KARNATAK LAW SOCIETY’S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

**(APPROVED BY AICTE, NEW DELHI)**

Department of **Electronics and Communication**



*Mini Project Report*

**“GUI FOR ADDRESS BOOK DATABASE”**

*Submitted in the partial fulfillment for the academic requirement**of*

***7th Semester B.E.***

***in***

***DATABASE AND MANAGEMENT SYSTEMS***

***(16EC754)***

*Submitted by*

**MOHAMMADFAIZ BAWA**

**USN:2GI16EC078**

Guide

**Ms. Pratijnya S Ajawan**

**2019 – 2020**

KARNATAK LAW SOCIETY’S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

**(APPROVED BY AICTE, NEW DELHI)**

Department of **Electronics and Communication**

**CERTIFICATE**

This is to certify that Mr. **MOHAMMADFAIZ BAWA** of **7th** semester and bearing USN **2GI16EC078** has satisfactorily completed the course activity (Seminar/Project) in **DATABASE AND MANAGEMENT SYSTEM**S course ( Course code:**16EC754**) . It can be considered as a bonafide work carried out in partial fulfillment for the academic requirement of **7th** Semester B.E. (**Electronics and Communication**) prescribed by KLS Gogte Institute of Technology, Belagavi during the academic year 2019- 2020.

The report has been approved as it satisfies the academic requirements in respect of Assignment ( Mini Project) prescribed for the said Degree.

Signature of the Faculty Member        Signature of the HOD

Date:13/11/2019

**GUI FOR ADDRESS BOOK DATABASE**

|  |  |  |
| --- | --- | --- |
| **S NO.** | **CONTENT** | **PAGE NO.** |
|  | Introduction |  |
|  | Address Book Management System |  |
|  | Database Selection |  |
|  | Create Database(address\_book) |  |
|  | Tkinter for GUI |  |
|  | System GUI Design |  |
|  | Conclusion |  |

**GUI FOR ADDRESS BOOK DATABASE**

1. **Introduction**

Graphical User Interface (GUI) for Address Book Management System deals with all kind of details like first name, last name, address details, city, state and zip code. It tracks all the details of a person from the name to the zip code which can be used for the information on the individual. The design can facilitate us to explore the information to know the address of the people listed in the database. The Address book management system is an automated version of manual Yellow Pages. In case of manual system they need a lot of time, manpower etc. Here almost all work is computerized. So the accuracy is maintained. Maintaining backup is very easy. It can do with in a few minutes. This GUI has mainly four operations ,they are, Add Records to the database, Delete Record to delete the content in the database system by using primary key, Update Record to edit an existing record and Show record to list the contents of the database.

1. **Address Book Management System**

Address Book Management System can be said a software version of ‘Yellow Pages’ which records the addresses of the people in a region. It was often used by travelers and people new to the region to find a person’s address. Here this database manages simple address with first and last names along with state , city and zip code. These information are stored in the database using a simple GUI. The GUI has separate text boxes for each field and four functional buttons which performs operation of adding, deleting, updating and displaying the contents of the database.

1. **Database Selection**

There are a variety of databases that we can select from the market. The widely used databases are SQLite3, Microsoft SQL, Oracle and MySQL. In this project we use SQLite3 database which is easy to use and is already built-in in Python3, user friendly, reliable and in this case for a simple user interface which uses Tkinter in Python. Moreover Python can access SQLite3 database directly and Python script is able to run faster with database and the processing time will definitely be shorter. The pre-school does not require complex and costly software for its database management system hence SQLite3 is the database for this project.

1. **Create the Database (address\_book)**

The database address\_book is created using the SQLite3 in PyCharm and also table is created as addresses as follows

# Create a database or connect to one  
conn = sqlite3.connect('address\_book.db')  
  
# Create cursor  
c = conn.cursor()  
  
# Create Table  
c.execute("""CREATE TABLE addresses (  
 first\_name text,  
 last\_name text,  
 address text,  
 city text,  
 state text,  
 zipcode integer)  
""")

conn.commit()  
  
# Close Connection  
conn.close()

This database file is stored as .db in the directory.

1. **Tkinter for GUI**

The user interface for the address book uses the platform Tkinter which is built-in in python3 and is easy to access in any python IDE (in this case PyCharm). Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with Tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using Tkinter is an easy task. To create a tkinter:

* Importing the module – tkinter
* Create the main window (container)
* Add any number of widgets to the main window
* Apply the event Trigger on the widgets.

1. **System GUI Design**

The GUI window that accepts the entries and details of the persons is designed by using the Tkinter. The design has two window as ‘Address Book App’ as the main window which has the following text boxes as shown

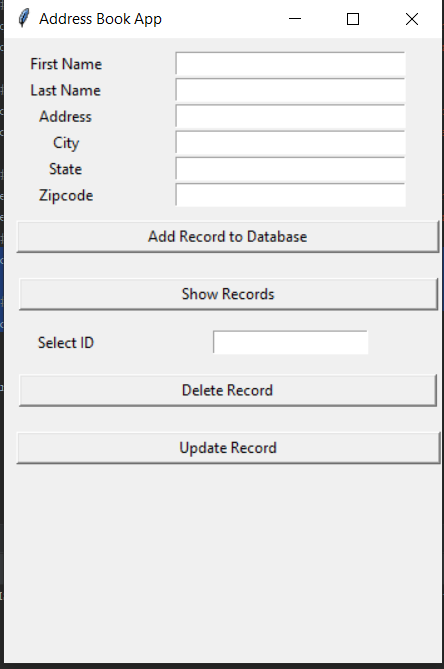


Fig.1

Add record to database adds the information filled in the text boxes to the address\_book database while the Show Records displays the list of content in the table addresses. Also, the deletion of records is done by Delete Record button. Firstly, at the Address book app page , all the fields except the Select ID are filled for Addition of record to the database. This is to ensure that all the relevant information are collected and inserted into the database in an orderly manner. As for the Update record button it has different window as shown

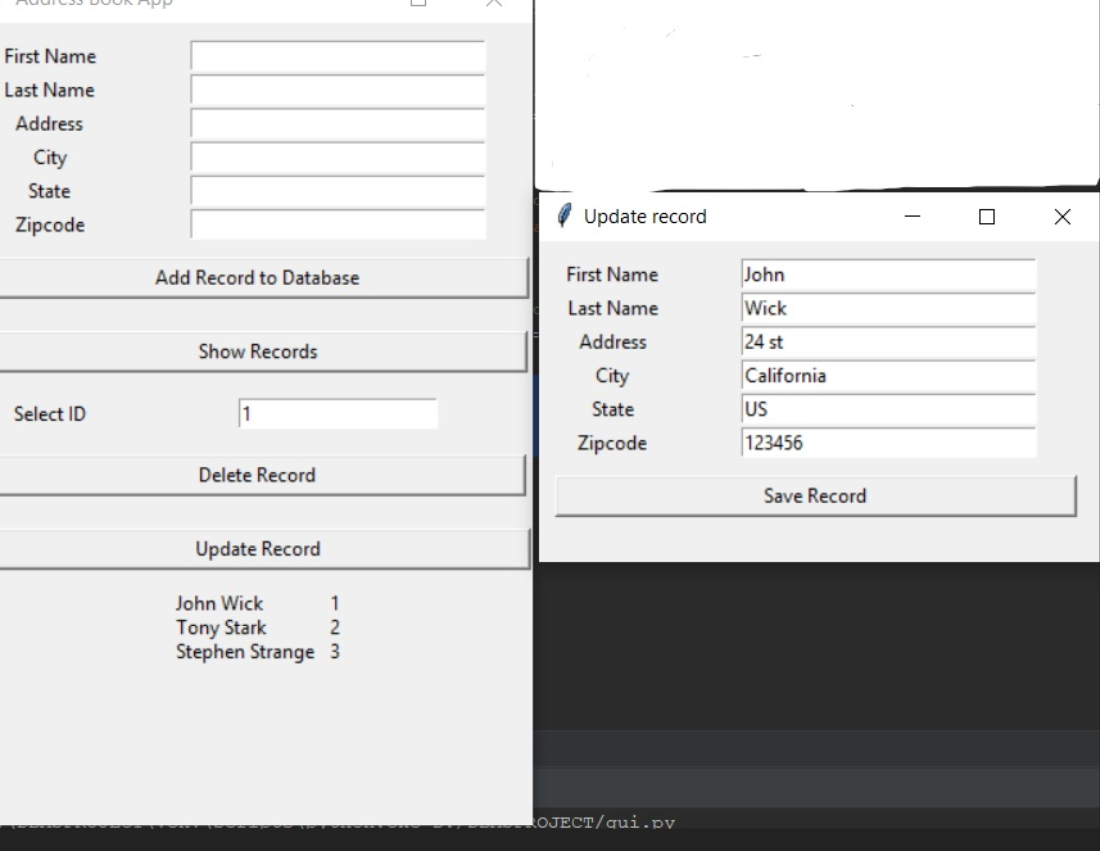


Fig.2

1. **Python Code**

---------------------GUI for address book-------------------------------------

from tkinter import \*

from tkinter import Entry

from PIL import Image,ImageTk

import sqlite3

root = Tk()

root.geometry("350x500")

root.title("Address Book App")

# Database

# Create a database or connect to one

conn = sqlite3.connect('address\_book.db')

# Create cursor

c = conn.cursor()

# Create Table

'''

c.execute("""CREATE TABLE addresses (

first\_name text,

last\_name text,

address text,

city text,

state text,

zipcode integer)

""")

'''

# create a edit func to update record

def update():

# Create a database or connect to one

conn = sqlite3.connect('address\_book.db')

# Create cursor

c = conn.cursor()

record\_id = delete\_box.get()

c.execute('''UPDATE addresses SET

first\_name = :first,

last\_name = :last,

address = :address,

city = :city,

state = :state,

zipcode = :zipcode

WHERE oid = :oid''',

{

'first': f\_name.get(),

'last': l\_name.get(),

'address': address.get(),

'city': city.get(),

'state': state.get(),

'zipcode': zipcode.get(),

'oid': record\_id

}

)

# Commit Changes

conn.commit()

# Close Connection

conn.close()

editor.destroy()

# Create Update record

def edit():

global editor

editor = Tk()

editor.geometry("350x200")

editor.title("Update record")

# Create a database or connect to one

conn = sqlite3.connect('address\_book.db')

# Create cursor

c = conn.cursor()

record\_id = delete\_box.get()

# QUery the Database

c.execute("SELECT \* FROM addresses WHERE oid = " + record\_id)

records = c.fetchall()

# Create Global Variables for text box names to use in update func

global f\_name

global l\_name

global address

global city

global state

global zipcode

# Create text Boxes

f\_name = Entry(editor, width=30)

f\_name.grid(row=0, column=1, padx=20, pady=(10, 0))

l\_name = Entry(editor, width=30)

l\_name.grid(row=1, column=1, padx=20)

address = Entry(editor, width=30)

address.grid(row=2, column=1, padx=20)

city = Entry(editor, width=30)

city.grid(row=3, column=1, padx=20)

state = Entry(editor, width=30)

state.grid(row=4, column=1, padx=20)

zipcode = Entry(editor, width=30)

zipcode.grid(row=5, column=1, padx=20)

# Create textbox labels

f\_name1 = Label(editor, text="First Name")

f\_name1.grid(row=0, column=0, pady=(10, 0))

l\_name1 = Label(editor, text="Last Name")

l\_name1.grid(row=1, column=0)

address1 = Label(editor, text="Address")

address1.grid(row=2, column=0)

city1 = Label(editor, text="City")

city1.grid(row=3, column=0)

state1 = Label(editor, text="State")

state1.grid(row=4, column=0)

zipcode1 = Label(editor, text="Zipcode")

zipcode1.grid(row=5, column=0)

# Loop thru results

for record in records:

f\_name.insert(0, record[0])

l\_name.insert(0, record[1])

address.insert(0, record[2])

city.insert(0, record[3])

state.insert(0, record[4])

zipcode.insert(0, record[5])

# create a save button

edit\_btn = Button(editor, text="Save Record", command=update)

edit\_btn.grid(row=11, column=0, columnspan=2, padx=10, pady=10, ipadx=125)

# Create function to delete records

def delete():

# Create a database or connect to one

conn = sqlite3.connect('address\_book.db')

# Create cursor

c = conn.cursor()

# Delete a record

c.execute("DELETE FROM addresses WHERE oid = " + delete\_box.get())

# Commit Changes

conn.commit()

# Close Connection

conn.close()

delete\_box.delete(0, END)

# Create Submit function

def submit():

# Create a database or connect to one

conn = sqlite3.connect('address\_book.db')

# Create cursor

c = conn.cursor()

# Insert into table

c.execute("INSERT INTO addresses VALUES (:f\_name, :l\_name, :address, :city, :state, :zipcode)",

{

'f\_name': f\_name.get(),

'l\_name': l\_name.get(),

'address': address.get(),

'city': city.get(),

'state': state.get(),

'zipcode': zipcode.get()

}

# Commit Changes

conn.commit()

# Close Connection

conn.close()

# Clear text boxes

f\_name.delete(0, END)

l\_name.delete(0, END)

address.delete(0, END)

city.delete(0, END)

state.delete(0, END)

zipcode.delete(0, END)

# create query Function

def query():

# Create a database or connect to one

conn = sqlite3.connect('address\_book.db')

# Create cursor

c = conn.cursor()

# QUery the Database

c.execute("SELECT \*, oid FROM addresses")

records = c.fetchall()

#print(records)

# Loop thruogh results

print\_records = ''

for record in records:

print\_records += str(record[0]) + " " + str(record[1]) + "\t" + str(record[6]) + "\n"

query\_label = Label(root,text=print\_records)

query\_label.grid(row=12, column=0, columnspan=2)

# Commit Changes

conn.commit()

# Close Connection

conn.close()

# Create text Boxes

f\_name = Entry(root,width=30)

f\_name.grid(row=0,column=1,padx=20, pady=(10,0))

l\_name = Entry(root,width=30)

l\_name.grid(row=1,column=1,padx=20)

address = Entry(root,width=30)

address.grid(row=2,column=1,padx=20)

city = Entry(root,width=30)

city.grid(row=3,column=1,padx=20)

state = Entry(root,width=30)

state.grid(row=4,column=1,padx=20)

zipcode = Entry(root,width=30)

zipcode.grid(row=5,column=1,padx=20)

delete\_box = Entry(root, width =20)

delete\_box.grid(row=9, column=1, padx=20)

# Create textbox labels

f\_name1 = Label(root, text="First Name")

f\_name1.grid(row=0,column=0, pady=(10,0))

l\_name1 = Label(root, text="Last Name")

l\_name1.grid(row=1,column=0)

address1 = Label(root, text="Address")

address1.grid(row=2,column=0)

city1 = Label(root, text="City")

city1.grid(row=3,column=0)

state1 = Label(root, text="State")

state1.grid(row=4,column=0)

zipcode1 = Label(root, text="Zipcode")

zipcode1.grid(row=5,column=0)

delete\_box\_label = Label(root,text="Select ID")

delete\_box\_label.grid(row=9,column=0,pady = 5)

# Create Submit Button

submit\_btn = Button(root, text="Add Record to Database", command = submit)

submit\_btn.grid(row=6, column=0, columnspan=2, pady=10, padx=10, ipadx=100)

# create a query button

query\_btn = Button(root, text="Show Records", command=query)

query\_btn.grid(row=7, column=0, columnspan=2, padx=10, pady=10, ipadx=125)

# create a delete button

delete\_btn = Button(root, text="Delete Record", command=delete)

delete\_btn.grid(row=10, column=0, columnspan=2, padx=10, pady=10, ipadx=125)

# create an Update button

edit\_btn = Button(root, text="Update Record", command=edit)

edit\_btn.grid(row=11, column=0, columnspan=2, padx=10, pady=10, ipadx=125)

# Commit Changes

conn.commit()

# Close Connection

conn.close()

root.mainloop()

1. **Conclusion**

Simplicity is never simple. As we have seen in this project, the process of creating a user friendly and straightforward platform that facilitates the administrator’s job is one filled with complexity. From understanding user requirements to system design and finally system prototype and finalization, every step requires in-depth understanding and commitment towards achieving the objectives of the project. Although the database management module is not fully integrated to the system and used on real time, the system prototype demonstrates easy navigation and data are stored in a systematic way. Overall, efficiency has improved and work processes simplified. Although all the objectives have been met, the system still has room for improvement. The system is robust and flexible enough for future upgrade using advanced technology and devices.