



UNIVERSITI TEKNOLOGI MARA

KEDAH BRANCH

SCHOOL OF INFORMATION SCIENCE

COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS

DIPLOMA IN LIBRARY INFORMATION (CDIM144)

IML208: PROGRAMMING FOR LIBRARIES

INDIVIDUAL ASSIGNMENT

Prepared by:

PRISCILLA ANN VINCENT (2022679716)

GROUP KIM1443F

Prepared for:

SIR AIRUL SHAZWAN BIN NORHASHIMI

Submission date:

4 JANUARY 2024

INDIVIDUAL ASSIGNMENT

PREPARED BY:

PRISCILLA ANN VINCENT (2022679716)

GROUP KIM1443F

IM144 – DIPLOMA IN LIBRARY INFORMATION

SCHOOL OF INFORMATION SCIENCE

COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS

UNIVERSITI TEKNOLOGI MARA (UITM)

KEDAH BRANCH

ACKNOWLEDGEMENT

I like to express my deepest thanks to my IML208 lecturer Sir Airul Shazwan Bin Norhashimi for giving guidance and support me to do this assignment. Thanks to him guidance I get to finish this assignment on time and learn something new from this assignment. Besides, through her motivation I get to complete this assignment in time before the submission.

I would also like to extend my grateful to my classmates for showing me the right way to do this assignment without their help I won't finish this assignment in the right format. I am very thankful to my parents for their prayers, supports, and give advice to me every day so that I can finish this assignment with success.

My special thanks to my roommates for their motivation in finishing this assignment and for accompanying me until late of nights. Without my roommates help maybe I would continue slacker person.

TABLE OF CONTENT

ACKNOWLEDGEMENT	i
1.0 INTRODUCTION	1
2.0 FLOWCHART	2
3.0 SCREENSHOT OF GUI	3
4.0 SCREENSHOT OF CODE	4-6
5.0 SCREENSHOT OF DATABASE	7
6.0 CONCLUSION	8

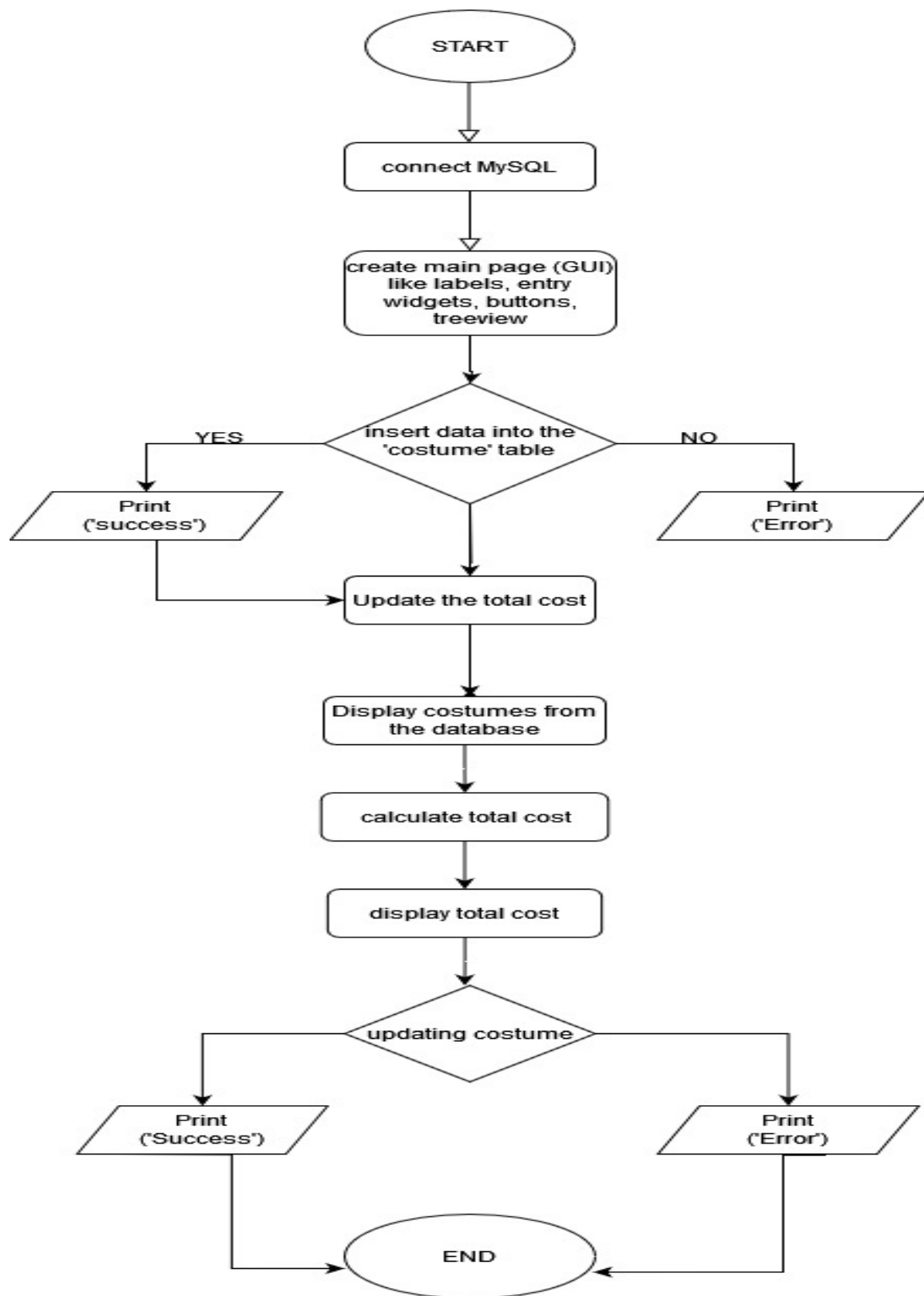
1.0 INTRODUCTION

For this assignment I have choose costume rental as my coding assignment for subject IML208 and this costume rental can help costumer to rent various type of costumes for events. It provides a cost-effective way to purchasing the costume by using this coding. For my calculation I'm using multiply between costume price and costume quantity.

Costume rental businesses typically maintain a diverse inventory of costumes spanning different eras, genres, characters, and themes. These costumes may include fantasy or sci-fi costumes, iconic characters from movies or literature, and more. The range of available costumes can cater to a wide variety of needs and preferences.

It also enables customers to explore different characters or eras, express their creativity, and enhance their event experience. Additionally, costume rental services are often more sustainable and environmentally friendly compared to purchasing new costumes for every occasion.

2.0 FLOWCHART



3.0 SNAPSHOT OF GUI

This is my main page which is the GUI for the costumer to rent the costume that they want from the display costume if they click it will appear with the price too. Its convenient for them to see rather than asking how much is the price for specific costumes.

Costume Rental System

Costume Name:

Costume Price:

Costume Quantity:

Add Costume

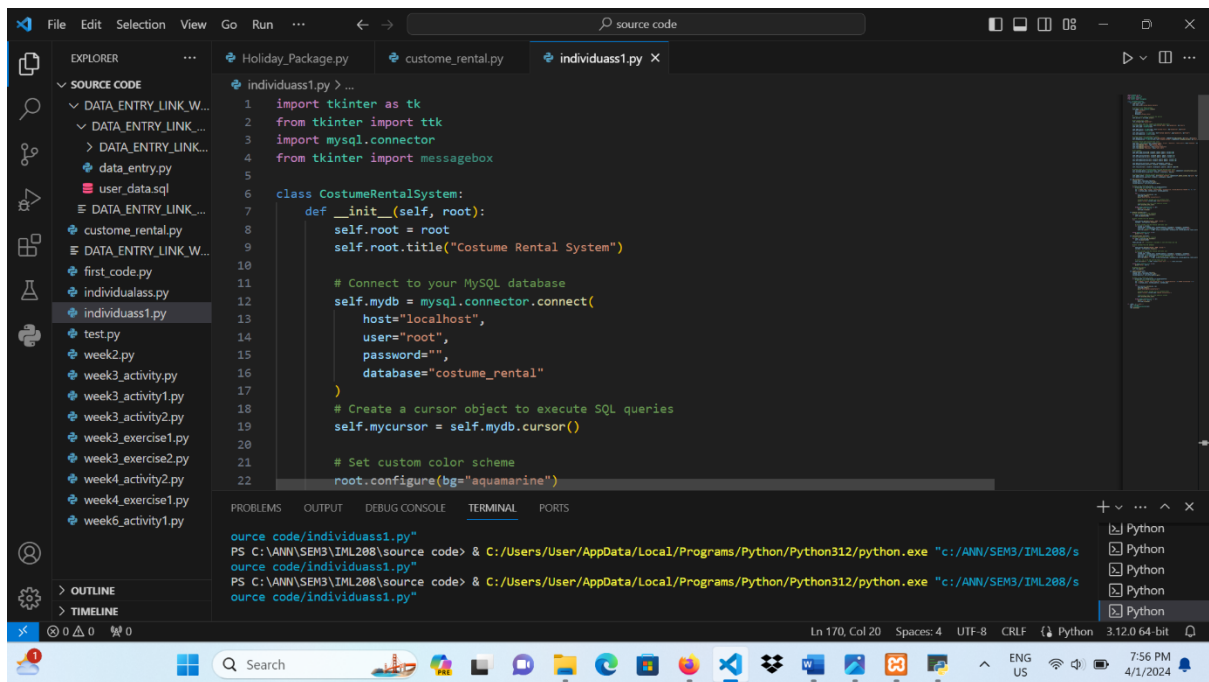
Display Costumes

Costume Name	Costume Price	Costume Quantity	Total Cost
--------------	---------------	------------------	------------

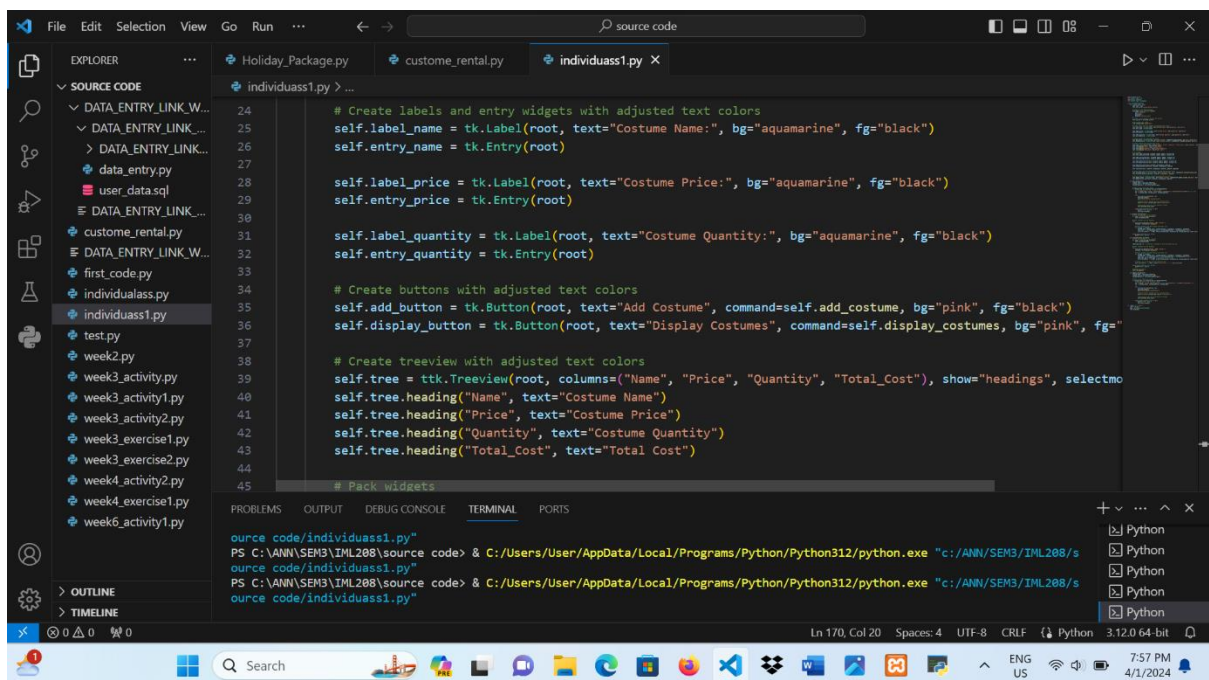
Calculate Total Cost

Update Costume

4.0 SNAPSHOT OF CODE



```
1 import tkinter as tk
2 from tkinter import ttk
3 import mysql.connector
4 from tkinter import messagebox
5
6 class CostumerRentalSystem:
7     def __init__(self, root):
8         self.root = root
9         self.root.title("Costume Rental System")
10
11         # Connect to your MySQL database
12         self.mydb = mysql.connector.connect(
13             host="localhost",
14             user="root",
15             password="",
16             database="costume_rental"
17         )
18
19         # Create a cursor object to execute SQL queries
20         self.mycursor = self.mydb.cursor()
21
22         # Set custom color scheme
23         root.configure(bg="aquamarine")
```



```
24 # Create labels and entry widgets with adjusted text colors
25 self.label_name = tk.Label(root, text="Costume Name:", bg="aquamarine", fg="black")
26 self.entry_name = tk.Entry(root)
27
28 self.label_price = tk.Label(root, text="Costume Price:", bg="aquamarine", fg="black")
29 self.entry_price = tk.Entry(root)
30
31 self.label_quantity = tk.Label(root, text="Costume Quantity:", bg="aquamarine", fg="black")
32 self.entry_quantity = tk.Entry(root)
33
34 # Create buttons with adjusted text colors
35 self.add_button = tk.Button(root, text="Add Costume", command=self.add_costume, bg="pink", fg="black")
36 self.display_button = tk.Button(root, text="Display Costumes", command=self.display_costumes, bg="pink", fg="black")
37
38 # Create treeview with adjusted text colors
39 self.tree = ttk.Treeview(root, columns=("Name", "Price", "Quantity", "Total_Cost"), show="headings", selectmode="browse")
40 self.tree.heading("Name", text="Costume Name")
41 self.tree.heading("Price", text="Costume Price")
42 self.tree.heading("Quantity", text="Costume Quantity")
43 self.tree.heading("Total_Cost", text="Total Cost")
44
45 # Pack widgets
```



```
45 # Pack widgets
46 self.label_name.grid(row=0, column=0, padx=5, pady=5, sticky=tk.W)
47 self.entry_name.grid(row=0, column=1, padx=5, pady=5, sticky=tk.W)
48
49 self.label_price.grid(row=1, column=0, padx=5, pady=5, sticky=tk.W)
50 self.entry_price.grid(row=1, column=1, padx=5, pady=5, sticky=tk.W)
51
52 self.label_quantity.grid(row=2, column=0, padx=5, pady=5, sticky=tk.W)
53 self.entry_quantity.grid(row=2, column=1, padx=5, pady=5, sticky=tk.W)
54
55 self.add_button.grid(row=3, column=0, columnspan=2, pady=10)
56 self.display_button.grid(row=4, column=0, columnspan=2, pady=10)
57
58 self.tree.grid(row=5, column=0, columnspan=2, padx=10, pady=10, ipadx=50)
59
60 # Create button for calculating total cost with adjusted text colors
61 self.calculate_button = tk.Button(root, text="Calculate Total Cost", command=self.calculate_total_cost, bg="p
62 self.calculate_button.grid(row=6, column=0, columnspan=2, pady=10)
63
64 # Create button for updating costume with adjusted text colors
65 self.update_button = tk.Button(root, text="Update Costume", command=self.update_costume, bg="pink", fg="black
66 self.update_button.grid(row=7, column=0, columnspan=2, pady=10)
67
68 def add_costume(self):
69     costume_name = self.entry_name.get()
70     costume_price = self.entry_price.get()
71     costume_quantity = self.entry_quantity.get()
72
73     # Check if all fields are filled
74     if costume_name and costume_price and costume_quantity:
75         # Inserting data into the 'costumes' table
76         sql = "INSERT INTO 'costumes' (Costume_Name, Costume_Price, Costume_Quantity) VALUES (%s, %s, %s)"
77         val = (costume_name, costume_price, costume_quantity)
78
79         try:
80             self.mycursor.execute(sql, val)
81             self.mydb.commit()
82             print("Data inserted successfully!")
83
84             # Display success message (you can customize this)
85             print(f"Costume {costume_name} added successfully!")
86
87             # Recalculate total cost after adding a costume
88             self.calculate_total_cost()
89
90 except mysql.connector.Error as err:
91     print(f"Error: {err}")
92     self.mydb.rollback()
93
94 def display_costumes(self):
95     # Clear existing data in the treeview
96     for item in self.tree.get_children():
97         self.tree.delete(item)
98
99     # Fetch costumes from the database
100     try:
101         self.mycursor.execute("SELECT * FROM 'costume'")
102         costumes = self.mycursor.fetchall()
103
104         # Insert costumes into the treeview with total cost
105         for costume in costumes:
106             costume_name, costume_price, costume_quantity = costume[1], costume[2], costume[3]
107             total_cost = float(costume_price) * int(costume_quantity) if costume_quantity else 0
108             self.tree.insert("", tk.END, values=(costume_name, costume_price, costume_quantity, total_cost))
```

```
68 def add_costume(self):
69     costume_name = self.entry_name.get()
70     costume_price = self.entry_price.get()
71     costume_quantity = self.entry_quantity.get()
72
73     # Check if all fields are filled
74     if costume_name and costume_price and costume_quantity:
75         # Inserting data into the 'costumes' table
76         sql = "INSERT INTO 'costumes' (Costume_Name, Costume_Price, Costume_Quantity) VALUES (%s, %s, %s)"
77         val = (costume_name, costume_price, costume_quantity)
78
79         try:
80             self.mycursor.execute(sql, val)
81             self.mydb.commit()
82             print("Data inserted successfully!")
83
84             # Display success message (you can customize this)
85             print(f"Costume {costume_name} added successfully!")
86
87             # Recalculate total cost after adding a costume
88             self.calculate_total_cost()
89
90 except mysql.connector.Error as err:
91     print(f"Error: {err}")
92     self.mydb.rollback()
93
94 def display_costumes(self):
95     # Clear existing data in the treeview
96     for item in self.tree.get_children():
97         self.tree.delete(item)
98
99     # Fetch costumes from the database
100     try:
101         self.mycursor.execute("SELECT * FROM 'costume'")
102         costumes = self.mycursor.fetchall()
103
104         # Insert costumes into the treeview with total cost
105         for costume in costumes:
106             costume_name, costume_price, costume_quantity = costume[1], costume[2], costume[3]
107             total_cost = float(costume_price) * int(costume_quantity) if costume_quantity else 0
108             self.tree.insert("", tk.END, values=(costume_name, costume_price, costume_quantity, total_cost))
```

```
94 def display_costumes(self):
95     # Clear existing data in the treeview
96     for item in self.tree.get_children():
97         self.tree.delete(item)
98
99     # Fetch costumes from the database
100     try:
101         self.mycursor.execute("SELECT * FROM 'costume'")
102         costumes = self.mycursor.fetchall()
103
104         # Insert costumes into the treeview with total cost
105         for costume in costumes:
106             costume_name, costume_price, costume_quantity = costume[1], costume[2], costume[3]
107             total_cost = float(costume_price) * int(costume_quantity) if costume_quantity else 0
108             self.tree.insert("", tk.END, values=(costume_name, costume_price, costume_quantity, total_cost))
```

```

110         except mysql.connector.Error as err:
111             print(f"Error: {err}")
112
113     def calculate_total_cost(self):
114         # Clean existing data in the treeview
115         for item in self.tree.get_children():
116             self.tree.delete(item)
117
118         total_cost_sum = 0 # Initialize a variable to store the total cost sum
119
120         # Fetch costumes from the database
121         try:
122             self.mycursor.execute("SELECT * FROM 'costume'")
123             costumes = self.mycursor.fetchall()
124
125             # Insert costumes into the treeview with total cost
126             for costume in costumes:
127                 costume_name, costume_price, costume_quantity = costume[1], costume[2], costume[3]
128                 total_cost = float(costume_price) * int(costume_quantity) if costume_quantity else 0
129                 total_cost_sum += total_cost # Accumulate the total cost
130                 self.tree.insert("", tk.END, values=(costume_name, costume_price, costume_quantity, total_cost))
131
132     source code/individuass1.py"
133     PS C:\ANN\SEM3\IML208\source code> & C:/Users/User/AppData/Local/Programs/Python/Python312/python.exe "c:/ANN/SEM3/IML208/s
134     source code/individuass1.py"
135     PS C:\ANN\SEM3\IML208\source code> & C:/Users/User/AppData/Local/Programs/Python/Python312/python.exe "c:/ANN/SEM3/IML208/s
136     source code/individuass1.py"

```

```

132     # Insert a row at the end with the total cost sum
133     self.tree.insert("", tk.END, values=("Total Cost", "", "", total_cost_sum))
134
135     except mysql.connector.Error as err:
136         print(f"Error: {err}")
137
138     # Update the treeview
139     self.tree.update()
140
141     def update_costume(self):
142         costume_name = self.entry_name.get()
143         costume_price = self.entry_price.get()
144         costume_quantity = self.entry_quantity.get()
145
146         # Check if all fields are filled
147         if costume_name and costume_price and costume_quantity:
148             # Updating data in the 'costumes' table
149             sql = "UPDATE 'costume' SET Costume_Price = %, Costume_Quantity = %s WHERE Costume_Name = %s"
150             val = (costume_price, costume_quantity, costume_name)
151
152             try:
153                 self.mycursor.execute(sql, val)
154
155     source code/individuass1.py"
156     PS C:\ANN\SEM3\IML208\source code> & C:/Users/User/AppData/Local/Programs/Python/Python312/python.exe "c:/ANN/SEM3/IML208/s
157     source code/individuass1.py"
158     PS C:\ANN\SEM3\IML208\source code> & C:/Users/User/AppData/Local/Programs/Python/Python312/python.exe "c:/ANN/SEM3/IML208/s
159     source code/individuass1.py"

```

```

148     # Updating data in the 'costumes' table
149     sql = "UPDATE 'costume' SET Costume_Price = %, Costume_Quantity = %s WHERE Costume_Name = %s"
150     val = (costume_price, costume_quantity, costume_name)
151
152     try:
153         self.mycursor.execute(sql, val)
154         self.mydb.commit()
155         print("Data updated successfully!")
156
157         # Display success message (you can customize this)
158         print(f"Costume {costume_name} updated successfully!")
159
160         # Recalculate total cost after adding a costume
161         self.calculate_total_cost()
162
163     except mysql.connector.Error as err:
164         print(f"Error: {err}")
165         self.mydb.rollback()
166
167     if __name__ == "__main__":
168         root = tk.Tk()
169         app = CostumeRentalSystem(root)
170         root.mainloop()
171
172     source code/individuass1.py"

```

5.0 SNAPSHOT OF DATABASE

The screenshot shows the phpMyAdmin interface with the 'costume' table selected. The table structure is displayed with the following columns:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Costume_Name	text	utf8mb4_general_ci		No	None			Change Drop More
2	Costume_Price	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
3	Costume_Quantity	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
4	Total_Cost	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More

Below the table structure, there are options to 'Check all', 'With selected', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', and 'Fulltext'. There is also a 'Print' button and a 'Propose table structure' button. A 'Move columns' section shows 'Add 1' and 'column(s) after Total_Cost' with a 'Go' button. The 'Indexes' section shows 'No index defined!' and a 'Create an index on' section with '1' and 'columns' and a 'Go' button.

The screenshot shows the phpMyAdmin interface with the 'costume' table selected. The query results are displayed for the query 'SELECT * FROM `costume`'. The results show 6 rows of data:

Costume_Name	Costume_Price	Costume_Quantity	Total_Cost
Fairy Costume	RM35	1	
Disney Princess Costume	RM40	1	
Mermaid Costume	RM50	1	
Animal Costume	RM70	1	
Superhero Costume	RM85	1	
Sanrio costume	RM40	1	

Below the query results, there are options to 'Show all', 'Number of rows: 25', and 'Filter rows: Search this table'. There is also a 'Query results operations' section with buttons for 'Print', 'Copy to clipboard', 'Export', 'Display chart', and 'Create view'.

6.0 CONCLUSION

This costume rental system is a simple code from python and use MySQL to retrieve the database from it so that it can connect to calculate the total cost. With display costume button we can easily see what types of costume in the database which is from MySQL and the update costume button also make it easy to update or edit our mistake after adding it into database.

From this assignment, I get to learn many new things that I never do in daily life and I also encounter some problem during running the python code but I keep on trying to find the solutions to fix it. It is a new knowledge to learn how to build a GUI which is the main page of the system and need to connect it with the MySQL so that the database can be used to calculate the total cost for costumes. I have a long way to learn python and I hope I will get better in learning it for future use.



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 : PRISCILLA ANN VINCENT

Matric Number : 2022679716

Course Code : IML208

Programme Code :-

Faculty / Campus : UiTM Kampus Sungai Petani