FACULTY OF COLLEGE OF COMPUTING, INFORMATICS AND MEDIA

UNIVERSITI TEKNOLOGI MARA (UiTM)

MERBOK, KEDAH

DIPLOMA IN LIBRARY INFORMATICS

(CDIM144)

PROGRAMMING FOR LIBRARIES

(IML208)

GROUP PROJECT: ROOM RENTAL

PREPARED BY:

NAME	STUDENT ID
MUHAMMAD DANISH ALFIAN BIN MOHD ZULLKIFLI	2022625348
MUHAMMAD HAFIZUL ZAHEEN BIN MULIADI	2022661936
NURUL SYAFIQAH BINTI MAHAYUDDIN	2022863286
SYASYA SYAMIMI BINTI AZMAN	2022878606

GROUP: KCDIM1443B

PREPARED FOR:

ENCIK AIRUL SHAZWAN BIN NORSHAHIMI

SUBMISSION DATE: 17TH JANUARY 2024

GROUP ASSIGNMENT

"ROOM RENTAL"

MUHAMMAD DANISH ALFIAN BIN MOHD ZULLKIFLI

2022625348

MUHAMMAD HAFIZUL ZAHEEN BIN MULIADI

2022661936

NURUL SYAFIQAH BINTI MAHAYUDDIN

2022863286

SYASYA SYAMIMI BINTI AZMAN

2022878606

FACULTY OF INFORMATION SCIENCE STUDIES,

COLLEGE OF COMPUTING,

INFORMATICS AND MEDIA STUDIES

17TH JANUARY 2024

ACKNOWLEDGEMENT

Greetings,

First of all, we would like to express our gratitude for being given good health during the period of completing this given assignment. With God’s permission, we was able to complete it in the given period. This work was done with our own efforts and the help from lecturer and the other friends of us.

Secondly, we would like to show respect to our lecturer, sir Airul, because he always gave us guidance in completing this group assignment and we was very thankful to him. In the meantime, we was fascinated by the way he taught in his own unique styles. He is very professional and sporting. W also want to thank him for teaching us and our other coursemate in this course.

Last but not least, we would like to thanks to our loving family for always support us until now. Because of them, we became more confident and ready to achieve something. we hope all the efforts we have made are not buried just like that while we are at this university. Getting good grades is not easy. To be honest, after making this assignment, we have learned many new things that we did not know before. we promise to use this knowledge in the future.

TABLE OF CONTENT

1.0 INTRODUCTION

Room rental refers to the practice of leasing or renting out a room within a property to an individual or group for a specific period. This arrangement is common in various settings, including residential, commercial, or hospitality environments. Room rentals are prevalent for a variety of reasons, such as providing temporary accommodation for travelers, students, or individuals seeking short-term living arrangements.

In residential contexts, homeowners or tenants may choose to rent out a spare room to generate additional income or share living expenses. This can be particularly appealing in urban areas where housing costs are high. Additionally, room rentals can offer a more affordable housing option for individuals looking for temporary or flexible living arrangements.

Room rental arrangements usually involve a formal agreement outlining the terms and conditions of the rental, including rental amount, length of stay, house furnishings, house rules and any other relevant details. The rise of online platforms has simplified the process of finding and booking room rentals, making it easier for both property owners and tenants to connect and transact.

Whether for residential or commercial purposes, room rentals play an important role in providing flexible housing solutions and contributing to the sharing economy. They cater to diverse needs, from affordable housing alternatives to convenient short-term stays, offering a versatile and dynamic approach to accommodation.

For this project, we had make the room rental system because this system is a comprehensive and user-friendly platform designed to streamline the process of renting individual rooms within a property. Whether you are a landlord looking to efficiently manage your rental spaces or a tenant searching for a convenient and transparent way to secure accommodations, this system aims to meet your needs.

1.1 OBJECTIVE

The objectives of a room rental system can vary based on the specific goals and requirements of the platform or service. However, here are some common objectives that a **room rental system** might aim to achieve:

- **Efficient Property Management:**

Streamline the process of managing rental properties, including room availability, bookings, and lease agreements.

- **User-Friendly Interface**

Develop an intuitive and user-friendly interface for both landlords and tenants to easily navigate the system.

- **Automated Booking and Reservations**

Implement an automated booking system that allows tenants to easily check room availability, make reservations, and complete the booking process online.

- **Transparent Pricing and Policies**

Ensure transparency in pricing and rental policies to build trust between landlords and tenants.

- **Payment Processing**

Facilitate secure and efficient online payment processing for rental transactions, including rent payments, security deposits, and other related fees.

When it comes to renting a room, whether you are a landlord or someone looking for a tenant, it is important to set clear objectives to ensure a successful and mutually beneficial arrangement.

1.2 PROBLEM STATEMENT FOR ROOM RENTAL SYSTEM

Renting rooms within a property should be a seamless and transparent process, benefiting both landlords and tenants. However, the current landscape presents several challenges that hinder the efficiency and convenience of room rentals.

The Room Rental System aims to address the following key problems:

1. **Lack of Centralized Management**

Many landlords struggle with the manual organization of room availability, lease agreements, and tenant details. This often leads to inefficiencies, confusion, and a lack of a centralized system for effective property management.

2. **Insecure Payment Processing**

Security concerns surrounding online payment processing can deter tenants from making payments through digital platforms. Landlords face challenges in providing a secure payment environment, impacting the overall trustworthiness of the rental system.

3. **Communication Gaps**

The lack of efficient communication tools within existing systems results in delays and misunderstandings between landlords and tenants. Improved communication channels are necessary to facilitate timely and clear interaction.

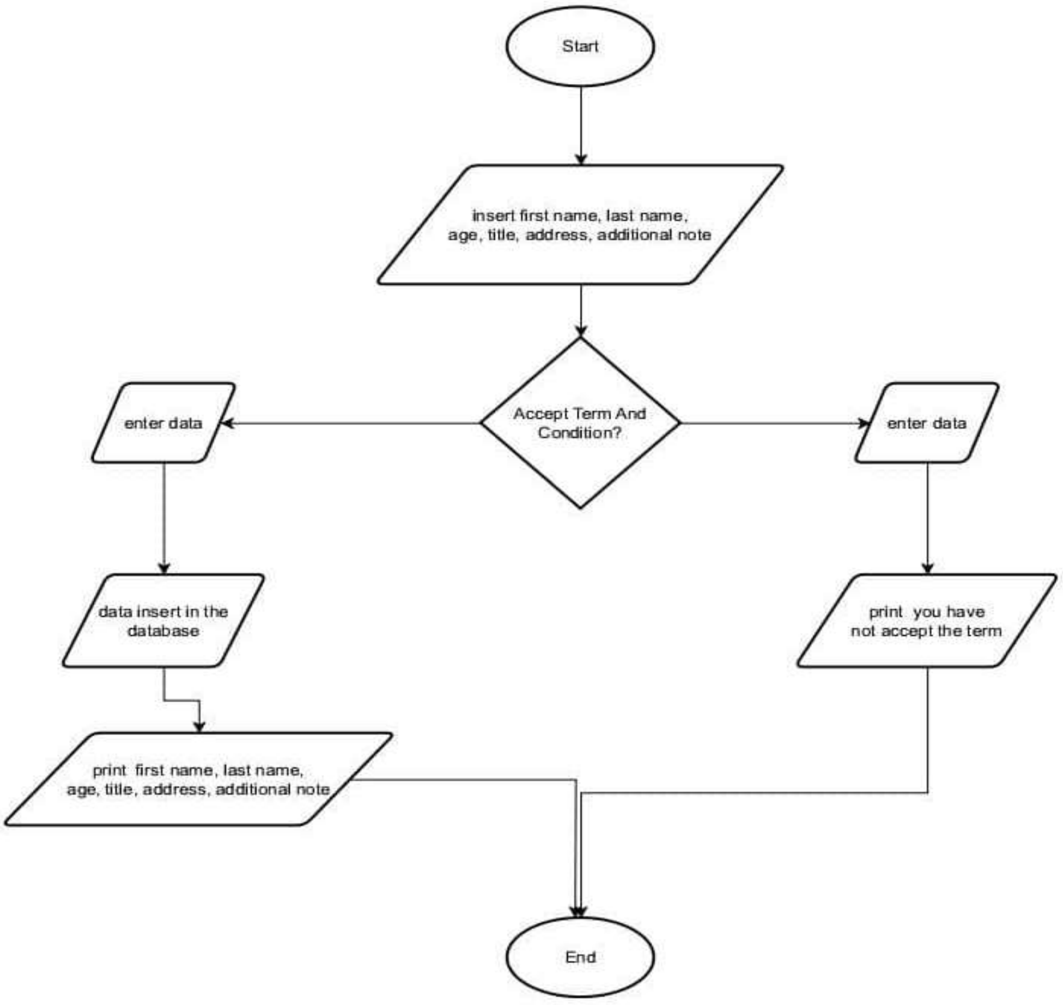
4. **Complex Booking Procedures**

Existing booking procedures for renting individual rooms are often convoluted and time-consuming. Tenants may face difficulties in checking real-time availability, making reservations, and completing the booking process online.

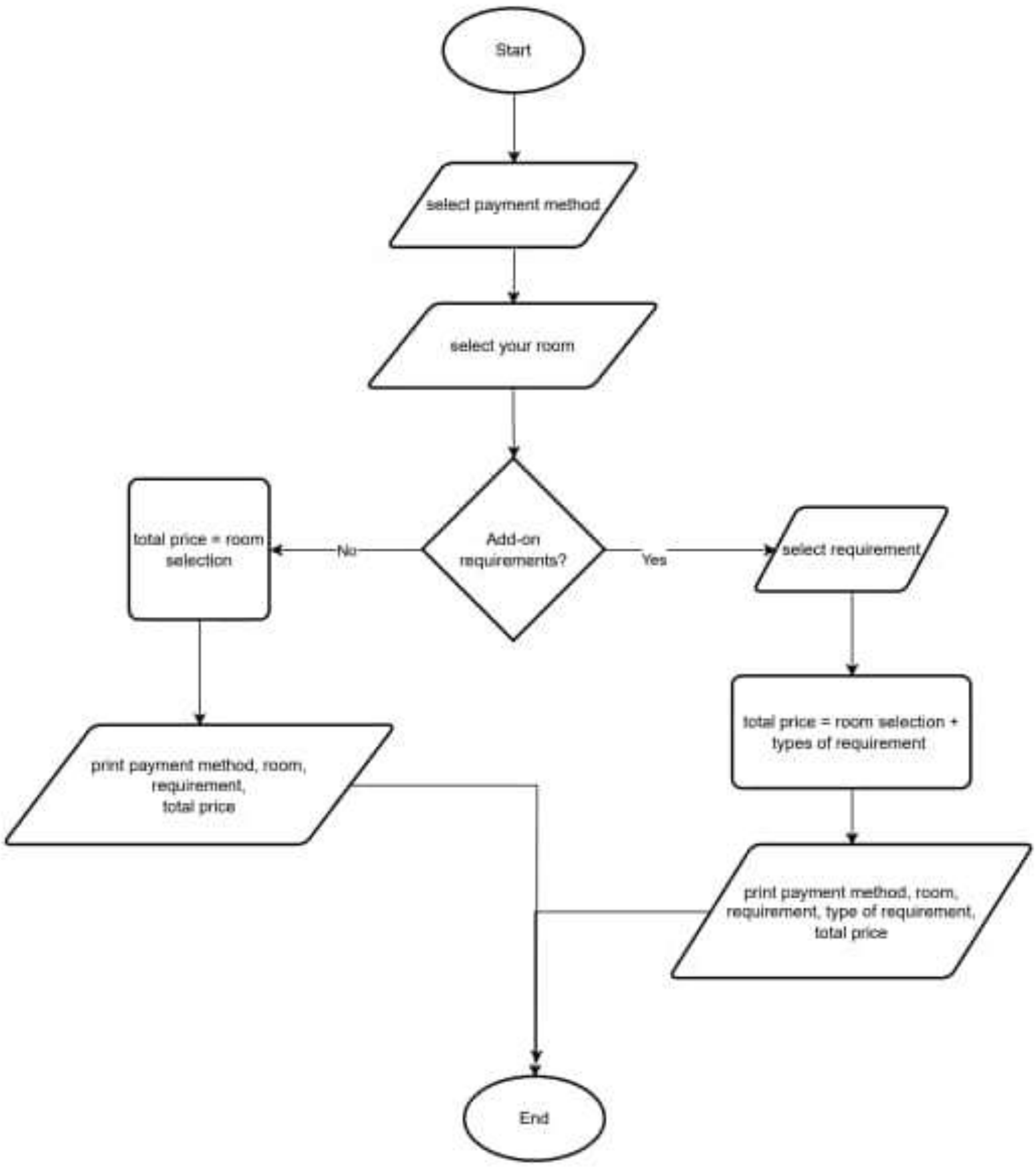
By addressing these challenges, the Room Rental System aims to revolutionize the room rental experience, offering a solution that enhances efficiency, transparency, and security for both landlords and tenants. This platform seeks to overcome existing shortcomings and provide a user-friendly, centralized, and reliable system for managing room rentals.

2.0 FLOWCHART

2.1 REGISTRATION FLOWCHART


Start
insert first name, last name, age, title, address, additional note
Accept Term And Condition?
enter data
enter data
data insert in the database
print you have not accept the term
print first name, last name, age, title, address, additional note
End

2.2 PAYMENT DETAIL PAYMENT


Start
select payment method
select your room
Add-on requirements?
No
Yes
total price = room selection
select requirement
total price = room selection + types of requirement
print payment method, room, requirement, total price
print payment method, room, requirement, type of requirement, total price
End

3.0 CODE FOR ROOM RENTAL

```
import tkinter as tk
from tkinter import *
from PIL import Image, ImageTk
from tkinter import ttk
import mysql.connector
from tkinter import messagebox



def open_about():
    about_window = Toplevel(root)
    about_window.title("About Us")
    about_window.geometry("1198x6000")



    image= ImageTk.PhotoImage(Image.open("house.png"))

    #img = img.resize((400, 200), Image.ANTIALIAS)

    photo=tk.Label(about_window, image=image, bg='white')
    photo.pack(fill='x' )



    about=tk.Label(about_window, text="\nRENT A ROOM. your hassle-free solution for affordable and comfortable room rentals.\nWe specialize in simplifying your accommodation search, offering a diverse range of\noptions for travelers, students, and professionals. Our mission is to prioritize your\ncomfort and provide a seamless renting experience. Choose Rent A Room for easy\nand affordable living solutions. Your ideal room is just a click away!\n\n\n"
            ,bg='#FEF0CA', font=('century', 17)
    )

    about.pack(fill='both')
    about_window.mainloop()
```

```
#user registration

def open_registration():
    registration_window = Toplevel(root)
    registration_window.title("Registration")
    registration_window.geometry("750x575")
    mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="room_rental"
)

    mycursor = mydb.cursor()

    def update_entry_data():
        mycursor.execute("SELECT * FROM registration")
        entries = mycursor.fetchall()

        # Clear existing items in the listbox
        entry_data.delete(0, tk.END)

        # Insert fetched data into the listbox
        for entry in entries:
            data = f'First Name: {entry[0]}, Last Name: {entry[1]}, Title: {entry[2]}, Age:
{entry[3]},\n' \
                f'Address: {entry[4]}\nNote: {entry[5]}'
            entry_data.insert(tk.END, data)
    def enter_data():
        first=name_entry.get()
        last=lastname_entry.get()
        title=title_combobox.get()
        age=age_entry.get()
        address=address_entry.get("1.0", tk.END).strip()
        note=note_desc_entry.get("1.0", tk.END).strip()
        status= check_status_var.get()

        if status != 'Accept':
            messagebox.showwarning("Terms not Accepted", "Please accept the terms and
conditions.")
            return

        print('First Name:',first, 'Last Name:',last,
'Title:',title,'Age:',age,'Address:',address,'Additional Note:',note )
        data = f'First Name: {first}, Last Name: {last}, Title: {title}, Age: {age},\n' \
            f'Address: {address}\nNote: {note}'

        entry_data.insert(tk.END, data)

        sql = "INSERT INTO registration (first, last, age,title , address, note) VALUES (%s, %s,
%s,%s,%s, %s)"
        val = (first, last, title, age, address,note,)
        mycursor.execute(sql, val)
        mydb.commit()

    def deleting():
        selected_index = entry_data.curselection()
        if selected_index:
            selected_item = entry_data.get(selected_index)
```

```
  # Extract first name from the selected item (you may need to adjust this based on your
data structure)
          first_name = selected_item.split(":")[1].split(",")[0].strip()


          # Delete from the database
          delete_sql = "DELETE FROM registration WHERE first = %s"
          mycursor.execute(delete_sql, (first_name,))
          mydb.commit()


          # Clear the selected item from the listbox
          entry_data.delete(selected_index)


   def update():
      selected_index = entry_data.curselection()
      if selected_index:
          selected_item = entry_data.get(selected_index)
          # Extract first name from the selected item (you may need to adjust this based on
your data structure)
          first_name = selected_item.split(":")[1].split(",")[0].strip()


          # Retrieve the existing data from the database
          select_sql = "SELECT * FROM registration WHERE first = %s"
          mycursor.execute(select_sql, (first_name,))
          existing_entry = mycursor.fetchone()


          # Update the data with new values
          first = name_entry.get()
          last = lastname_entry.get()
          title = title_combobox.get()
          age = age_entry.get()
          address = address_entry.get("1.0", tk.END).strip()
          note = note_desc_entry.get("1.0", tk.END).strip()
# Update the data in the database
          update_sql = "UPDATE registration SET first=%s, last=%s, age=%s, title=%s,
address=%s, note=%s WHERE first=%s"
          val = (first, last, age, title, address, note, first_name)
          mycursor.execute(update_sql, val)
          mydb.commit()

          # Update the data in the listbox
          updated_data = f'First Name: {first}, Last Name: {last}, Title: {title}, Age: {age},\n' \
                      f'Address: {address}\nNote: {note}'
          entry_data.delete(selected_index)
          entry_data.insert(selected_index, updated_data)



#book entry frame
   entry_frame= tk.LabelFrame(registration_window, text="REGISTRATION FORM",
pady=30, padx=25, font= ( "Arial Black",  ), bg="#B87C4C", height=90)
   entry_frame.pack(fill='both')

   title_label=tk.Label(entry_frame, text='FIRST NAME', font=( 'Bahnschrift', 12),
bg='#B87C4C')
   title_label.grid(row=0, column=0)

   author_label=tk.Label(entry_frame, text='LAST NAME', font=( 'Bahnschrift',
12),bg='#B87C4C')
   author_label.grid(row=0, column=1)

   name_entry=tk.Entry(entry_frame,  bg='#FFF6E8', )
   lastname_entry=tk.Entry(entry_frame, bg='#FFF6E8')
   name_entry.grid(row=1, column=0)
   lastname_entry.grid(row=1, column=1)
   title_label=tk.Label(entry_frame, text='TITLE',font=( 'Bahnschrift', 12), bg='#B87C4C')
   title_combobox=ttk.Combobox(entry_frame, values=['Mr.', 'Mrs.', 'Dr.', 'Datuk', 'Datin', 'Tan Sri', 'Puan Sri',],)
   title_label.grid(row=0, column=4, padx=50)
   title_combobox.grid(row=1, column=4, padx=100)
#genre_combobox.set('select genre')



   age_label=tk.Label(entry_frame, text='AGE',font=( 'Bahnschrift', 12), bg='#B87C4C', padx=100)
   age_entry=tk.Entry(entry_frame, bg='#FFF6E8')
   age_label.grid(row=0, column=3)
   age_entry.grid(row=1,column=3)

   address_label=tk.Label(entry_frame, text='ADDRESS', font=( 'Bahnschrift', 12),bg='#B87C4C')
   address_entry=tk.Text(entry_frame, height=4, width=25, bg='#FFF6E8')
   address_label.grid(row=6, column=0, columnspan=2)
   address_entry.grid(row=7,column=0, columnspan=2)


   note_desc=tk.Label(entry_frame, text='ADDITIONAL NOTE',font=( 'Bahnschrift', 12), bg='#B87C4C')
   note_desc_entry=tk.Text(entry_frame, height=4, width=25, bg='#FFF6E8')
   note_desc.grid(row=6, column=3, columnspan=5)
   note_desc_entry.grid(row=7, column=3, columnspan=5)

   check_status_var=tk.StringVar()
   check_button=tk.Checkbutton(entry_frame, text='Agree To Term And Condition',font=( 'Bahnschrift', 12), variable=check_status_var, onvalue='Accept' , offvalue='Decline', bg='#B87C4C')
   check_button.grid(row=8, column=0, sticky='ew', columnspan=2)
```

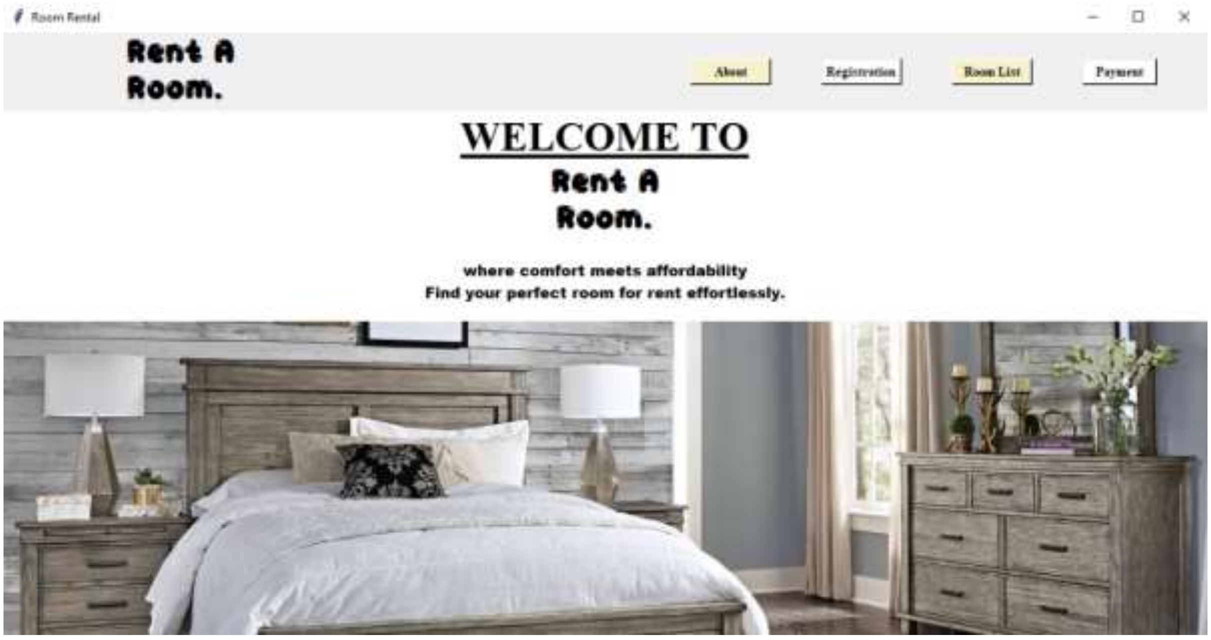
```
#button

    button=tk.Button(entry_frame, text='Enter Data', pady=5, bg='#D17C30', font=(
'Bahnschrift'),width=21 ,command= enter_data )
    button.grid(row=8, column=3,columnspan=5 )


    entry_data=tk.Listbox(entry_frame, height=10, width=82, bg='#FFF6E8' )
    entry_data.grid(row=9, column=0, columnspan=5, )


    update_entry_data()


    edit=tk.Button(entry_frame, text='Edit', width=21,  bg='#D17C30', command= update)
    edit.grid(row=10, column=0, columnspan=3)


    delete=tk.Button(entry_frame, text='Delete', width=21,  bg='#D17C30', command=
deleting)
    delete.grid(row=10, column=3, columnspan=5)



    for widget in entry_frame.winfo_children():
        widget.grid_configure(padx=10, pady=5)
        button.grid_configure(padx=30, pady=30)



    registration_window.mainloop()




def open_room_list():
    list_window= Toplevel(root)
    list_window.title('Room List')
    list_window.geometry('800x600')
    title = tk.Label(list_window, text='ROOM LIST', font=('times new roman', 40))
    title.pack()


# Frame for Listbox
    box = tk.Frame(list_window)
    box.pack()


# Listbox
    room_list = tk.Listbox(box, width=100, height=10, selectmode=tk.EXTENDED, fg='black',
font=('times new roman', 12, 'bold'),bd=5,relief="sunken")
    room_list.pack()


    # Sample room entries
    room_list.insert(tk.END, "Choice Of Room and Price\n\n")
    room_list.insert(tk.END, "Room A: 2 Single be,d  \nAdd-on Requirements: Stand
Fan,Bookshelves, Iron Board,   \nPrice: RM 140\n\n")
    room_list.insert(tk.END, "Room B: 1 Queen bed,  \nAdd-on Requirements: Stand
Fan,Bookshelves, Iron Board,  \nPrice: RM 200\n\n")
    room_list.insert(tk.END, "Room C: 1 Queen bed, 1 Single bed,  \nAdd-on Requirements:
Stand Fan,Bookshelves, Iron Board,  \nPrice: RM 300\n\n")
    room_list.configure(state='disable')


    room_list = tk.Listbox(box, width=50, height=5, selectmode=tk.EXTENDED, fg='black',
font=('times new roman', 12, 'bold'),bd=4,relief="groove")
    room_list.pack()
    room_list.insert(tk.END, "Types Of Requirements and Price\n\n")
    room_list.insert(tk.END, "Stand Fan  \nPrice: RM 30\n\n")
    room_list.insert(tk.END, "Bookshelves  \nPrice: RM 30\n\n")
    room_list.insert(tk.END, "Iron Board  \nPrice: RM 10\n\n")
    room_list.configure(state='disable')


    list_window.mainloop()
```

```
#payment database



def open_payment():
    payment_window = Toplevel(root)
    payment_window.title("Payment")
    payment_window.geometry("800x500")

     #CONNECT TO MYSQL DATABASE
    mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="room_rental"
    )



    mycursor = mydb.cursor()

    def collect_data():
        Payment_Method = Payment_Method_combobox.get()
        Room_Selection = Room_Selection_combobox.get()
        Add_On_Requirements = Add_On_Requirements_combobox.get()
        Types_Of_Requirements = Types_Of_Requirements_combobox.get()

        Price_Per_Room = {
            "Room A": 140,
            "Room B": 200,
            "Room C": 380,
        }
        Types_Of_Requirements_Price = {
            "Stand Fan": 30,
            "Bookshelves": 50,
            "Iron Board": 10,
        }

        # Default price if the type is not found
        Rent_Total_Price = 0

        # Check different cases using if, elif, and else
        if Types_Of_Requirements == "Stand Fan":
            Rent_Total_Price = Price_Per_Room[Room_Selection] +
int(Types_Of_Requirements_Price["Stand Fan"])

        elif Types_Of_Requirements == "Bookshelves":
            Rent_Total_Price = Price_Per_Room[Room_Selection] +
int(Types_Of_Requirements_Price["Bookshelves"])

        elif Types_Of_Requirements == "Iron Board":
            Rent_Total_Price = Price_Per_Room[Room_Selection] +
int(Types_Of_Requirements_Price["Iron Board"])

        else:
            Rent_Total_Price = Price_Per_Room[Room_Selection]

        # TO INSERT DATA TO DATABASE
        sql = "INSERT INTO payment_detail (Payment_Method, Room_Selection,
Add_On_Requirements,Types_Of_Requirements , Rent_Total_Price) VALUES (%s, %s,
%s,%s,%s)"
        val = (Payment_Method, Room_Selection, Add_On_Requirements,
Types_Of_Requirements, Rent_Total_Price)
        mycursor.execute(sql, val)
        mydb.commit()

        # Displaying the collected data
        output_label.configure(text=f"Payment_Method: {Payment_Method}, Room_Selection:
{Room_Selection}, Add_On_Requirements: {Add_On_Requirements},
Type_Of_Requirements:{Types_Of_Requirements_Price} Rent_Total_Price:
RM{Rent_Total_Price}")


    # GUI Interface


    payment_window.configure(bg="#AFC1D0")


    frame = tk.Frame(payment_window)

    frame.pack()


    label = tk.Label(payment_window, text="PAYMENT DETAIL", font=("Segoe Script", 15,
"bold"),bg="#C3E0E5",bd=4,relief="groove")

    label.pack(ipadx=10, ipady=20, fill='x', )


    frame = tk.Frame(payment_window, bg='#AFC1D0')

    frame.pack()


    # Saving customer payment

    Customer_payment_detail_frame = tk.LabelFrame(frame, text="RENTAL PAYMENT",
font=("Bahnschrift SemiLight Condensed",20),bg="#AFC1D0",bd=3,relief="solid", )

    Customer_payment_detail_frame.grid(row=0, column=0, pady=30)


    Payment_Method_label = tk.Label(Customer_payment_detail_frame, text="Payment
Method",font=("Bahnschrift SemiLight Condensed",20),bg="#AFC1D0",bd=3,relief="ridge")

    Payment_Method_combobox = ttk.Combobox(Customer_payment_detail_frame,
values=["Debit/Credit Card", "Cash"])

    Payment_Method_label.grid(row=6, column=0)

    Payment_Method_combobox.grid(row=6, column=4)



    for widget in Customer_payment_detail_frame.winfo_children():

        widget.grid_configure(padx=15, pady=10)
    Customer_payment_detail_frame = tk.LabelFrame(frame, text="SELECTION AND ADD-
ON", font=("Bahnschrift SemiLight Condensed",20 ),bg="#AFC1D0",bd=3,relief="solid")


 Customer_payment_detail_frame.grid(row=10, column=0)


    Room_Selection_label = tk.Label(Customer_payment_detail_frame, text="Select Your
Room",font=("Bahnschrift SemiLight Condensed",20),bg="#AFC1D0",bd=3,relief="ridge")

    Room_Selection_combobox = ttk.Combobox(Customer_payment_detail_frame,
values=["Room A", "Room B", "Room C"])

    Room_Selection_label.grid(row=12, column=0)

    Room_Selection_combobox.grid(row=12, column=4)


    Add_On_Requirements_label = tk.Label(Customer_payment_detail_frame, text="Add-on
Requirements",font=("Bahnschrift SemiLight Condensed",
20),bg="#AFC1D0",bd=3,relief="ridge")

    Add_On_Requirements_combobox = ttk.Combobox(Customer_payment_detail_frame,
values=["Yes", "No"])

    Add_On_Requirements_label.grid(row=14, column=0)

    Add_On_Requirements_combobox.grid(row=14, column=4)


    Types_Of_Requirements_label = tk.Label(Customer_payment_detail_frame, text="Select
Requirements",font=("Bahnschrift SemiLight
Condensed",20),bg="#AFC1D0",bd=3,relief="ridge")

    Types_Of_Requirements_combobox = ttk.Combobox(Customer_payment_detail_frame,
values=["Stand Fan", "Bookshelves", "Iron Board"])

    Types_Of_Requirements_label.grid(row=16, column=0)

    Types_Of_Requirements_combobox.grid(row=16, column=4)


    for widget in Customer_payment_detail_frame.winfo_children():

        widget.grid_configure(padx=15, pady=10)



    frame1=tk.Frame(payment_window, bg='#C3E0E5')

    frame1.pack(fill='both')
    # Calculate button

    Total_button = tk.Button(frame1, text="Calculate
Total!",bg="#C3E0E5",bd=3,relief="raised", command=collect_data)

    Total_button.pack(pady=15)


    label = tk.Label(frame1, text='Rental Total Price', font=("Courier New", 15, "underline",
"bold"),bg="#C3E0E5",bd=3,relief="groove")

    label.pack(ipadx=10, ipady=10)

    output_label = tk.Label(payment_window, text="",font=("MV
Boli",10),bg="#D4F1F4",bd=5,relief="sunken")

    output_label.pack()


    payment_window.mainloop()


def register_action():

    print("Registration Button Clicked")


def payment_action():

    print("Payment Button Clicked")


root = Tk()

root.title('Room Rental')

root.geometry('1198x600')


header = Frame(root)

header.pack()


title = Label(header, text='Rent A                          \nRoom.                          ',
font=('Valoon', 25, ))

title.pack(side='left', padx=120)


about = Button(header, text='About', width=10, bg='#F6F2CB', font=('times new roman', 10,
'bold'), command=open_about)

about.pack(side='left')
```

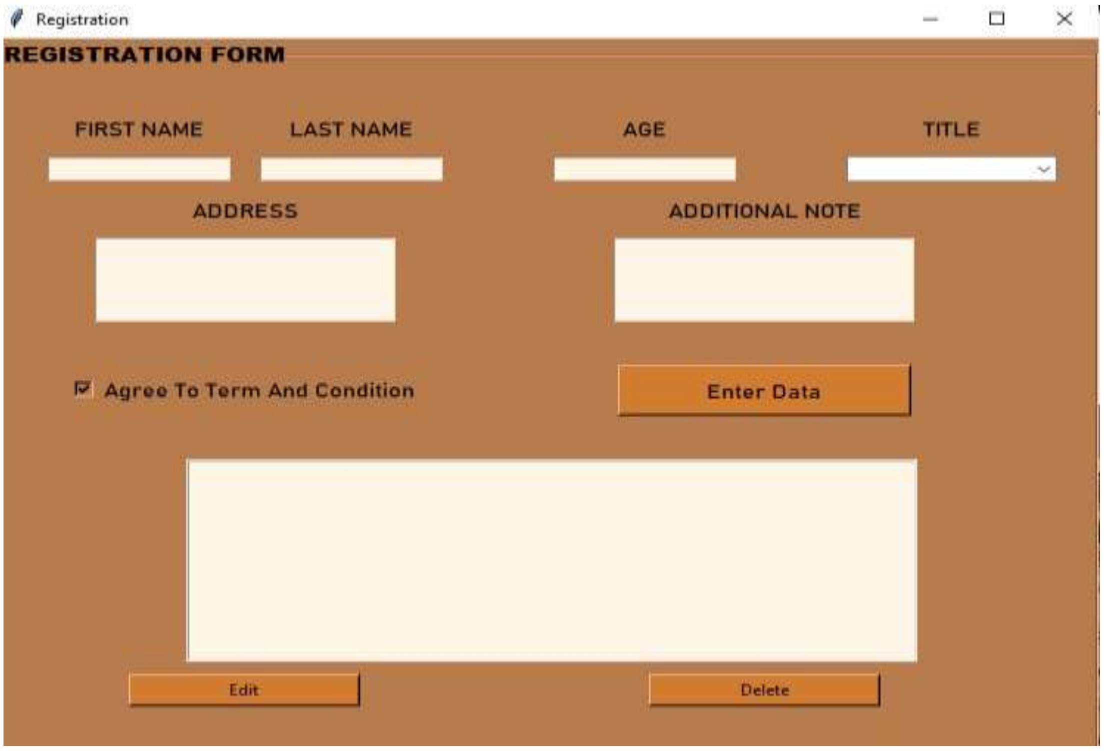
```
registration_button = Button(header, text='Registration', width=10, bg='WHITE', font=('times new roman', 10, 'bold'), command=open_registration)

registration_button.pack(side='left', padx=50)


list_button = Button(header, text='Room List', width=10, bg='#F6F2CB', font=('times new roman', 10, 'bold'), command=open_room_list)

list_button.pack(side='left')


payment_button = Button(header, text='Payment', width=10, bg='WHITE', font=('times new roman', 10, 'bold'), command=open_payment)

payment_button.pack(side='right', padx=50)


mid = Frame(root, bg='#FFFFFF')

mid.pack()


welcome = Label(mid, text='WELCOME TO', font=('times new roman', 30, 'bold', 'underline'), bg='#FFFFFF')

welcome.pack()


name = Label(mid, text='Rent A\nRoom.', font=('valoon', 25), bg='#FFFFFF')

name.pack()


desc = Label(mid, text='where comfort meets affordability\nFind your perfect room for rent effortlessly.', font=('arial black', 11), bg='#FFFFFF')

desc.pack(pady=15)


img = ImageTk.PhotoImage(Image.open("8d425aa19f74daa5d1663e936adc.jpeg"))

photo = Label(mid, image=img)

photo.pack()


root.mainloop()
```

4.0 GUI FOR ROOM RENTAL SYSTEM

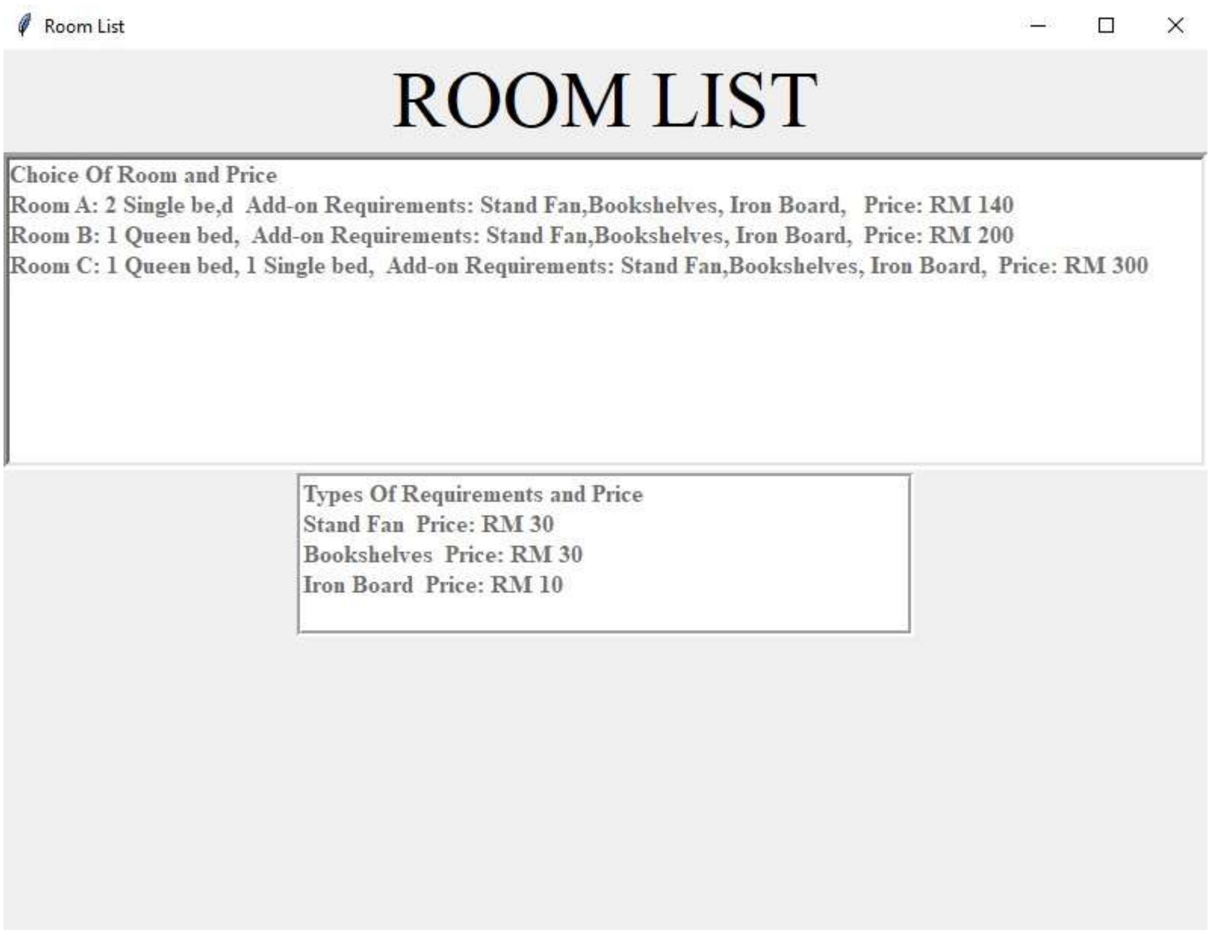
4.1 MAIN INTERFACE


Room Rental
Rent A Room.
About
Registration
Room List
Payment
WELCOME TO
Rent A Room.
where comfort meets affordability
Find your perfect room for rent effortlessly.

4.2 REGISTRATION GUI



4.3 ROOM LIST



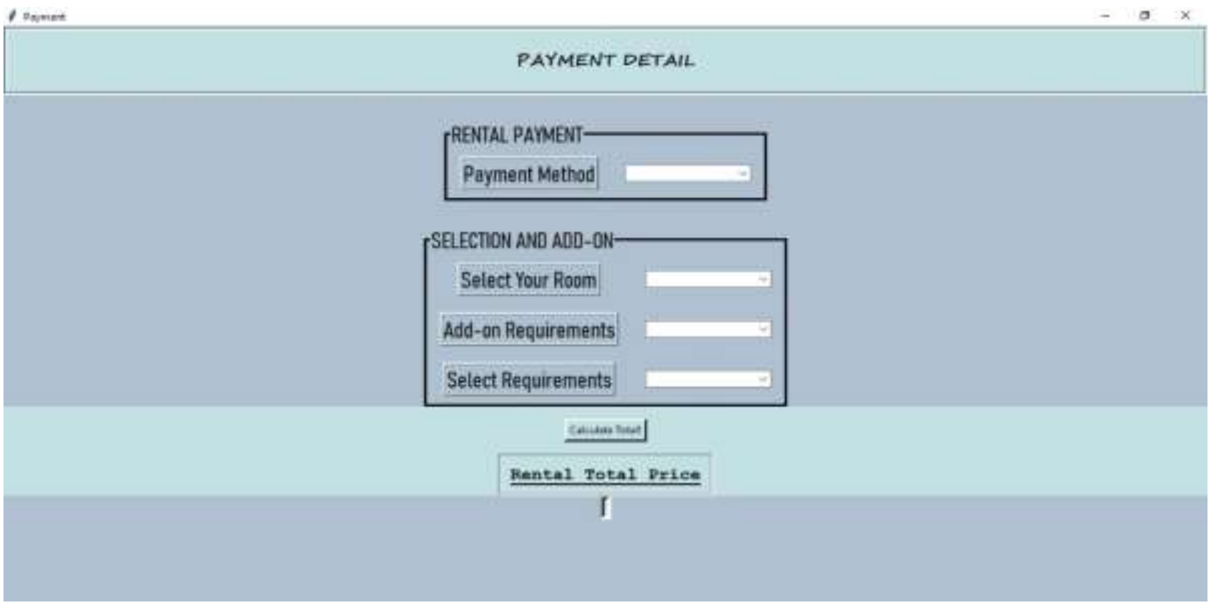
Room List

ROOM LIST

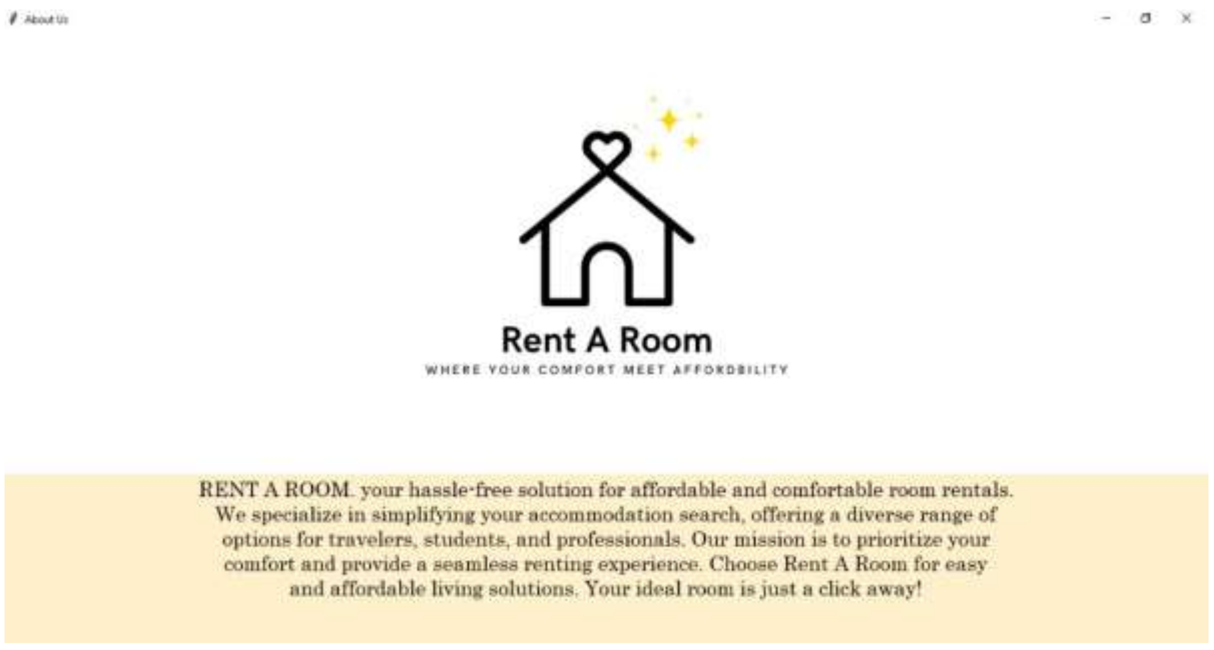
Choice Of Room and Price
Room A: 2 Single be,d Add-on Requirements: Stand Fan,Bookshelves, Iron Board, Price: RM 140
Room B: 1 Queen bed, Add-on Requirements: Stand Fan,Bookshelves, Iron Board, Price: RM 200
Room C: 1 Queen bed, 1 Single bed, Add-on Requirements: Stand Fan,Bookshelves, Iron Board, Price: RM 300

Types Of Requirements and Price
Stand Fan Price: RM 30
Bookshelves Price: RM 30
Iron Board Price: RM 10

4.4 PAYMENT DETAIL



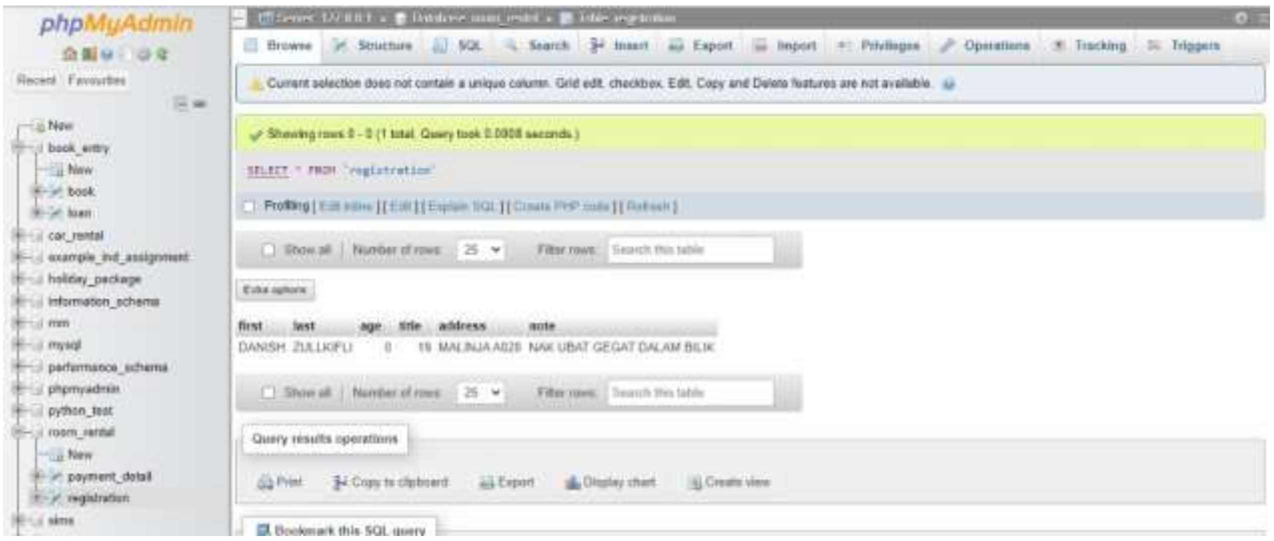
4.5 ABOUT GUI



RENT A ROOM. your hassle-free solution for affordable and comfortable room rentals. We specialize in simplifying your accommodation search, offering a diverse range of options for travelers, students, and professionals. Our mission is to prioritize your comfort and provide a seamless renting experience. Choose Rent A Room for easy and affordable living solutions. Your ideal room is just a click away!

5.0 DATABASE FOR ROOM RENTAL SYSTEM

5.1 DATABASE BROWSE FOR REGISTRATION



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

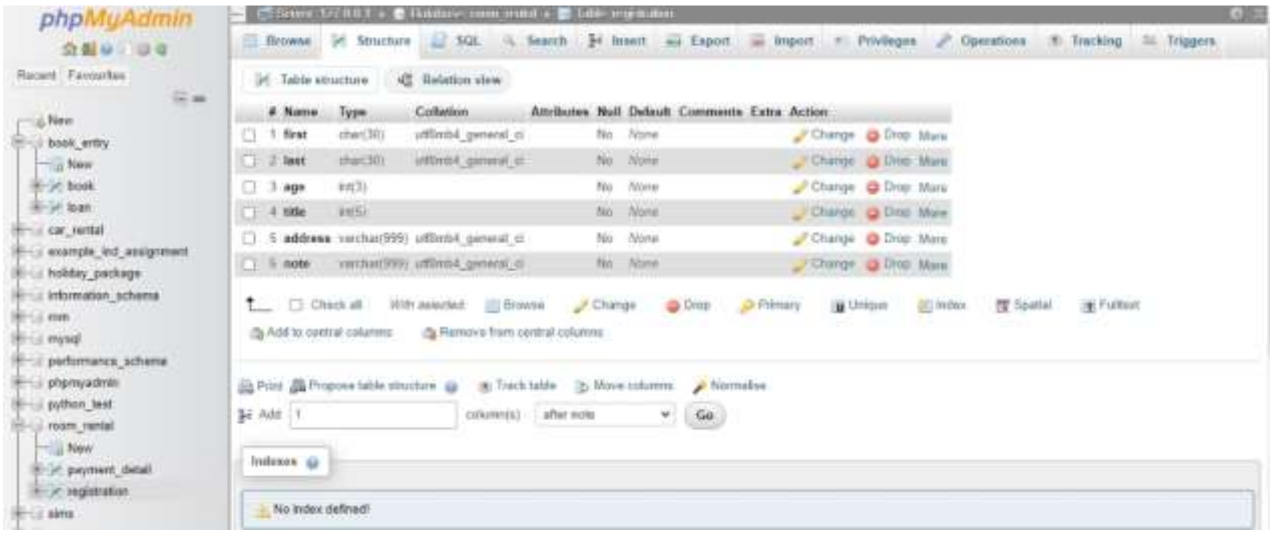
Showing rows 0 - 0 (1 total, Query took 0.0008 seconds.)

```
SELECT * FROM `registration`
```

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

first	last	age	title	address	note
DANISH	ZULLKIFLI	0	19	MALINJA A028	NAK UBAT GEGAT DALAM BILIK

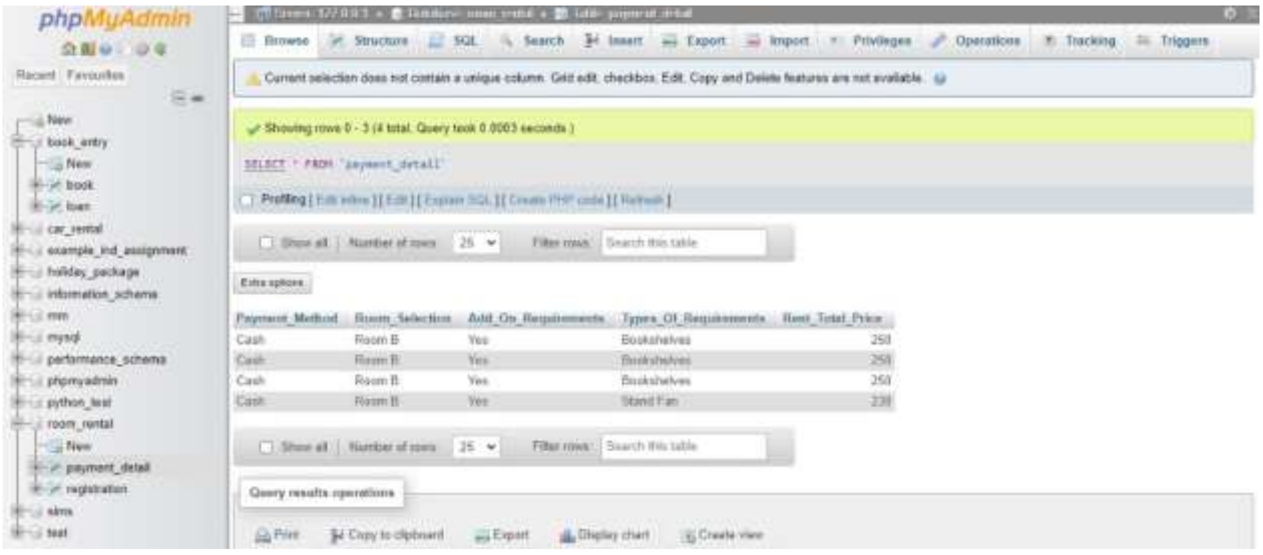
5.2 DATABASE STRUCTURE FOR REGISTRATION



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	first	char(30)	utf8mb4_general_ci		No	None			Change Drop More
2	last	char(30)	utf8mb4_general_ci		No	None			Change Drop More
3	age	int(3)			No	None			Change Drop More
4	title	int(5)			No	None			Change Drop More
5	address	varchar(999)	utf8mb4_general_ci		No	None			Change Drop More
6	note	varchar(999)	utf8mb4_general_ci		No	None			Change Drop More

No index defined!

5.3 DATABASE BROWSE FOR PAYMENT DETAIL



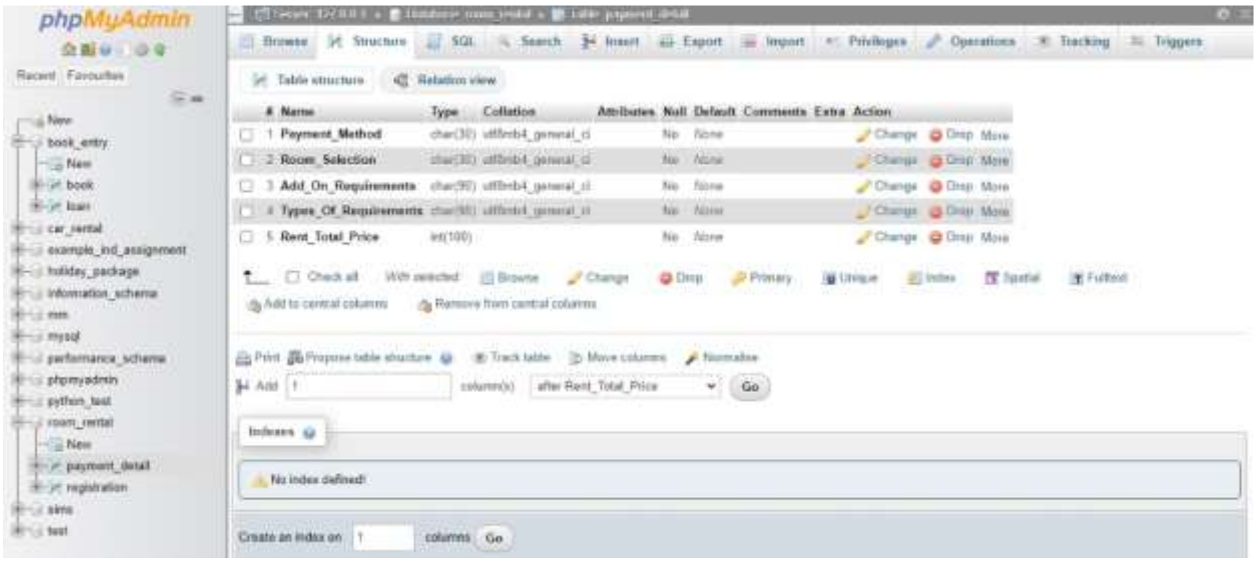
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

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

```
SELECT * FROM `payment_detail`
```

Payment_Method	Room_Selection	Add_On_Requirements	Types_Of_Requirements	Rent_Total_Price
Cash	Room B	Yes	Bookshelves	250
Cash	Room B	Yes	Bookshelves	250
Cash	Room B	Yes	Bookshelves	250
Cash	Room B	Yes	Stand Fan	230

5.4 DATABASE STRUCTURE FOR PAYMENT DETAIL



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Payment_Method	char(30)	utf8mb4_general_ci		No	None			Change Drop More
2	Room_Selection	char(30)	utf8mb4_general_ci		No	None			Change Drop More
3	Add_On_Requirements	char(90)	utf8mb4_general_ci		No	None			Change Drop More
4	Types_Of_Requirements	char(90)	utf8mb4_general_ci		No	None			Change Drop More
5	Rent_Total_Price	int(100)			No	None			Change Drop More

No index defined!

6.0 CONCLUSION

In conclusion, the Room Rental System represents a significant step forward in addressing the challenges associated with renting individual rooms within a property. By providing a comprehensive and user-friendly platform, this system aims to transform the room rental experience for both landlords and tenants. Through the incorporation of key features, such as centralized property management, streamlined booking procedures, and transparent communication channels, the Room Rental System seeks to enhance efficiency and foster trust in the rental process.

The system's commitment to clear pricing and policies, along with secure payment processing, addresses common pain points related to transparency and financial transactions. By offering tools for tenant screening, maintenance issue resolution, and legal compliance, the platform aims to create a secure and well-managed living environment for all parties involved. Furthermore, the Room Rental System recognizes the importance of data-driven decision-making and provides landlords with reporting and analytics tools. This empowers property owners to gain insights into property performance, occupancy rates, and financial metrics, enabling informed decision-making for property optimization.

In essence, the Room Rental System strives to simplify and streamline the room rental experience, making it more accessible, secure, and efficient. By doing so, the platform seeks to foster positive relationships between landlords and tenants, ultimately contributing to a more harmonious and successful room rental ecosystem. As the system continues to evolve and adapt to the changing needs of the rental market, it stands as a promising solution to the challenges faced by both property owners and tenants in the realm of room rentals.