completing academic assessments. This include but not limited to copying from another, allowing others to copy, unauthorized collaboration on an assignment or open book tests, or engaging in any act or conduct that can be construed as cheating.
b. **Plagiarism:** Using or attempts to use the work of others (ideas, design, words, art, music, etc.) without acknowledging the source; using or purchasing materials prepared by another person or agency or engaging in other behavior that a reasonable person would consider as plagiarism.
c. **Fabrication:** Falsifying data, information, or citations in any academic assessment and evaluation.
d. **Deception:** Providing false information with intend to deceive an instructor concerning any academic assessment and evaluation.
e. **Furnishing false information:** Providing false information or false representation to any UiTM official, instructor, or office.

With this pledge, I am fully aware that I am obliged to conduct myself with utmost honesty and integrity. I fully understand that a disciplinary action can be taken against me if I, in any manner, violate this pledge.

**Name : AMEERA NAFEESA BINTI AZHARI**
**Matric Number : 2022629292**
**Course Code : IML208**
**Programme Code :-**
**Faculty / Campus : UiTM Kampus Sungai Petani**

**STUDENT PLEDGE OF ACADEMIC INTEGRITY**

As a student of Universiti Teknologi MARA (UiTM), it is my responsibility to act in accordance with UiTM's academic assessment and evaluation policy. I hereby pledge to act and uphold academic integrity and pursue scholarly activities in UiTM with honesty and responsible manner. I will not engage or tolerate acts of academic dishonesty, academic misconduct, or academic fraud including but not limited to:

a. **Cheating:** Using or attempt to use any unauthorized device, assistance, sources, practice or materials while completing academic assessments. This include but not limited to copying from another, allowing others to copy, unauthorized collaboration on an assignment or open book tests, or engaging in any act or conduct that can be construed as cheating.
b. **Plagiarism:** Using or attempts to use the work of others (ideas, design, words, art, music, etc.) without acknowledging the source; using or purchasing materials prepared by another person or agency or engaging in other behavior that a reasonable person would consider as plagiarism.
c. **Fabrication:** Falsifying data, information, or citations in any academic assessment and evaluation.
d. **Deception:** Providing false information with intend to deceive an instructor concerning any academic assessment and evaluation.
e. **Furnishing false information:** Providing false information or false representation to any UiTM official, instructor, or office.

With this pledge, I am fully aware that I am obliged to conduct myself with utmost honesty and integrity. I fully understand that a disciplinary action can be taken against me if I, in any manner, violate this pledge.

**Name : HANI RASYIQAH IRDINA BINTI HASSANAL ZAIRI**
**Matric Number : 2022863864**
**Course Code : IML208**
**Programme Code :-**
**Faculty / Campus : UiTM Kampus Sungai Petani**

**STUDENT PLEDGE OF ACADEMIC INTEGRITY**

As a student of Universiti Teknologi MARA (UiTM), it is my responsibility to act in accordance with UiTM's academic assessment and evaluation policy. I hereby pledge to act and uphold academic integrity and pursue scholarly activities in UiTM with honesty and responsible manner. I will not engage or tolerate acts of academic dishonesty, academic misconduct, or academic fraud including but not limited to:

a. **Cheating:** Using or attempt to use any unauthorized device, assistance, sources, practice or materials while completing academic assessments. This include but not limited to copying from another, allowing others to copy, unauthorized collaboration on an assignment or open book tests, or engaging in any act or conduct that can be construed as cheating.
b. **Plagiarism:** Using or attempts to use the work of others (ideas, design, words, art, music, etc.) without acknowledging the source; using or purchasing materials prepared by another person or agency or engaging in other behavior that a reasonable person would consider as plagiarism.
c. **Fabrication:** Falsifying data, information, or citations in any academic assessment and evaluation.
d. **Deception:** Providing false information with intend to deceive an instructor concerning any academic assessment and evaluation.
e. **Furnishing false information:** Providing false information or false representation to any UiTM official, instructor, or office.

With this pledge, I am fully aware that I am obliged to conduct myself with utmost honesty and integrity. I fully understand that a disciplinary action can be taken against me if I, in any manner, violate this pledge.

**Name : NURUL SYAKIRAH BINTI ASRIL**
**Matric Number : 2022618038**
**Course Code : IML208**
**Programme Code :-**
**Faculty / Campus : UiTM Kampus Sungai Petani**