A
Project
Report on

Steganography - Hide a Secret Text Message in an Image

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

by
Mohd Faziel (2200970100107)

Under the Supervision of
Prof. Mr. Rajwantbir Singh Kohli_RK

Galgotias College of Engineering & Technology Greater Noida, Uttar Pradesh
India-201306
Affiliated to

Dr. A.P.J. Abdul Kalam Technical University
Lucknow, Uttar Pradesh,
India-226031
2023-24

TABLE OF CONTENTS

GALGOTIAS COLLEGE OFENGINEERING & TECHNOLOGY
GREATER NOIDA, UTTAR PRADESH, INDIA- 201306 .

CERTIFICATE

This is to certify that the project report entitled “Steganography - Hide a Secret Text Message in an Image” submitted by Mr. **Mohd Faziel (2200970100107)** to the Galgotias College of Engineering & Technology, Greater Noida, Uttar Pradesh, affiliated to Dr. A.P.J. Abdul Kalam Technical University Lucknow, Uttar Pradesh inpartial fulfillmentfor the award of DegreeofBachelorofTechnology in Computer science & Engineering is a bonafide record of the mini project work carried out by them under my supervision during the year 2023-24

SIGNATURE	**SIGNATURE**
Mr. Rajwantbir Singh Kohli_RK	**Dr. Vishnu Sharma**
Professor	**Professor and Head**
Dept. of CSE	**Dept. of CSE& Allied Branches**

Abstract

Image-Stegnanography is a Python-based application integrating Tkinter for an intuitive GUI, enabling users to seamlessly select images and input secret messages. Employing steganographic techniques, the application facilitates secure message concealment within images. Key features include versatile message input, flexible image selection, and robust error handling. TkinterSteg empowers users with diverse steganography algorithms, ensuring a personalized and secure communication experience. The project harmonizes image processing, cryptography, and GUI development, delivering an accessible and efficient tool for concealed information exchange.

Chapter - 1: Introduction

In an era marked by an increasing emphasis on secure communication, the fusion of technology and cryptographic principles has given rise to innovative solutions. The project, titled "Steganography-Hides a secret text message in an image" emerges at the intersection of image processing, cryptography, and graphical user interface (GUI) development. Steganography, the practice of concealing messages within seemingly innocuous data, finds a practical and accessible manifestation in TkinterSteg.

This Python-based application leverages the Tkinter library to create an intuitive GUI, providing users with a seamless platform for secure communication. TkinterSteg aims to bridge the gap between technical sophistication and user accessibility by offering a straightforward yet powerful tool for concealing and extracting confidential information within images.

The project's core functionalities include image selection, message input, and the application of steganographic techniques for message concealment. Users are empowered to select any image of their choice as a cover for hiding secret messages, ensuring a personalized and flexible approach to secure communication. The versatile message input accommodates various forms of information, catering to the diverse needs of users.

TkinterSteg's versatility is further underscored by its support for various steganography algorithms, ranging from the widely used Least Significant Bit (LSB) to more advanced techniques like F5. This allows users to tailor the level of security for their concealed messages, offering a spectrum of options to suit individual preferences.

Chapter - 2 : Problem Statement

In the digital age, the secure exchange of information is increasingly vital, prompting the need for advanced techniques that guarantee confidentiality. Conventional communication methods often fall short in providing robust data security. This project addresses the pressing challenge of secure information exchange by formulating a steganographic application. The problem lies in the inadequacy of existing communication channels to safeguard sensitive data effectively. Consequently, there is a crucial need for an intuitive and secure solution that seamlessly integrates steganography into everyday communication, ensuring the covert transmission of confidential text messages within images.

Chapter - 3 : Proposed work

The proposed work involves the development of a comprehensive steganographic application, titled "Steganography: Hiding a Secret Text Message in an Image." The project will focus on the integration of advanced techniques to ensure the secure embedding and extraction of text messages within images. The key components of the proposed work include:

1. **User-Friendly Interface Design:**
   - Create an intuitive graphical user interface (GUI) using Tkinter in Python, ensuring ease of use for a diverse user base.

2. **Image Selection and Message Input:**
   - Implement functionality allowing users to select an image as the carrier and input a text message to be concealed.

5. **Error Handling Mechanisms:**
   - Develop robust error-handling mechanisms to address potential issues, such as file not found errors, or invalid inputs.

6. **Testing and Validation:**
   - Conduct thorough testing to validate the effectiveness and reliability of the steganographic application under different scenarios and input conditions.

7. **User Documentation:**
   - Prepare comprehensive user documentation to guide users through the application's features, ensuring a smooth and informed user experience.

By meticulously executing these components, the proposed work aims to deliver a fully functional and secure steganographic application that meets the project's objectives. This comprehensive approach ensures the application's usability, security, and adherence to ethical standards in facilitating covert communication through image-based text message concealment.

Chapter - 4 : Requirements

1. **Python:** Make sure you have Python installed on your system

2. **Tkinter:** Tkinter is the standard GUI library for Python. It should be included with your Python installation, so you usually don't need to install it separately.

3. **Image Processing Library:** You'll need a library to handle image processing tasks, such as reading and modifying image files. The Python Imaging Library (PIL) or its successor, Pillow, is commonly used for this purpose. You can install Pillow using pip:

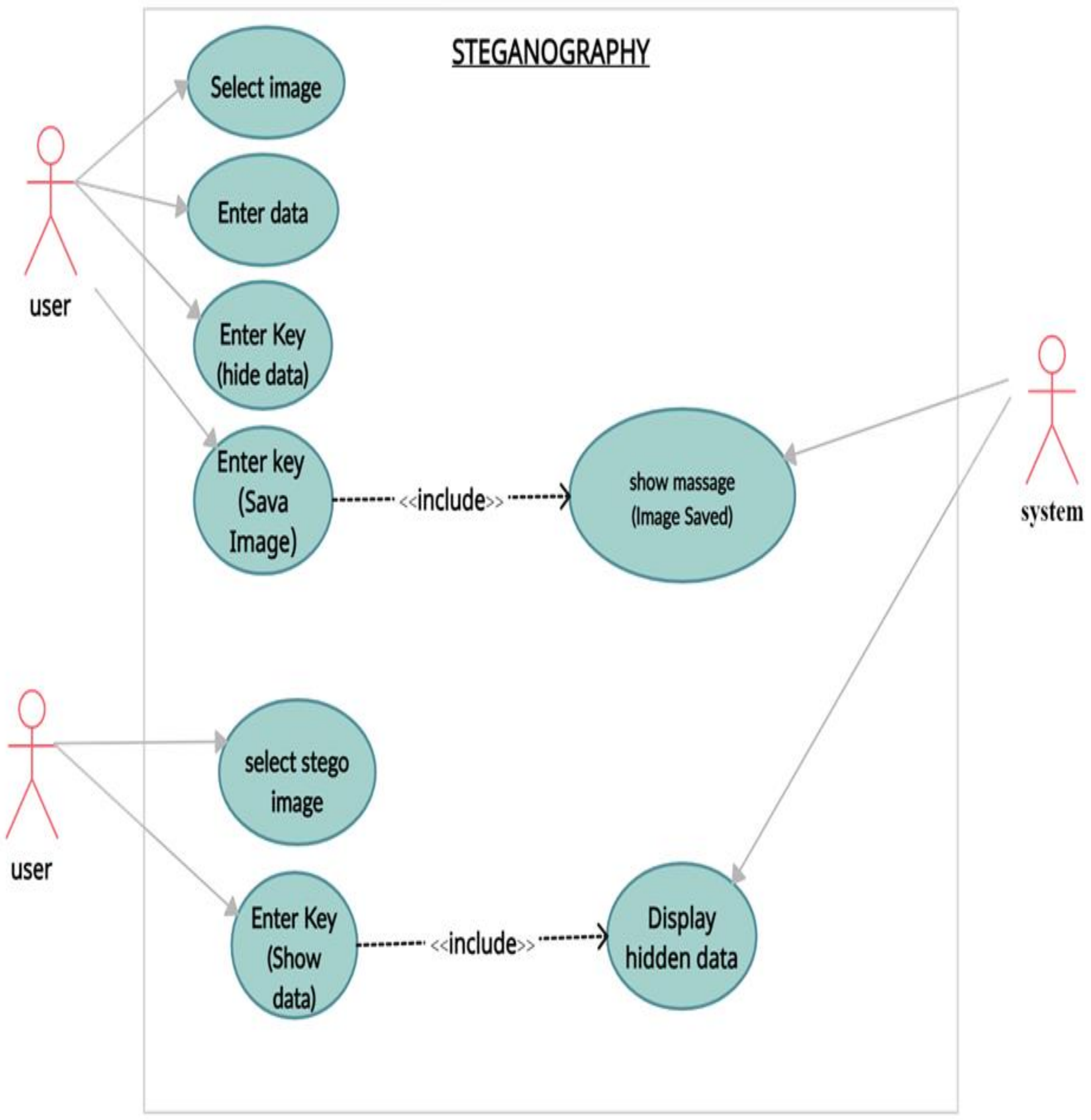
4. **Secret Message Handling:** You'll need to write code to handle the encoding and decoding of secret messages into and from images. This will involve manipulating the pixel values in the image. You'll use Python to perform these tasks.

5. **Steganography Algorithms:** You can implement various steganography algorithms to hide and retrieve data from images. Common algorithms include LSB (Least Significant Bit) and more advanced techniques like F5. You'll need to choose and implement the algorithm of your choice.

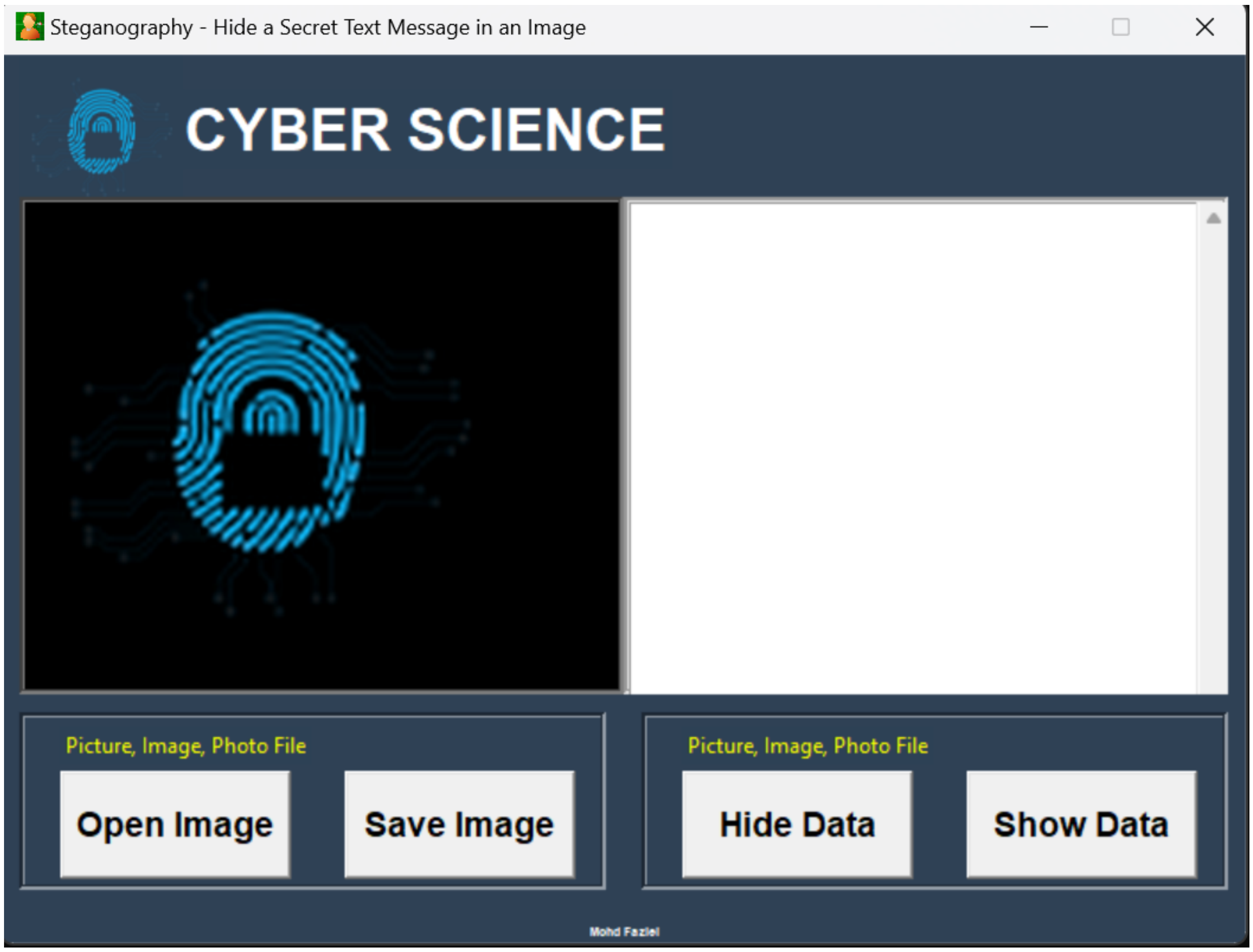
6. **User Interface (Tkinter):** Design and create a graphical user interface (GUI) using Tkinter. This GUI should provide options for users to select an image, hide a message, and extract a hidden message.

7. **Error Handling:** Implement error handling to deal with potential issues, such as file not found errors, invalid input, or insufficient space in the image for the secret message

Use-case Diagram:



Chapter - 5 : GUI Design


Steganography - Hide a Secret Text Message in an Image
CYBER SCIENCE
Picture, Image, Photo File
Open Image
Save Image
Picture, Image, Photo File
Hide Data
Show Data
Mohd Fazlel

Chapter - 6 : System Design

Chapter - 7 : Testing

Test Case No.	**Step No.**	**Description**	**Input**	**Expected Result**	**Actual Result**	**Status (Passed/Failed)**	**Requirement No.**
1	1	….	….	….	….	….	….
	2	….	….	….	….	….	….
2	1	….	….	….	….	….	….

Chapter - 8 : Source code and Limitations

Source Code:

```
# Imports all classes and functions from the tkinter module. The * means everything will be imported.
#used for creating GUI
from tkinter import *

#provides dialog windows for file-related operations.
from tkinter import filedialog

#Imports the Image and ImageTk classes from the Python Imaging Library (PIL), which is used for working with images.
from PIL import Image, ImageTk

#Imports the os module, which provides a way to interact with the operating system, used here for getting the current working directory.
import os

# Imports the least significant bit (LSB) method from the stegano library for hiding and revealing information in images.
from stegano import lsb

#Imports the messagebox module from tkinter for displaying message boxes.
from tkinter import messagebox

#Defines a function resize_image that takes an image and resizes it to the specified width and height using the Lanczos resampling algorithm.
def resize_image(image, width, height):
    return image.resize((width, height), Image.LANCZOS)



# Defines a function showimage that opens a file dialog for selecting an image file.
# It sets the filename global variable to the selected file's path.
# If a file is selected, it opens the image, resizes it, converts it to a
Tkinter PhotoImage, and displays it in a label (lbl).
# If no file is selected, it shows an error message.
# Exception handling is included to catch any errors during this
process.
def showimage():
    global filename
    try:
        filename = filedialog.askopenfilename(
            initialdir=os.getcwd(),
            title='Select Image File',
            filetypes=(("PNG files", "*.png"), ("JPG files", "*.jpg"), ("All
files", "*.*"))
        )
        if filename:
            img = Image.open(filename)
            img = resize_image(img, 250, 250)
            img = ImageTk.PhotoImage(img)
            lbl.configure(image=img)
            lbl.image = img
        else:
            messagebox.showerror("Error", "Please select an image.")
    except Exception as e:
        messagebox.showerror("Error", f"An error occurred: {str(e)}")


# Defines a function Hide that hides a message obtained from a
Text widget into the selected image using the LSB method.
# Sets the secret global variable to the steganographic object.
# Displays a success message and clears the text widget.
# Shows an error if no image is selected.
# Exception handling is included.
def Hide():
    global secret
    global filename
    message = text1.get(1.0, END)
    try:
        if filename:
            secret = lsb.hide(filename, message)
            messagebox.showinfo("Success",     "Message     hidden
successfully.")
```

```
            text1.delete(1.0, END)
        else:
            messagebox.showerror("Error",  "Please  select  an  image
first.")
    except Exception as e:
        messagebox.showerror("Error", f"An error occurred: {str(e)}")


#Defines a function Show that reveals a hidden message from the
selected image and displays it in a Text widget.
# Shows an error if no image is selected.
# Exception handling is included.
def Show():
    global filename
    try:
        if filename:
            clear_message = lsb.reveal(filename)
            text1.delete(1.0, END)
            text1.insert(END, clear_message)
        else:
            messagebox.showerror("Error",  "Please  select  an  image
first.")
    except Exception as e:
        messagebox.showerror("Error", f"An error occurred: {str(e)}")



#Defines a function save that saves the image with the hidden
message to a file named "encrypted.png".
# Shows an error if there is no image to save.
# Exception handling is included.
def save():
    try:
        if 'secret' in globals():
            secret.save("encrypted.png")
            messagebox.showinfo("Success",        "Image        saved
successfully.")
        else:
            messagebox.showerror("Error", "No image to save.")
    except Exception as e:
messagebox.showerror("Error", f"An error occurred: {str(e)}")
```

```
#This block creates the main window with specific characteristics( title, geometry, and background color.)
root = Tk()
root.title("Steganography - Hide a Secret Text Message in an Image")
root.geometry("700x500+150+180")
root.resizable(False, False)
root.configure(bg="#2f4155")


#Global variables to store the selected filename and the steganographic object.
filename = ""
secret = None

#Sets the window icon and loads a logo for the GUI.
image_icon = PhotoImage(file="logo.jpg")
root.iconphoto(False, image_icon)
logo = PhotoImage(file="logo.png")

#Several frames and labels are created to organize and display different elements in the GUI.

#Creates and places labels with a logo and a title in the Tkinter window.
Label(root, image=logo, bg="#2d4155").place(x=10, y=0)
Label(root, text="CYBER SCIENCE", bg="#2d4155",
fg="white", font="arial 25 bold").place(x=100, y=20)

#Creates a frame (f) and a label (lbl) within it, where the image will be displayed.
f = Frame(root, bd=3, bg="black", width=340, height=280, relief=GROOVE)
f.place(x=10, y=80)
lbl = Label(f, bg="black")
lbl.place(x=10, y=10)  # Adjusted the placement of the label


#Creates a frame (frame2), a Text widget (text1) within it for displaying text, and a vertical scrollbar for the Text widget.
frame2 = Frame(root, bd=3, bg="white", width=340, height=280,
relief=GROOVE)
frame2.place(x=350, y=80)
text1 = Text(frame2, font="Robote 20", bg="white", fg="black",
relief=GROOVE)
text1.place(x=0, y=0, width=320, height=295)
scrollbar1 = Scrollbar(frame2)
scrollbar1.place(x=320, y=0, height=295)  # Adjusted the height
scrollbar1.configure(command=text1.yview)
text1.configure(yscrollcommand=scrollbar1.set)

#Creates a frame (frame3). It is positioned at coordinates (10, 370)
within the main Tkinter window.
frame3 = Frame(root, bd=3, bg="#2f4155", width=330, height=100,
relief=GROOVE)
frame3.place(x=10, y=370)

#Buttons of frame3
# The first button ("Open Image") is associated with the showimage
function when clicked. It is positioned at (20, 30).
Button(frame3, text="Open Image", width=10, height=2, font="arial
14 bold", command=showimage).place(x=20, y=30)

# The second button ("Save Image") is similar to the first button but
is associated with the save function. It is positioned at (180, 30).
Button(frame3, text="Save Image", width=10, height=2, font="arial
14 bold", command=save).place(x=180, y=30)


# The label displays the text "Picture, Image, Photo File" with a
background color of #2f4115 and text color of yellow. It is
positioned at (20, 5).
Label(frame3, text="Picture, Image, Photo File", bg="#2d4155",
fg="yellow").place(x=20, y=5)

# Creates a label to display developers name at the bottom center
of the main frame.
Label(root, text="Mohd Faziel", bg="#2f4155", fg="white",
font=("arial 5 bold")).place(relx=0.5, rely=1, anchor='s')


# Creates a frame (frame4) with similar characteristics to frame3. It
```

```
is positioned at coordinates (360, 370) within the main Tkinter
window.
frame4 = Frame(root, bd=3, bg="#2f4155", width=330, height=100,
relief=GROOVE)
frame4.place(x=360, y=370)

#Buttons of frame4
# The first button ("Hide Data") is associated with the Hide function
when clicked. It is positioned at (20, 30).
Button(frame4, text="Hide Data", width=10, height=2, font="arial 14
bold", command=Hide).place(x=20, y=30)

#The second button ("Show Data") is similar to the first button but
is associated with the Show function. It is positioned at (180, 30).
Button(frame4, text="Show Data", width=10, height=2, font="arial
14 bold", command=Show).place(x=180, y=30)

#The label displays the text "Picture, Image, Photo File" with a
background color of #2f4115 and text color of yellow. It is
positioned at (20, 5).
Label(frame4, text="Picture, Image, Photo File", bg="#2d4155",
fg="yellow").place(x=20, y=5)

#This starts the Tkinter event loop, allowing the GUI to interact with
the user.
root.mainloop()
```

Limitations of the Project:

1. **Image Size Constraints:**

- The project may face limitations concerning the size of images suitable for effective steganographic embedding. Large messages in small images could potentially compromise the concealment process.

2. **Security Level:**

- While efforts are made to incorporate cryptographic techniques, the project may not achieve the same level of security as more complex, specialized encryption methods. Users should be aware of the application's limitations in high-stakes security scenarios.

3. **File Format Compatibility:**

- The project's compatibility with a diverse range of image file formats may be limited. Ensuring broad compatibility requires additional testing and optimization for various formats.

4. **User Awareness:**

- Users must be cautious about the ethical and legal implications of steganography. The project's success relies on responsible use, and users should be educated about potential misuse or unintended consequences.

5. **Platform Dependency:**

- The application's performance may vary across different operating systems and environments, potentially leading to inconsistencies in user experience.

Understanding and acknowledging these limitations is crucial for both the developers and users, ensuring realistic expectations and responsible use of the steganographic application.

Future Work:
1**. **Enhanced Security Measures:****
- Investigate and implement advanced cryptographic techniques to further enhance the security of the hidden messages, ensuring resistance to emerging threats and sophisticated attacks.

2. **Algorithm Optimization:**
- Explore and develop optimization strategies for steganographic algorithms to improve efficiency and reduce the impact on image quality, allowing for more seamless integration with a broader range of images.

3. **Multimedia Support:**
- Extend the application's capabilities to support hiding messages not only in images but also in other multimedia formats, such as audio or video, expanding the scope of secure communication.

4. **Cross-Platform Compatibility:**
- Optimize the application for cross-platform compatibility, ensuring consistent performance and user experience across various operating systems and devices.

5. **User Authentication Mechanisms:**
- Implement additional layers of security, such as user authentication mechanisms, to enhance the overall security posture of the application and restrict unauthorized access.

6. **Educational Resources:**
- Develop educational resources within the application to inform users about responsible steganography use, ethical considerations, and potential legal implications.

By addressing these future work areas, the project can evolve to meet the growing demands of secure communication, incorporating cutting-edge technologies and refining user experiences for a more robust and versatile steganographic application.

Conclusion:

In conclusion, "Steganography: Hiding a Secret Text Message in an Image" has successfully realized a user-friendly platform for secure communication. The Tkinter-based application seamlessly conceals and extracts confidential messages within images. While recognizing certain limitations, the project sets the stage for future enhancements, including advanced cryptography and multimedia support. Through an intuitive interface and responsible technology use advocacy, the project stands as a robust solution, ready for ongoing refinement and adaptation to evolving security demands.

References:

- **tkinter Documentation:** https://docs.python.org/3/library/tkinter.html
- **Python Imaging Library (PIL) Documentation:** https://pillow.readthedocs.io/en/stable/
- Stegano Library Documentation: https://pypi.org/project/stegano/
  Python Official Documentation: https://docs.python.org/3/