



COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS

UNIVERSITI TEKNOLOGI MARA (UiTM)
CAWANGAN KEDAH, KAMPUS SUNGAI PETANI

DIPLOMA IN LIBRARY INFORMATIC (CDIM144)

PROGRAMMING FOR LIBRARIES (IML208)

GROUP PROJECT:
“CINEMA BOOKING SYSTEM REPORT”

PREPARED BY:

NAME	MATRIC NO.
ADAM ISKANDAR BIN ZULKARNINE	2022642664
NURRUL HAIRIENA BINTI ARZA	2022829308
UMAIRAH WAJIEHAH BINTI MOHAMAD	2022607006
UMMI KALSUM TAQIAH BINTI SABUDIN	2022820722

CLASS: KCDIM144 3E

PREPARED FOR:
SIR AIRUL SHAZWAN BIN NORSHAHIMI

SUBMISSION DATE:

16 JANUARY 2024

GROUP RPROJECT:
“CINEMA BOOKING SYSTEM REPORT”

PREPARED BY:

NAME	MATRIC NO.
ADAM ISKANDAR BIN ZULKARNINE	2022642664
NURRUL HAIRIENA BINTI ARZA	2022829308
UMAIRAH WAJIEHAH BINTI MOHAMAD	2022607006
UMMI KALSUM TAQIAH BINTI SABUDIN	2022820722

CLASS: KCDIM144 3E

COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS

UNIVERSITI TEKNOLOGI MARA (UiTM)
CAWANGAN KEDAH, KAMPUS SUNGAI PETANI

SUBMISSION DATE:

16 JANUARY 2024

ACKNOWLEDGEMENT

Assalamualaikum, warahmatullahi wabarakatuh. Salam UiTM Dihatiku.

First and foremost, we would like to give never-ending praise to Allah, The All Mighty, for having blessed us with the ability and good health to complete this pairing assignment. Special thanks to our lecturer for IML208, Sir Airul Shazwan bin Norshahimi, whose guidance and moral support have been major keys throughout our journey in finishing this assignment. His invaluable contributions carried us through all the stages of writing our project. We could not have undertaken this journey without our parents and siblings, to whom we are deeply indebted. Their love and appreciation for our studies are what keep us going through hard times. Their prayers for us are what have sustained us this far without us wanting to give it all up. Words can never express our gratitude enough to all of us mentioned above.

Second, our deepest appreciation goes to our dearest groupmates, who have given us all they can for putting up with us even at our absolute worst and for never doubting us even once. Despite all the pressures we have on our shoulders, they have never stopped being our helping hands. Thank you so much.

Our only hope for this assignment is that it will soon be helpful, no matter how big or small. Again, thanks to everyone who was directly or indirectly involved in the making of this assignment.

TABLE OF CONTENT

1.0 INTRODUCTION.....	1
2.0 PROBLEM STATEMENTS	2
3.0 OBJECTIVES	3
4.0 FLOWCHART	4
4.1 MODULE FIRST: FLOWCHART FOR MEMBERSHIP MODULE.....	4
4.2 SECOND MODULE: FLOWCHART FOR MOVIE MODULE.....	5
4.3 THIRD MODULE: FLOWCHART FOR TICKET MODULE	6
5.0 SNAPSHOT OF CODING	7
6.0 SNAPSHOT OF PROJECT (GUI).....	15
6.1 Main Menu.....	15
6.2 Membership Module.....	15
6.3 Movie Module.....	16
6.4 Ticket Module	16
6.5 After insert the data.....	17
7.0 SNAPSHOT OF DATABASE (XAMPP).....	19
8.0 CONCLUSION	21

1.0 INTRODUCTION

For our group assignment, we wanted to create a systematic cinema booking system. In this system, we have included three different modules, which are the membership module, the movie module, and the ticket module. Each of these modules contains its own attributes and functions. The membership module contains attributes such as IC number, name, e-mail, state, age, and category. After the user inserts their IC number, it will automatically generate their age. The calculation involved in this step is taking the first two numbers from the IC number and subtracting them to 24, which represents the year 2024. Based on age, we used the selection method to determine their categories, whether child, teenager, or adult. After these data are submitted, they will be saved in the database. However, users can also choose to delete or update their data.

Next is the movie module. This module contains attributes such as movie name, duration, genre, and classification. We used the combobox to display our movie choices and price information for each movie. In this module, users will just need to select their movie of interest. After the movie has been selected, other attributes like the duration, genre, and classification will automatically appear, and users can click submit.

The last module would be the ticket module. The ticket module contains attributes such as IC number, selected movie, date, seat number, and pax. The selected movie from the movie module will be brought to the ticket module, which is in the selected movie attribute. Then, users just need to insert their IC number, date, seat number, and pax. The price of the selected movie will be multiplied by the number of passengers that users wish for. After the data is submitted, a receipt will be generated, containing the attributes completed with users' data together with a special booking ID. This booking ID uses the looping method, precisely while looping. Then, these data will be submitted to the database. However, users can also choose to delete or update their data.

2.0 PROBLEM STATEMENTS

Customer Service Issues:

- 1) Long Queue Times: Extended wait times for customers at the cinema, whether for purchasing tickets or collecting pre-booked tickets.
- 2) Poor Customer Support: Ineffective or unresponsive customer support channels can frustrate customers seeking assistance.

Inventory Management:

- 1) Overbooking: Selling more tickets than available seats due to poor inventory management.
- 2) Outdated Information: Failure to update showtimes, movie schedules, or seat availability in real-time.

Customer Experience:

- 3) Unpleasant User Interface: Poorly designed or confusing interfaces can negatively impact the user experience.
- 4) Lack of Personalization: Failing to offer personalized recommendations or loyalty rewards to frequent customers.

3.0 OBJECTIVES

1) Facilitate Ticket Sales:

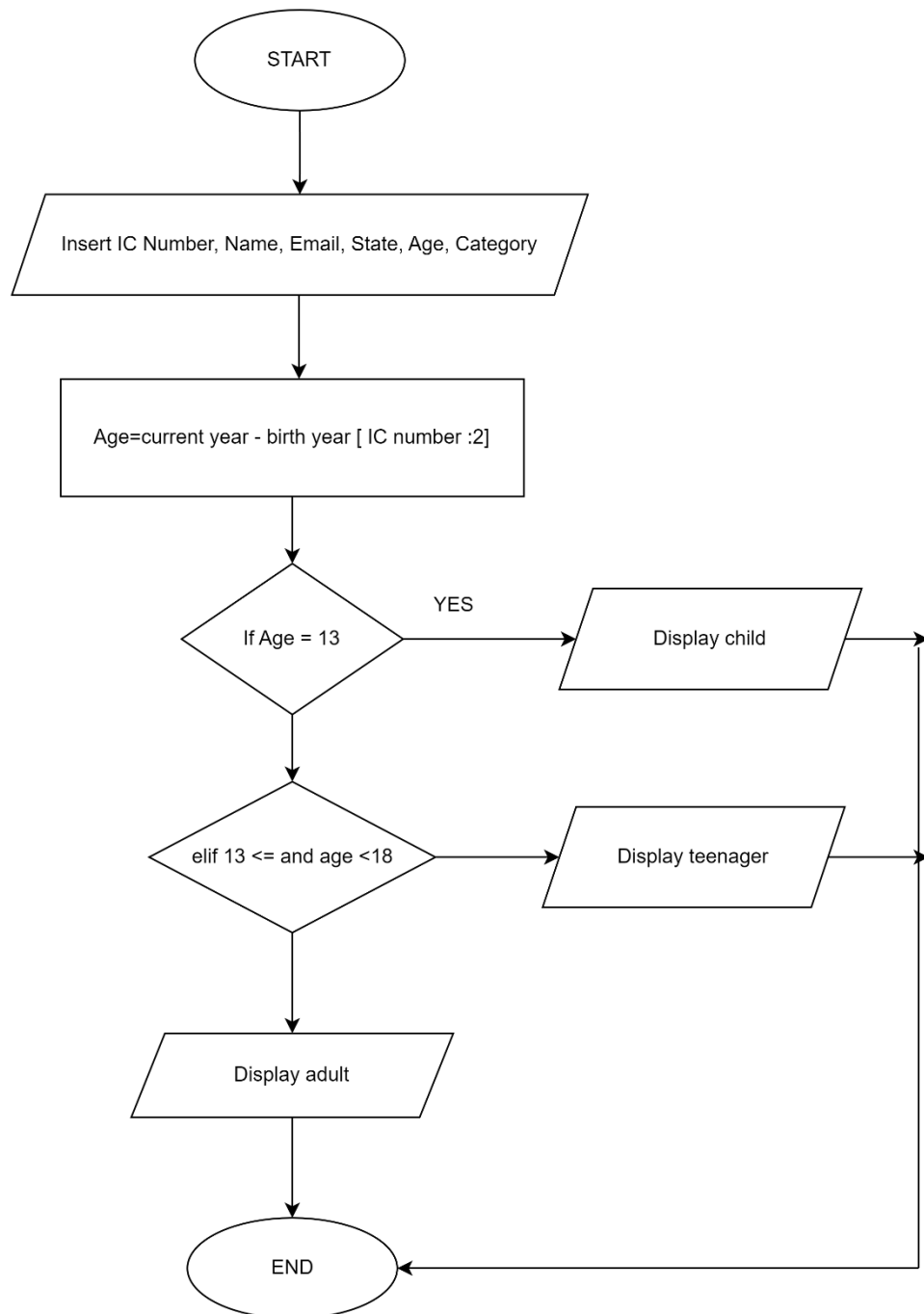
Enable customers to easily purchase tickets for movie screenings through multiple channels, including online platforms, mobile apps, kiosks, and on-site ticket counters.

2) Streamline Booking Process:

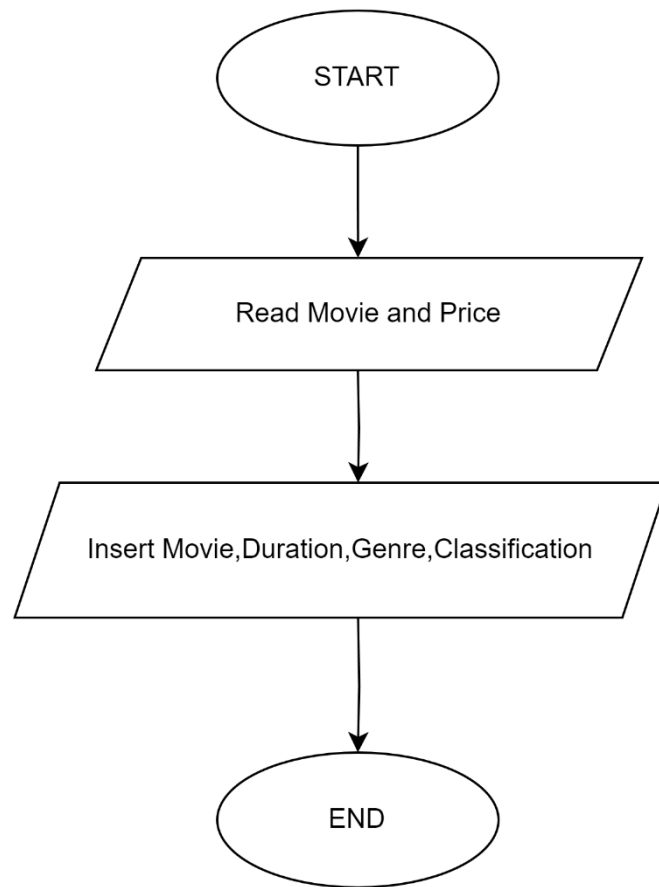
Provide a user-friendly and efficient booking process to minimize wait times and enhance the overall customer experience.

4.0 FLOWCHART

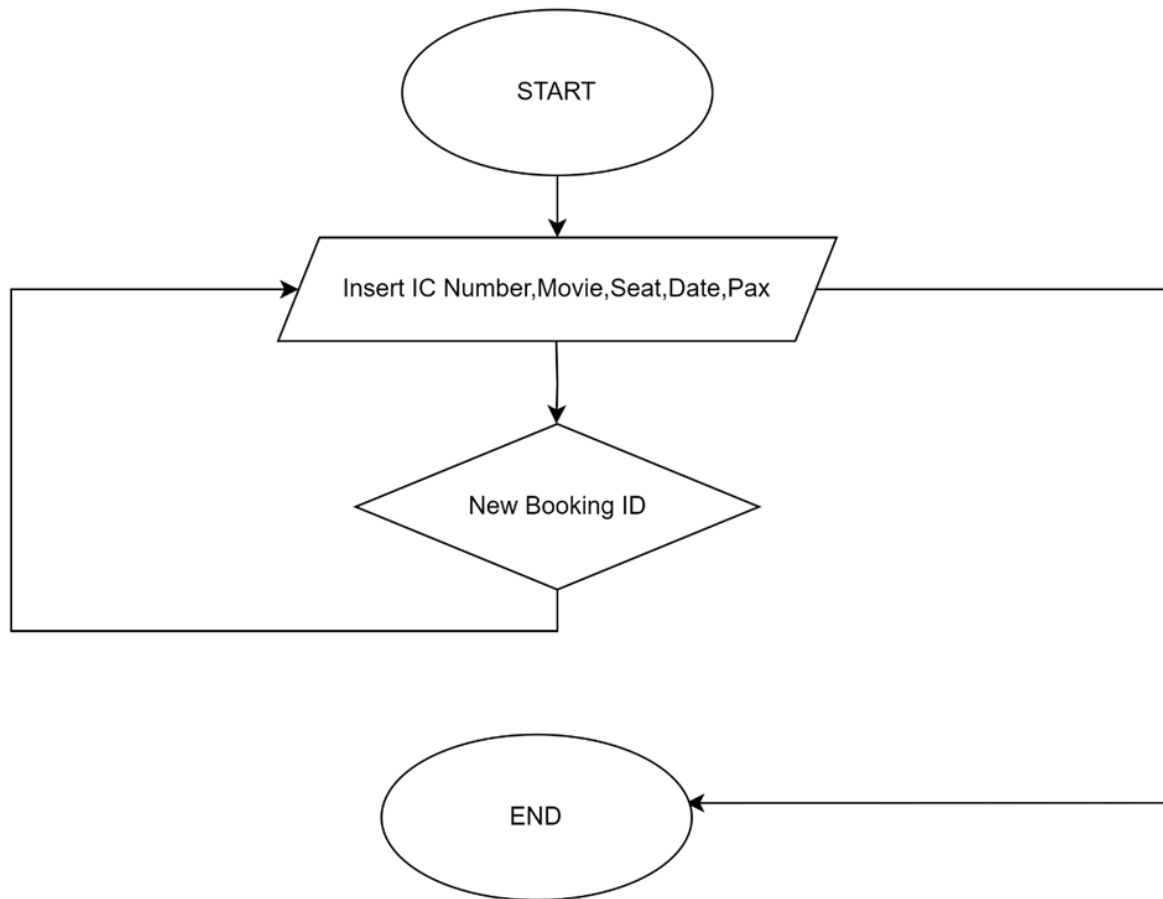
4.1 MODULE FIRST: FLOWCHART FOR MEMBERSHIP MODULE



4.2 SECOND MODULE: FLOWCHART FOR MOVIE MODULE



4.3 THIRD MODULE: FLOWCHART FOR TICKET MODULE



5.0 SNAPSHOT OF CODING

```
File Edit Selection View Go ... ← → Untitled (Workspace)
gp_assign.py × anotherexmp.py kiki.py
IML 208 > gp_assign.py > ...
1 import tkinter as tk
2 from tkinter import ttk
3 from tkinter import messagebox
4 import mysql.connector
5 import random
6
7 # Connect to your MySQL database
8 mydb = mysql.connector.connect(
9     host="localhost",
10    user="root",
11    password="",
12    database="cinema_system"
13 )
14
15 # Create a cursor object to execute SQL queries
16 mycursor = mydb.cursor()
17
18 class CinemaSystemApp:
19     def __init__(self, root):
20         self.root = root
21         self.root.title("Cinema System")
22         self.root.geometry("500x700")
23         self.root.configure(bg="lightgoldenrod1") # Set the background color for the main window
24
25         # GUI for Main Menu
26         self.root_frame = ttk.Frame(root)
27         self.root_frame.pack()
28         self.root_label = tk.Label(self.root_frame, text="WELCOME TO HAUU CINEMA!", font=("Impact", 20), bg="navajowhite2")
29         self.root_label.grid(row=0, column=0, columnspan=3, pady=10)
30
31         # Buttons for Membership, Ticket, and Movie
32         self.membership_button = tk.Button(self.root_frame, text="Membership", command=self.show_membership, bg="moccasin")
33         self.membership_button.grid(row=1, column=0, padx=5)
34
35         self.movie_button = tk.Button(self.root_frame, text="Movie", command=self.show_movie, bg="moccasin")
36         self.movie_button.grid(row=1, column=1, padx=5)
```

```
File Edit Selection View Go ... ← → Untitled (Workspace)
gp_assign.py × anotherexmp.py kiki.py
IML 208 > gp_assign.py > ...
35 self.movie_button = tk.Button(self.root_frame, text="Movie", command=self.show_movie, bg="moccasin")
36 self.movie_button.grid(row=1, column=1, padx=5)
37
38 self.ticket_button = tk.Button(self.root_frame, text="Ticket", command=self.show_ticket, bg="moccasin")
39 self.ticket_button.grid(row=1, column=2, padx=5)
40
41 self.current_module_frame = None
42
43 # Variable to store the selected movie
44 self.selected_movie = tk.StringVar()
45
46 # Dictionaries to store data
47 self.membership_data = {}
48 self.movie_data = {}
49 self.ticket_data = {}
50
51 def show_membership(self):
52     self.destroy_current_module()
53     self.current_module_frame = ttk.Frame(self.root_frame)
54     self.current_module_frame.grid(row=2, column=0, columnspan=3)
55     self.create_membership_tab(self.current_module_frame)
56
57 def show_ticket(self):
58     self.destroy_current_module()
59     self.current_module_frame = ttk.Frame(self.root_frame)
60     self.current_module_frame.grid(row=2, column=0, columnspan=3)
61     self.create_ticket_tab(self.current_module_frame)
62
63 def show_movie(self):
64     self.destroy_current_module()
65     self.current_module_frame = ttk.Frame(self.root_frame)
66     self.current_module_frame.grid(row=2, column=0, columnspan=3)
67     self.create_movie_tab(self.current_module_frame)
```

```
File Edit Selection View Go ... ← → Untitled (Workspace)
gp_assign.py x anotherxmp.py kiki.py
IML 208 > gp_assign.py > ...
68
69 # to destroy the current frame for example from membership frame to ticket frame
70 def destroy_current_module(self):
71     if self.current_module_frame:
72         self.current_module_frame.destroy()
73
74 def calculate_age(self):
75     # Retrieve the entered IC number
76     ic_number = self.ic_number_entry.get()
77     category = self.category_entry.get()
78
79     # Extract birth year from IC number
80     birthyear = int(ic_number[:2])
81     currentyear = 24 # Assuming the current year is 2024
82     age = currentyear - birthyear
83
84     # Store the calculated age as an integer attribute in the class
85     self.calculated_age = age
86
87     # Categorize age into groups
88     if age < 13:
89         category = "Child"
90     elif 13 <= age < 18:
91         category = "Teenager"
92     else:
93         category = "Adult"
94
95     # Update the age entry widget with the calculated age and category
96     self.age_entry.delete(0, tk.END)
97     self.age_entry.insert(0, int(age))
98     self.category_entry.delete(0, tk.END)
99     self.category_entry.insert(0, category)
100
```

```
File Edit Selection View Go ... ← → Untitled (Workspace)
gp_assign.py x anotherxmp.py kiki.py
IML 208 > gp_assign.py > ...
101
102 #module 1 (membership)
103
104 def create_membership_tab(self, parent):
105     membership_tab = ttk.Frame(parent)
106
107     # Membership Module Widgets
108     self.ic_number_label = ttk.Label(membership_tab, text="IC Number (generate age):")
109     self.ic_number_entry = ttk.Entry(membership_tab)
110     self.ic_number_label.grid(row=0, column=0, padx=5, pady=5, sticky="w")
111     self.ic_number_entry.grid(row=0, column=1, padx=5, pady=5, sticky="w")
112
113     self.name_label = ttk.Label(membership_tab, text="Name:")
114     self.name_entry = ttk.Entry(membership_tab)
115     self.name_label.grid(row=1, column=0, padx=5, pady=5, sticky="w")
116     self.name_entry.grid(row=1, column=1, padx=5, pady=5, sticky="w")
117
118     self.email_label = ttk.Label(membership_tab, text="E-mail:")
119     self.email_entry = ttk.Entry(membership_tab)
120     self.email_label.grid(row=2, column=0, padx=5, pady=5, sticky="w")
121     self.email_entry.grid(row=2, column=1, padx=5, pady=5, sticky="w")
122
123     self.state_label = ttk.Label(membership_tab, text="State:")
124     self.state_entry = ttk.Entry(membership_tab)
125     self.state_label.grid(row=3, column=0, padx=5, pady=5, sticky="w")
126     self.state_entry.grid(row=3, column=1, padx=5, pady=5, sticky="w")
127
128     self.age_label = ttk.Label(membership_tab, text="Age:")
129     self.age_entry = ttk.Entry(membership_tab)
130     self.age_label.grid(row=4, column=0, padx=5, pady=5, sticky="w")
131     self.age_entry.grid(row=4, column=1, padx=5, pady=5, sticky="w")
132
133     self.category_label = ttk.Label(membership_tab, text="Category:")
134     self.category_entry = ttk.Entry(membership_tab)
135     self.category_label.grid(row=5, column=0, padx=5, pady=5, sticky="w")
136     self.category_entry.grid(row=5, column=1, padx=5, pady=5, sticky="w")
```

```

File Edit Selection View Go ... ← → Untitled (Workspace)
gp_assign.py × anotherxmp.py kikiki.py
IML 208 > gp_assign.py > ...
138
139 calculate_age_button = ttk.Button(membership_tab, text="calculate Age", command=self.calculate_age)
140 submit_button = ttk.Button(membership_tab, text="Submit", command=self.submit_membership)
141 calculate_age_button.grid(row=6, column=0, columnspan=2, pady=5)
142 submit_button.grid(row=7, columnspan=2, pady=10)
143
144 update_button = ttk.Button(membership_tab, text="Update", command=self.update_membership)
145 delete_button = ttk.Button(membership_tab, text="Delete", command=self.delete_membership)
146 update_button.grid(row=8, column=0, pady=5)
147 delete_button.grid(row=8, column=1, pady=5)
148
149 # Add a label to display success message
150 self.success_label = ttk.Label(membership_tab, text="")
151 self.success_label.grid(row=9, column=0, columnspan=2, pady=10)
152
153 membership_tab.grid()
154
155 def submit_membership(self):
156     # Retrieve the entered data from membership module
157     ic_number = self.ic_number_entry.get()
158     name = self.name_entry.get()
159     email = self.email_entry.get()
160     state = self.state_entry.get()
161     age = int(self.age_entry.get())
162     category = self.category_entry.get()
163
164     # Store the collected data in the instance variable
165     self.membership_data = {
166         "IC": ic_number,
167         "Name": name,
168         "Email": email,
169         "State": state,
170         "Age": age,
171         "Category": category
172     }
173

```

```

File Edit Selection View Go ... ← → Untitled (Workspace)
gp_assign.py × anotherxmp.py kikiki.py
IML 208 > gp_assign.py > ...
173
174 sql = "INSERT INTO membership (IC_Number, Name, Email, State, Age, Category) VALUES (%s, %s, %s, %s, %s, %s)"
175 val = (ic_number, name, email, state, age, category)
176 mycursor.execute(sql, val)
177 mydb.commit()
178
179 # Display success message
180 self.success_label.config(text="Your data has been successfully saved")
181
182 def update_membership(self):
183     # Retrieve the entered data from membership module
184     ic_number = int(self.ic_number_entry.get())
185     name = self.name_entry.get()
186     email = self.email_entry.get()
187     state = self.state_entry.get()
188     age = int(self.age_entry.get())
189
190     # Update the collected data in the instance variable
191     self.membership_data = {
192         "IC": ic_number,
193         "Name": name,
194         "Email": email,
195         "State": state,
196         "Age": age,
197     }
198
199     # Update the record in the database
200     sql = "UPDATE membership SET Name=%s, Email=%s, State=%s, Age=%s WHERE IC_Number=%s"
201     val = (name, email, state, age, ic_number)
202     mycursor.execute(sql, val)
203     mydb.commit()
204
205     # Display success message
206     self.success_label.config(text="Your data has been successfully updated")
207

```

```

File Edit Selection View Go ... ← → Untitled (Workspace)
gp_assign.py x anotherxmp.py kiki.py
IML 208 > gp_assign.py > ...
207
208 def delete_membership(self):
209     # Retrieve the entered data from membership module
210     ic_number = self.ic_number_entry.get()
211
212     # Delete the record from the database
213     sql = "DELETE FROM membership WHERE IC_Number=%s"
214     val = (ic_number,)
215     mycursor.execute(sql, val)
216     mydb.commit()
217
218     # Display success message
219     self.success_label.config(text="Your data has been successfully deleted")
220
221     # Clear the entry fields after deletion
222     self.ic_number_entry.delete(0, tk.END)
223     self.name_entry.delete(0, tk.END)
224     self.email_entry.delete(0, tk.END)
225     self.state_entry.delete(0, tk.END)
226     self.age_entry.delete(0, tk.END)
227
228 #-----
229 #module 2 (movie)
230
231 def create_movie_tab(self, parent):
232     movie_tab = ttk.Frame(parent)
233     movieinfo_text = tk.Text(movie_tab, height=8, width=35)
234     movieinfo_text.grid(row=0, column=1, columnspan=7, pady=10)
235     movieinfo_text.insert(tk.END, "Movie & Price: \n\n")
236     movieinfo_text.insert(tk.END, "Trolls: RM 15 \n\n")
237     movieinfo_text.insert(tk.END, "Wish: RM 16 \n\n")
238     movieinfo_text.insert(tk.END, "The Marvels: RM 17 \n\n")
239     movieinfo_text.configure(state='disabled')
240

```

```

File Edit Selection View Go ... ← → Untitled (Workspace)
gp_assign.py x anotherxmp.py kiki.py
IML 208 > gp_assign.py > ...
240
241 # Movie Module Widgets
242 self.movie_name_label = ttk.Label(movie_tab, text="Movie:")
243 self.movie_names = ["Trolls", "Wish", "The Marvels"]
244 self.movie_name_entry = ttk.Combobox(movie_tab, values=self.movie_names, textvariable=self.selected_movie)
245 self.movie_name_label.grid(row=1, column=1, padx=5, pady=5, sticky="w")
246 self.movie_name_entry.grid(row=1, column=2, padx=5, pady=5, sticky="w")
247
248 self.duration_label = ttk.Label(movie_tab, text="Duration:")
249 self.duration_entry = ttk.Entry(movie_tab)
250 self.duration_label.grid(row=2, column=1, padx=5, pady=5, sticky="w")
251 self.duration_entry.grid(row=2, column=2, padx=5, pady=5, sticky="w")
252
253 self.genre_label = ttk.Label(movie_tab, text="Genre:")
254 self.genre_entry = ttk.Entry(movie_tab)
255 self.genre_label.grid(row=3, column=1, padx=5, pady=5, sticky="w")
256 self.genre_entry.grid(row=3, column=2, padx=5, pady=5, sticky="w")
257
258 self.classification_label = ttk.Label(movie_tab, text="Classification:")
259 self.classification_entry = ttk.Entry(movie_tab)
260 self.classification_label.grid(row=4, column=1, padx=5, pady=5, sticky="w")
261 self.classification_entry.grid(row=4, column=2, padx=5, pady=5, sticky="w")
262
263 submit_button = ttk.Button(movie_tab, text="Submit", command=self.submit_movie)
264 submit_button.grid(row=6, columnspan=5, pady=10)
265
266 # Bind the update_movie_info method to the <<ComboboxSelected>> event
267 self.movie_name_entry.bind("<<ComboboxSelected>>", self.update_movie_info)
268
269 # Add a label to display success message
270 self.success_label = ttk.Label(movie_tab, text="")
271 self.success_label.grid(row=8, column=0, columnspan=5, pady=5)
272
273 movie_tab.grid()
274

```

```

File Edit Selection View Go ... ← → Untitled (Workspace)
gp_assign.py x anotherxmp.py kikki.py
IML 208 > gp_assign.py > ...
274
275 def update_movie_info(self, event):
276     selected_movie = self.selected_movie.get()
277
278     # Movie information dictionary
279     movie_info = {
280         "Trolls": {"duration": "93 minutes", "genre": "Musical/Animation", "classification": "PG13"},
281         "Wish": {"duration": "95 minutes", "genre": "Animation/Comedy", "classification": "P12"},
282         "The Marvels": {"duration": "105 minutes", "genre": "Action/Fantasy", "classification": "PG13"}
283     }
284
285     # movie information based on the selected movie
286     if selected_movie in movie_info:
287         info = movie_info[selected_movie]
288         self.duration_entry.delete(0, tk.END)
289         self.duration_entry.insert(0, str(info["duration"]))
290         self.genre_entry.delete(0, tk.END)
291         self.genre_entry.insert(0, info["genre"])
292         self.classification_entry.delete(0, tk.END)
293         self.classification_entry.insert(0, info["classification"])
294
295 def submit_movie(self):
296     # Retrieve the entered data from movie module
297     selected_movie = self.selected_movie.get()
298     duration = self.duration_entry.get()
299     genre = self.genre_entry.get()
300     classification = self.classification_entry.get()
301
302     # Store the collected data in the instance variable
303     self.movie_data = {
304         "Selected Movie": selected_movie,
305         "Duration": duration,
306         "Genre": genre,
307         "Classification": classification
308     }
309

```

```

File Edit Selection View Go ... ← → Untitled (Workspace)
gp_assign.py x anotherxmp.py kikki.py
IML 208 > gp_assign.py > ...
310     # To insert your Data to your database
311     sql = "INSERT INTO movie (Movie_Name, Duration, Genre, Classification) VALUES (%s, %s, %s, %s)"
312     val = (selected_movie, duration, genre, classification)
313     mycursor.execute(sql, val)
314     mydb.commit()
315
316     # Display success message
317     self.success_label.config(text="Movie Added Successfully!")
318
319 #-----
320 #module 3 (ticket)
321
322 def create_ticket_tab(self, parent):
323     ticket_tab = ttk.Frame(parent)
324
325     #Serve as the container for the seat layout.
326     self.seat_layout_frame = tk.Canvas(ticket_tab, width=250, height=70, bg="white")
327     self.seat_layout_frame.grid(row=0, column=0, columnspan=2, pady=10)
328
329     seat_width = 40 # Adjust the width of each seat
330     seat_height = 20 # Adjust the height of each seat
331     rows = 2
332     columns = 5
333

```

```

File Edit Selection View Go ... ← → Untitled (Workspace)
gp_assign.py X anotherxmp.py kiki.py
IML 208 > gp_assign.py CinemaSystemApp > submit_movie
333
334     for row in range(rows):
335         for seat_number in range(1, columns + 1):
336             x1 = 10 + (seat_number - 1) * (seat_width + 7)
337             y1 = 10 + row * (seat_height + 7)
338             x2 = x1 + seat_width
339             y2 = y1 + seat_height
340
341             #10 is for margin # 7 is refererd to padding
342             #x1 for top left corner of the seat
343             #y1 for top side of the canvas
344             #x2 for bottom right corner of the seat(width of the seat)
345             #y2 for bottom right corner of the seat(height of the seat)
346
347             self.seat_layout_frame.create_rectangle(x1, y1, x2, y2, fill="orange")
348             label_text = f"Seat {row * columns + seat_number}"
349             label_x = x1 + seat_width / 2
350             label_y = y1 + seat_height / 2
351
352             #dividing by 2 help in finding the midpoint of the width and height of the seat rectangle
353
354             self.seat_layout_frame.create_text(label_x, label_y, text=label_text, fill="white")
355
356     # Ticket Module Widgets
357     self.ic_number_label = ttk.Label(ticket_tab, text="IC Number:")
358     self.ic_number_entry = ttk.Entry(ticket_tab)
359     self.ic_number_label.grid(row=1, column=0, padx=5, pady=5, sticky="w")
360     self.ic_number_entry.grid(row=1, column=1, padx=5, pady=5, sticky="w")
361
362     self.selected_movie_label = ttk.Label(ticket_tab, text="Selected Movie:")
363     self.selected_movie_entry = ttk.Entry(ticket_tab, state='readonly', textvariable=self.selected_movie)
364     self.selected_movie_label.grid(row=2, column=0, padx=5, pady=5, sticky="w")
365     self.selected_movie_entry.grid(row=2, column=1, padx=5, pady=5, sticky="w")
366

```

```

File Edit Selection View Go ... ← → Untitled (Workspace)
gp_assign.py X anotherxmp.py kiki.py
IML 208 > gp_assign.py CinemaSystemApp > submit_movie
367     self.date_label = ttk.Label(ticket_tab, text="Date (YY/MM/DD):")
368     self.date_entry = ttk.Entry(ticket_tab)
369     self.date_label.grid(row=3, column=0, padx=5, pady=5, sticky="w")
370     self.date_entry.grid(row=3, column=1, padx=5, pady=5, sticky="w")
371
372     self.seat_number_label = ttk.Label(ticket_tab, text="Seat Number:")
373     self.seat_number_combobox = ttk.Combobox(ticket_tab, values=["Seat 1", "Seat 2", "Seat 3", "Seat 4", "Seat 5", "Seat 6", "Seat 7", "Seat 8", "Seat 9", "Seat 10"])
374     self.seat_number_label.grid(row=4, column=0, padx=5, pady=5, sticky="w")
375     self.seat_number_combobox.grid(row=4, column=1, padx=5, pady=5, sticky="w")
376
377     self.pax_label = ttk.Label(ticket_tab, text="Pax:")
378     self.pax_entry = ttk.Entry(ticket_tab)
379     self.pax_label.grid(row=5, column=0, padx=5, pady=5, sticky="w")
380     self.pax_entry.grid(row=5, column=1, padx=5, pady=5, sticky="w")
381
382     submit_button = ttk.Button(ticket_tab, text="Submit", command=self.submit_ticket)
383     submit_button.grid(row=6, columnspan=2, pady=10)
384
385     update_button = ttk.Button(ticket_tab, text="Update", command=self.update_ticket)
386     delete_button = ttk.Button(ticket_tab, text="Delete", command=self.delete_ticket)
387     update_button.grid(row=7, column=0, pady=5)
388     delete_button.grid(row=7, column=1, pady=5)
389
390     # Output Label & result
391     label = tk.Label(ticket_tab, text='RECEIPT', font=("Impact",13))
392     output_label = tk.Label(ticket_tab, text="")
393     label.grid(row=8, columnspan=2, pady=10)
394     output_label.grid()
395
396     # Add a label to display success message
397     self.success_label = tk.Label(ticket_tab, text="")
398     self.success_label.grid(row=9, column=0, columnspan=5, pady=5)
399
400     ticket_tab.grid()
401

```



```

File Edit Selection View Go ... < -> Untitled (Workspace)
gp_assign.py X anotherxmp.py kikki.py
IML 208 > gp_assign.py > CinemaSystemApp > submit_movie

402 def update_ticket(self):
403     # Retrieve the entered data from membership module
404     ic_number = self.ic_number_entry.get()
405     selected_movie = self.selected_movie_entry.get()
406     date = self.date_entry.get()
407     seat_number = self.seat_number_combobox.get()
408     pax = int(self.pax_entry.get())
409
410     # The price below is to defined the value from your selections
411     prices = {
412         "Trolls": 15,
413         "Wish": 16,
414         "The Marvels": 17,
415     }
416
417     # Calculate the total price. This will be derived from your selection (Package, Pack).
418     total_price = (prices[selected_movie] * pax)
419
420
421     # Update the collected data in the instance variable
422     self.ticket_data = {
423         "IC_Number": ic_number,
424         "Selected_Movie": selected_movie,
425         "Date": date,
426         "Seat_Number": seat_number,
427         "Pax": pax,
428         "Price": total_price,
429     }
430
431     # Update the record in the database
432     sql = "UPDATE ticket SET Selected_Movie=%s, Date=%s, Seat_Number=%s, Pax=%s, Price=%s WHERE IC_Number=%s"
433     val = (selected_movie, date, seat_number, pax, total_price, ic_number)
434     mycursor.execute(sql, val)
435     mydb.commit()
436

```

```

File Edit Selection View Go ... < -> Untitled (Workspace)
gp_assign.py X anotherxmp.py kikki.py
IML 208 > gp_assign.py > CinemaSystemApp > submit_movie

436
437     # Display a message indicating the update has been performed
438     messagebox.showinfo("Update", "Ticket information updated successfully")
439
440 def clear_ticket_tab(self):
441     # Clear the entry fields in the ticket tab
442     self.ic_number_entry.delete(0, tk.END)
443     self.selected_movie.set("") # Clear the selected movie
444     self.date_entry.delete(0, tk.END)
445     self.seat_number_combobox.delete(0, tk.END)
446     self.pax_entry.delete(0, tk.END)
447
448 def delete_ticket(self):
449     # Retrieve the entered data from ticket module
450     ic_number = self.ic_number_entry.get()
451
452     # Delete the record from the database
453     sql = "DELETE FROM ticket WHERE IC_Number=%s"
454     val = (ic_number,)
455     mycursor.execute(sql, val)
456     mydb.commit()
457
458     # Display a message indicating the deletion has been performed
459     messagebox.showinfo("Delete", "Ticket information deleted successfully")
460
461     # Clear the entry fields after deletion
462     self.clear_ticket_tab()
463

```

```

File Edit Selection View Go ... < > Untitled (Workspace)
gp_assign.py x anotherxmp.py kiki.py
IML 208 > gp_assign.py > CinemaSystemApp > submit_movie

464 def submit_ticket(self):
465     while True:
466         # Retrieve the entered data from ticket module
467         ic_number = self.ic_number_entry.get()
468         selected_movie = self.selected_movie.get()
469         date = self.date_entry.get()
470         seat_number = self.seat_number_combobox.get()
471         pax = int(self.pax_entry.get())
472
473         # Store the collected data in the instance variable
474         self.ticket_data = {
475             "IC": ic_number,
476             "Selected Movie": selected_movie,
477             "Date": date,
478             "Seat Number": seat_number,
479             "Pax": pax
480         }
481
482         # The price below is to defined the value from your selections
483         prices = {
484             "Trolls": 15,
485             "Wish": 16,
486             "The Marvels": 17,
487         }
488
489         # Calculate the total price
490         total_price = (prices[selected_movie] * pax)
491
492         booking_id = random.randint(100, 999)
493
494         sql = "INSERT INTO ticket (IC_Number, Selected_Movie, Date, Seat_Number, Pax, Price, Booking_ID) VALUES (%s, %s, %s, %s, %s, %s, %s)"
495         val = (ic_number, selected_movie, date, seat_number, pax, total_price, booking_id)
496         mycursor.execute(sql, val)
497         mydb.commit()
498

```

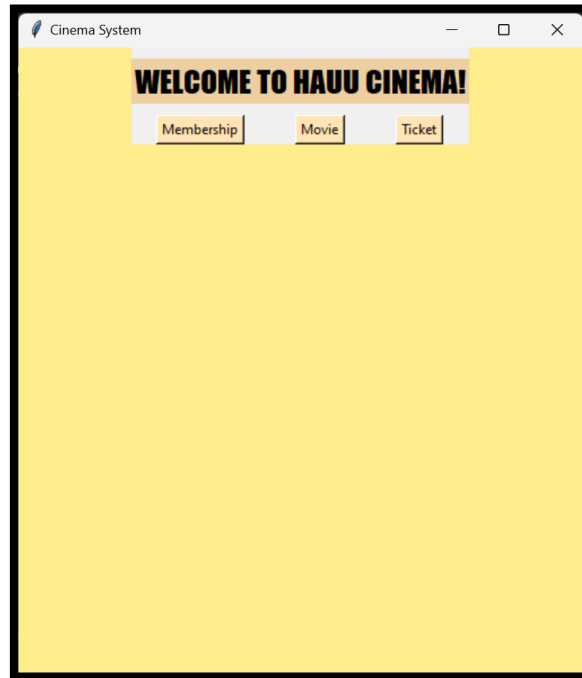
```

498
499         self.success_label.config(text=f"IC Number: {ic_number}\n"
500         f"Selected Movie: {selected_movie}\n"
501         f"Date: {date}\n"
502         f"Seat Number: {seat_number}\n"
503         f"Pax: {pax}\n"
504         f"Price: RM{total_price}\n"
505         f"Booking ID: {booking_id}\n\n"
506         f"THANK YOU!")
507
508         messagebox.showinfo("Booking ID", f"Your Booking ID: {booking_id}")
509
510         break
511
512 if __name__ == "__main__":
513     root = tk.Tk()
514     app = CinemaSystemApp(root)
515     root.mainloop()

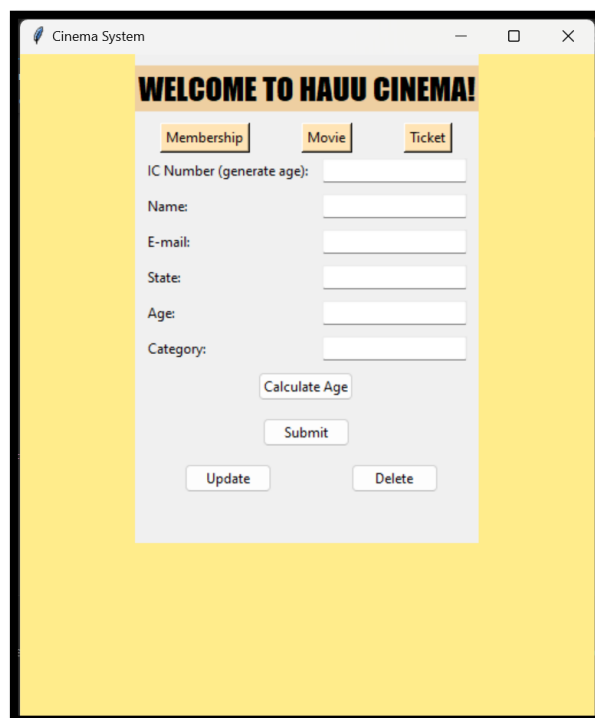
```

6.0 SNAPSHOT OF PROJECT (GUI)

6.1 Main Menu



6.2 Membership Module

A screenshot of a web browser window titled "Cinema System". The page has a yellow background. At the top, there is a black banner with the text "WELCOME TO HAUU CINEMA!" in white. Below the banner, there are three buttons: "Membership", "Movie", and "Ticket". The "Membership" button is highlighted with a black border. Below the buttons, there is a form with the following fields: "IC Number (generate age):", "Name:", "E-mail:", "State:", "Age:", and "Category:". Each field has a corresponding input box. Below the input boxes, there are three buttons: "Calculate Age", "Submit", and "Update". The "Update" button is highlighted with a black border.

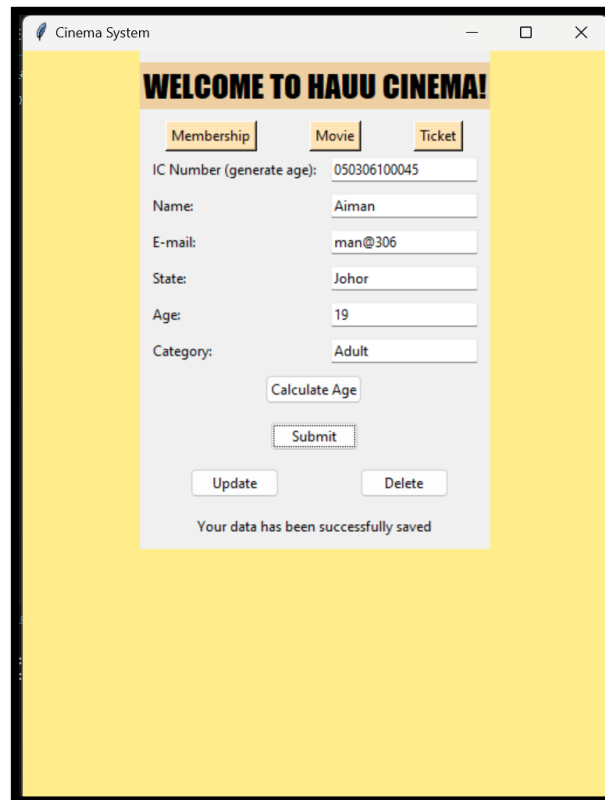
6.3 Movie Module

The screenshot shows a web application window titled "Cinema System". The main heading is "WELCOME TO HAUU CINEMA!". Below the heading are three tabs: "Membership", "Movie", and "Ticket". The "Movie" tab is selected. The content area displays "Movie & Price:" followed by a list of movies and their prices: "Trolls: RM 15", "Wish: RM 16", and "The Marvels: RM 17". Below this list are four input fields: "Movie:" (a dropdown menu), "Duration:", "Genre:", and "Classification:". A "Submit" button is located at the bottom of the form.

6.4 Ticket Module

The screenshot shows the same "Cinema System" window, but the "Ticket" tab is selected. The heading "WELCOME TO HAUU CINEMA!" remains. Below the tabs, there is a grid of 10 seat selection buttons labeled "Seat 1" through "Seat 10". Below the grid are five input fields: "IC Number:", "Selected Movie:", "Date (YY/MM/DD):", "Seat Number:" (a dropdown menu), and "Pax:". There are three buttons: "Submit", "Update", and "Delete". At the bottom, the word "RECEIPT" is displayed in bold.

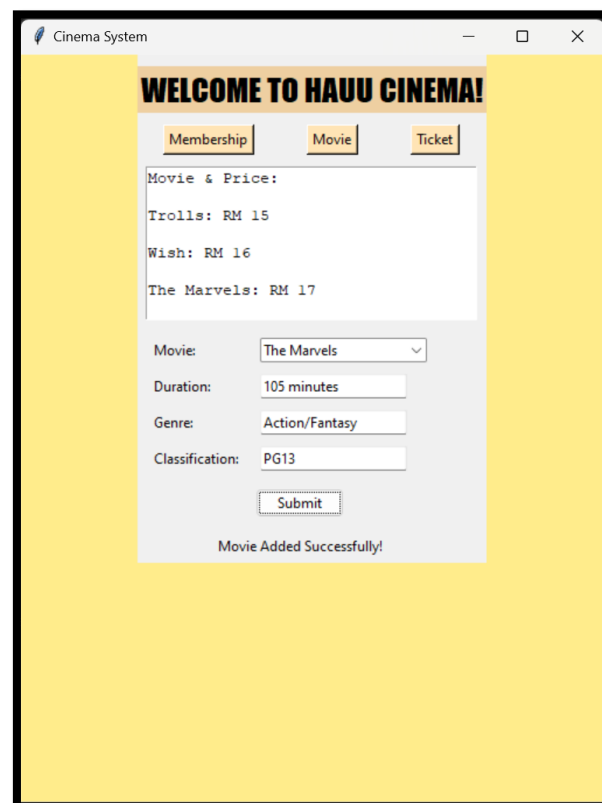
6.5 After insert the data



The screenshot shows a web application window titled "Cinema System". The main heading is "WELCOME TO HAUU CINEMA!". Below the heading are three tabs: "Membership", "Movie", and "Ticket". The "Membership" tab is active. The form contains the following fields and values:

- IC Number (generate age): 050306100045
- Name: Aiman
- E-mail: man@306
- State: Johor
- Age: 19
- Category: Adult

Below the fields are three buttons: "Calculate Age", "Submit", and "Update". The "Submit" button is highlighted with a dashed border. At the bottom of the form, a message states: "Your data has been successfully saved".



The screenshot shows the same "Cinema System" window, but the "Movie" tab is now active. The form displays the following information:

Movie & Price:

- Trolls: RM 15
- Wish: RM 16
- The Marvels: RM 17

Below this list are the following fields and values:

- Movie: The Marvels (dropdown menu)
- Duration: 105 minutes
- Genre: Action/Fantasy
- Classification: PG13

At the bottom of the form is a "Submit" button with a dashed border. A message at the very bottom states: "Movie Added Successfully!".

Cinema System

Ur

WELCOME TO HAUU CINEMA!

Membership

Movie

Ticket

Seat 1

Seat 2

Seat 3

Seat 4

Seat 5

Seat 6

Seat 7

Seat 8

Seat 9

Seat 10

IC Number:

050306100045

Selected Movie:

The Marvels

Date (YY/MM/DD):

24/01/25

Seat Number:

Seat 4

Pax:

2

Submit

Update

Delete

RECEIPT

IC Number: 050306100045

Selected Movie: The Marvels

Date: 24/01/25

Seat Number: Seat 4

Pax: 2

Price: RM34

Booking ID: 754

THANK YOU!

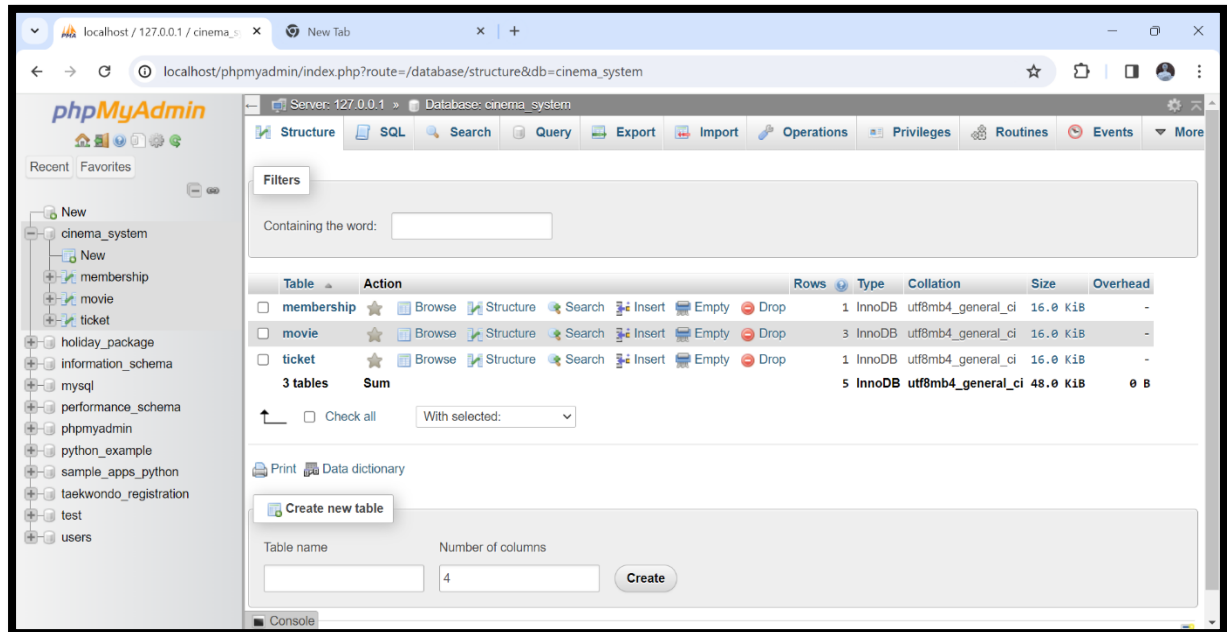
Booking ID

Your Booking ID: 754

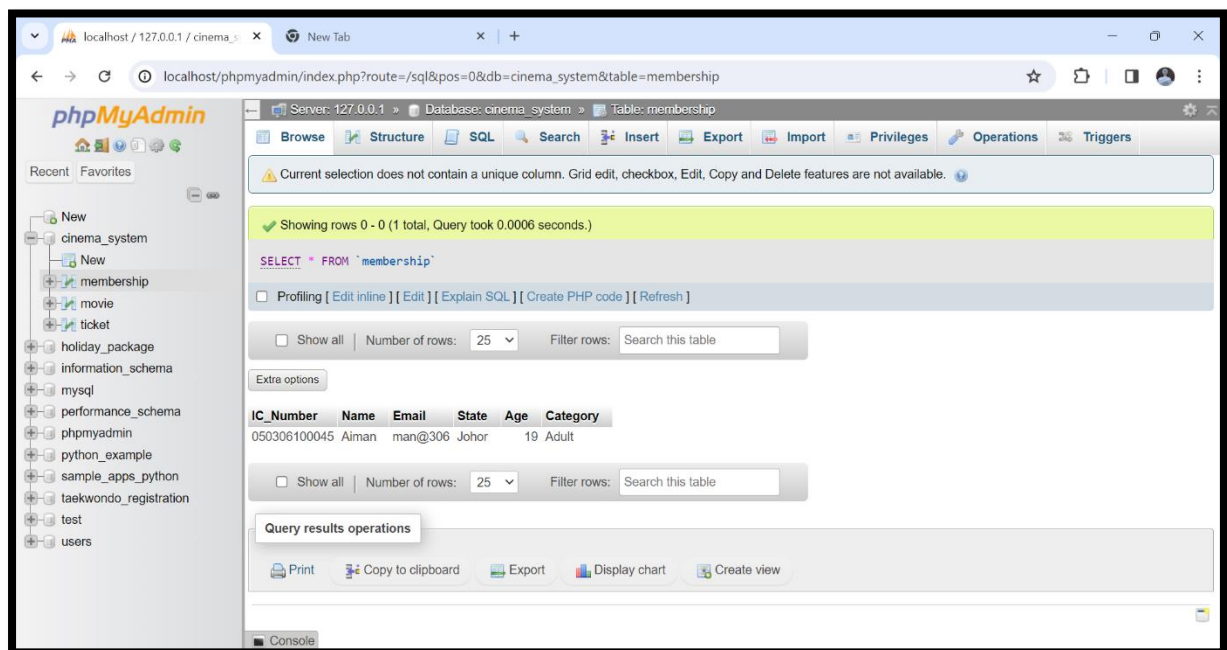
OK

7.0 SNAPSHOT OF DATABASE (XAMPP)

Cinema_System Database



Membership Table



Movie Table

The screenshot shows the phpMyAdmin interface for the 'cinema_system' database. The 'movie' table is selected, and the 'Structure' tab is active. The table has columns: Movie_Name, Duration, Genre, and Classification. The data row shows 'The Marvels' with a duration of 105 minutes, genre of Action/Fantasy, and classification of PG13.

Server: 127.0.0.1 > Database: cinema_system > Table: movie

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

Showing rows 0 - 2 (3 total, Query took 0.0003 seconds.)

`SELECT * FROM `movie``

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

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

Extra options

Movie_Name	Duration	Genre	Classification
The Marvels	105 minutes	Action/Fantasy	PG13

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

Query results operations

Print Copy to clipboard Export Display chart Create view

Ticket Table

The screenshot shows the phpMyAdmin interface for the 'cinema_system' database. The 'ticket' table is selected, and the 'Structure' tab is active. The table has columns: IC_Number, Selected_Movie, Date, Seat_Number, Pax, Price, and Booking_ID. The data row shows a ticket for 'The Marvels' on 24/01/25, seat 4, for 2 pax at a price of 34, with booking ID 754.

Server: 127.0.0.1 > Database: cinema_system > Table: ticket

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

Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

`SELECT * FROM `ticket``

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

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

Extra options

IC_Number	Selected_Movie	Date	Seat_Number	Pax	Price	Booking_ID
050306100045	The Marvels	24/01/25	Seat 4	2	34	754

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

Query results operations

Print Copy to clipboard Export Display chart Create view

8.0 CONCLUSION

To sum up, our team has created an organized movie ticketing system that consists of three unique modules: Membership, Movie, and Ticket. Every module provides a variety of features and capabilities designed to improve consumers' movie-going experiences. These modules are the prime examples of user interface efficiency, enabling movie fans to quickly submit their choices and add to the cinema database. In addition to fulfilling the main requirements of our group project, our Cinema System is an example of how collaborative technologies like Tkinter and MySQL can be used to build efficient and friendly cinema management systems. The smooth flow of data and operations is ensured by proper integration of various modules, which eventually helps to make movie ticketing more effective and pleasurable.

On top of that, adding a while loop to the Ticket module of the system not only improves user experience but also shows how we value customer convenience and adaptability. The loop makes the system easier and user-friendly by allowing users to submit multiple ticket transactions quickly and without having to restart it. This feature demonstrates our commitment to providing a solution that not only satisfies the original assignment objectives but also takes note of realistic user circumstances for a complete cinema booking experience.

In short, our movie ticketing system not only satisfies the group assignment's requirements but is also a flexible and adaptable solution that meets the demands of movie theatre management systems in the future.



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 : ADAM ISKANDAR BIN ZULKARNINE

Matric Number : 2022642664

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 : NURRUL HAIRIENA BINTI ARZA
Matric Number : 2022829308
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 : UMAIRAH WAJIEHAH BINTI MOHAMAD
Matric Number : 2022607006
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 : UMMI KALSUM TAQIAH BINTI SABUDIN

Matric Number : 2022820722

Course Code : IML208

Programme Code :-

Faculty / Campus : UiTM Kampus Sungai Petani