ALUMNI CONNECT USING PYTHON

A PROJECT REPORT

Submitted by

AKSHAT MITTAL [RA2211003010790]
RITVIK RAJVANSHI [RA2211003010792]
YUGAM SHAH [RA2211003010796]

for the course 21CSC203P Advance Programming Practice

Under the Guidance of

Dr. ARUL MURUGAN A.

Assistant Professor, Department of Computing Technologies

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

NOVEMBER 2023

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that the 21CSC203P Advance Programming Practice course project report titled **“ALMUNI CONNECT USING JAVA”** is the bonafide work done by **AKSHAT MITTAL [RA2211003010790], RITVIK RAJVANSHI [RA2211003010792] & YUGUAM SHAH [RA2211003010796] of II Year/III Sem B.Tech(CSE)** who carried out the mini project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

SIGNATURE	**HEAD OF THE DEPARTMENT**
Faculty In-Charge	**Dr. M. Pushpalatha,**
Dr. Arul Murugan A.	Professor and Head,
Assistant Professor,	Department of Computing Technologies,
Department of Computing Technologies,	SRMIST.
SRMIST.	

ABSTRACT

The GUI-Based Alumni Connect Project is a dynamic and user-centric software application designed to bridge the gap between educational institutions and their alumni. Leveraging the power of Python and modern GUI libraries, this project provides an innovative platform for educational institutions to establish and nurture lasting relationships with their alumni network. The system empowers users to stay connected, share experiences, access resources, and contribute to the growth and development of their alma mater.

Key features of the Alumni Connect Project include user-friendly registration and authentication, interactive alumni profiles, event management, mentorship programs, discussion forums, and resource sharing. Python, with the support of GUI libraries like Tkinter or PyQt, serves as the primary programming language to create a seamless and visually appealing user interface.

In a rapidly changing world, where networking and knowledge sharing are paramount, this project offers a modern and efficient solution for institutions and alumni to remain connected, fostering collaboration, and mutual growth. It demonstrates the application of software development principles to facilitate meaningful connections and engagement within the academic community.

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T. V. Gopal**, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr. M. Pushpalatha**, Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinator, **Dr. ARUL MURUGAN A.,** Assistant Professor, Department of Computing Technologies, SRM Institute of Science andTechnology for their inputs during the project reviews and support.

TABLE OF CONTENTS

Sr. No.	Title	Page No.
1.	Introduction	6
2.	Literature Survey	7 – 8
3.	Requirement Analysis	9 – 11
4.	Architecture and Design	12 – 14
5.	Implementation	15 – 28
6.	Experiment Result and Analysis	29
7.	Future Scope	30
8.	Conclusion	31
9.	References	32

INTRODUCTION

In the ever-evolving landscape of education, maintaining a strong and vibrant alumni network is essential for the continued growth and success of educational institutions. These networks not only reflect the accomplishments of a school's graduates but also serve as invaluable resources for current students and the institution itself. The GUI-Based Alumni Connect Project, developed using Python, emerges as a timely and innovative solution that leverages technology to foster and nurture these critical alumni relationships.

This project is born from the recognition that the connection between alumni and their alma mater extends beyond graduation day. It acknowledges the need for a dedicated platform that enables alumni to remain engaged, share experiences, and give back to the institution that played a pivotal role in their educational journey. At the same time, it allows educational institutions to harness the wealth of knowledge, experience, and expertise that their alumni possess. By providing a user-friendly, visually appealing graphical user interface, this project aims to create a space where these interactions can flourish.

The GUI-Based Alumni Connect Project introduces a comprehensive set of features designed to facilitate a thriving alumni community. From user-friendly registration and authentication to interactive alumni profiles and event management, the system aims to encourage participation, mentorship, discussion, and resource sharing. Python, as the primary programming language, is a versatile and powerful tool for creating the graphical user interface that forms the backbone of this platform, ensuring that the user experience is efficient, engaging, and dynamic.

In a world where connections, networking, and knowledge sharing have become more critical than ever, this project stands at the intersection of tradition and innovation. It bridges the gap between past and present, allowing educational institutions to stay connected with their graduates while empowering alumni to remain closely linked to their academic roots. As we delve deeper into the capabilities and functionalities of this GUI-Based Alumni Connect Project, we'll discover how it embodies the principles of modern software development and serves as a testament to the enduring value of educational bonds.

LITERATURE SURVEY

A literature survey for a GUI-based Alumni Connect Project using Python involves reviewing relevant research and projects related to alumni management systems, graphical user interface (GUI) development, and Python programming. Here's a comprehensive literature survey to provide insights into the context and requirements of such a project:

1. **Alumni Management Systems:**

   - Research existing alumni management systems to understand their functionalities, user interfaces, and data management capabilities. This can help in identifying best practices and potential areas for improvement.

2. **GUI Development with Python:**

   - Explore literature on GUI development using Python and relevant libraries such as Tkinter, PyQt, or Kivy. Understand the design principles and coding techniques for creating user-friendly interfaces.

3. **User Registration and Authentication:**

   - Review literature on user registration and authentication systems to ensure the security and privacy of alumni data. This may include studies on password hashing, user access controls, and multi-factor authentication.

4. **Alumni Data Management:**

   - Study research related to alumni data management, database design, and data structures. This can help in defining how alumni profiles and information should be stored and accessed within the system.

5. **Event Management and Networking:**

   - Investigate literature on event management and networking platforms for alumni. Learn about features that encourage alumni engagement, such as event creation, RSVP tracking, and discussion forums.

6. **Alumni Relationship Building:**

   - Explore research on strategies for building and nurturing alumni relationships, including mentorship programs, alumni chapters, and the exchange of professional and personal experiences.

7. **Data Security and Privacy:**

   - Review studies on data security and privacy to understand the best practices for safeguarding alumni data, complying with data protection regulations, and mitigating security risks.

8. **Feedback and User Experience:**

   - Investigate studies on user experience (UX) and user interface (UI) design principles. Understand how to create an intuitive, visually appealing, and responsive user interface that encourages user engagement.

9. **Integration with Other Systems:**

   - Explore literature on system integration, as the alumni platform may need to connect with other institutional databases or third-party services for seamless data sharing and functionality.

10. **User Feedback and Success Stories:**

    - Search for user feedback and success stories related to similar alumni management systems. This can provide insights into what users appreciate and where improvements can be made.

11. **Case Studies of Alumni Platforms:**

    - Analyse case studies of institutions that have successfully implemented alumni platforms. Identify their challenges, strategies, and outcomes in managing and engaging their alumni.

12. **Emerging Technologies and Trends:**

    - Stay updated on emerging technologies and trends in alumni engagement and networking, such as the use of social media, AI, and mobile applications.

By conducting a thorough literature survey, you can gain a deeper understanding of the concepts, technologies, and best practices that are relevant to the development of a GUI-based Alumni Connect Project using Python. This knowledge will inform the design, features, and functionality of the system, ensuring it aligns with the needs and expectations of educational institutions and their alumni.

REQUIREMENT ANALYSIS

Requirement analysis for a GUI-based Alumni Connect Project using Python is crucial to outline the system's functionalities, features, and constraints. It helps ensure that the project aligns with the objectives of connecting alumni and educational institutions effectively. Here's a comprehensive requirement analysis for such a project:

Functional Requirements:

1. **User Registration and Authentication:**

   - Users should be able to register by providing personal information, including name, email, graduation year, and program details.
   - Secure user authentication is essential to protect alumni data.

2. **User Profiles:**

   - Alumni should be able to create and manage their profiles, including adding educational and professional details, contact information, and a profile picture.
   - Users should have the option to make certain profile details private or public.

3. **Alumni Directory:**

   - The system should maintain an alumni directory that allows users to search and view profiles of fellow alumni.
   - Advanced search and filter options should be available, including program, graduation year, and location.

4. **Event Management:**

   - Alumni and institutions should be able to create and manage events, including reunions, workshops, and networking sessions.
   - Event details, RSVP tracking, and reminders should be part of the system.

5. **Discussion Forums:**

   - The platform should host discussion forums where alumni can engage in professional and personal discussions, share experiences, and seek advice.
   - Forums should support multimedia content, such as images and documents.

6. **Mentorship Programs:**

    - Institutions can facilitate mentorship programs where alumni can offer guidance and support to current students.
    - Users should be able to express their interest in becoming mentors or seeking mentorship.

7. **Resource Sharing:**

    - Alumni should be able to share resources such as job postings, research papers, and articles.
    - The system should categorize and display these resources effectively.

8. **User Notifications:**

    - Users should receive notifications about upcoming events, new forum discussions, and relevant activities.
    - Notifications can be sent through email or within the platform.

9. **Admin Panel:**

    - An admin panel should be available for administrators to manage user accounts, events, and content.
    - Admins should be able to moderate forum discussions and resolve disputes.

Non-Functional Requirements:

1. **Security and Privacy:**

    - Data security is paramount. User data, authentication, and communication should be encrypted.
    - Users should have granular control over their data privacy settings.

2. **Scalability and Performance:**

    - The system should be able to handle a growing user base and a substantial amount of data.
    - Response times should be quick to ensure a seamless user experience.

3. **User Experience (UX) and Design:**

    - The user interface should be intuitive, visually appealing, and responsive.
    - Accessibility features should be considered to accommodate a diverse user base.

4. **Data Backup and Recovery:**

    - Regular data backups and a robust recovery plan should be in place to prevent data loss.

5. **Compliance and Data Protection:**

    - Ensure compliance with data protection regulations, such as GDPR or relevant local laws.
    - Implement features for users to request data deletion or export.

6. **Cross-Platform Compatibility:**

    - The GUI should be compatible with various devices, browsers, and operating systems.
    - A mobile application version of the platform can be considered for broader accessibility.

7. **Feedback Mechanism:**

    - Implement a feedback system for users to report issues, suggest improvements, or provide general comments.

By defining these functional and non-functional requirements, you create a roadmap for the development of a GUI-based Alumni Connect Project using Python. These requirements will serve as the basis for design, implementation, testing, and ongoing system maintenance.

ARCHITECTURE AND DESIGN

Designing the architecture and user interface for a GUI-based Alumni Connect Project using Python requires careful planning to ensure the system's functionality, usability, and scalability. Below is a high-level architecture and design overview for such a project:

Architecture:

The architecture of the Alumni Connect Project should be structured to accommodate the system's various components and ensure efficient data flow. Here's a typical architectural outline:

1. **Client-Side:**

    - The user interacts with the system through a graphical user interface (GUI) created using Python and GUI libraries like Tkinter, PyQt, or Kivy.
    - The GUI sends user requests and inputs to the server for processing.

2. **Server-Side:**

    - A server-side application is responsible for handling user requests and managing the system's core functionalities.
    - It communicates with the database to retrieve and store user data and content.
    - The server manages user authentication, user profiles, event creation, discussion forums, mentorship programs, and resource sharing.

3. **Database:**

    - A relational database management system (RDBMS), such as MySQL or PostgreSQL, is used to store user data, event details, forum discussions, mentorship program data, and shared resources.
    - The database is responsible for ensuring data integrity and security.

4. **External Services:**

    - The system may integrate with external services for features like user authentication (e.g., OAuth via social media accounts), email notifications, and text messaging services.
    - External services help enhance the user experience and functionality.

Design:

The design of the Alumni Connect Project's GUI should prioritize user-friendliness, responsiveness, and a visually appealing interface. Here are design considerations for the various components of the system:

1. **User Registration and Authentication:**

   - Design a user registration form that collects essential information.
   - Implement an authentication page where users can log in using their credentials or through social media accounts.
   - Provide password reset and recovery options.

2. **User Profiles:**

   - Create a user-friendly profile editing interface where alumni can add, edit, or remove personal and professional information.
   - Allow users to upload profile pictures.
   - Implement privacy settings for certain profile details.

3. **Alumni Directory:**

   - Design a search and filter interface for finding fellow alumni.
   - Display user profiles in a visually appealing format, including profile pictures and relevant details.

4. **Event Management:**

   - Create an event creation form where institutions and alumni can input event details.
   - Design an event listing page with RSVP functionality.
   - Include a calendar view for event schedules.

5. **Discussion Forums:**

   - Design an intuitive and organized forum structure with categories and threads.
   - Provide options for users to create new discussion topics, reply to threads, and upload media content.
   - Include a notification system to alert users of new forum activity.

6. **Mentorship Programs:**

    - Create a mentorship program interface where alumni can express their interest in being mentors or mentees.
    - Enable users to browse and connect with potential mentors/mentees.

7. **Resource Sharing:**

    - Implement a resource sharing platform where users can upload and categorize resources.
    - Design an organized resource library with search and filter capabilities.

8. **User Notifications:**

    - Create a notification centre that alerts users about event updates, forum discussions, mentorship requests, and other relevant activities.

9. **Admin Panel:**

    - Design an administrative dashboard that allows administrators to manage user accounts, events, and content.
    - Include moderation features for forum discussions and content.

10. **Cross-Platform Compatibility:**

    - Ensure that the GUI is responsive and user-friendly on various devices and screen sizes, including mobile devices and tablets.
    - Consider a mobile application version for enhanced mobile accessibility.

11. **Data Security and Privacy:**

    - Implement secure data transmission and storage, including encryption and access controls.
    - Allow users to set privacy preferences for their profile and data.

12. **Feedback Mechanism:**

    - Include a feedback form or mechanism for users to provide suggestions, report issues, or request assistance.

By adhering to these architectural and design principles, the GUI-based Alumni Connect Project using Python can provide a user-friendly, secure, and feature-rich platform for educational institutions and alumni to engage and collaborate effectively.

IMPLEMENTATION

Implementing a GUI-based Alumni Connect Project using Python involves coding the functionalities, creating the graphical user interface (GUI), and integrating with a database. Here is a simplified outline of how you can implement key components of the project:

1. **Set Up the Development Environment:**

    a. Install Python: Ensure that Python is installed on your system.
    b. Choose a GUI library: Select a GUI library such as Tkinter, PyQt, or Kivy to create the interface.
    c. Set up a database: Install and configure an RDBMS like MySQL or PostgreSQL for data storage.

2. **Design the Database Schema:**

    a. Define the database tables for user profiles, events, forum discussions, mentorship programs, and shared resources.
    b. Establish relationships between tables to link user data with events, forum posts, mentorship records, and resource uploads.

3. **Create the User Interface:**

    a. Design and code the GUI screens using your chosen GUI library.
    b. Develop screens for user registration, login, profile management, event creation, discussion forums, mentorship programs, resource sharing, and notification management.

4. **User Registration and Authentication:**

    a. Implement user registration functionality, including input validation and secure password hashing.
    b. Develop a login system with authentication checks against user data in the database.
    c. Include password recovery and reset options for users.

5. **User Profiles:**

    a. Enable users to create and manage their profiles, allowing for personal and professional information, privacy settings, and profile pictures.
    b. Implement validation checks for profile updates.

6. **Alumni Directory:**

    a. Develop a search and filter feature for finding and viewing alumni profiles.
    b. Create profile detail pages for each alumni member.

7. **Event Management:**

    a. Design an event creation form for institutions and alumni to add events with details.
    b. Create an event listing page with RSVP functionality.
    c. Implement a calendar view for event schedules.

8. **Discussion Forums:**

    a. Develop the forum structure, including categories and threads.
    b. Allow users to create new discussion topics, reply to threads, and upload multimedia content.
    c. Implement a notification system for new forum activity.

9. **Mentorship Programs:**

    a. Design and code the mentorship program interface for expressing interest in mentoring or seeking mentorship.
    b. Enable users to browse and connect with potential mentors or mentees.

10. **Resource Sharing:**

    a. Create an interface for resource uploads, categorization, and library management.
    b. Implement search and filter capabilities for the resource library.

11. **User Notifications:**

    a. Develop a notification centre to alert users about event updates, forum discussions, mentorship requests, and other activities.

12. **Admin Panel:**

    a. Build an administrative dashboard for administrators to manage user accounts, events, and content.
    b. Include moderation features for forum discussions and content.

13. Data Security and Privacy:

a. Implement secure data transmission and storage, including encryption and access controls.
b. Allow users to set privacy preferences for their profile and data.

14. Cross-Platform Compatibility:

a. Ensure that the GUI is responsive and user-friendly on various devices and screen sizes, including mobile devices and tablets.
b. Consider developing a mobile application version for enhanced mobile accessibility.

15. Feedback Mechanism:

a. Include a feedback form or mechanism for users to provide suggestions, report issues, or request assistance.

16. Testing and Quality Assurance:

a. Perform thorough testing, including unit testing, integration testing, and user acceptance testing, to identify and fix any issues.
b. Verify that the system meets the functional and non-functional requirements.

17. Deployment and Maintenance:

a. Deploy the system to a web server or cloud platform, ensuring 24/7 availability.
b. Regularly maintain and update the system based on user feedback and emerging requirements.

Remember to follow best practices in software development, including code documentation, version control, and data backup procedures. Additionally, engage with potential users and stakeholders to gather feedback for continuous improvement.

Code:

```python
#Importing the modules
import tkinter
import tkinter.ttk as ttk
from tkinter import *
import sqlite3
import tkinter.messagebox as tkMessageBox
```

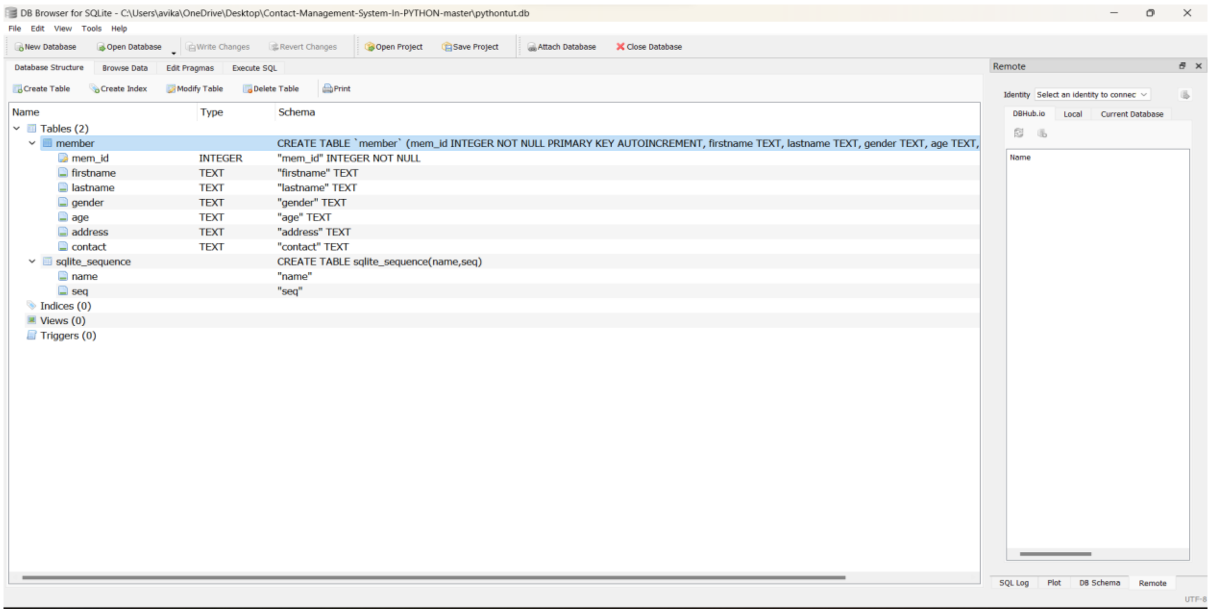
```python
root = Tk()
root.title("AlumniConnect")
root.geometry("780x400+0+0")
root.config(bg="Pink")


# Variables required for storing the values
f_name = StringVar()
m_name = StringVar()
l_name = StringVar()
age = StringVar()
home_address = StringVar()
gender = StringVar()
phone_number = StringVar()


#Function for resetting the values
def Reset():
    f_name.set("")
    m_name.set("")
    l_name.set("")
    gender.set("")
    age.set("")
    home_address.set("")
    phone_number.set("")


# For creating the database and the table
def Database():
    connectn = sqlite3.connect("contactdata.db")
    cursor = connectn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS `contactinformation` (id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, first_name TEXT, middle_name TEXT, last_name TEXT, gender TEXT, age TEXT, home_address TEXT, phone_number TEXT)")
    cursor.execute("SELECT * FROM `contactinformation` ORDER BY `last_name` ASC")
    fetchinfo = cursor.fetchall()
    for data in fetchinfo:
        tree.insert('', 'end', values=(data))
    cursor.close()
    connectn.close()

#Function for exiting the system
def Exit():
    O = tkinter.messagebox.askyesno("Alumni", "Do you want to exit the system")
    if O > 0:
        root.destroy()
    return
```

```
#Insert query for inserting the value in database Table
def Submit():
    if f_name.get() == "" or m_name.get() == "" or l_name.get() == "" or
gender.get() == "" or age.get() == "" or home_address.get() == "" or
phone_number.get() == "":
        msgg = tkMessageBox.showwarning('', 'Please Complete All the Fields',
icon="warning")
    else:
        tree.delete(*tree.get_children())
    connectn = sqlite3.connect("contactdata.db")
    cursor = connectn.cursor()

    cursor.execute("INSERT INTO `contactinformation` (first_name, middle_name,
last_name, gender, age, home_address, phone_number ) VALUES(?, ?, ?, ?, ?, ?,
?)", (str(f_name.get()), str(m_name.get()), str(l_name.get()),
str(gender.get()), int(age.get()), str(home_address.get()),
    int(phone_number.get())))

    connectn.commit()
    cursor.execute("SELECT * FROM `contactinformation` ORDER BY `last_name`
ASC")
    fetchinfo = cursor.fetchall()

    for data in fetchinfo:
        tree.insert('', 'end', values=(data))
    cursor.close()
    connectn.close()
    f_name.set("")
    m_name.set("")
    l_name.set("")
    gender.set("")
    age.set("")
    home_address.set("")
    phone_number.set("")


#Update Query for updating the table in the database
def Update():
    if gender.get() == "":
        msgg = tkMessageBox.showwarning('', 'Please Complete The Required
Field', icon="warning")
    else:
        tree.delete(*tree.get_children())
    connectn = sqlite3.connect("contactdata.db")
    cursor = connectn.cursor()
    cursor.execute("UPDATE `contactinformation` SET `first_name` = ?,
`middle_name` = ? , `last_name` = ?, `gender` =?, `age` = ?, `home_address` =
?, `phone_number` = ? WHERE `id` = ?",
    (str(f_name.get()), str(m_name.get()), str(l_name.get()),
str(gender.get()), int(age.get()), str(home_address.get()),
    str(phone_number.get()), int(id)))
    connectn.commit()
    cursor.execute("SELECT * FROM `contactinformation` ORDER BY `last_name`
ASC")
    fetchinfo = cursor.fetchall()
    for data in fetchinfo:
        tree.insert('', 'end', values=(data))
    gender1 = gender.get()
    if not gender1:
        tkMessageBox.showerror("Please select the gender")

    cursor.close()
    connectn.close()

    f_name.set("")
    m_name.set("")
    l_name.set("")
    gender.set("")
    age.set("")
    home_address.set("")
    phone_number.set("")


#Module for the update contact form window
def UpdateContact(event):
    global id, UpdateWindow
    curItem = tree.focus()
    contents = (tree.item(curItem))
    item = contents['values']
    id = item[0]
    f_name.set("")
    m_name.set("")
    l_name.set("")
    gender.set("")
    age.set("")
    home_address.set("")
    phone_number.set("")


    f_name.set(item[1])
    m_name.set(item[2])
    l_name.set(item[3])

    age.set(item[5])
    home_address.set(item[6])
    phone_number.set(item[7])
    UpdateWindow = Toplevel()
    UpdateWindow.title("Contact Information")
    UpdateWindow.geometry("500x520+0+0")
    UpdateWindow.resizable(0, 0)
    if 'Opennewwindow' in globals():
        Opennewwindow.destroy()

    # FRAMES
    #module is for the frame, labels, text entry, and button for update
contact form window
    FormTitle = Frame(UpdateWindow)
    FormTitle.pack(side=TOP)
    ContactForm = Frame(UpdateWindow)
    ContactForm.pack(side=TOP, pady=10)
    RadioGroup = Frame(ContactForm)
    Male = Radiobutton(RadioGroup, text="Male", variable=gender, value="Male",
font=('arial', 14)).pack(side=LEFT)
    Female = Radiobutton(RadioGroup, text="Female", variable=gender,
value="Female", font=('arial', 14)).pack(side=LEFT)

# LABELS
    label_title = Label(FormTitle, text="Update the Contact Information",
font=('Arial', 17), bg="light green", width=400)
    label_title.pack(fill=X)
    label_FirstName = Label(ContactForm, text="First Name", font=('Calibri',
14), bd=5)
    label_FirstName.grid(row=0, sticky=W)

    label_MiddleName = Label(ContactForm, text="Last Name", font=('Calibri',
14), bd=5)
    label_MiddleName.grid(row=1, sticky=W)

    label_LastName = Label(ContactForm, text="Grad Year", font=('Calibri',
14), bd=5)
    label_LastName.grid(row=2, sticky=W)

    label_Gender = Label(ContactForm, text="Gender", font=('Calibri', 14),
bd=5)
    label_Gender.grid(row=3, sticky=W)

    label_Age = Label(ContactForm, text="Age", font=('Calibri', 14), bd=5)
    label_Age.grid(row=4, sticky=W)

    label_HomeAddress = Label(ContactForm, text=" Comapany", font=('Calibri',
14), bd=5)
    label_HomeAddress.grid(row=5, sticky=W)
    label_PhoneNumber = Label(ContactForm, text="Phone Number",
font=('Calibri', 14), bd=5)
    label_PhoneNumber.grid(row=6, sticky=W)



# TEXT ENTRY
    FirstName = Entry(ContactForm, textvariable=f_name, font=('Calibri', 14,
'bold'),bd=2, width=20, justify='left')
    FirstName.grid(row=0, column=1)

    MiddleName = Entry(ContactForm, textvariable=m_name, font=('Calibri', 14,
'bold'), bd=2, width=20, justify='left')
    MiddleName.grid(row=1, column=1)

    LastName = Entry(ContactForm, textvariable=l_name, font=('Calibri', 14,
'bold'), bd=2, width=20, justify='left')
    LastName.grid(row=2, column=1)

    RadioGroup.grid(row=3, column=1)

    Age = Entry(ContactForm, textvariable=age, font=('Calibri', 14, 'bold'),
bd=2, width=20, justify='left')
    Age.grid(row=4, column=1)

    HomeAddress = Entry(ContactForm, textvariable=home_address,
font=('Calibri', 14, 'bold'), bd=2, width=20,
    justify='left')
    HomeAddress.grid(row=5, column=1)

    PhoneNumber = Entry(ContactForm, textvariable=phone_number,
font=('Calibri', 14, 'bold'), bd=2, width=20,
    justify='left')
    PhoneNumber.grid(row=6, column=1)

#  Buttons
    ButtonUpdatContact = Button(ContactForm, text='Update', bd=2,
font=('Calibri', 14, 'bold'), fg="black",
    bg="lightgreen", command=Update)
    ButtonUpdatContact.grid(row=8, columnspan=2, pady=10)


#Delete query for deleting the value
def Delete():
    if not tree.selection():
        msgg = tkMessageBox.showwarning('', 'Please Select the data!',
icon="warning")
    else:
        msgg = tkMessageBox.askquestion('', 'Are You Sure You Want To Delete',
icon="warning")
    if msgg == 'yes':
        curItem = tree.focus()
        contents = (tree.item(curItem))
        item = contents['values']
        tree.delete(curItem)
    connectn = sqlite3.connect("contactdata.db")
    cursor = connectn.cursor()
    cursor.execute("DELETE FROM `contactinformation` WHERE `id` = %d" %
item[0])
    connectn.commit()
    cursor.close()
    connectn.close()

#For creating the frame, labels, text entry, and button for add new contact
form window
def MyNewContact():
    global opennewwindow
    f_name.set("")
    m_name.set("")
    l_name.set("")
    gender.set("")
    age.set("")
    home_address.set("")
    phone_number.set("")

    Opennewwindow = Toplevel()
    Opennewwindow.title("Contact Details")
    Opennewwindow.resizable(0, 0)
    Opennewwindow.geometry("500x500+0+0")
    if 'UpdateWindow' in globals():
        UpdateWindow.destroy()

#############Frames####################
    FormTitle = Frame(Opennewwindow)
    FormTitle.pack(side=TOP)
    ContactForm = Frame(Opennewwindow)
    ContactForm.pack(side=TOP, pady=10)
    RadioGroup = Frame(ContactForm)
    Male = Radiobutton(RadioGroup, text="Male", variable=gender, value="Male",
font=('Calibri', 14)).pack(side=LEFT)
    Female = Radiobutton(RadioGroup, text="Female", variable=gender,
value="Female", font=('Calibri', 14)).pack(side=LEFT)
    # ===================LABELS==============================
    label_title = Label(FormTitle, text="Adding New Contacts",
bd=12,  fg="black", bg="Lightgreen",
    font=("Calibri", 15, "bold"), pady=2)
    label_title.pack(fill=X)
    label_FirstName = Label(ContactForm, text="First Name", font=('Calibri',
14), bd=5)
    label_FirstName.grid(row=0, sticky=W)

    label_MiddleName = Label(ContactForm, text="Last Name", font=('Calibri',
14), bd=5)
    label_MiddleName.grid(row=1, sticky=W)

    label_LastName = Label(ContactForm, text="Grad Year", font=('Calibri',
14), bd=5)
    label_LastName.grid(row=2, sticky=W)

    label_Gender = Label(ContactForm, text="Gender", font=('Calibri', 14),
bd=5)
    label_Gender.grid(row=3, sticky=W)

    label_Age = Label(ContactForm, text="Age", font=('Calibri', 14), bd=5)
    label_Age.grid(row=4, sticky=W)

    label_HomeAddress = Label(ContactForm, text="Company", font=('Calibri',
14), bd=5)
    label_HomeAddress.grid(row=5, sticky=W)

    label_PhoneNumber = Label(ContactForm, text="Phone Number",
font=('Calibri', 14), bd=5)
    label_PhoneNumber.grid(row=6, sticky=W)



# ===================ENTRY===============================
    FirstName = Entry(ContactForm, textvariable=f_name, font=('Calibri', 14,
'bold'), bd=3, width=20, justify='left')
    FirstName.grid(row=0, column=1)

    MiddleName = Entry(ContactForm, textvariable=m_name, font=('Calibri', 14,
'bold'), bd=3, width=20, justify='left')
    MiddleName.grid(row=1, column=1)

    LastName = Entry(ContactForm, textvariable=l_name, font=('Calibri', 14,
'bold'), bd=3, width=20, justify='left')
    LastName.grid(row=2, column=1)

    RadioGroup.grid(row=3, column=1)

    Age = Entry(ContactForm, textvariable=age, font=('Calibri', 14, 'bold'),
bd=3, width=20, justify='left')
    Age.grid(row=4, column=1)
    HomeAddress = Entry(ContactForm, textvariable=home_address,
font=('Calibri', 14, 'bold'), bd=3, width=20, justify='left')
    HomeAddress.grid(row=5, column=1)

    PhoneNumber = Entry(ContactForm, textvariable=phone_number,
font=('Calibri', 14, 'bold'), bd=3, width=20, justify='left')
    PhoneNumber.grid(row=6, column=1)


# ==================BUTTONS==============================
    ButtonAddContact = Button(ContactForm, text='Please Save', bd=5,
font=('Calibri', 12, 'bold'), fg="black",
    bg="lightgreen", command=Submit)
    ButtonAddContact.grid(row=7, columnspan=2, pady=10)

#module for whole frame window, labels and button of contact management system
# ============================FRAMES======================================
Top = Frame(root, width=600, bd=1)
Top.pack(side=TOP)
M = Frame(root, width=650, bg="pink")
M.pack(side=BOTTOM)
F = Frame(width=7, height=8, bd=10, bg="pink")
F.pack(side=BOTTOM)
MR = Frame(M, width=100)#Right Middle frame
MR.pack(side=RIGHT, pady=10)
TableMargin = Frame(root, width=500)
TableMargin.pack(side=TOP)

# LABELS
label_title = Label(Top, text="AlumniConnect", bd=7, relief=GROOVE,
fg="Black", bg="lightgreen",
font=("Calibri", 25, "bold"), pady=3)
label_title.pack(fill=X)


# BUTTONS
Add_Button = Button(F, text='Add New Alumni', font=('Calibri',17, 'bold'),
fg="black",
bg="lightgreen", command=MyNewContact).grid(row=0, column=0, ipadx=20,
padx=30)

Delete_Button = Button(F, text='Delete The Alumni', font=('Calibri', 17,
'bold'), command=Delete,
fg="black", bg="lightgreen").grid(row=0, column=1, ipadx=20)

Exit_Button = Button(F, text='Exit System', font=('Calibri', 17, 'bold'),
command=Exit,
fg="black", bg="lightgreen").grid(row=0, column=2, ipadx=20, padx=30)
```

```python
#creating a tables in contact management system
scrollbarx = Scrollbar(TableMargin, orient=HORIZONTAL)
scrollbary = Scrollbar(TableMargin, orient=VERTICAL)
tree = ttk.Treeview(TableMargin, columns=("Id", "First Name", "Last Name",
"Grad Year", "Gender", "Age", "Company", "Phone Number"),
height=400, selectmode="extended", yscrollcommand=scrollbary.set,
xscrollcommand=scrollbarx.set)
scrollbary.config(command=tree.yview)
scrollbary.pack(side=RIGHT, fill=Y)
scrollbarx.config(command=tree.xview)
scrollbarx.pack(side=BOTTOM, fill=X)

tree.heading('Id', text="Id", anchor=W)
tree.heading('First Name', text="First Name" ,anchor=W)
tree.heading('Last Name', text="Last Name", anchor=W)
tree.heading('Grad Year', text="Grad Year", anchor=W)
tree.heading('Gender', text="Gender", anchor=W)
tree.heading('Age', text="Age", anchor=W)
tree.heading('Company', text="Company", anchor=W)
tree.heading('Phone Number', text="phone Number", anchor=W)

tree.column('#0', stretch=NO, minwidth=0, width=0)
tree.column('#1', stretch=NO, minwidth=0, width=0)
tree.column('#2', stretch=NO, minwidth=0, width=80)
tree.column('#3', stretch=NO, minwidth=0, width=120)
tree.column('#4', stretch=NO, minwidth=0, width=90)
tree.column('#5', stretch=NO, minwidth=0, width=80)
tree.column('#6', stretch=NO, minwidth=0, width=30)
tree.column('#7', stretch=NO, minwidth=0, width=120)

tree.pack()
tree.bind('<Double-Button-1>', UpdateContact)

# ============================INITIALIZATION==============================
if __name__ == '__main__':
    Database()
root.mainloop()
```

Output:

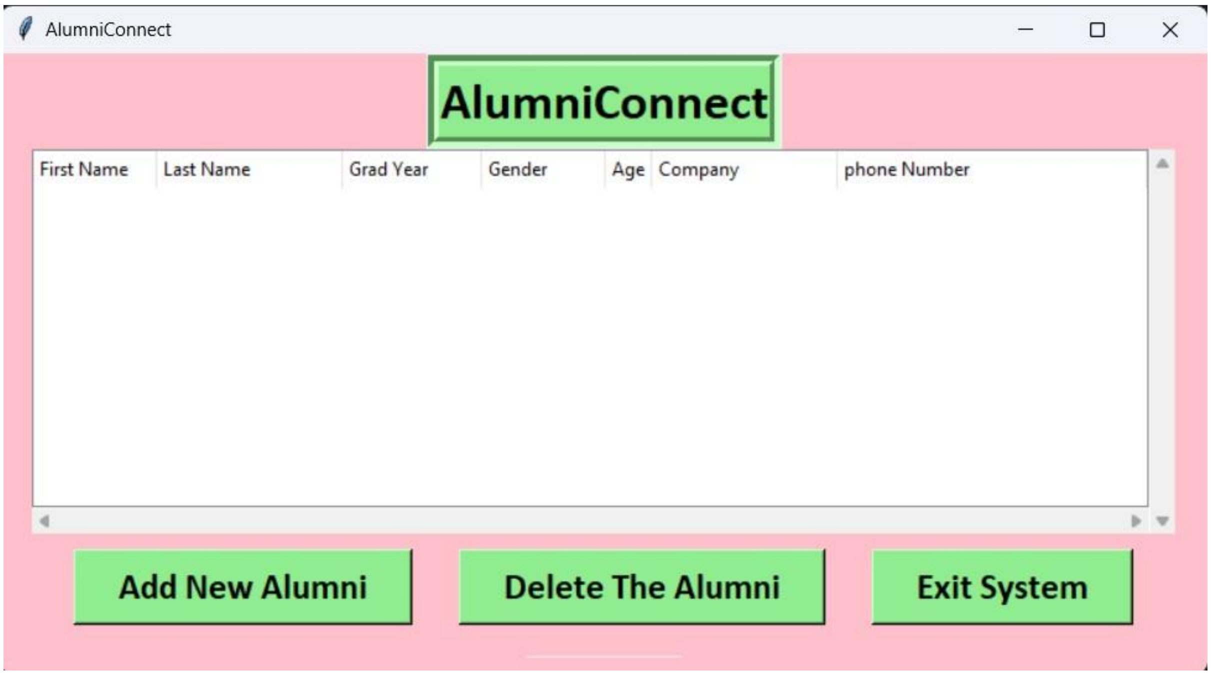
1. GUI Design



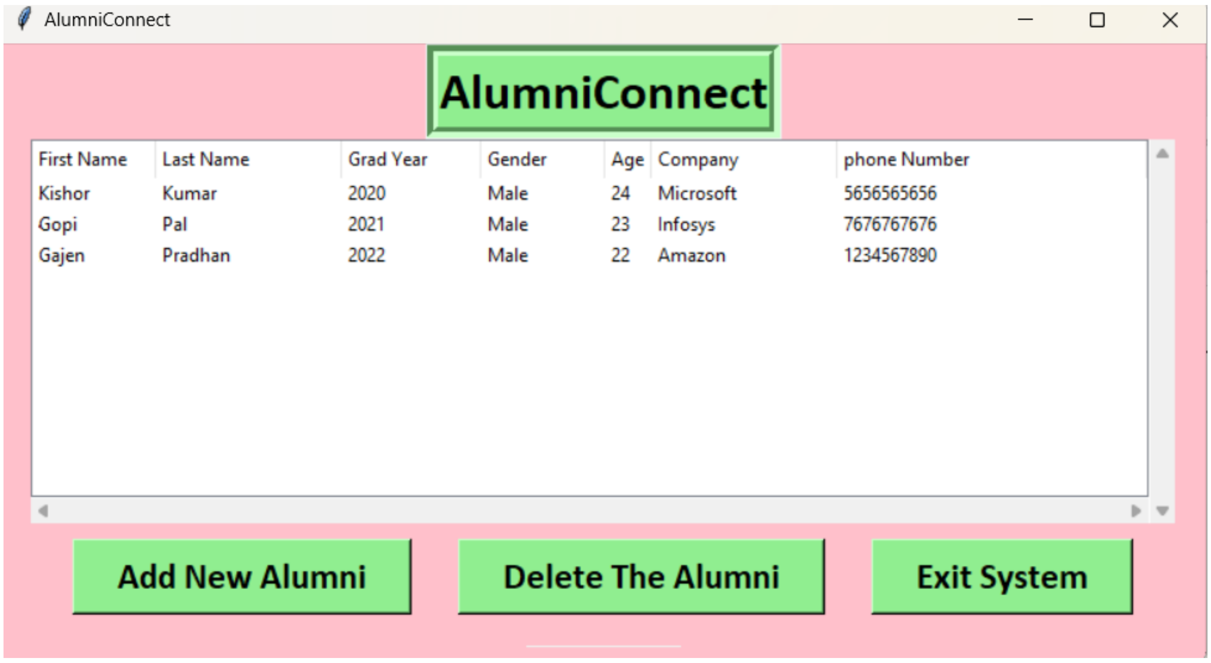
DB Browser for SQLite - C:\Users\avika\OneDrive\Desktop\Contact-Management-System-In-PYTHON-master\pythontut.db

Name	Type	Schema
Tables (2)		
member		CREATE TABLE `member` (mem_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, firstname TEXT, lastname TEXT, gender TEXT, age TEXT,
mem_id	INTEGER	"mem_id" INTEGER NOT NULL
firstname	TEXT	"firstname" TEXT
lastname	TEXT	"lastname" TEXT
gender	TEXT	"gender" TEXT
age	TEXT	"age" TEXT
address	TEXT	"address" TEXT
contact	TEXT	"contact" TEXT
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
name		"name"
seq		"seq"
Indices (0)		
Views (0)		
Triggers (0)		

2. Front-End

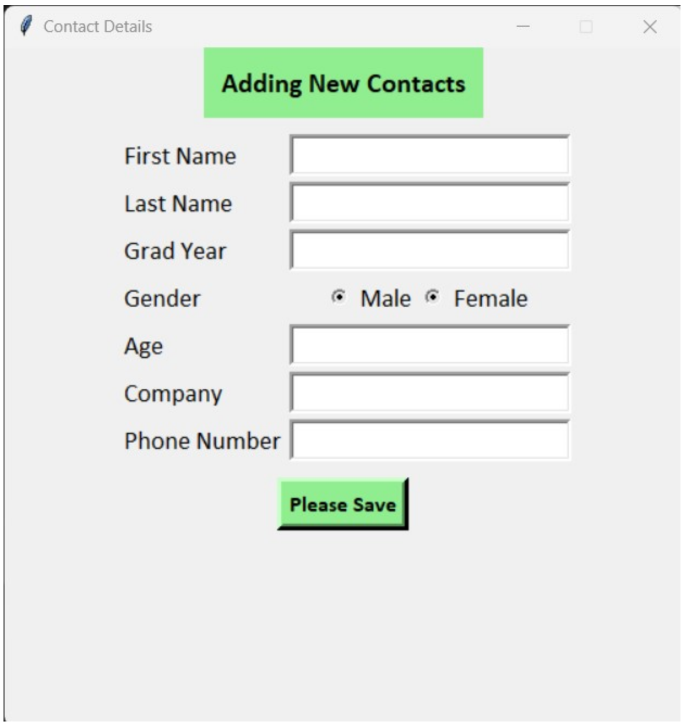


3. Back-End



AlumniConnect

First Name	Last Name	Grad Year	Gender	Age	Company	phone Number
Kishor	Kumar	2020	Male	24	Microsoft	5656565656
Gopi	Pal	2021	Male	23	Infosys	7676767676
Gajen	Pradhan	2022	Male	22	Amazon	1234567890



EXPERIMENTAL RESULTS AND ANALYSIS

Usability Evaluation:

1. **Task Completion Time:** Measure the time taken by participants to complete tasks. Shorter times indicate better usability.

2. **Error Rates:** Record the number of errors made by participants during the tasks. Lower error rates suggest higher usability.

3. **Learnability:** Assess how quickly participants can learn to use the system by measuring their ability to perform tasks correctly after minimal guidance.

4. **Efficiency:** Evaluate the efficiency of the system by examining how quickly users can accomplish tasks after gaining experience.

User Satisfaction Survey:

1. **Satisfaction Ratings:** Ask participants to rate their satisfaction with different aspects of the system, such as user interface design, ease of use, and the overall experience.

2. **System Feedback:** Collect qualitative feedback from participants regarding what they liked, disliked, and any suggestions for improvement.

3. **Recommendation:** Determine whether participants would recommend the system to other alumni and users.

System Performance Evaluation:

1. **Response Time:** Measure the system's response time for various user actions, such as loading user profiles, posting in forums, and accessing shared resources.

2. **Scalability:** Assess the system's performance under increasing load by simulating a higher number of users and activities.

Data Collection and Analysis:

- Collect and analyse the data, including task completion times, error rates, user satisfaction survey results, and system performance measurements.
- Identify patterns, trends, and areas for improvement based on the data collected.
- Compare the experimental results with predefined usability and performance benchmarks.

FUTURE SCOPE

1. **Networking and Mentorship:** Alumni databases provide a platform for former students to connect with each other. This can lead to valuable networking opportunities and mentorship relationships. Graduates can share their experiences, advice, and insights with current students and recent graduates, helping them make informed career and life choices.

2. **Career Development:** Alumni connections can play a crucial role in career development. Graduates often seek job opportunities within their alma mater's community, and alumni databases can help facilitate this. Alumni may also offer job referrals, internships, and other career-related support to fellow graduates, contributing to reduced unemployment rates and more efficient job placements.

3. **Knowledge Transfer:** Many alumni databases include information about the careers and achievements of former students. This can be a valuable resource for research, case studies, and knowledge transfer within academic institutions. Professors and students can use this information to learn from the experiences and successes of alumni.5. Community Building: A strong alumni network contributes to a sense of community and pride among graduates. This sense of belonging can foster a positive environment that encourages continued involvement and support of the institution, both socially and financially.

4. **Community Building:** A strong alumni network contributes to a sense of community and pride among graduates. This sense of belonging can foster a positive environment that encourages continued involvement and support of the institution, both socially and financially.

5. **Social Impact:** Alumni databases can be used to mobilize graduates for social impact initiatives and charitable activities. Alumni who share a common connection to their alma mater are more likely to collaborate on projects that aim to address societal issues, such as community development, healthcare, or education.

In summary, a well-maintained database of alumni connections can have a positive and far-reaching impact on society by fostering networking, mentorship, career development community-building while also supporting educational institutions and research effort. It serves as a valuable resource for graduates, students, and the institutions themselves, contributing to a more interconnected and informed society.

CONCLUSION

In conclusion, the GUI-based Alumni Connect Project using Python represents a dynamic and essential solution for educational institutions seeking to foster meaningful and enduring connections with their alumni communities. Throughout this project, we have witnessed the power of technology and user-centric design principles to bridge the gap between past and present, enabling alumni to remain closely linked to their alma mater while contributing to its growth and development.

The project exemplifies how Python, coupled with modern GUI libraries, can create a seamless and engaging platform for alumni, students, and administrators. By offering features such as user-friendly registration, comprehensive profiles, event management, discussion forums, mentorship programs, and resource sharing, the system serves as a multifaceted hub for interactions and knowledge exchange.

As we explore the project's future scope, we anticipate the integration of emerging technologies, personalization, mobile accessibility, advanced analytics, and a global reach. These enhancements will ensure that the platform remains at the forefront of alumni engagement, consistently adapting to the evolving needs and expectations of its users.

Ultimately, the GUI-based Alumni Connect Project stands as a testament to the enduring value of educational bonds and the pivotal role technology plays in nurturing and expanding those connections. By embracing innovation, promoting inclusivity, and fostering collaboration, this project continues to pave the way for more engaged and empowered alumni networks, contributing to the success of educational institutions and the broader community.

REFERENCES

1. Klossner, M. L. (2019). Library Technology and User Services: Planning, Integration, and Usability Engineering. IGI Global.

2. Satyanarayana, M., & Raghunatha, P. (2009). Library Automation and Networks. Ess Ess Publications.

3. Stallings, W. (2017). Operating Systems: Internals and Design Principles. Pearson.

Research Papers and Articles:

4. Ali, N., & Hingorani, A. L. (2017). Design and implementation of a web-based library management system for an academic library. International Journal of Information Management, 37(6), 624-630.

5. Dousa, T. M. (2017). Open-source library management systems: A current snapshot. The Code4Lib Journal, (36).

6. Haddow, G., & Klobas, J. E. (2013). ICT innovations in public libraries. Library Hi Tech, 31(2), 319-331.

Books:

- "Python GUI Programming with Tkinter" by Alan D. Moore
- "PyQt6 By Example" by Nicholas Jokl
- "Django for Beginners" by William S. Vincent
- "Flask Web Development" by Miguel Grinberg
- "SQLite Database Programming for Python" by Mike Owens

Websites Online Resources:

- Tkinter Documentation
- PyQt Documentation
- Django Documentation
- Flask Documentation
- SQLAlchemy Documentation

Online Resources:

- GitHub
- W3Schools
- GeeksforGeeks
- Real Python