Development of Hospital Management using Python Language

# CONTENTS

**S.No Topics Page No.**

1. **Introduction**
2. **Block Diagram**
3. **Description of the Project**
4. **Algorithm**
5. **Project Code**
6. **Result**
7. **Conclusion**

**Objective of the Project:**

The main objective of the Hospital Management System using Python is to develop an efficient, reliable, and user-friendly software application that helps hospital staff manage patient-related information in a digital format. This system is designed to handle tasks such as registering new patients, storing and updating their medical and personal information, and providing a summary of their data in the form of a formatted prescription. The aim is to reduce the dependency on manual paperwork and improve the speed and accuracy of operations within the hospital. By integrating a MySQL database and a graphical user interface using Tkinter, the project ensures real-time data entry, retrieval, and management. Additionally, the system supports essential operations such as modifying existing records, deleting outdated entries, and clearing or exiting the program securely. Overall, this project strives to automate hospital record-keeping processes and contribute to improved healthcare service delivery.

Beyond just creating a digital system for hospital data entry, this project also aims to enhance the decision-making process for healthcare providers. By ensuring that patient information is well-organized and readily accessible, doctors and hospital staff can provide timely and accurate diagnoses and treatments. The system is designed to improve not just operational efficiency, but also patient satisfaction by reducing wait times and minimizing administrative errors. It also promotes data integrity and can serve as a foundation for analytics or future integration with larger hospital networks, including cloud-based health information systems. These advancements have significantly improved the efficiency of local health information systems across various care environments.

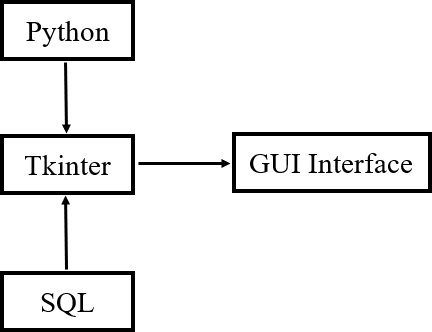
**Introduction:**

In today’s fast-paced healthcare environment, managing hospital operations manually can be both inefficient and error-prone. With increasing numbers of patients and growing demands for better health services, hospitals require a digital solution to streamline their administrative processes. Hospital management encompasses a broad range of activities such as maintaining patient records, managing appointments, handling billing and prescriptions, and ensuring proper communication between doctors and patients. Traditional methods of storing patient information using physical files are not only outdated but also vulnerable to data loss, misplacement, and unauthorized access.

To address these challenges, a computerized Hospital Management System becomes essential. This project introduces a software-based approach using Python for the interface and MySQL for the backend database to manage patient data effectively. With the help of this system, healthcare professionals can input, view, and update patient information quickly. The GUI built with Tkinter ensures ease of use even for users who may not be technically proficient, and the use of a structured database ensures that all information is stored securely and can be retrieved when needed. This project showcases how technology can be leveraged to improve hospital workflow and patient care.

The healthcare industry is increasingly adopting digital tools to overcome the limitations of traditional manual systems. With the help of programming and databases, even small clinics can manage data professionally and securely. This Hospital Management System project reflects a real-world scenario where technology simplifies complex administrative tasks. As healthcare grows more data-driven, having structured digital records helps in tracking patient history, reducing redundancy, and ensuring better coordination among healthcare workers. The integration of a simple yet functional user interface with a reliable backend highlights how even basic technologies can significantly uplift the standards of medical   
record management systems widely used in local healthcare environments.

**Block Diagram:**

****

**Description of the Project:**

The Hospital Management System is a Python-based desktop application that allows hospitals to manage patient records digitally. It uses the Tkinter library for the graphical user interface, which provides a clean and intuitive environment for users to input and manage data. The backend is powered by MySQL, a popular relational database that stores patient records securely and efficiently. The system is divided into various functional modules that work together to provide a seamless experience.

The first module involves setting up the MySQL database. A database named hospitaldemo is created, and within it, a table called patients is used to store all patient-related information such as patient ID, name, date of birth, gender, contact number, disease, referred by, and date of admission. The patient ID is set as the primary key to ensure that each record is unique.

The second module is the graphical user interface, which is divided into several frames for better organization. The data entry section allows users to input patient details through labeled entry fields. The application includes a Treeview widget that displays all stored records in a tabular format, making it easy to view and select specific entries. When a record is selected, its details are automatically populated into the input fields, allowing for easy updates or deletion.

Another important feature is the prescription module, which generates a formatted text area displaying the patient's summary, simulating a medical prescription or visit note. This can be printed or saved for hospital records. Additionally, the system includes utility buttons such as Save, Update, Delete, Clear, Exit, and Generate Prescription, each programmed to handle the corresponding database or UI action.

What makes this system practical is its interactivity and real-time database communication. By using the mysql.connector package, the application connects to the MySQL server and performs operations like insertion, deletion, and data fetching instantly. The system also includes basic input validation to ensure that no field is left empty during registration. Overall, this project combines GUI programming and database management to deliver a useful tool for hospital administration.

**Algorithm:**

Step1: Start

Step2: Set up MySQL database and create required table

Step3: Connect Python to MySQL using mysql-connector

Step4: Create GUI using Tkinter and organize layout

Step5: Implement functions to add, update, delete, and display patient data

Step6: Display table and bind selection events.

Step7: Use formatted text output to show patient details

Step8: End

**Project Code:**

from tkinter import \*

from tkinter import ttk, messagebox

import mysql.connector

# Setup Database

def setup\_database():

try:

con = mysql.connector.connect(host="localhost", user="root", password="admin")

cursor = con.cursor()

cursor.execute("CREATE DATABASE IF NOT EXISTS hospitaldemo")

con.close()

con = mysql.connector.connect(host="localhost", user="root", password="admin", database="hospitaldemo")

cursor = con.cursor()

cursor.execute("""

CREATE TABLE IF NOT EXISTS patients (

name VARCHAR(100),

dob VARCHAR(20),

age VARCHAR(10),

gender VARCHAR(10),

phone VARCHAR(15),

patient\_id VARCHAR(20) PRIMARY KEY,

admission\_date VARCHAR(20),

disease VARCHAR(100),

referred\_by VARCHAR(100)

)

""")

con.commit()

con.close()

except mysql.connector.Error as err:

messagebox.showerror("Database Error", f"Error: {err}")

setup\_database()

# GUI Start

win = Tk()

win.title("Hospital Management System")

screen\_width = win.winfo\_screenwidth()

screen\_height = win.winfo\_screenheight()

win.geometry(f"{screen\_width}x{screen\_height}")

win.config(bg='black')

# Variables

nameofthepatient = StringVar()

dateofbirth = StringVar()

age = StringVar()

gender = StringVar()

phnum = StringVar()

patientid = StringVar()

dateofadd = StringVar()

disease = StringVar()

reff = StringVar()

# Functions

def pd():

if nameofthepatient.get() == "" or dateofbirth.get() == "":

messagebox.showerror("Error", "All fields are required")

else:

try:

con = mysql.connector.connect(host="localhost", user="root", password="admin", database="hospitaldemo")

my\_cursor = con.cursor()

my\_cursor.execute("INSERT INTO patients VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s)", (

nameofthepatient.get(), dateofbirth.get(), age.get(), gender.get(),

phnum.get(), patientid.get(), dateofadd.get(), disease.get(), reff.get()

))

con.commit()

fetch\_data()

con.close()

messagebox.showinfo("Success", "Record has been successfully recorded")

except mysql.connector.Error as e:

messagebox.showerror("Database Error", f"Error: {e}")

def fetch\_data():

con = mysql.connector.connect(host="localhost", user="root", password="admin", database="hospitaldemo")

my\_cursor = con.cursor()

my\_cursor.execute('SELECT \* FROM patients')

rows = my\_cursor.fetchall()

if rows:

table.delete(\*table.get\_children())

for item in rows:

table.insert('', END, values=item)

con.close()

def get\_data(event=''):

selected = table.focus()

values = table.item(selected, 'values')

if values:

nameofthepatient.set(values[0])

dateofbirth.set(values[1])

age.set(values[2])

gender.set(values[3])

phnum.set(values[4])

patientid.set(values[5])

dateofadd.set(values[6])

disease.set(values[7])

reff.set(values[8])

def pre():

txt\_frame.delete(1.0, END)

txt = (

f"{'Name of the Patient':22}: {nameofthepatient.get()}\n"

f"{'Date of Birth':22}: {dateofbirth.get()}\n"

f"{'Age':22}: {age.get()}\n"

f"{'Gender':22}: {gender.get()}\n"

f"{'Phone Number':22}: {phnum.get()}\n"

f"{'Patient ID':22}: {patientid.get()}\n"

f"{'Date of Admission':22}: {dateofadd.get()}\n"

f"{'Disease/Issue':22}: {disease.get()}\n"

f"{'Referred By':22}: {reff.get()}\n"

)

txt\_frame.insert(END, txt)

def delete():

if patientid.get() == "":

messagebox.showerror("Error", "Please select a patient record to delete using Patient ID")

return

try:

con = mysql.connector.connect(host="localhost", user="root", password="admin", database="hospitaldemo")

my\_cursor = con.cursor()

my\_cursor.execute("DELETE FROM patients WHERE patient\_id=%s", (patientid.get(),))

con.commit()

con.close()

fetch\_data()

clear()

messagebox.showinfo("Deleted", "Patient data has been deleted successfully")

except Exception as e:

messagebox.showerror("Error", f"Something went wrong:\n{e}")

def clear():

nameofthepatient.set("")

dateofbirth.set("")

age.set("")

gender.set("")

phnum.set("")

patientid.set("")

dateofadd.set("")

disease.set("")

reff.set("")

txt\_frame.delete(1.0, END)

def exit():

if messagebox.askyesno("Exit", "Do you really want to exit?"):

win.destroy()

def modify():

if patientid.get() == "":

messagebox.showerror("Error", "Please select a patient to update")

return

try:

con = mysql.connector.connect(host="localhost", user="root", password="admin", database="hospitaldemo")

my\_cursor = con.cursor()

query = """

UPDATE patients SET

name=%s,

dob=%s,

age=%s,

gender=%s,

phone=%s,

admission\_date=%s,

disease=%s,

referred\_by=%s

WHERE patient\_id=%s

"""

values = (

nameofthepatient.get(),

dateofbirth.get(),

age.get(),

gender.get(),

phnum.get(),

dateofadd.get(),

disease.get(),

reff.get(),

patientid.get()

)

my\_cursor.execute(query, values)

con.commit()

con.close()

fetch\_data()

messagebox.showinfo("Success", "Record updated successfully")

except mysql.connector.Error as e:

messagebox.showerror("Database Error", f"Error: {e}")

# Header

Label(win, text='HOSPITAL MANAGEMENT SYSTEM', font='algerian 33', bg='blue', fg='white').pack(fill=X)

# Top Frame for Inputs and Patient Details

top\_frame = Frame(win)

top\_frame.pack(fill=BOTH, expand=True)

# Left Input Frame

input\_frame = LabelFrame(top\_frame, text="Patient Information", font='ariel 15 bold', bg='pink', bd=10)

input\_frame.pack(side=LEFT, fill=BOTH, expand=True, padx=5, pady=5)

fields = [

("Name of the Patient", nameofthepatient),

("Date of Birth", dateofbirth),

("Age", age),

("Gender", gender),

("Phone Number", phnum),

("Patient ID", patientid),

("Date of Admission", dateofadd),

("Disease/Issue", disease),

("Referred By", reff),

]

for idx, (label\_text, var) in enumerate(fields):

Label(input\_frame, text=label\_text, bg='pink', font='ariel 12').grid(row=idx, column=0, sticky='w', padx=10, pady=5)

Entry(input\_frame, textvariable=var, width=30).grid(row=idx, column=1, padx=10, pady=5)

# Right Patient Details Frame

prescription\_frame = LabelFrame(top\_frame, text="Patient Details", font='ariel 15 bold', bg='pink', bd=10)

prescription\_frame.pack(side=LEFT, fill=BOTH, expand=True, padx=5, pady=5)

txt\_frame = Text(prescription\_frame, font=('Courier', 12), bg='light yellow')

txt\_frame.pack(fill=BOTH, expand=True)

# Table Frame

table\_frame = Frame(win, bd=10, relief=RIDGE)

table\_frame.pack(fill=BOTH, expand=True)

scroll\_x = Scrollbar(table\_frame, orient=HORIZONTAL)

scroll\_y = Scrollbar(table\_frame, orient=VERTICAL)

scroll\_x.pack(side=BOTTOM, fill=X)

scroll\_y.pack(side=RIGHT, fill=Y)

table = ttk.Treeview(table\_frame, columns=('notp', 'dob', 'age', 'gen', 'pn', 'pi', 'doa', 'di', 'rb'),

xscrollcommand=scroll\_x.set, yscrollcommand=scroll\_y.set)

scroll\_x.config(command=table.xview)

scroll\_y.config(command=table.yview)

for col in ('notp', 'dob', 'age', 'gen', 'pn', 'pi', 'doa', 'di', 'rb'):

table.heading(col, text=col.upper())

table.column(col, width=150)

table['show'] = 'headings'

table.pack(fill=BOTH, expand=True)

table.bind("<ButtonRelease-1>", get\_data)

# Bottom Button Frame

button\_frame = Frame(win, bg='gray')

button\_frame.pack(fill=X)

buttons = [

("Save Patient Data", pd, 'green'),

("Patient Details", pre, 'purple'),

("Modify", modify, 'orange'),

("Delete", delete, 'brown'),

("Clear", clear, 'blue'),

("Exit", exit, 'red')

]

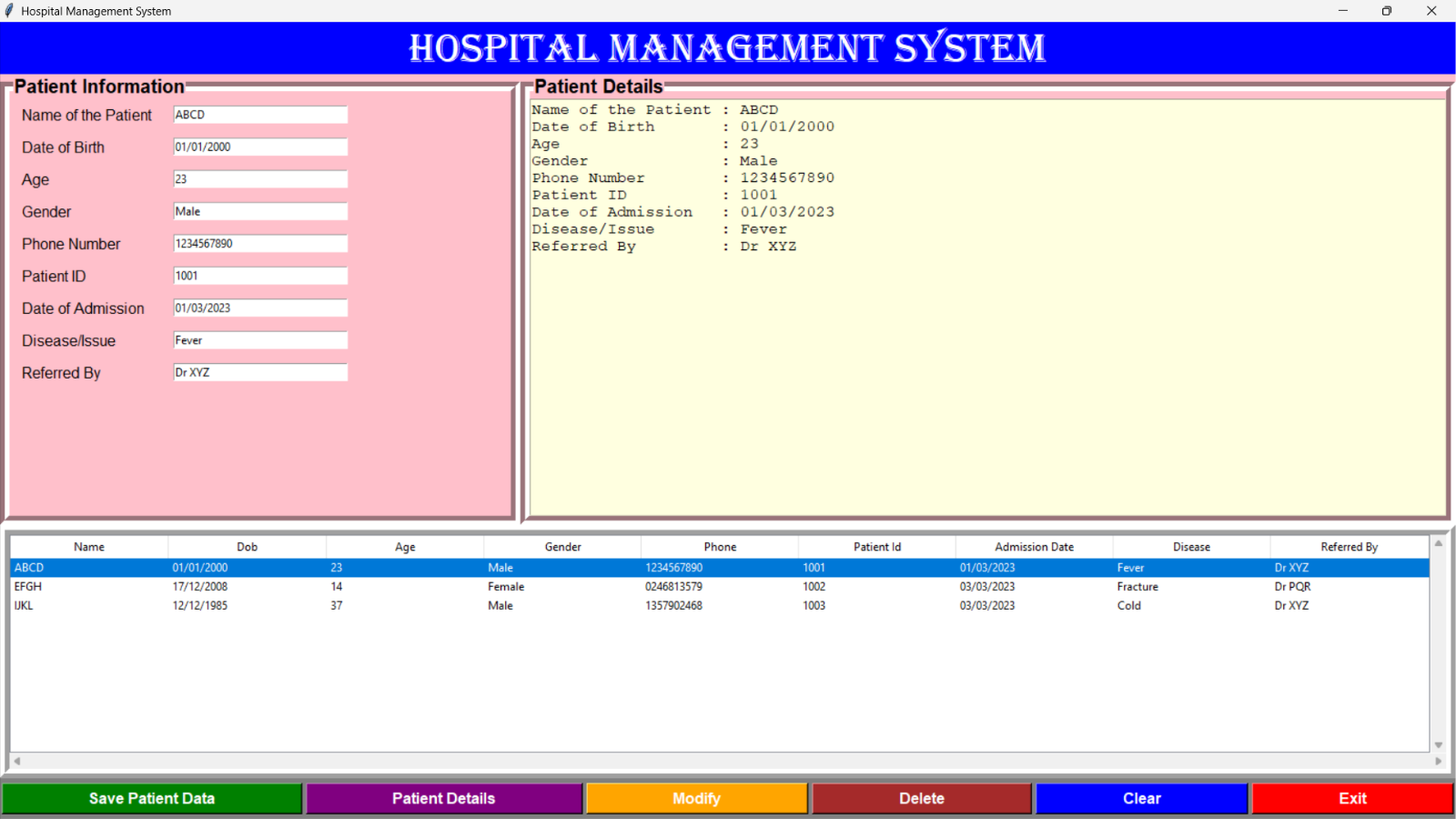
for text, cmd, color in buttons:

Button(button\_frame, text=text, font='ariel 12 bold', bg=color, fg='white', command=cmd, bd=4).pack(side=LEFT, fill=X, expand=True, padx=2, pady=5)

fetch\_data()

win.mainloop()

**Result:**



**Conclusion:**

The Hospital Management System developed using Python and MySQL provides an efficient and scalable solution for managing patient data in hospitals. It reduces the reliance on manual record-keeping, minimizes the chances of errors, and improves the overall efficiency of hospital administration. With features like adding, updating, deleting, and displaying patient records, along with a user-friendly interface, the system is well-suited for small to mid-sized healthcare facilities. It not only enhances data organization but also ensures that crucial patient information is always available at the click of a button.

Furthermore, this project demonstrates the effective use of Python's GUI capabilities and database integration in real-world applications. It lays the foundation for future improvements such as incorporating modules for appointment scheduling, staff management, billing systems, electronic health records, and patient history tracking. In an era where digital transformation is essential for every sector, including healthcare, this project serves as a small yet meaningful step toward smarter and more connected hospital management systems.