Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved; NAAC Accredited A++
Karunya Nagar, Coimbatore - 641 114, Tamil Nadu, India.

DIVISION OF DATA SCIENCE AND CYBER SECURITY

SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

A SKILL BASED EVALUATION REPORT

SUBMITTED BY

HARIHARAN K (URK22AI1048)

COURSE CODE
20CS2016

COURSE NAME

DATABASE MANAGEMENT SYSTEMS

APRIL 2024

BIRTHDAY ALERT SYSTEM

A REAL TIME APPLICATION REPORT

Submitted by

DHURUV SWAMY R (URK22AI1016)

HARIHARAN K (URK22AI1048)

DIVISION OF DATA SCIENCE AND CYBER SECURITY

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed-to-be-under Sec-3 of the UGC Act, 1956) Karunya Nagar, Coimbatore - 641 114. INDIA

APRIL 2024

ABSTRACT

The Birthday Alert System represents a significant advancement in personalized communication technology, particularly in the realm of social messaging platforms like WhatsApp. By harnessing the capabilities of Python programming language and SQLite database management, this system automates the process of sending heartfelt birthday greetings to contacts. Users can conveniently input birthday dates, which are then stored and managed efficiently within the system. Through sophisticated data retrieval and analysis mechanisms implemented in Python scripts, the system is adept at tracking upcoming birthdays and triggering timely dispatch of greetings. Seamless integration with the WhatsApp API ensures that messages are delivered swiftly and reliably to recipients.

Furthermore, the project emphasizes continual refinement and improvement through iterative development processes. This iterative approach has enabled the system to evolve, addressing any potential issues and enhancing its usability and reliability. As a result, users can rely on the system to consistently deliver personalized birthday greetings, thereby enhancing their satisfaction and fostering deeper connections with friends and family.

Overall, the Birthday Alert System exemplifies the transformative potential of technology in nurturing interpersonal relationships within the digital age. By automating the process of sending birthday greetings via a widely used messaging platform, the system underscores how technology can facilitate meaningful interactions and strengthen bonds between individuals, even in an increasingly digitalized world.

PROBLEM STATEMENT

In today's fast-paced digital age, where schedules are often hectic, the Birthday Reminder System emerges as a crucial tool for modern individuals striving to maintain meaningful connections amidst the whirlwind of daily life. With the prevalence of digital communication, traditional methods of remembering birthdays have become increasingly inadequate. However, this system transcends these limitations by offering users a convenient and efficient way to stay on top of important dates.

By seamlessly integrating with WhatsApp, the Birthday Reminder System revolutionizes the way users interact with birthday notifications. Gone are the days of relying on memory or handwritten calendars; instead, the system delivers instantaneous alerts directly to users' preferred messaging platform. This not only ensures that no birthday goes unnoticed but also streamlines the process of sending heartfelt wishes to loved ones, colleagues, and friends.

At its core, the project embodies the ethos of modernity and connectivity, empowering individuals to nurture relationships and celebrate special occasions with ease. With its user-friendly interface and robust functionality, the system fosters a sense of community and belonging in an increasingly interconnected world. By leveraging cutting-edge technology, it not only simplifies birthday management but also enriches lives and strengthens social bonds.

The Birthday Reminder System seeks to revolutionize how users manage important dates by providing a seamless platform for storing and organizing birthday information. Through automation and WhatsApp integration, the project aims to simplify the process of remembering birthdays and foster stronger interpersonal connections. By offering a reliable means of staying connected with loved ones and colleagues, it addresses a pressing need in today's digital landscape, where meaningful interactions are often overlooked amidst the chaos of everyday life.

CONTENT

The Birthday Reminder System utilizes modules such as sqlite3, datetime, os, win10toast, random, webbrowser, and time to manage database operations, handle dates, display notifications, generate random messages, open URLs, and handle time delays.

Our entity relationship model consists of a single entity "Friends" with attributes Sno (serial number), Name, DOB (date of birth), Number (WhatsApp number), and Last_Wish. This model represents the relationship between individuals and their birthdays, facilitating efficient management of birthday reminders.

The entity relationship model is converted into an SQLite database by creating a table named "Friends" with columns corresponding to the attributes in the model. Attributes are defined with appropriate data types and constraints to ensure data integrity.

Our table, "Friends," is at least in the first normal form (1NF) as it contains atomic values in each cell and there are no repeating groups. Further normalization may be applied based on specific requirements to achieve higher normal forms.

Modules Used:

In the first script:

sqlite3: Used for database operations, such as connecting to the SQLite database and executing SQL queries.

datetime: Utilized for working with dates and times, especially for obtaining today's date and formatting dates.

os: Employed for working with file paths, specifically for getting the path of the icon file used for notifications.

win10toast: Used for displaying toast notifications on Windows operating systems.

random: Utilized for generating random choices, specifically for selecting random birthday wishes.

web browser: Utilized for opening URLs in the default web browser, specifically for sending WhatsApp messages.

time: Used for introducing delays in the program, such as adding pauses between actions.

In the second script:

tkinter: Used for creating the graphical user interface (GUI) of the application, including windows, labels, buttons, and entry fields.

messagebox: A submodule of tkinter used for displaying message boxes, such as error messages or informational pop-ups.

datetime: Similar to its usage in the first script, used for working with dates and times, especially for validating and formatting dates.

sqlite3: Similar to its usage in the first script, used for database operations, such as connecting to the SQLite database, executing SQL queries, and fetching data.

os: Similar to its usage in the first script, used for working with file paths.

time: Similar to its usage in the first script, used for handling time-related operations.

Entity-Relationship Model:

An entity-relationship (ER) model is a graphical representation used to describe the relationships between entities in a database. In the context of our project, the ER model would likely describe the structure of our database schema, including the entities (tables), attributes (columns), and relationships between them.

Since our project involves managing friends' birthdays, a simple ER model might consist of the following entities:

Friends: Represents the entity for storing information about friends, including their names, dates of birth, WhatsApp numbers, and last wish dates.

Attributes: Sno (serial number), Name, DOB (date of birth), Number, Last_Wish.

This entity would be the primary one in our database since it holds the main information about our friends.

The relationships between entities would include:

No explicit relationships are visible in the provided scripts. However, in a more complex database design, you might establish relationships between entities, such as establishing a one-to-many relationship between a Friend entity and another entity representing messages sent to friends.

It's important to note that the ER model might evolve based on the specific requirements and functionalities of our project. You may need to refine and expand the model as you develop our application further.

Entity Relationship Conversion into SQLite Database:

To convert an entity-relationship (ER) model into an SQLite database, you'll need to create tables in the database corresponding to the entities in my ER model.

Each entity in my ER model will become a table in the SQLite database.

Attributes of each entity will become columns in the corresponding table.

Relationships between entities will be represented using foreign keys.

For example, if you have an entity Friends with attributes Sno, Name, DOB, Number, and Last_Wish, you would create a table in SQLite named Friends with columns corresponding to these attributes.

Normalization Form of Our Table:

Normalization is the process of organizing data in a database to avoid redundancy and dependency. It involves dividing large tables into smaller tables and defining relationships between them.

The level of normalization achieved by a table is described by its normal form.

Without seeing the complete structure of our table and its dependencies, it's difficult to determine its exact normal form.

However, based on the provided attributes (Sno, Name, DOB, Number, Last_Wish), our table appears to be in at least the first normal form (1NF), as it contains atomic values in each cell and there are no repeating groups.

Connecting Frontend to Backend:

To connect the frontend (GUI created using tkinter) to the backend (SQLite database), you'll need to perform database operations in response to user actions in the GUI.

In our tkinter application, you'll call functions/methods that interact with the SQLite database to perform tasks such as adding entries, retrieving records, deleting records, etc.

You can use the sqlite3 module in Python to connect to and execute SQL queries on the SQLite database from our tkinter application.

Normal Form Stored in Our Database:

The normal form stored in your database depends on how well the database schema is designed and how normalized your tables are.

As mentioned earlier, without knowing the complete structure of your tables and their relationships, it's challenging to determine the exact normal form stored in your database.

Generally, the aim is to achieve at least the third normal form (3NF) to minimize redundancy and dependency in the data.

Implementation of Our Table:

The implementation of your table involves creating a SQLite database file and defining the table structure using SQL queries.

You'll use the CREATE TABLE statement in SQLite to define the structure of your Friends table, specifying the column names, data types, and any constraints such as primary keys and foreign keys.

Syntax:

```
    CREATE TABLE Friends (
        Sno INTEGER PRIMARY KEY,
        Name TEXT NOT NULL,
        DOB DATE,
        Number TEXT,
        Last_Wish TEXT
);
```

Creates a table named Friends with columns Sno, Name, DOB, Number, and Last_Wish. The Sno column is designated as the primary key.

METHODOLOGY / ARCHITECTURE

Methodology:

Requirement Analysis: The project begins with a thorough analysis of user requirements, identifying key functionalities and features essential for an effective Birthday Reminder System.

Design and Planning: Based on the requirements gathered, the system architecture is designed, outlining the database structure, user interface components, and integration with WhatsApp for automated alerts. A detailed project plan is formulated, specifying tasks, timelines, and resource allocation.

Database Setup: The SQLite database is set up to store birthday information, including fields for name, date of birth, and contact details. Appropriate indexing and normalization techniques are employed to ensure efficient data retrieval and management.

User Interface Development: A user-friendly interface is developed to facilitate easy interaction with the system. Users can add, update, or delete birthday records, as well as configure notification preferences.

WhatsApp Integration: The Twilio API is integrated to enable communication with WhatsApp. Authentication credentials are obtained, and functionality is implemented to send automated messages containing birthday reminders to users' WhatsApp accounts.

Automated Alert System: Scheduled tasks are implemented to periodically check the database for upcoming birthdays. When a birthday is detected, the system generates a personalized message and sends it to the respective contact via WhatsApp.

Testing and Validation: The system undergoes rigorous testing to ensure reliability, accuracy, and seamless functionality. Test cases are designed to validate various scenarios, including data input, message delivery, and error handling.

Architecture:

Frontend Interface: The user interacts with the system through a frontend interface, which provides functionalities for adding, updating, and deleting birthday records. The interface also allows users to configure notification settings.

Backend Processing: The backend of the system handles data processing and communication with external services such as the SQLite database and the Twilio API. It executes scheduled tasks to check for upcoming birthdays and triggers automated alerts.

SQLite Database: The SQLite database serves as the primary data repository, storing information about birthdays, including names, dates of birth, and contact details. It is structured to facilitate efficient data retrieval and management.

Twilio Integration: The system integrates with the Twilio API to send automated messages to users' WhatsApp accounts. Twilio provides the necessary infrastructure for message delivery, including authentication and communication protocols.

Scheduled Tasks: Scheduled tasks are implemented to periodically scan the database for upcoming birthdays. When a birthday is detected, the system generates a personalized message and initiates the delivery process via Twilio.

Error Handling: Robust error handling mechanisms are incorporated to address potential issues such as database connectivity failures, message delivery errors, and other unforeseen circumstances. Logs are generated to aid in debugging and troubleshooting.

Security Measures: The system implements security measures to safeguard users' personal data and ensure confidentiality. Access controls and encryption techniques are employed to mitigate potential security risks.

SQLITE DATABASE:

	Sno	Name	DOB	Number	Last_Wish
	Search column...	Search column...	Search column...	Search column...	Search column...
1	3	dhuruv	2024-04-07	7418945601	0,2024,
2	4	hari	2024-04-07	7092003788	0,2024,
3	5	soorya	2024-04-07	9943051504	0,2024,
4	6	dd	2024-04-07	7708323025	0,2024,
5	7	anto	2024-04-07	97896523327	0,2024,
6	8	ARAVIND	2024-04-07	9344516381	0,2024,
7	9	sam	2024-04-07	6369000186	0,2024,
8	10	tamil	2024-04-07	6379021954	0,2024,
9	11	aron	2024-04-07	9585325511	0,2024,
10	12	presana	2024-04-07	6380401992	0,2024,

IMPLEMENTATION – CODING AND OUTPUT SCREENSHOT

APPENDIX -1

SOURCE CODE:

```
import tkinter as tk
from tkinter import messagebox
from datetime import datetime
import sqlite3 as sq

class FriendGUI:
    def __init__(self, master):
        self.master = master
        master.title("Birthday Wisher")

        # Connect to the database
        self.conn = sq.connect(r"Database\database.sqlite")
        self.cur = self.conn.cursor()

        # Add a background image
        self.bg_image = tk.PhotoImage(file="rd.png")
        self.background = tk.Label(master, image=self.bg_image)
        self.background.place(x=1, y=1, relwidth=1, relheight=1 )  # Set full background
        self.setup_styles()
        self.create_widgets()


    def setup_styles(self):
        # Define custom styles
        self.heading_font = ("Helvetica", 24, "bold")
        self.button_font = ("Helvetica", 14)
        self.entry_font = ("Helvetica", 12)

        self.heading_bg = "#ffcccc"
        self.button_bg = "#ffd699"
        self.entry_bg = "white"

    def create_widgets(self):
        # Heading label
        self.label = tk.Label(self.master, text="Welcome to Birthday Wisher!",
font=self.heading_font, bg=self.heading_bg)
        self.label.pack(pady=20)

        # Add an entry button
        self.add_button = tk.Button(self.master, text="Add an Entry", font=self.button_font,
bg=self.button_bg, command=self.add_entry)
        self.add_button.pack(pady=10)

        # See all records button
        self.see_all_button = tk.Button(self.master, text="See All Records",
font=self.button_font, bg=self.button_bg, command=self.see_all_records)
        self.see_all_button.pack(pady=10)
```

```
        # See records of a particular month button
        self.see_month_button = tk.Button(self.master, text="See Records of a Particular
Month", font=self.button_font, bg=self.button_bg, command=self.see_month_records)
        self.see_month_button.pack(pady=10)

        # Delete a record button
        self.delete_button = tk.Button(self.master, text="Delete a Record",
font=self.button_font, bg=self.button_bg, command=self.delete_record)
        self.delete_button.pack(pady=10)

        # Exit button
        self.exit_button = tk.Button(self.master, text="Exit", font=self.button_font,
bg=self.button_bg, command=self.master.quit)
        self.exit_button.pack(pady=10)

    def add_entry(self):
        top = tk.Toplevel()
        top.title("Add New Entry")

        # Name entry
        self.name_label = tk.Label(top, text="Name:", font=self.entry_font)
        self.name_label.grid(row=0, column=0, sticky="e")
        self.name_entry = tk.Entry(top, font=self.entry_font, bg=self.entry_bg)
        self.name_entry.grid(row=0, column=1)

        # Date of Birth entry
        self.dob_label = tk.Label(top, text="Date of Birth (YYYY-MM-DD):",
font=self.entry_font)
        self.dob_label.grid(row=1, column=0, sticky="e")
        self.dob_entry = tk.Entry(top, font=self.entry_font, bg=self.entry_bg)
        self.dob_entry.grid(row=1, column=1)

        # WhatsApp number entry
        self.number_label = tk.Label(top, text="WhatsApp Number (91XXXXXXXXXX):",
font=self.entry_font)
        self.number_label.grid(row=2, column=0, sticky="e")
        self.number_entry = tk.Entry(top, font=self.entry_font, bg=self.entry_bg)
        self.number_entry.grid(row=2, column=1)

        # Submit button
        self.submit_button = tk.Button(top, text="Submit", font=self.button_font,
bg=self.button_bg, command=self.submit_entry)
        self.submit_button.grid(row=3, columnspan=2, pady=10)

    def submit_entry(self):
        name = self.name_entry.get()
        dob = self.dob_entry.get()
        number = self.number_entry.get()
```

```
        try:
            dob_date = datetime.strptime(dob, "%Y-%m-%d").date()
            number = int(number)
        except ValueError:
            messagebox.showerror("Error", "Invalid input")
            return

        query = "INSERT INTO Friends (Name, DOB, Number) VALUES (?, ?, ?)"
        self.cur.execute(query, (name, dob_date, number))
        self.conn.commit()  # Commit changes to the database

        messagebox.showinfo("Success", "Entry added successfully")

    def see_all_records(self):
        top = tk.Toplevel()
        top.title("All Records")

        query = "SELECT * FROM Friends"
        self.cur.execute(query)
        data = self.cur.fetchall()

        self.format_print(top, data)

    def see_month_records(self):
        top = tk.Toplevel()
        top.title("Records of a Particular Month")

        self.month_label = tk.Label(top, text="Month (MM):", font=self.entry_font)
        self.month_label.grid(row=0, column=0, sticky="e")
        self.month_entry = tk.Entry(top, font=self.entry_font, bg=self.entry_bg)
        self.month_entry.grid(row=0, column=1)

        self.submit_month_button = tk.Button(top, text="Submit", font=self.button_font,
bg=self.button_bg, command=lambda: self.submit_month(top))
        self.submit_month_button.grid(row=1, columnspan=2, pady=10)

    def submit_month(self, top):
        month = self.month_entry.get()
        query = f"SELECT * FROM Friends WHERE strftime('%m', DOB) = ?"
        self.cur.execute(query, (month,))
        data = self.cur.fetchall()
        self.format_print(top, data)

    def delete_record(self):
        top = tk.Toplevel()
        top.title("Delete a Record")

        query = "SELECT * FROM Friends"
        self.cur.execute(query)
        data = self.cur.fetchall()
```

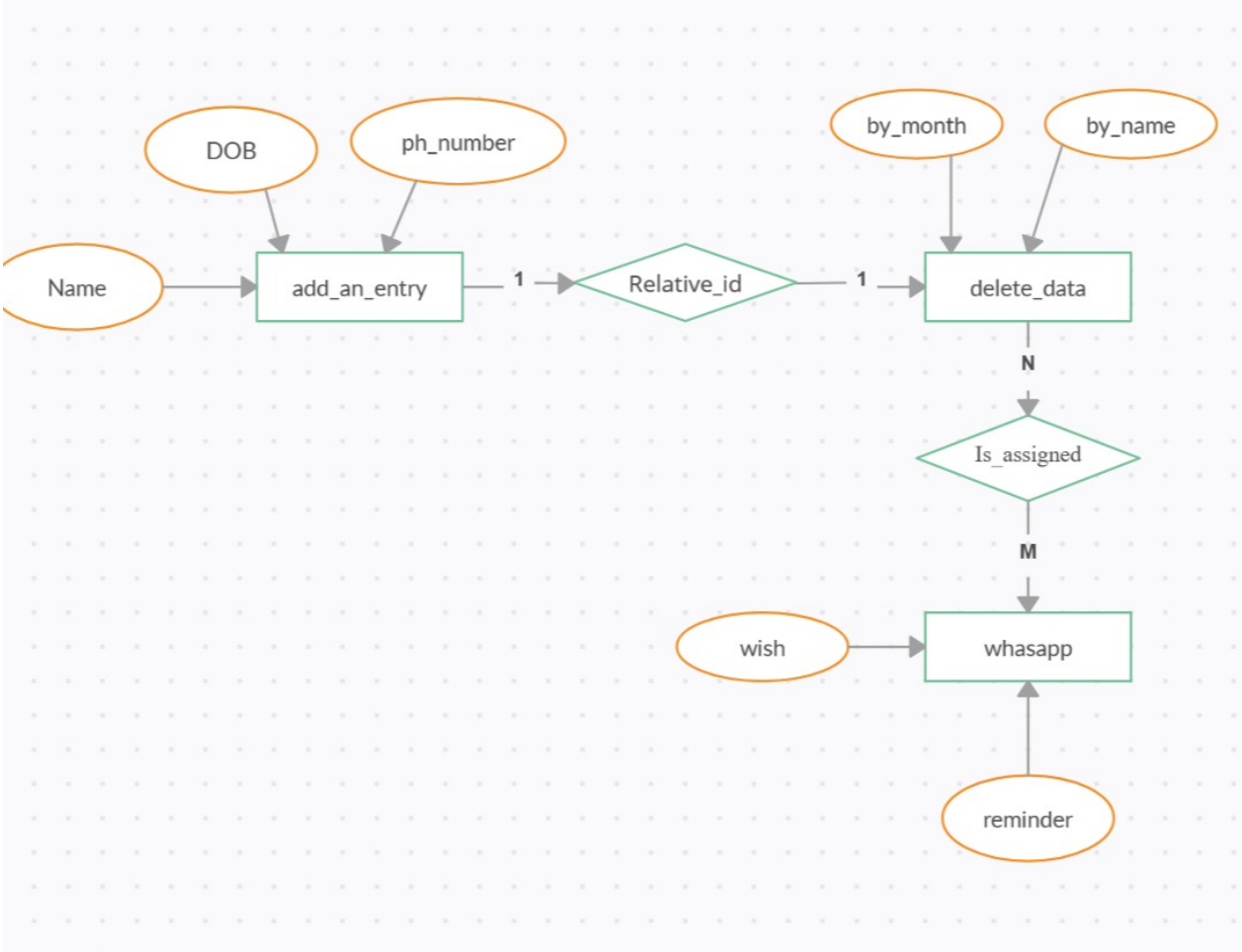
```
        self.format_print(top, data)

        self.sno_label = tk.Label(top, text="Enter the Sno to delete:", font=self.entry_font)
        self.sno_label.grid(row=len(data)+1, column=0, sticky="e")
        self.sno_entry = tk.Entry(top, font=self.entry_font, bg=self.entry_bg)
        self.sno_entry.grid(row=len(data)+1, column=1)

        self.delete_button = tk.Button(top, text="Delete", font=self.button_font,
bg=self.button_bg, command=lambda: self.submit_delete(top))
        self.delete_button.grid(row=len(data)+2, columnspan=2, pady=10)

    def submit_delete(self, top):
        sno = self.sno_entry.get()
        query = "DELETE FROM Friends WHERE Sno = ?"
        self.cur.execute(query, (sno,))
        self.conn.commit()  # Commit changes to the database

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

        top.destroy()

    def format_print(self, top, data):
        if data:
            header = ("Sno", "Name", "DOB", "Number", "Last_Wish")
            for i, header_item in enumerate(header):
                label = tk.Label(top, text=header_item, font=self.entry_font)
                label.grid(row=0, column=i)

            for i, row in enumerate(data):
                for j, cell in enumerate(row):
                    label = tk.Label(top, text=cell, font=self.entry_font)
                    label.grid(row=i+1, column=j)

def main():
    root = tk.Tk()
    app = FriendGUI(root)
    root.mainloop()

if __name__ == "__main__":
    main()
```

Another Code (WhatsApp Integration):

```
import sqlite3 as sq
from datetime import date, datetime
import os
from win10toast import ToastNotifier
import random as ran
import webbrowser as web
import time

class Wish:
    def __init__(self):
        try:
            # Connecting to the Database and initializing a cursor with any unexpected error
handling
            self.conn = sq.connect(r"Database\database.sqlite")
            self.cur = self.conn.cursor()
        except Exception as e:
            print(e)

        self.checkToday()
        self.cur.close()
        self.conn.close()

    def checkToday(self):
        '''Function to check if today's anyone birthday'''
        today = datetime.now().strftime("%m-%d") # Getting today's date
        # Data retrival from the database commands
        query = (f"select Sno, Name, Last_Wish from Friends where DOB like '%-
%{today}';")
        self.cur.execute(query)
        data = self.cur.fetchall()
        # Check if there is anybody's birthday and then only do the process
        if data:
            temp = []
            name_list = []
            # Getting present year
            present_year = datetime.now().strftime("%Y")
            for i in data:
                # Appending only names of the person in a name_list and all the data in temp
List
                name_list.append(f"{i[1]} ")
                temp.append(i)
            # Converting name_list to string for the notification
            names = "".join(name_list)
            # Get the path of the Icon file for the notification
            iconpath = os.path.dirname(os.path.realpath(__file__))
            filename = r"Icon\birthday.ico"
            iconpath = os.path.join(iconpath, filename)
            # Message for the notification
```

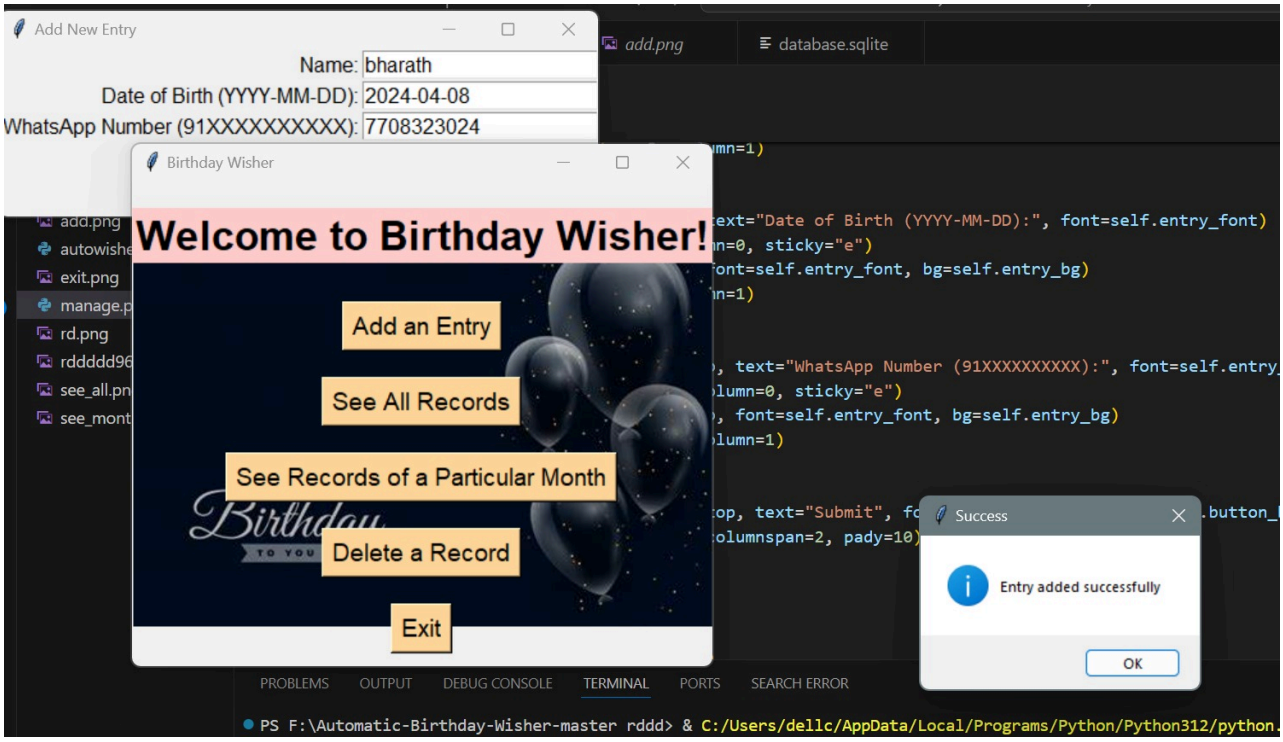
```
            msg = (f"We have Birthday of {names}. Wishing them on our behalf. Make sure
you are connected to Internet. You have 30 seconds.")
            noti = ToastNotifier() # Norification Initialiation
            noti.show_toast("Let's Make Our Loved Ones Birthday Special", msg, iconpath,
30) # Showing the notification
            # Data preparation for commiting it to the database
            for j in temp:
                sno = j[0]
                last_wish = str(j[2])
                # Check if the person is already wished or not: if not then wish and add current
year in the Last_Wish column of the database to avoid wishing multiple times
                if present_year not in last_wish:
                    # Fetching the contact no for the birthday person
                    query = (f"select Number from Friends where Sno = {sno};")
                    self.cur.execute(query)
                    contact = self.cur.fetchall()
                    self.sendMsg(contact) # sendmsg function call to send message to the person
passing his contact no as argument
                    # Updating the Last_Wish for the person already wished
                    query = (f"update Friends set Last_Wish = Last_Wish || '{present_year},'
where Sno = {sno}")
                    self.cur.execute(query)
                    self.conn.commit()
            time.sleep(5)
    def sendMsg(self, contact):
        '''Function for sending WhatsApp messages to the person'''
        wishes =
[r'Wishing+you+a+day+filled+with+happiness+and+a+year+filled+with+joy.+Happy+bi
rthday%21',
r'Sending+you+smiles+for+every+moment+of+our+special+day%E2%80%A6Have+a+
wonderful+time+and+a+very+happy+birthday%21',
r'On+our+birthday+we+wish+for+you+that+whatever+you+want+most+in+life+it+come
s+to+you+just+the+way+you+imagined+it+or+better.+Happy+birthday%21',
r'Sending+our+way+a+bouquet+of+happiness%E2%80%A6To+wish+you+a+very+happ
y+birthday%21',
r'Wishing+you+a+beautiful+day+with+good+health+and+happiness+forever.+Happy+bi
rthday%21'] # Custom Birthday messages url encoded
        # Preparing the url for each individual with a random custom message from the
wishes list
        for no in contact:
            msg = ran.choice(wishes)
            url =
(f"https://web.whatsapp.com/send?phone={no[0]}&text={msg}&source&data&app_abse
nt")
            web.open(url) # Opening url in the browser
            time.sleep(3)
if __name__ == "__main__":
    wish = Wish()
    wish
```

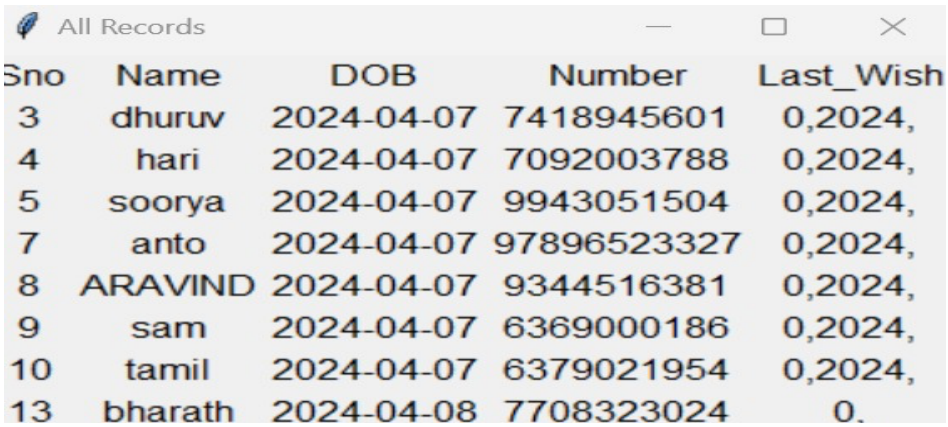
ER DIAGRAM:


DOB
ph_number
by_month
by_name
Name
add_an_entry
1
Relative_id
1
delete_data
N
Is_assigned
M
wish
whasapp
reminder

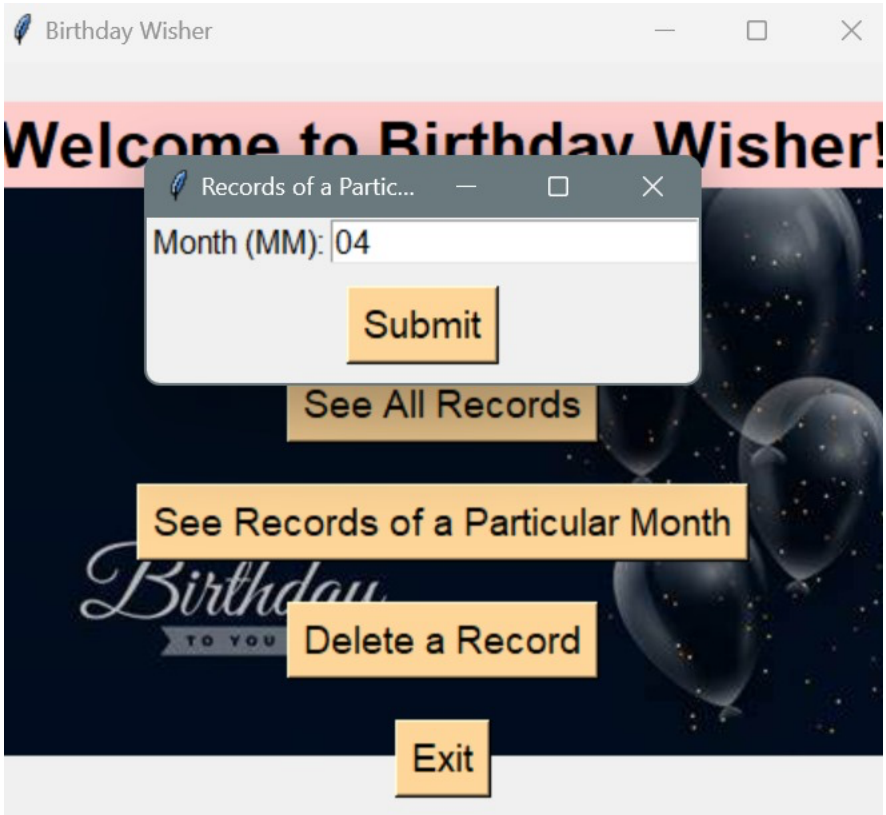
APPENDIX - 2

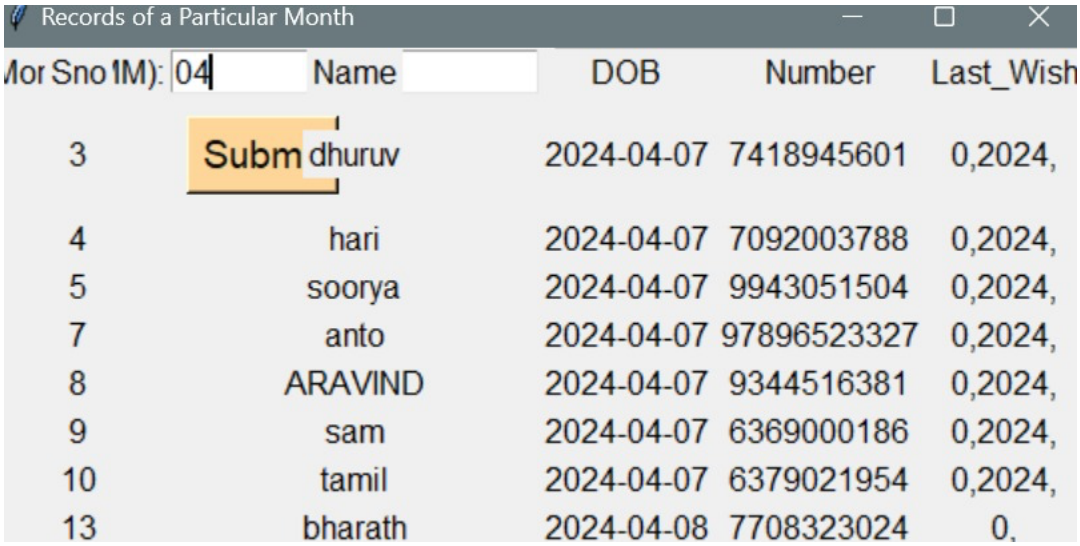
OUTPUT:

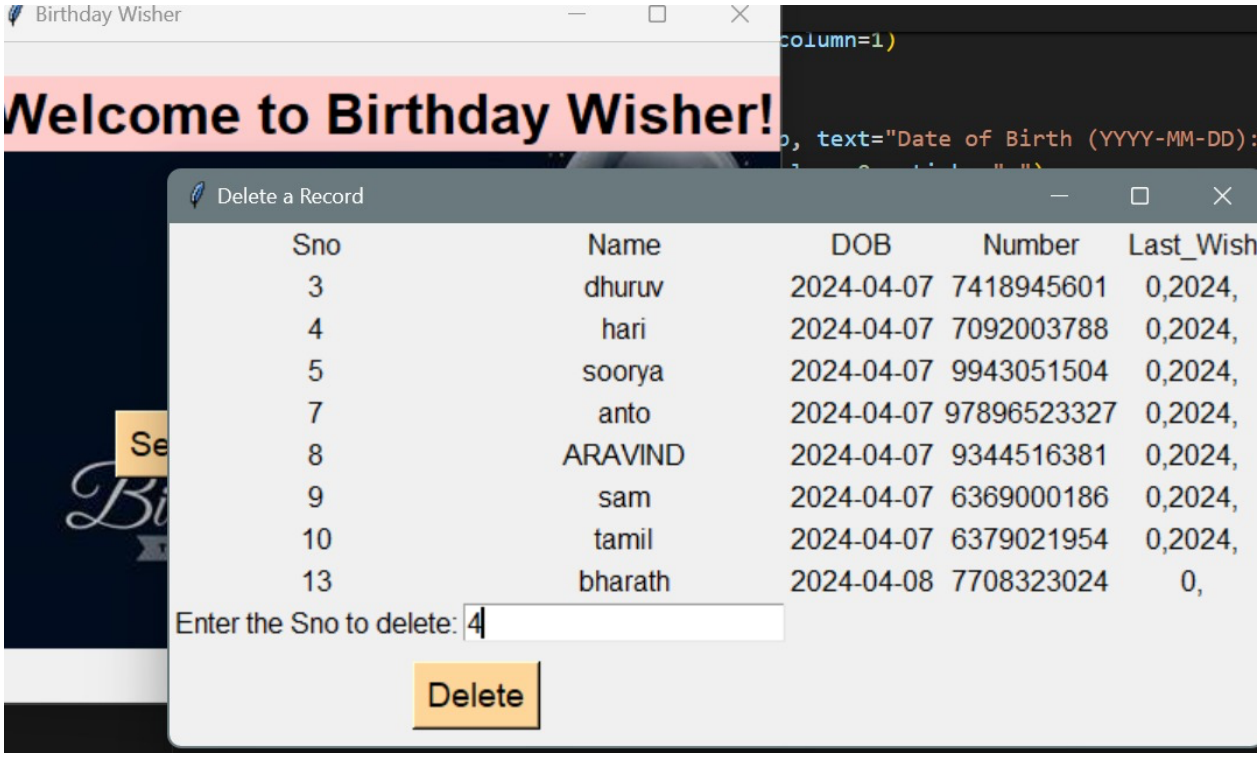


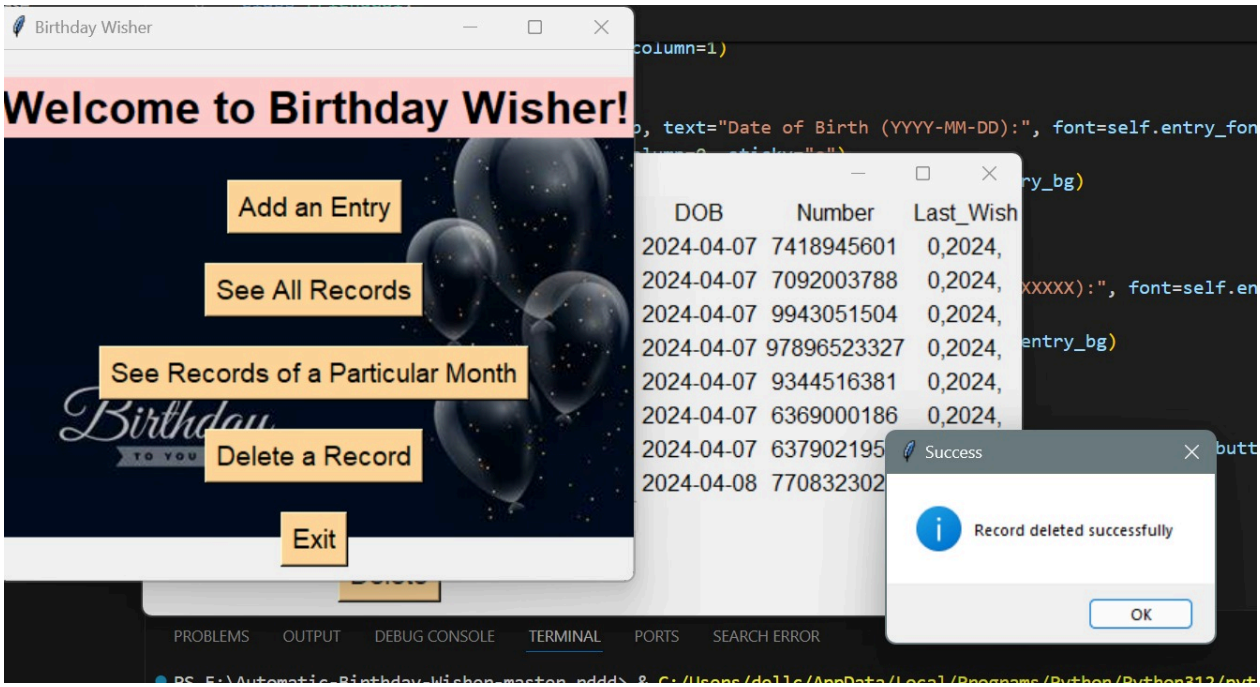


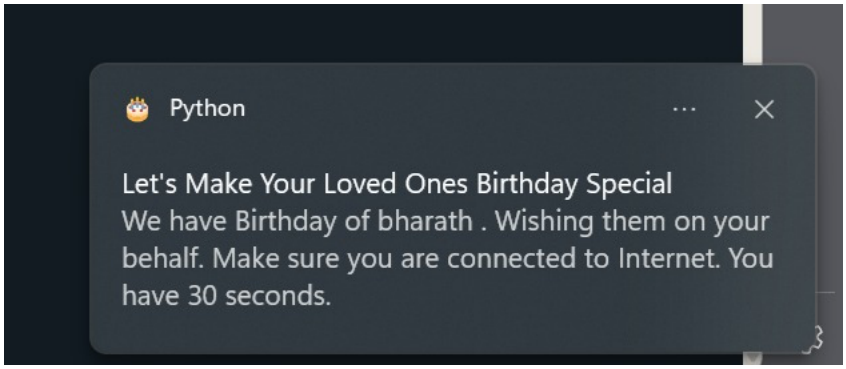
Sno	Name	DOB	Number	Last_Wish
3	dhuruv	2024-04-07	7418945601	0,2024,
4	hari	2024-04-07	7092003788	0,2024,
5	soorya	2024-04-07	9943051504	0,2024,
7	anto	2024-04-07	97896523327	0,2024,
8	ARAVIND	2024-04-07	9344516381	0,2024,
9	sam	2024-04-07	6369000186	0,2024,
10	tamil	2024-04-07	6379021954	0,2024,
13	bharath	2024-04-08	7708323024	0,

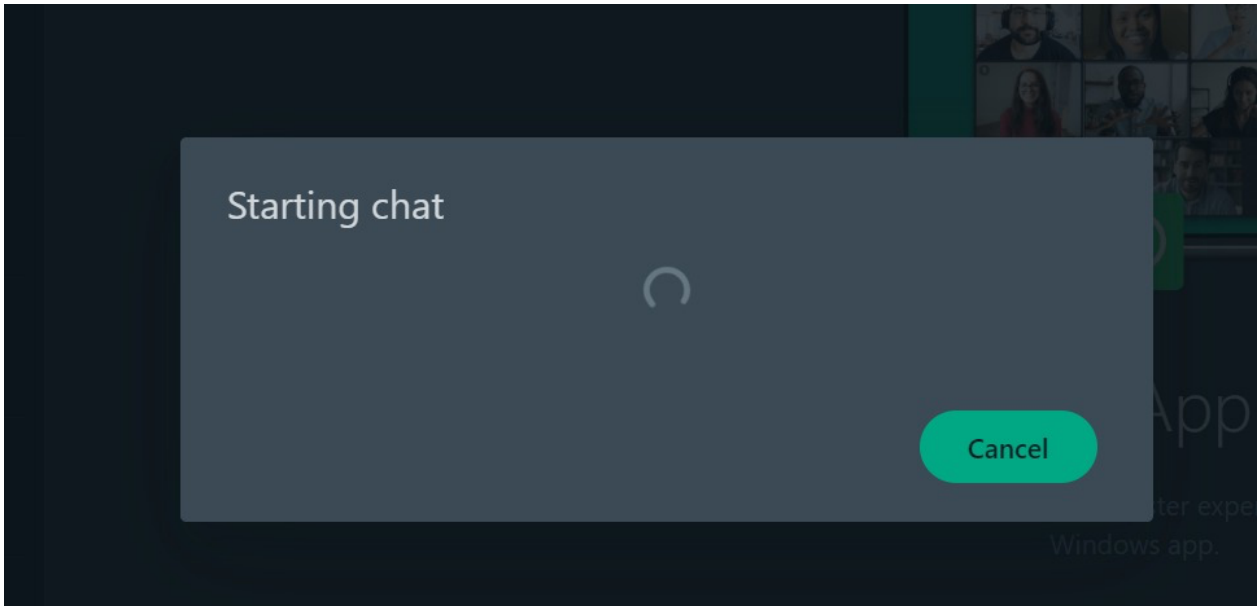


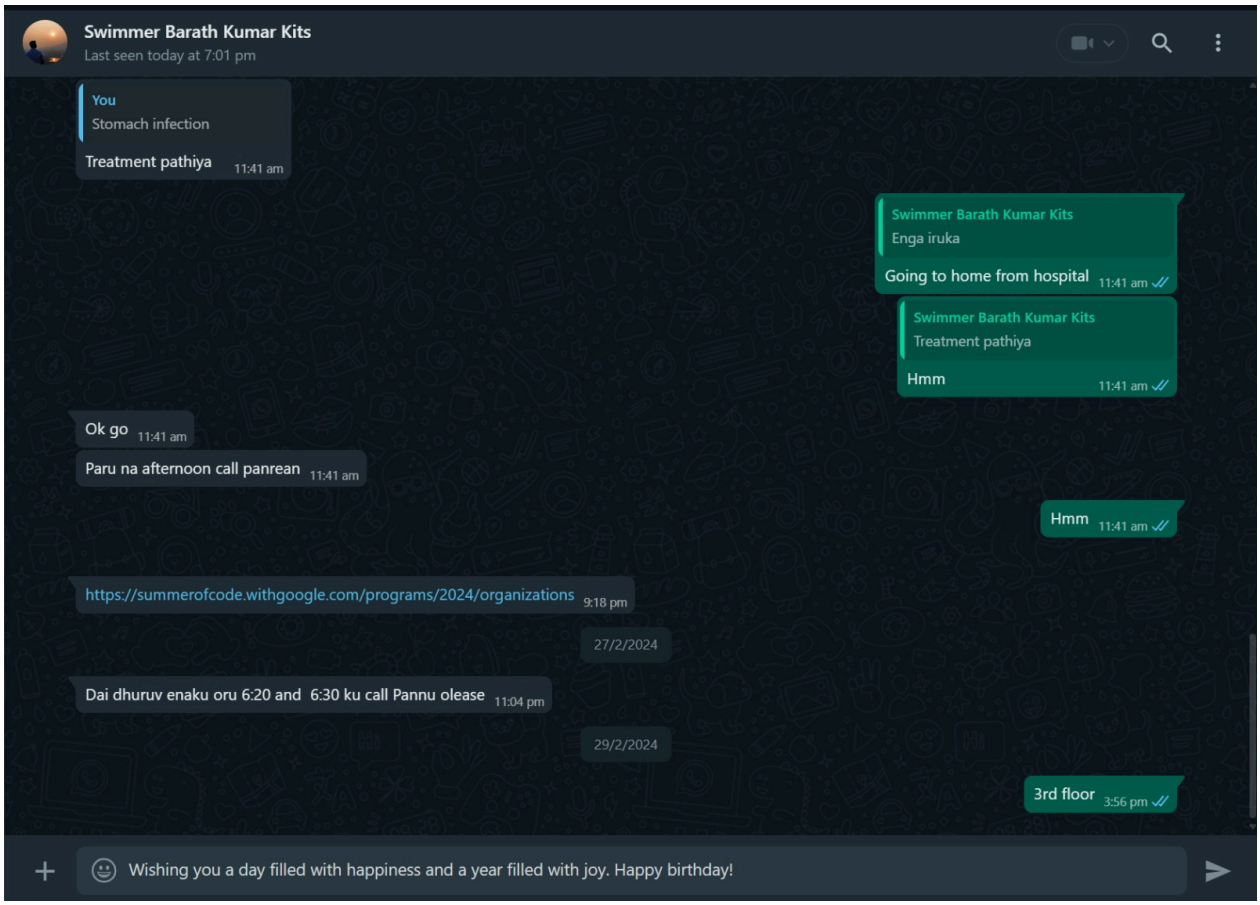

Records of a Particular Month
Mor Sno (M): 04
Name
DOB
Number
Last_Wish
Subm
3 dhuruv 2024-04-07 7418945601 0,2024,
4 hari 2024-04-07 7092003788 0,2024,
5 soorya 2024-04-07 9943051504 0,2024,
7 anto 2024-04-07 97896523327 0,2024,
8 ARAVIND 2024-04-07 9344516381 0,2024,
9 sam 2024-04-07 6369000186 0,2024,
10 tamil 2024-04-07 6379021954 0,2024,
13 bharath 2024-04-08 7708323024 0,


Birthday Wisher
Welcome to Birthday Wisher!
column=1)
text="Date of Birth (YYYY-MM-DD):
Delete a Record
Sno Name DOB Number Last_Wish
3 dhuruv 2024-04-07 7418945601 0,2024,
4 hari 2024-04-07 7092003788 0,2024,
5 soorya 2024-04-07 9943051504 0,2024,
7 anto 2024-04-07 97896523327 0,2024,
8 ARAVIND 2024-04-07 9344516381 0,2024,
9 sam 2024-04-07 6369000186 0,2024,
10 tamil 2024-04-07 6379021954 0,2024,
13 bharath 2024-04-08 7708323024 0,
Enter the Sno to delete: 4
Delete


Birthday Wisher
Welcome to Birthday Wisher!
Add an Entry
See All Records
See Records of a Particular Month
Delete a Record
Exit
DOB Number Last_Wish
2024-04-07 7418945601 0,2024,
2024-04-07 7092003788 0,2024,
2024-04-07 9943051504 0,2024,
2024-04-07 97896523327 0,2024,
2024-04-07 9344516381 0,2024,
2024-04-07 6369000186 0,2024,
2024-04-07 637902195
2024-04-08 770832302
Success
Record deleted successfully
OK
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR


Python
Let's Make Your Loved Ones Birthday Special
We have Birthday of bharath . Wishing them on your behalf. Make sure you are connected to Internet. You have 30 seconds.


Starting chat
Cancel


Swimmer Barath Kumar Kits
Last seen today at 7:01 pm
You
Stomach infection
Treatment pathiya 11:41 am
Swimmer Barath Kumar Kits
Enga iruka
Going to home from hospital 11:41 am
Swimmer Barath Kumar Kits
Treatment pathiya
Hmm 11:41 am
Ok go 11:41 am
Paru na afternoon call panrean 11:41 am
Hmm 11:41 am
https://summerofcode.withgoogle.com/programs/2024/organizations 9:18 pm
27/2/2024
Dai dhuruv enaku oru 6:20 and 6:30 ku call Pannu olease 11:04 pm
29/2/2024
3rd floor 3:56 pm
Wishing you a day filled with happiness and a year filled with joy. Happy birthday!

CONCLUSION

In conclusion, the development of the Birthday Reminder System with WhatsApp Integration represents a significant advancement in modernizing the process of managing and commemorating important dates. By leveraging technology and automation, the project offers users a streamlined solution for organizing birthday information and receiving timely alerts directly to their smartphones.

Through a comprehensive methodology encompassing requirement analysis, design, development, and testing, the system has been meticulously crafted to meet user needs and deliver a seamless user experience. The integration of SQLite for database management, Twilio for WhatsApp communication, and scheduled tasks for automated alerts ensures the reliability, efficiency, and accuracy of the system's functionalities.

The architecture of the system, characterized by a frontend interface, backend processing, database storage, Twilio integration, scheduled tasks, error handling mechanisms, and security measures, illustrates the intricacies involved in achieving the project's objectives. Together, these components form a cohesive framework that empowers users to effortlessly manage birthdays and nurture personal and professional relationships.

EVALUATION SHEET

Reg.No: URK22AI1048

Name: HARIHARAN K

Course code: 20CS2016

Course Name: Database Management Systems

S.No	Rubrics	Maximum Marks	Marks Obtained
1	Birthday Alert System	40	
Total		40	

Signature of the Faculty-in-charge

Signature of the Examiner1 **Signature of the Examiner2**