



اُنِيْوَرْسِيْتِيْ تِكْنُوْلُوْجِيْ مَارَا
UNIVERSITI
TEKNOLOGI
MARA

**COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS
UNIVERSITI TEKNOLOGI MARA (UiTM)
CAWANGAN KEDAH**

**DIPLOMA IN LIBRARY INFORMATICS
(CDIM144)
PROGRAMMING FOR LIBRARY
(IML 208)**

TITLE REPORT: PATIENT CLINIC REGISTRATION

PREPARED BY:

MUHAMMAD HAFIZUL ZAHEEN BIN MULIADI (2022661936)

GROUP: CDIM1443B

PREPARED FOR:

SIR AIRUL SHAZWAN BIN NORSHAHIMI

SUBMISSION DATE: WEEK 12

TITLE PAGE

TITLE REPORT: PATIENT CLINIC REGISTRATION

NAME: MUHAMMAD HAFIZUL ZAHEEN BIN MULIADI

MATRIC NO: 2022661936

CLASS: CDIM1443B

DIPLOMA IN LIBRARY INFORMATICS

(CDIM144)

UNIVERSITI TEKNOLOGI MARA (UiTM)

CAWANGAN KEDAH

SUBMISSION DATE: WEEK 12

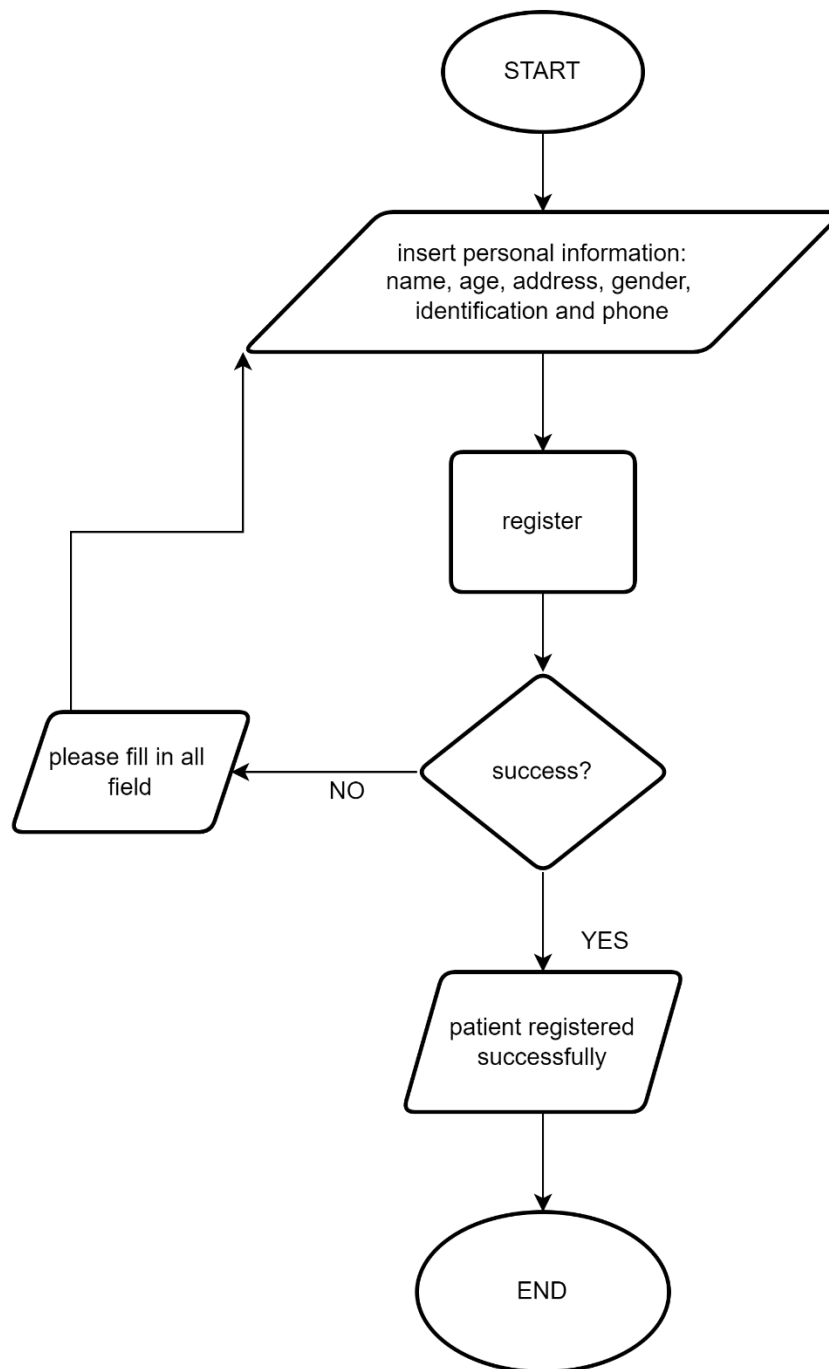
INTRODUCTION

This assignment aims to develop a Patient Clinic Registration System using programming concepts to enhance healthcare accessibility, efficacy, and usability. The system will provide a user-friendly interface for patients and clinic administrators, ensuring seamless registration processes. The project will transcend geographical barriers and prioritize scalability and security to ensure data privacy. The interface will be adaptive to patient-clinic interactions and accommodate evolving healthcare needs. The system will also be fortified with stringent security measures to protect patient information confidentiality and privacy. This project aims to bridge the gap between technology and humanitarian service, enhancing healthcare services for countless individuals in need.

HOW THE SYSTEM WORK

The system going to work with when a user fills up their information in the GUI and then the information is going to appear in the database linked. The project is mean to be for some people who want get check up in clinic early. This project also have a benefit for the clinic organization because they will have a organize work.

FLOW CHART



PYTHON CODE

```
assignment.py > create_patient
1  import tkinter as tk
2  from tkinter import messagebox
3  import mysql.connector
4
5  mydb = mysql.connector.connect(
6      host="localhost",
7      user="root",
8      password="",
9      database="patient_clinic_registration"
10 )
11
12 # Create a cursor object to execute SQL queries
13 mycursor = mydb.cursor()
14
15 # Temporary patient data store
16 patients = []
17
18 def create_patient():
19     name = name_entry.get()
20     age = age_entry.get()
21     gender = gender_entry.get()
22     address_label = address_entry.get()
23     identification = identification_entry.get()
24     phone = phone_entry.get()
25
26     print("name:", name)
27     print("age:", age)
28     print("gender:", gender)
29     print("address :", address_label)
30
31     print("id:", identification)
32     print("phone:", phone)
33
34     sql = "INSERT INTO patient_registration (name, age, gender, address_label, identification, phone) VALUES (%s, %s, %s, %s, %s, %s)"
35     val = (name, age, gender, address_label, identification, phone)
36     mycursor.execute(sql, val)
37     mydb.commit()
38
39     # Basic validation
40     if name and age and gender and identification and phone:
41         patients.append({"Name": name, "Age": age, "Gender": gender, "ID": identification, "Phone": phone})
42         messagebox.showinfo("Success", "Patient Registered Successfully!")
43         clear_entries()
44         update_patient_list()
45     else:
46         messagebox.showerror("Error", "Please fill in all fields.")
47
48 def clear_entries():
49     name_entry.delete(0, tk.END)
50     age_entry.delete(0, tk.END)
51     gender_entry.delete(0, tk.END)
52     address_entry.delete(0, tk.END)
53     identification_entry.delete(0, tk.END)
54     phone_entry.delete(0, tk.END)
55
56 def show_patients():
57     if patients:
58         patient_info = "\n".join([f"Name: {patient['Name']}, Age: {patient['Age']}, Gender: {patient['Gender']}, Address: {patient['Address']}, ID: {patient['ID']}, Phone: {patient['Phone']}" for patient in patients])
59         messagebox.showinfo("Patient List", patient_info)
```

```

59         else:
60             messagebox.showinfo("Patient List", "No Patients Registered Yet.")
61
62     def clear_patients():
63         global patients
64         patients = []
65         update_patient_list()
66         messagebox.showinfo("Success", "All Patients Cleared.")
67
68     def edit_patient():
69         selection = patient_listbox.curselection()
70         if selection:
71             selected_patient = patients[selection[0]]
72             edit_window = tk.Tk()
73             edit_window.title("Edit Patient")
74             edit_window.geometry('300x300')
75
76             edit_frame = tk.Frame(edit_window)
77             edit_frame.pack()
78
79             name_label = tk.Label(edit_frame, text="Name")
80             name_label.grid(row=0, column=0)
81             name_entry = tk.Entry(edit_frame)
82             name_entry.grid(row=0, column=1)
83             name_entry.insert(tk.END, selected_patient['Name'])
84
85             age_label = tk.Label(edit_frame, text="Age")
86             age_label.grid(row=1, column=0)
87             age_entry = tk.Entry(edit_frame)

```

```

88             age_entry.grid(row=1, column=1)
89             age_entry.insert(tk.END, selected_patient['Age'])
90
91             gender_label = tk.Label(edit_frame, text="Gender")
92             gender_label.grid(row=2, column=0)
93             gender_entry = tk.Entry(edit_frame)
94             gender_entry.grid(row=2, column=1)
95             gender_entry.insert(tk.END, selected_patient['Gender'])
96
97             address_label = tk.Label(edit_frame, text="Address")
98             address_label.grid(row=3, column=0)
99             address_entry = tk.Entry(edit_frame)
100             address_entry.grid(row=3, column=1)
101             address_entry.insert(tk.END, selected_patient['Address'])
102
103             identification_label = tk.Label(edit_frame, text="ID")
104             identification_label.grid(row=4, column=0)
105             identification_entry = tk.Entry(edit_frame)
106             identification_entry.grid(row=4, column=1)
107             identification_entry.insert(tk.END, selected_patient['ID'])
108
109             phone_label = tk.Label(edit_frame, text="Phone")
110             phone_label.grid(row=5, column=0)
111             phone_entry = tk.Entry(edit_frame)
112             phone_entry.grid(row=5, column=1)
113             phone_entry.insert(tk.END, selected_patient['Phone'])
114
115             def update_patient():
116                 patients[selection[0]] = {

```

```

117         "Name": name_entry.get(),
118         "Age": age_entry.get(),
119         "Gender": gender_entry.get(),
120         "Address": address_entry.get(),
121         "ID": identification_entry.get(),
122         "Phone": phone_entry.get()
123     }
124     messagebox.showinfo("Success", "Patient Details Updated Successfully!")
125     edit_window.destroy()
126     update_patient_list()
127
128     update_button = tk.Button(edit_window, text="Update", command=update_patient)
129     update_button.pack()
130
131 else:
132     messagebox.showerror("Error", "Please select a patient to edit.")
133
134 def update_patient_list():
135     patient_listbox.delete(0, tk.END)
136     for patient in patients:
137         patient_listbox.insert(tk.END, f"{patient['Name']}")
138
139 def create_registration_form():
140     global name_entry, age_entry, gender_entry, address_entry, identification_entry, phone_entry, patient_listbox
141
142     registration_window = tk.Tk()
143     registration_window.title("Patient Clinic Registration")
144     registration_window.geometry('450x400')
145     registration_window.configure(bg='#73C6B6')

```

```

146
147     frame = tk.Frame(registration_window, bg='#73C6B6')
148
149     name_label = tk.Label(frame, text="Name", bg='#73C6B6', fg="FFFFFF", font=("Arial", 18))
150     name_entry = tk.Entry(frame, font=("Arial", 12))
151
152     age_label = tk.Label(frame, text="Age", bg='#73C6B6', fg="FFFFFF", font=("Arial", 18))
153     age_entry = tk.Entry(frame, font=("Arial", 12))
154
155     gender_label = tk.Label(frame, text="Gender", bg='#73C6B6', fg="FFFFFF", font=("Arial", 18))
156     gender_entry = tk.Entry(frame, font=("Arial", 12))
157
158     address_label = tk.Label(frame, text="Address", bg='#73C6B6', fg="FFFFFF", font=("Arial", 18))
159     address_entry = tk.Entry(frame, font=("Arial", 12))
160
161     identification_label = tk.Label(frame, text="ID", bg='#73C6B6', fg="FFFFFF", font=("Arial", 18))
162     identification_entry = tk.Entry(frame, font=("Arial", 12))
163
164     phone_label = tk.Label(frame, text="Phone", bg='#73C6B6', fg="FFFFFF", font=("Arial", 18))
165     phone_entry = tk.Entry(frame, font=("Arial", 12))
166
167     register_button = tk.Button(frame, text="Register", bg="#FF3399", fg="FFFFFF", font=("Arial", 12), command=create_patient)
168     show_button = tk.Button(frame, text="Show Patients", bg="#FF3399", fg="FFFFFF", font=("Arial", 12), command=show_patients)
169     clear_patients_button = tk.Button(frame, text="Clear Patients", bg="#FF3399", fg="FFFFFF", font=("Arial", 12), command=clear_patients)
170     edit_button = tk.Button(frame, text="Edit Patient", bg="#FF3399", fg="FFFFFF", font=("Arial", 12), command=edit_patient)
171
172     name_label.grid(row=0, column=0, padx=10, pady=10)
173     name_entry.grid(row=0, column=1, padx=10, pady=10)
174     age_label.grid(row=1, column=0, padx=10, pady=10)

```

```

175     age_entry.grid(row=1, column=1, padx=10, pady=10)
176     gender_label.grid(row=2, column=0, padx=10, pady=10)
177     gender_entry.grid(row=2, column=1, padx=10, pady=10)
178     address_label.grid(row=3, column=0, padx=10, pady=10)
179     address_entry.grid(row=3, column=1, padx=10, pady=10)
180     identification_label.grid(row=4, column=0, padx=10, pady=10)
181     identification_entry.grid(row=4, column=1, padx=10, pady=10)
182     phone_label.grid(row=5, column=0, padx=10, pady=10)
183     phone_entry.grid(row=5, column=1, padx=10, pady=10)
184     register_button.grid(row=6, column=0, columnspan=2, pady=20)
185     show_button.grid(row=7, column=0, columnspan=2, pady=10)
186     clear_patients_button.grid(row=8, column=0, columnspan=2, pady=10)
187     edit_button.grid(row=9, column=0, columnspan=2, pady=10)
188
189     patient_listbox = tk.Listbox(frame, width=50)
190     patient_listbox.grid(row=10, column=0, columnspan=2)
191     frame.pack()
192
193     registration_window.mainloop()
194
195 if __name__ == "__main__":
196     create_registration_form()

```

GUI

Patient Clinic Registration

Name

Age

Gender

Address

ID

Phone

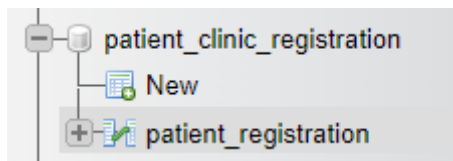
Register

Show Patients

Clear Patients

Edit Patient

DATABASE



STRUCTURE

Server: 127.0.0.1 » Database: patient_clinic_registration » Table: patient_registration

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 name	text	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	2 age	int(2)			No	None			Change Drop More
<input type="checkbox"/>	3 gender	text	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	4 address_label	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	5 identification	int(10)			No	None			Change Drop More
<input type="checkbox"/>	6 phone	varchar(12)	utf8mb4_general_ci		No	None			Change Drop More

☐ Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext

Add to central columns Remove from central columns

Print Propose table structure Track table Move columns Normalise

Add 1 column(s) after phone

Indexes

No index defined!

BROWSER

Server: 127.0.0.1 » Database: patient_clinic_registration » Table: patient_registration

Browser Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

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

✓ Showing rows 0 - 3 (4 total, Query took 0.0015 seconds.)

```
SELECT * FROM `patient_registration`
```

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

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

Extra options

name	age	gender	address_label	identification	phone
MUHAMMAD HANIF AIMAN BIN KAMARUZAMAN	20	MALE	KOLEJ MALINJA	2022871792	01121955708
JOHAN ISKANDAR BIN AHMAD TAMIMI	19	MALE	KOLEJ MALINJA	2147483647	0184063913
MUHAMMAD DANISH ALFIAN BIN MOHD ZULLKIFLI	19	MALE	A020	2147483647	0106547423
MUHAMMAD HAFIZUL ZAHEEN BIN MULIADI	19	MALE	KOLEJ MALINJA	2147483647	0192191194

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

Query results operations

Print Copy to clipboard Export Display chart Create view

Console

CONCLUSION

Project IML208 is a ground-breaking project that optimises patient clinic registration systems using Python programming. As part of this endeavour, an application was developed that integrates technology with human-centric aspects of patient care. Its user-friendly interface improves appointment scheduling and allows for smooth communication between healthcare practitioners and patients. The versatility and robustness of the application provide a smooth connection, improving the overall experience for both patients and healthcare practitioners. Administrators may use the data analysis skills to make educated choices, address emerging patterns, and optimise clinic resources for better patient management. The user-friendly features help to improve operational efficiency and provide a more patient centred healthcare experience. Project IML208 transforms patient clinic registration processes, improving patient care quality and showcasing the beneficial impact of well-designed technology.