**Encryption using BMP Images  
Masa Lubbadeh 12027847**

This application allows users to hide secret text within images using the least significant bit (LSB) method of steganography.

* ***Key Features:***
* Load and display BMP images.
* Enter secret text manually or load from a file.
* Hide secret text in an image using user-defined bit depth.
* Restore hidden text from an image.
* Save the modified image.
* Clear inputs and results.
* ***Setup Instructions:***

I used python language with some libraries, so first you need to setup the project correctly.

* Ensure you have Python and necessary libraries installed (tkinter, PIL, ttkbootstrap).
* Use this command to install them: pip install Pillow ttkbootstrap
* ***Code Overview:***
* ***Used Libraries:***
* tkinter: A standard Python library for creating GUI applications.
* filedialog and messagebox: I used them for when I want to deal with files and show message boxes to the user.
* Ttk and ttkbootstrap: used them for a more modern look and feel.
* PIL (Pillow): image processing library to deal with images.
* ***Function Descriptions***
* display\_image(image, area): This function shows a resized image in a specific part of the application window, making sure it fits nicely in the space provided.
* load\_image(): This function lets the user pick a BMP image from their computer. After selecting an image, it opens and displays it using the display\_image() function.
* load\_text\_from\_file(): This function allows users to choose a text file to load secret text into the application. It updates the text area with the content from the selected file.
* save\_result\_image(): This function saves the modified image that has the hidden secret text in it. If the user hasn't loaded an image, it shows an error message.
* clear\_all(): This function resets everything in the app. It clears the loaded image and text, returns any placeholder text to the text area, and resets the image display areas.
* hide\_text():This function hides the secret text in the cover image, it does this by changing the least significant bits (LSBs) of the image's pixel values. This is done by several steps:

-First, it converts the text into its binary representation.

-Second, it adds a delimiter at the end of the secret text to mark the end of the hidden message so the program will know where the message ends (it can also test if the character is between a-z but I thought this method is more functional).

-Then, the function goes through the pixels of the cover image. Each pixel is made up of three-color values: red, green, and blue (RGB)

The function modifies the RGB values of the image pixels according to binary data, allowing the text to be hidden within the image.

The choice of how many bits to use for hiding the text—1, 2, or 3 bits—affects the visibility of the changes in the image.

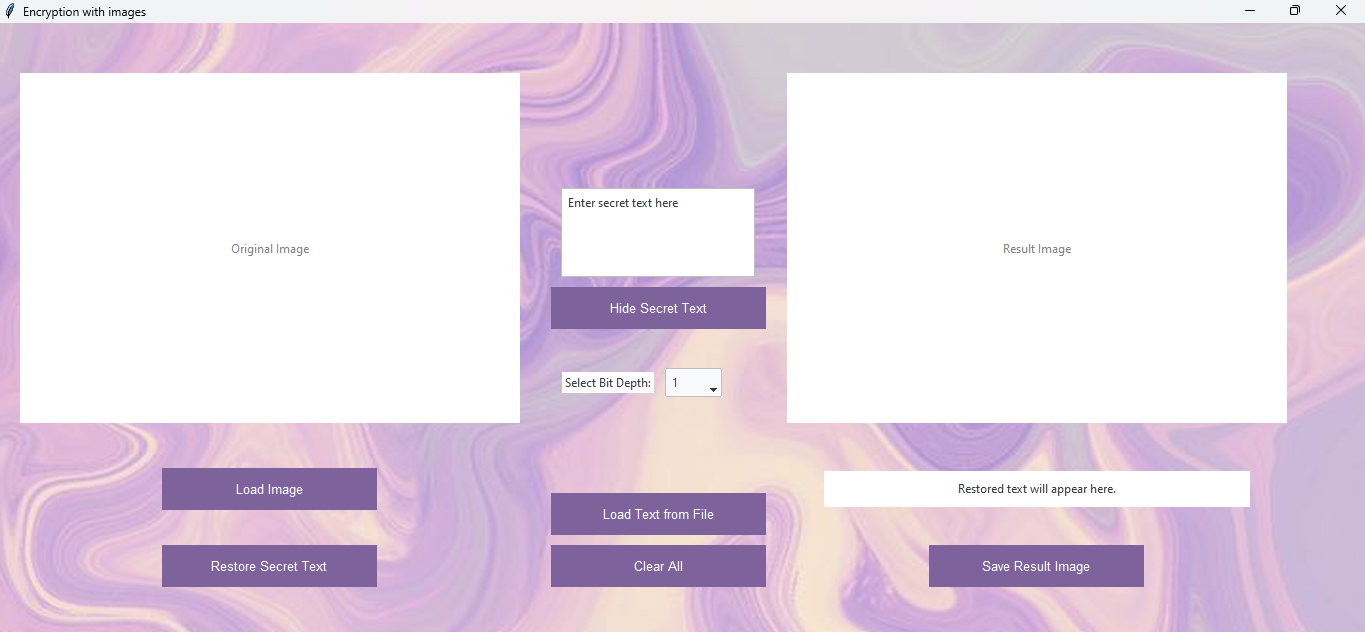
**--1 Bit**: Changes are very subtle. The image looks almost the same, making it hard to notice any difference.

**--2 Bits**: Introduces slightly more noticeable changes, but the image still retains much of its original appearance.

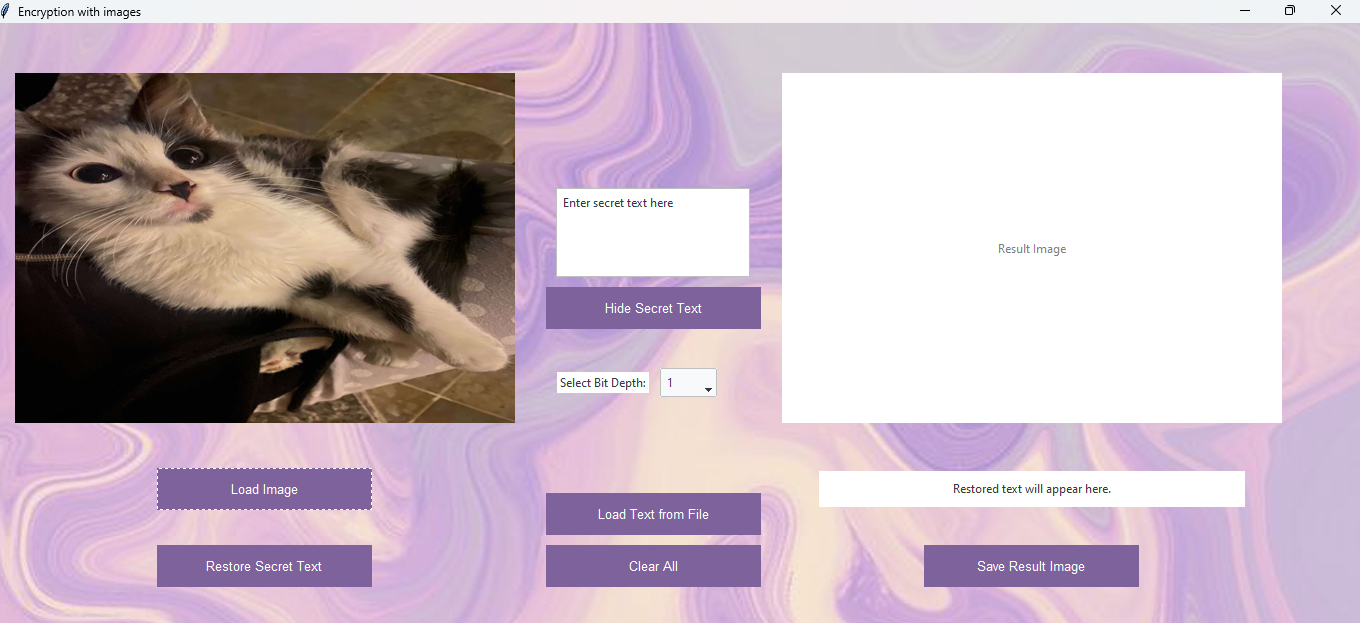
**--3 Bits**: This creates more visible alterations to the image, which may affect how it looks but allows for more data to be hidden.

* restore\_text(): retrieves the hidden message from the modified image by examining the least significant bits of each pixel’s red color value. First, it reads through these bits to gather a long string of binary numbers. Then, it groups these bits into bytes (8 bits each) and converts each byte back into a character, gradually reconstructing the original hidden message. The function looks for a special end signal made of eight "0" bits, which tells it when the message has fully ended. Once complete, the recovered secret message is displayed in the application for the user to view.
* ***Screenshots from the application:***

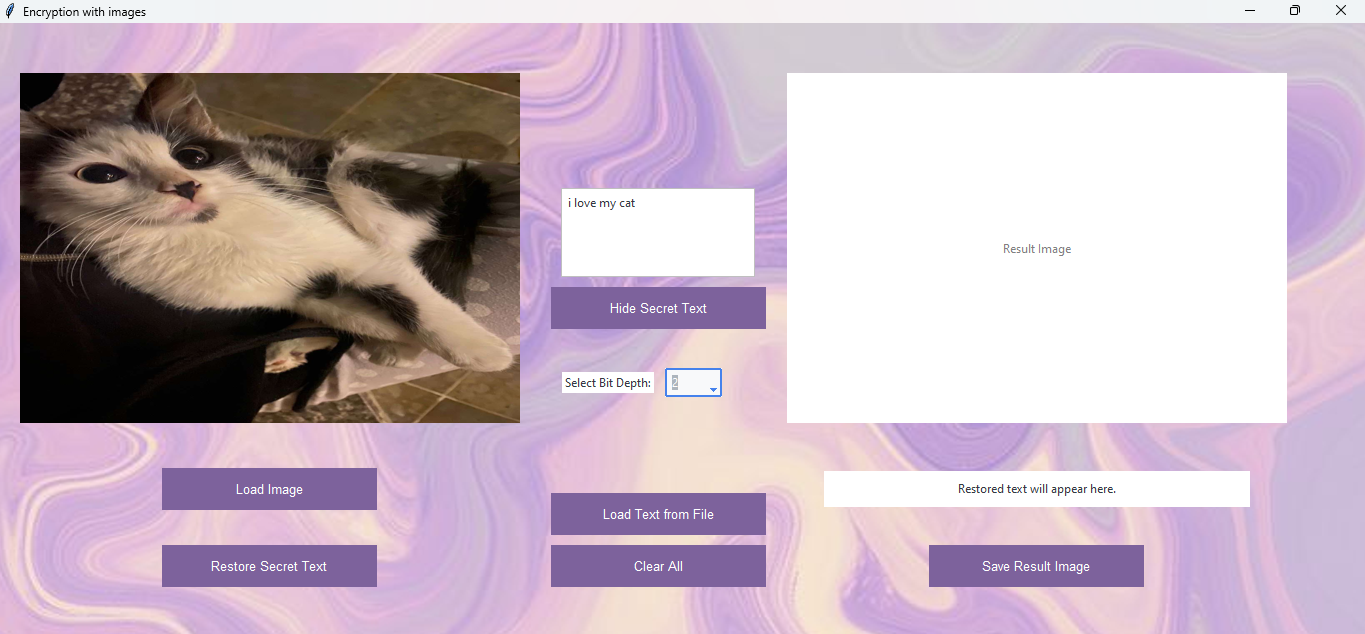
This is the main window, as you can see its divided into three sections. The first section is for uploading the BMP image, the second section is to enter the secret text and choose number of bits, last section is to view the result image and there you can view the hidden text.



Here, I uploaded a bmp image by pressing the Load Image button, it only accepts bmp images.

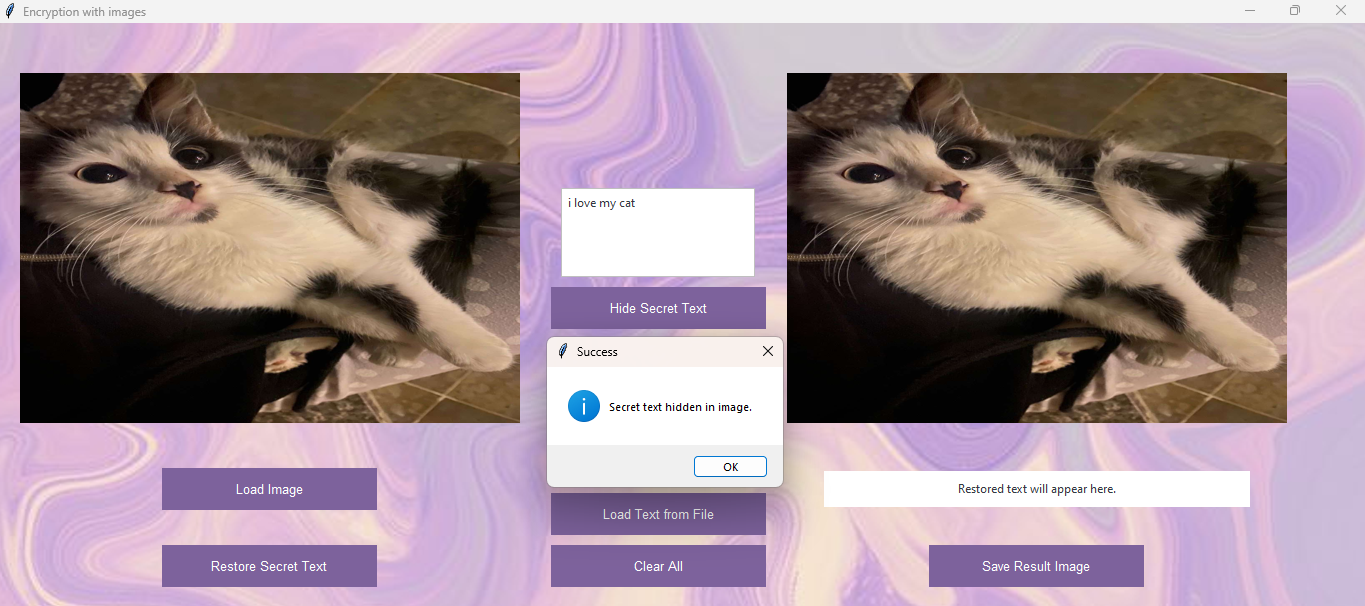


Then, in the second section I entered the text I want to hide within the image and selected number of bits.

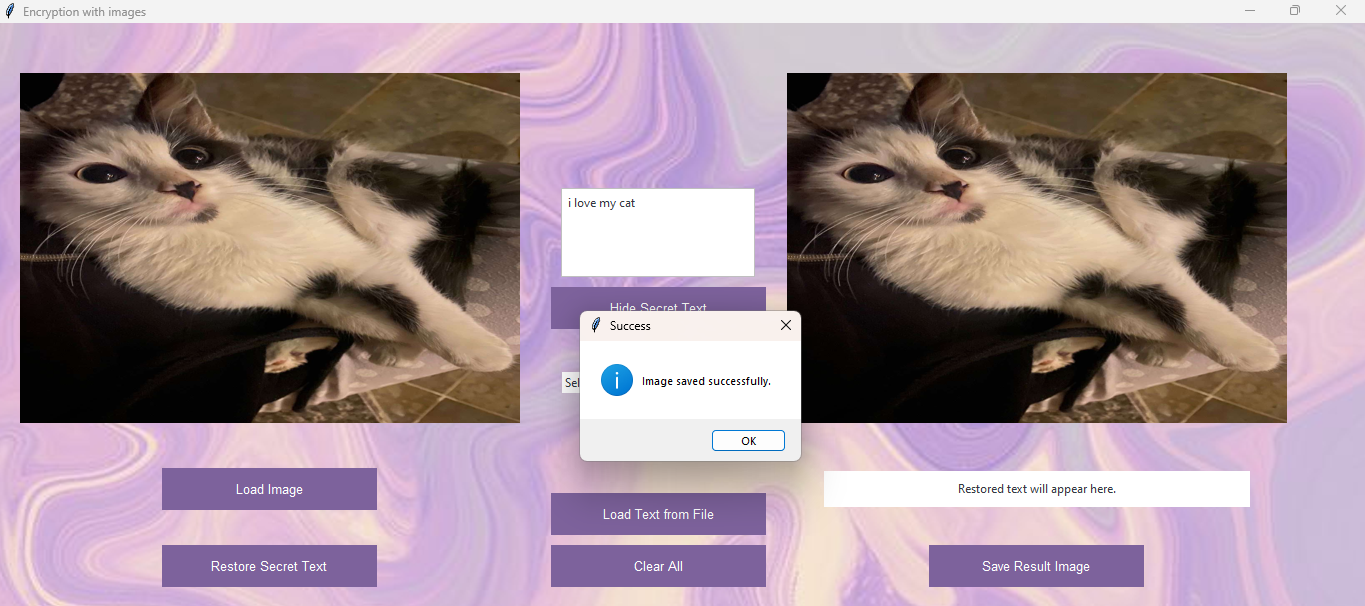


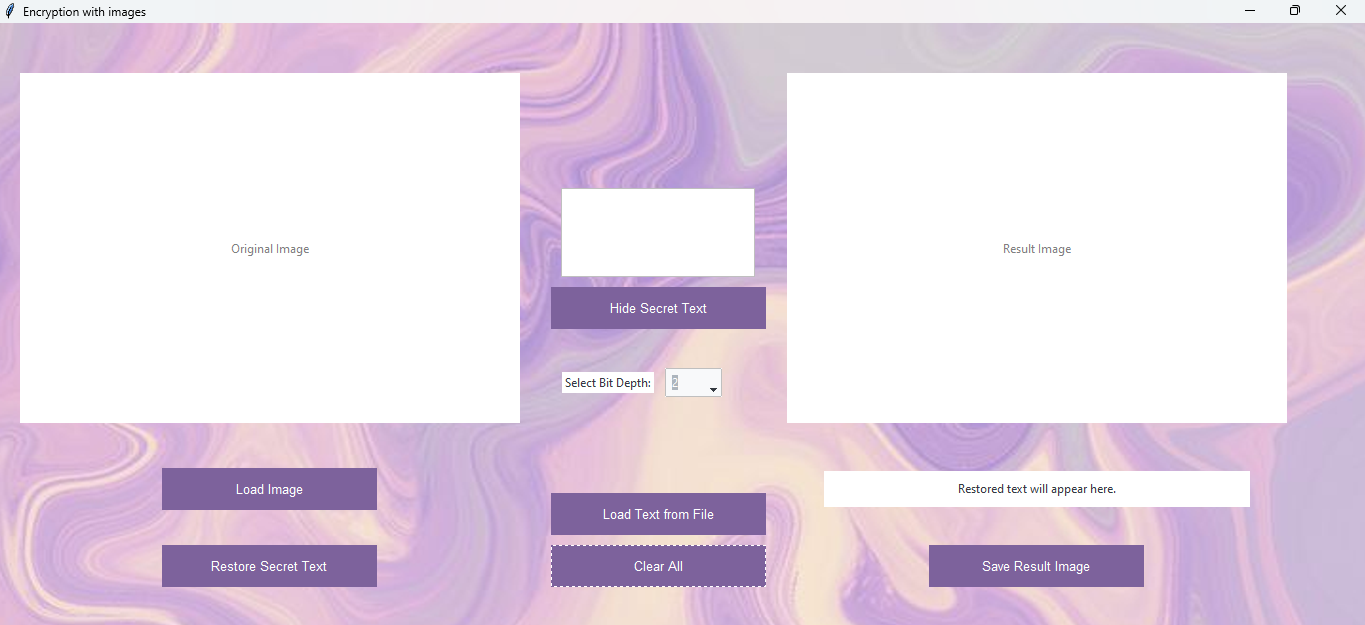
I also can upload text from a file by pressing on the Load Text from File button

After pressing the Hide Secret Text button, the result image will be displayed and I can save it.

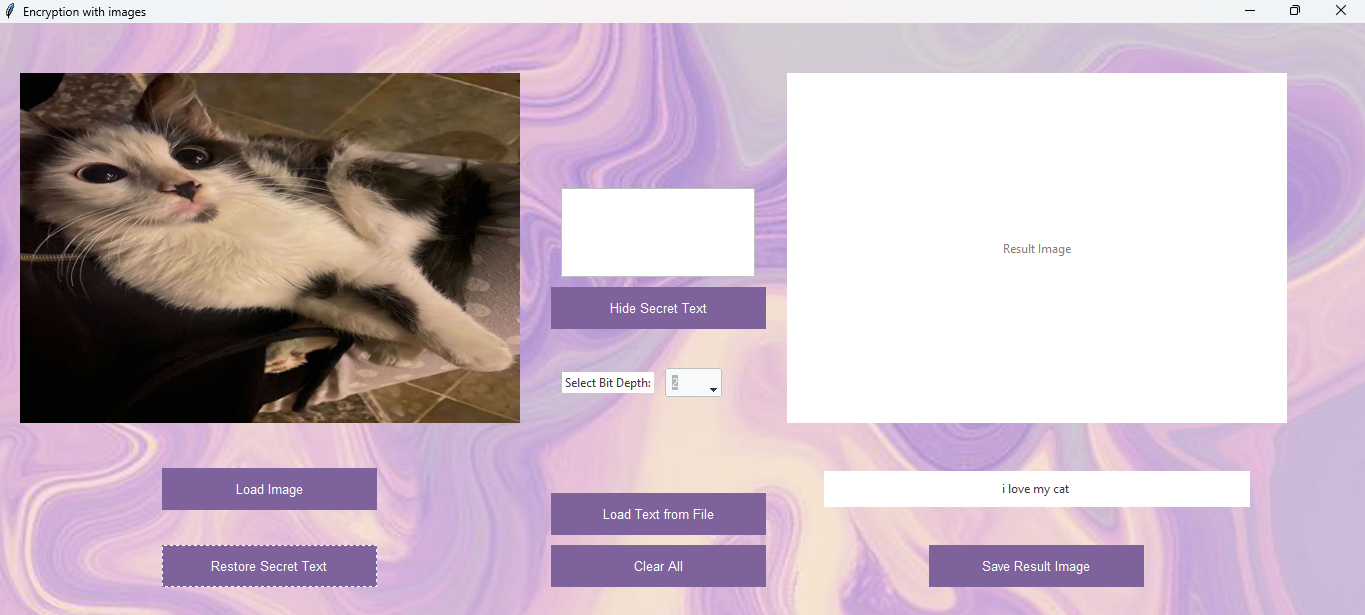


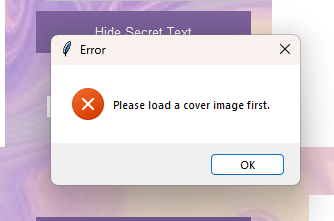
After pressing the Save Result Image button:



Then, I pressed the Clear All button:

Now, I’m going to upload the result image to restore the secret text from it, the result will be displayed in the label in the third section



If I tried to press the hide or restore buttons without uploading an image, a message box with an error will pop out:

You can check the project repository at:

https://github.com/MasaLubbadeh/Encryption-Using-BMP-Images.git