**IMAGE STEGNOGRAPHY**

**INFORMATION HIDING IN IMAGES**

**A Project Report Submitted in partial fulfilment of the requirements for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted By**

**MALLIDI PAVAN KUMAR REDDY (20221A0580)**

**NAGABHERI RAKESH KISHORE (21225A0509)**

**VATHADI YUVARAJU (21225A0512)**

**VINNA ABHI RAM (20221A05C6)**

**Under the Guidance of**

**Ms. N JEEVANA JYOTHI M. Tech**

**Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BONAM VENKATA CHALAMAYYA ENGINEERING COLLEGE**

**(AUTONOMOUS)**

(Approved by A I C T E, New Delhi & Permanently Affiliated to J N T U K, Kakinada)

(Accredited by N.B.A & NAAC with ‘A’ Grade)

**ODALAREVU – 533210**

**2020-24**

**BONAM VENKATA CHALAMAYYA ENGINEERING COLLEGE**

**(AUTONOMOUS)**

(Approved by A I C T E, New Delhi & Permanently Affiliated to J N T U K, Kakinada)

(Accredited by N.B.A & NAAC with ‘A’ Grade)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that the project work entitled “**IMAGE STEGNOGRAPHY INFORMATION HIDING IN IMAGES**” is being submitted for the partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering, at BVC Engineering College, Odalarevu**, is a bonafide work done by **Mallidi Pavan Kumar Reddy (20221A0580), Nagabheri Rakesh Kishore (21225A0509), Vathadi Yuvaraju (21225A0512), Vinna Abhi Ram (20221A05c6)** for the academic year 2023-24 and it has been found suitable for acceptance according to the requirement of University. The results embodied in this thesis have not been submitted to any other University Institute for the award of any degree.

**Project Guide Head of the Department**

**Ms. N JEEVANA JYOTHI Dr. B S N Murthy**

**M.Tech M.Tech, Ph.D, MISTE**

**Assistant Professor**

**External Examiner**

**ACKNOWLEDGEMENT**

We express our deep sense of gratitude and respect towards our guide Ms. **N JEEVANA JYOTHI Associate Professor**, BVC Engineering College, Odalarevu for her excellent guidance right from selection of the Project work. Her constant encouragement and support have been the cause of our success in completing this thesis in the college.

We are thankful to **Dr. P. DEVABALAN,** Professor of CSE and Project Coordinator for facilitating, training, counselling and support to carry out the project work.

We are thankful To **Dr. B S N Murthy, Professor and Head of the Department of CSE** for facilitating, training, counselling, and support to carry out the project work.

We are also thankful to **Dr. Chandramouli V S A**, Principal, BVC Engineering College, Odalarevu, for his support during and till the completion of the project.

We are thankful to all the **teaching and non-teaching staff** of Computer Science & Engineering Department, **Management**, BVC Engineering College, Odalarevu, and our friends for their direct and indirect help provided to us in completing the project.

**PROJECT ASSOCIATES**

**MALLIDI PAVAN KUMAR REDDY (20221A0580)**

**NAGABHERI RAKESH KISHORE (21225A0509)**

**VATHADI YUVARAJU (21225A0512)**

**VINNA ABHI RAM (20221A05C6)**

**ABSTRACT**

The proliferation of digital communication channels necessitates the development of secure methods for transmitting sensitive information. Image steganography emerges as a clandestine technique to embed confidential data within innocuous images, ensuring covert transmission while preserving the integrity of the communication medium. This project delves into the realm of image steganography by implementing a Python-based solution for information hiding within images. Leveraging the power of Python libraries such as OpenCV and NumPy, the project facilitates the seamless integration of hidden messages into image files. Through the utilization of various steganographic techniques like LSB (Least Significant Bit) embedding and its variants, the project provides a comprehensive exploration into the methodologies of concealing data within digital images. Additionally, it offers functionalities for message extraction, enabling the retrieval of concealed information from steganographic images. The project serves as an educational tool for understanding the principles and practical applications of steganography, empowering users to enhance their comprehension of information security concepts.

LIST OF CONTENTS

**CONTENTS PAGE NO**

[CHAPTER - 1 INTRODUCTION 1](#_Toc165458381)

[1.1 What is Python 1](#_Toc165458382)

[1.2 Advantages of Python 2](#_Toc165458383)

[1.3 Disadvantages of Python 2](#_Toc165458384)

[1.4 Install Python on Windows Step by Step 3](#_Toc165458385)

[1.5 Installation of Python 5](#_Toc165458386)

[1.6 Verify the Python Installation 6](#_Toc165458387)

[1.7 Check how the Python IDLE works 6](#_Toc165458388)

[1.8 What is VS Code: 7](#_Toc165458389)

[1.9 Steps to Install Visual Studio Code on Windows 7](#_Toc165458390)

[CHAPTER - 2 SYSTEM ANALYSIS 10](#_Toc165458391)

[2.1 Existing System 10](#_Toc165458392)

[2.1.1 Drawbacks 10](#_Toc165458393)

[2.2 Proposed System 11](#_Toc165458394)

[2.2.1 Advantages 11](#_Toc165458395)

[2.3 Modules 12](#_Toc165458396)

[2.4 Feasibility Study 13](#_Toc165458397)

[2.4.1 Technical Feasibility 13](#_Toc165458398)

[2.4.2 Economic Feasibility 13](#_Toc165458399)

[2.4.3 Operational Feasibility 14](#_Toc165458400)

[2.4.4 Legal and Ethical Feasibility 14](#_Toc165458401)

[CHAPTER - 3 REQUIREMENT ANALYSIS 15](#_Toc165458402)

[3.1 Software Requirements 15](#_Toc165458403)

[3.2 Functional Requirements 15](#_Toc165458404)

[3.3 Hardware Requirements 15](#_Toc165458405)

[3.4 Libraries Required 15](#_Toc165458406)

[CHAPTER - 4 SOFTWARE DESIGN 16](#_Toc165458407)

[4.1 Methodology 16](#_Toc165458408)

[4.1.1 User Interface Design 16](#_Toc165458409)

[4.1.2 Encoding Process 16](#_Toc165458410)

[4.1.3 Decoding Process 16](#_Toc165458411)

[4.1.4 Testing and Validation 16](#_Toc165458412)

[4.1.5 Documentation and User Guide 17](#_Toc165458413)

[4.1.6 Deployment and Maintenance 17](#_Toc165458414)

[4.2 Data Flow Diagram 17](#_Toc165458415)

[4.3 Use Case diagram 18](#_Toc165458416)

[4.4 Sequence Diagram 19](#_Toc165458417)

[4.5 State Diagram 20](#_Toc165458418)

[4.6 Package Diagram 21](#_Toc165458419)

[CHAPTER - 5 IMPLEMENTATION 22](#_Toc165458420)

[5.1 Steganography Technique 22](#_Toc165458421)

[5.1.1 Image Steganography 23](#_Toc165458422)

[5.1.2 Video Steganography: 23](#_Toc165458423)

[5.1.3 Audio Steganography: 23](#_Toc165458424)

[5.1.4 Text Steganography: 23](#_Toc165458425)

[5.2 Phases of Steganography 23](#_Toc165458426)

[5.3 Least significant bit (LSB) 24](#_Toc165458427)

[5.3.1 LSB Encoding 24](#_Toc165458428)

[5.3.2 LSB Decoding 25](#_Toc165458429)

[5.4 Image Steganography using Python 26](#_Toc165458430)

[5.4.1 Project Prerequisite 26](#_Toc165458431)

[5.4.2 Steps to Build a Python Image Steganography Project 27](#_Toc165458432)

[5.5 Applications of The Project 28](#_Toc165458433)

[5.6 Source Code 28](#_Toc165458434)

[CHAPTER - 6 TESTING 33](#_Toc165458435)

[6.1 System Test 33](#_Toc165458436)

[6.2 Types of Tests 33](#_Toc165458437)

[6.2.1 Unit testing 33](#_Toc165458438)

[6.2.2 Integrated Testing 33](#_Toc165458439)

[6.2.3 Functional Testing 33](#_Toc165458440)

[6.2.4 White Box Testing 34](#_Toc165458441)

[6.2.5 Black Box Testing 34](#_Toc165458442)

[6.3 Test strategy and approach 34](#_Toc165458443)

[6.3.1 Test objectives 34](#_Toc165458444)

[6.3.2 Features to be tested 34](#_Toc165458445)

[6.4 Acceptance Testing 34](#_Toc165458446)

[CHAPTER - 7 OUTPUT 35](#_Toc165458447)

[CHAPTER - 8 CONCLUSION & FUTURE ENHANCEMENT 42](#_Toc165458448)

[CHAPTER - 9 BIBLIOGRAPHY 43](#_Toc165458449)

**LIST OF FIGURES**

**FIGURE PAGE NO**

[Fig 1.1 Python Official Website 3](#_Toc165458530)

[Fig 1.2 Python Download Page 3](#_Toc165458531)

[Fig 1.3 Python Versions Available to Download 4](#_Toc165458532)

[Fig 1.4 Select the OS to download 4](#_Toc165458533)

[Fig 1.5 Python application Setup 5](#_Toc165458534)

[Fig 1.6 Setup Successful Message 5](#_Toc165458535)

[Fig 1.7 Open Command Prompt 6](#_Toc165458536)

[Fig 1.8 Check Whether Python is Installed 6](#_Toc165458537)

[Fig 1.9 Python IDLE Interface 6](#_Toc165458538)

[Fig 1.10 Visual Studio Code Official Download Site 7](#_Toc165458539)

[Fig 1.11 VS Code Licence agreement 8](#_Toc165458540)

[Fig 1.12 Add the VS Code to PATH 8](#_Toc165458541)

[Fig 1.13 VS Code Installation Screen 8](#_Toc165458542)

[Fig 1.14 VS Code Installation Complete 9](#_Toc165458543)

[Fig 1.15 VS Code Interface 9](#_Toc165458544)

[Fig 4.1 Data Flow Diagram 17](#_Toc165458545)

[Fig 4.2 Use Case Diagram 18](#_Toc165458546)

[Fig 4.3 Sequence Diagram 19](#_Toc165458547)

[Fig 4.4 State Diagram 20](#_Toc165458548)

[Fig 4.5 Package Diagram 21](#_Toc165458549)

[Fig 5.1 Different Mediums to Achieve Steganography 23](#_Toc165458550)

[Fig 5.2 Process of Embedding 3-Bit into Pixel 25](#_Toc165458551)

[Fig 5.3 LSB Substitution Architecture 26](#_Toc165458552)

[Fig 7.1 Open the file in the VSCODE 35](#_Toc165458553)

[Fig 7.2 Create an Virtual Environment By using the syntax python -m venv .venv 35](#_Toc165458554)

[Fig 7.3 Open Command Prompt and Activate Virtual environment using activate.bat 36](#_Toc165458555)

[Fig 7.4 Install the Requirements (Pillow) using pip install pillow 36](#_Toc165458556)

[Fig 7.5 Run the main file python image\_Stegno.py 37](#_Toc165458557)

[Fig 7.6 Interface of the Project 37](#_Toc165458558)

[Fig 7.7 Frame for selecting the image to encode 38](#_Toc165458559)

[Fig 7.8 Select an image to Encode the message 38](#_Toc165458560)

[Fig 7.9 Enter the Message to Encode to Image 39](#_Toc165458561)

[Fig 7.10 Choose the Location and File Name to save the Encoded Image 39](#_Toc165458562)

[Fig 7.11 Success message represent the successful encoding 40](#_Toc165458563)

[Fig 7.12 Frame for selecting the image to decode 40](#_Toc165458564)

[Fig 7.13 Select an image to Decode the message 41](#_Toc165458565)

[Fig 7.14 Message is decoded in image 41](#_Toc165458566)

# INTRODUCTION

In today's digital age, the need for secure communication methods has become paramount. With the rise of cyber threats and the increasing importance of protecting sensitive information, traditional encryption methods may not always suffice. This has led to the exploration and adoption of alternative techniques such as steganography, particularly in the realm of image manipulation.

Steganography, an ancient practice dating back to ancient Greece, involves concealing messages within other non-secret data to avoid detection. In the digital era, steganography finds its application in embedding secret messages within seemingly innocuous digital files such as images, audio files, or videos. Among these, image steganography stands out as one of the most intriguing and widely researched areas, owing to the popularity and ubiquity of digital images in modern communication channels.

The concept behind image steganography is deceptively simple: embedding hidden information within the pixels of an image without altering its perceptual quality. This process allows for the covert transmission of sensitive data, making it imperceptible to unintended recipients. However, achieving this delicate balance between concealment and transparency poses a significant challenge, requiring the development of sophisticated algorithms and techniques.

Through the implementation of various steganographic methods, such as Least Significant Bit (LSB) embedding and its variants, the project seeks to demonstrate the versatility and effectiveness of image steganography in concealing data. Additionally, the project will offer functionalities for message extraction, allowing users to retrieve hidden information from steganographic images.

By engaging in this project, participants will gain valuable insights into the principles and applications of steganography, thereby enhancing their understanding of information security concepts. Furthermore, the project serves as a foundation for further exploration and experimentation in the field of digital forensics, cryptography, and data security.

## 1.1 What is Python

These are some facts about Python. Python is present most widely used multi-purpose and high-level programming language.it allows programming in Object-Oriented and Procedural paradigms. Python programs commonly are smaller than other programming languages like Java. Programmers have to write relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by the almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc. The biggest strength of Python language is huge collection of standard libraries it can be used for the following

* Machine Learning.
* GUI Applications (such as Kivy, Tkinter, PyQt, etc.)
* Web frame works such as Django (used by YouTube, Instagram, Dropbox)
* Image processing (such as Open cv and Pillow)
* Web scraping (such as Scrapy, Beautiful So up and Selenium)

## 1.2 Advantages of Python

1. **Less Coding:** When performing the same task in other languages, all tasks performed in Python require less coding. Python is also excellent standard library support, so there is no need to search for third-party libraries to get the job done. This is the main reason why many people recommend that beginners learn Python.
2. **Affordable:** Python is a free resource, so individuals, small businesses, or large organizations can use available free resources to build applications, and Python is a popular and widely used community supply.
3. **Python is for Everyone:** Python can run in any environment, be it Linux, Mac or Windows. Programmers need to learn different languages for different jobs, but with Python, you can professionally use Python to create web applications, perform data analysis and machine learning, automate things, perform web scraping, and create powerful games and visualization initial later rain programming language.

## 1.3 Disadvantages of Python

1. **Speed Limitations**

We have seen that Python code executes line by line. But because Python is interpreted, its execution speed is very slow. This is not a problem, unless speed is the focus of the project work. In other words, the speed is necessary, and the benefits Python provides are enough to distract us from its speed limitations.

1. **Weak in Mobile Computing and Browsers**

Although Python is a good server-side language, Python is rarely seen on the client side. Most importantly, is rarely used to implement smartphone- based applications. One of these applications is called Carbon nelle. Although Bright on exists, it is not so famous because it is not so safe.

1. **Design Restrictions**

Python is a dynamically typed language. This means that you do not need to declare variable types when writing code content. It uses to duck typing. what is that? Well, it just means that if it looks like a duck, then it must be a duck. This is easy for programmers in the coding process, and it will generate runtime errors.

1. **Under developed data base access layer**

Compared to the most widely used technologies such as JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layer is a bit underdeveloped and it is used less in large companies.

1. **Simple**

We are not kidding. Python's simplicity is indeed a problem. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that Java code seems unnecessary.

## 1.4 Install Python on Windows Step by Step

**Step-1:** Go to the official site and use Google Chrome or any other web browser to download and install Python. Or click on the following link: <https://www.python.org/>

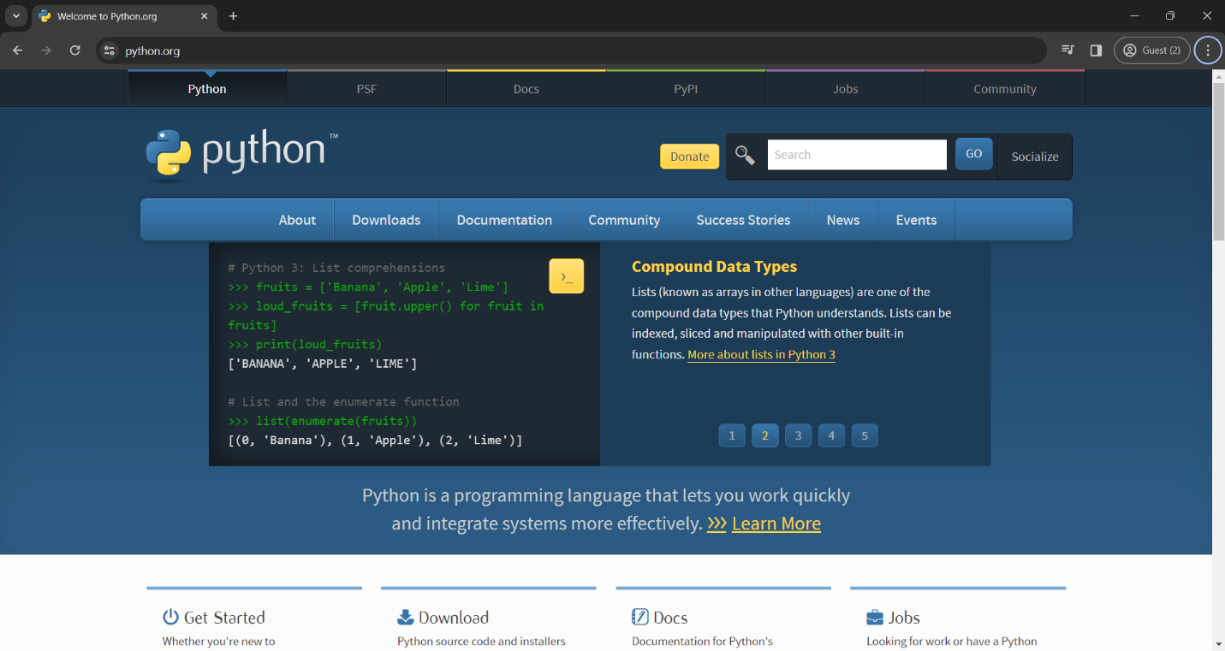


Fig 1.1 Python Official Website

Now check the latest and correct version of your operating system.

**Step-2:** Click on the Download Tab.



Fig 1.2 Python Download Page

**Step-3**: You can select the yellow Download Python for Windows3.7.4 button, or you can scroll download click the download of the corresponding version. Here, we are downloading the latest version of Python for Window 3.7.4.

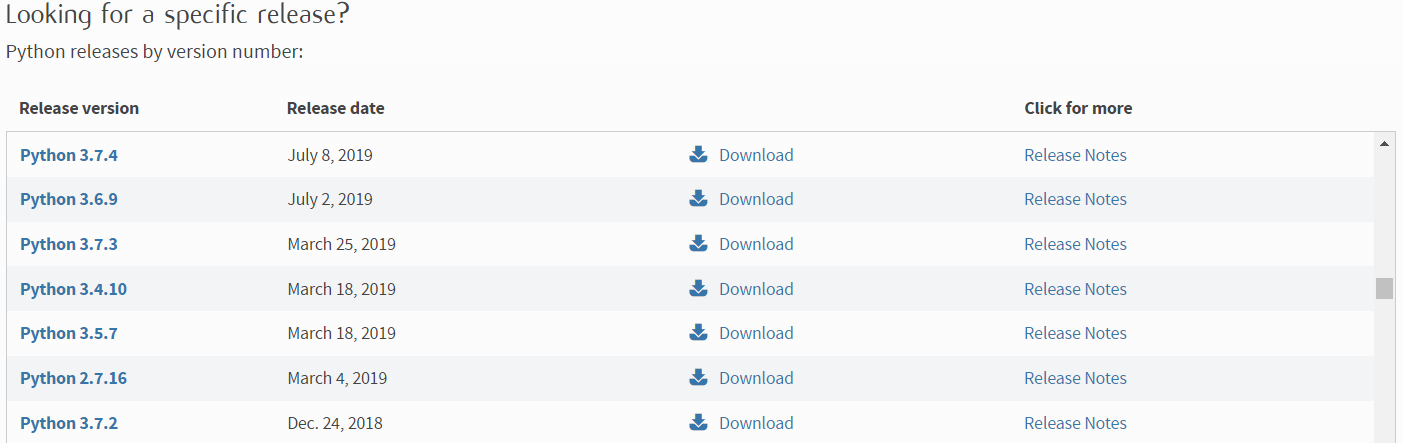


Fig 1.3 Python Versions Available to Download

**Step-4:** Scroll down the page until you find the "File" option.

**Step5:** Here you will see different versions of Python and operating systems.

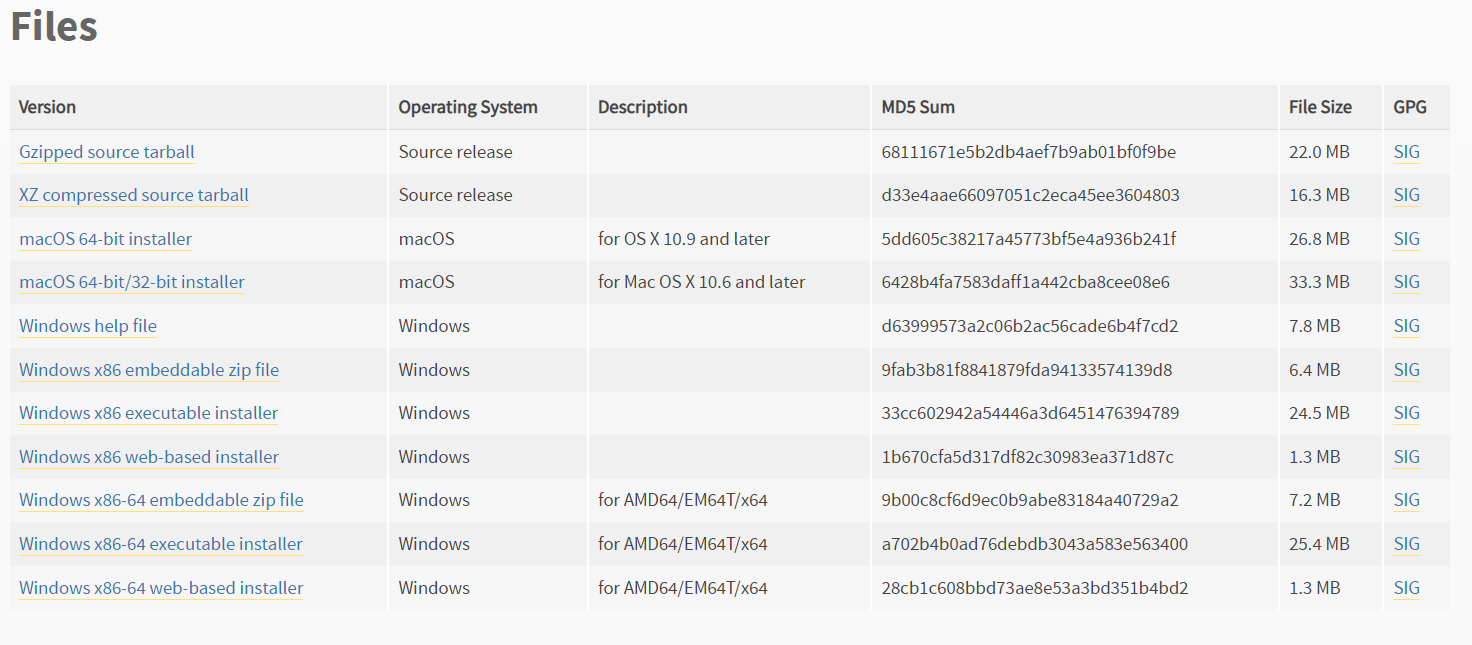


Fig 1.4 Select the OS to download

* To download Windows 32bit Python, you can select the built-in Windows X86 Zip file, WindowsX86 executable installer or WindowsX86 installer on the Web.
* To download Windows 64-bit Python, you can select any option of the three options. Zip File Embeddable Windows X866, Windows X8664 executable installer or Windows X8664 installer based on the web.

## 1.5 Installation of Python

**Step-1:** Goto Download and Open the downloaded python version to carry out the installation process.

**Step-2:** Before you click on Install Now, make sure to put a tick on Add Python 3.7 to Path



Fig 1.5 Python application Setup

**Step-3:** Click on Install Now After the installation is successful. Click on Close.



Fig 1.6 Setup Successful Message

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

**Note:** The installation process might take a couple of minutes.

## 1.6 Verify the Python Installation

**Step-1:** Click on Start

**Step-2:** In the Windows Run Command, type “cmd”.

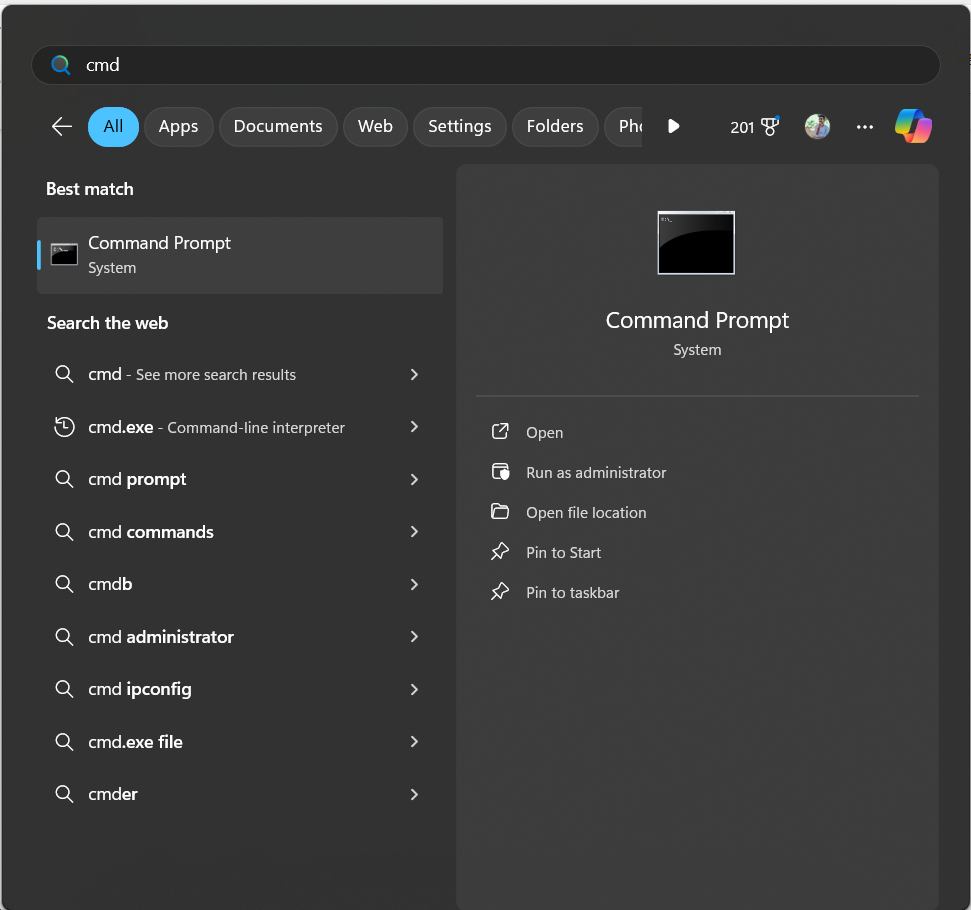


Fig 1.7 Open Command Prompt

**Step-3:** Open the Command prompt option.

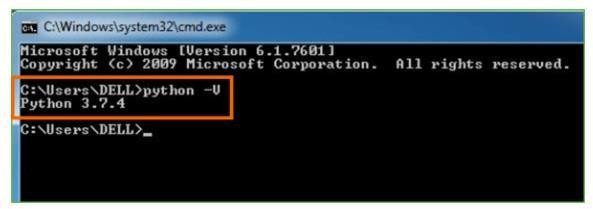


Fig 1.8 Check Whether Python is Installed

**Step-4:** Let us test whether the python is correctly installed. Type **python– V** and press Enter.

**Step-5:** You will get the answer as 3.7.4

**Note:** If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

## 1.7 Check how the Python IDLE works

**Step-1:** Click on Start

**Step-2:** In the Windows Run command, type “python idle”.

**Step-3**: Click on IDLE (Python3.7 64-bit) and launch the program

**Step-4:** To go ahead with working in IDLE

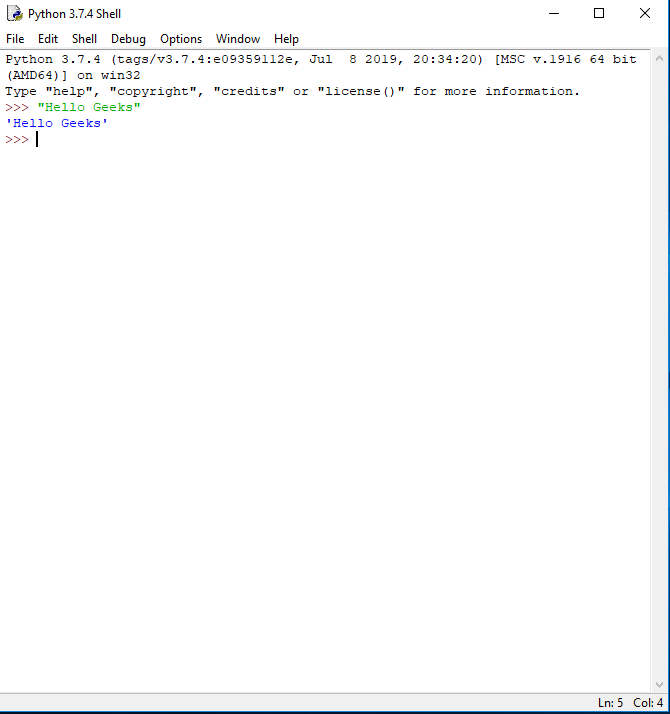


Fig 1.9 Python IDLE Interface

**Step-5:** Name the file and save as type should be Python files. Click on SAVE.

## 1.8 What is VS Code:

Visual Studio Code is the most popular code editor and the IDEs provided by Microsoft for writing different programs and languages. It allows the users to develop new code bases for their applications and allow them to successfully optimize them.

For its high popularity, individuals opt to Install Visual Studio Code on Windows over any other IDE. Installation of Windows Visual Studio Code is not a difficult matter. The Installation process starts with Downloading the Visual Studio Code EXE file to some on-screen instructions.

We are going to list all the steps required to Install VS Code on Windows in a detailed format.

* Quick Preview to Install Visual Studio Code on Windows
* Download the VS Code file from the Official Website.
* Execute the download file.
* Accept the Terms & Conditions.
* Click on the Install button.
* Wait for the installation to complete.
* Click on the Launch button to start it.

## 1.9 Steps to Install Visual Studio Code on Windows

**Step 1:** Visit the Official Website of the Visual Studio Code using any web browser like Google Chrome, Microsoft Edge, etc.



Fig 1.10 Visual Studio Code Official Download Site

**Step 2:** Press the “Download for Windows” button on the website to start the download of the Visual Studio Code Application.

**Step 3:** When the download finishes, then the Visual Studio Code Icon appears in the downloads folder.

**Step 4:** Click on the Installer icon to start the installation process of the Visual Studio Code.

**Step 5:** After the Installer opens, it will ask you to accept the terms and conditions of the Visual Studio Code. Click on I accept the agreement and then click the Next button.

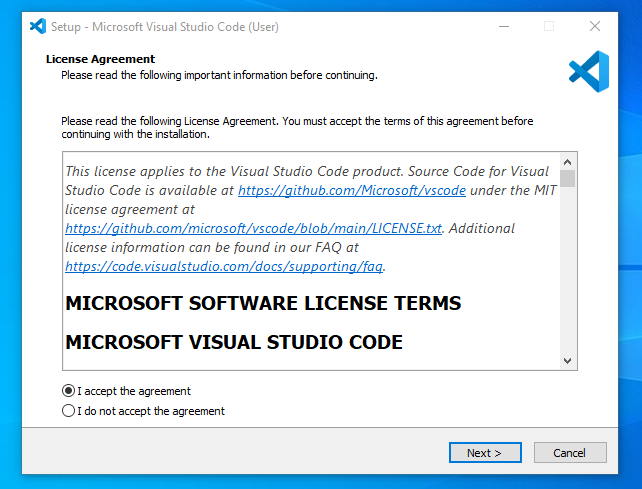


Fig 1.11 VS Code Licence agreement

**Step 6:** Choose the location data for running the Visual Studio Code. It will then ask you to browse the location. Then click on the Next button.

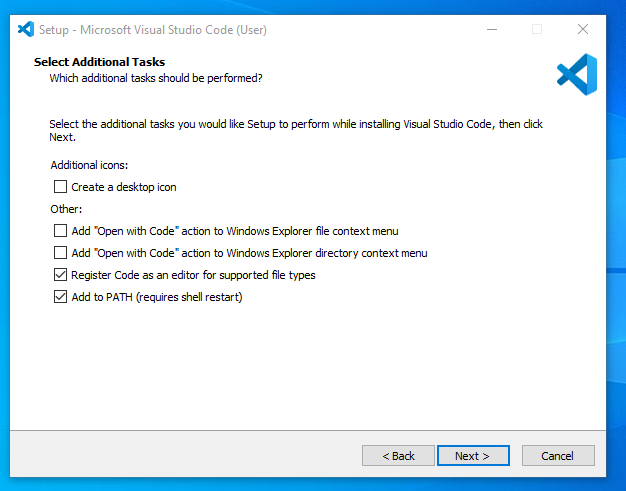


Fig 1.12 Add the VS Code to PATH

**Step 7:** Then it will ask to begin the installation setup. Click on the Install button.

**Step 8:** After clicking on Install, it will take about 1 minute to install the Visual Studio Code on your device.

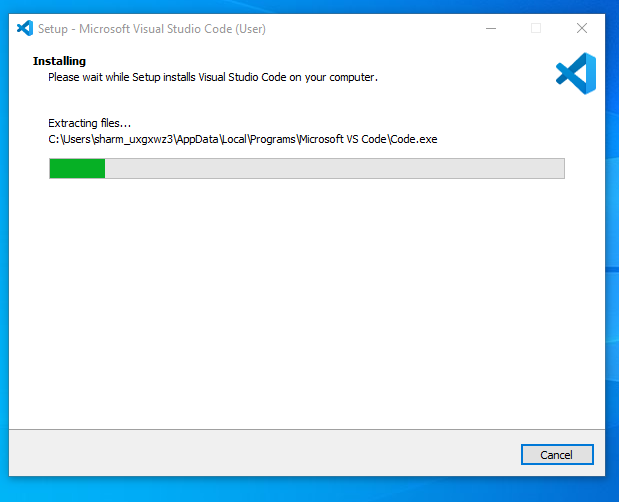


Fig 1.13 VS Code Installation Screen

**Step 9:** After the Installation setup for Visual Studio Code is finished, it will show a window like this below. Tick the “Launch Visual Studio Code” checkbox and then click Next.

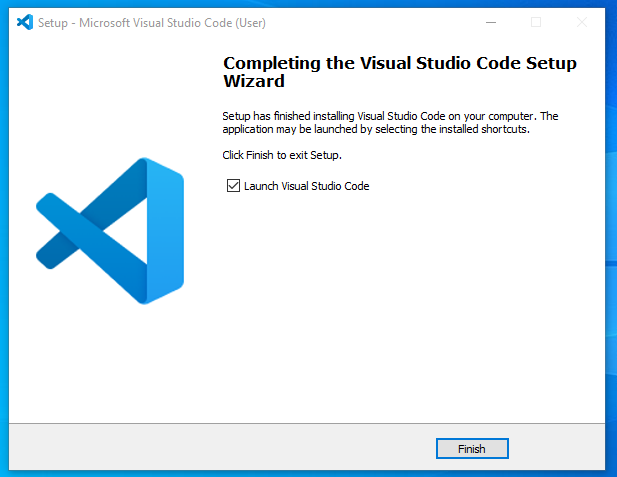


Fig 1.14 VS Code Installation Complete

**Step 10:** After the previous step, the Visual Studio Code window opens successfully. Now you can create a new file in the Visual Studio Code window and choose a language of yours to begin your programming journey!

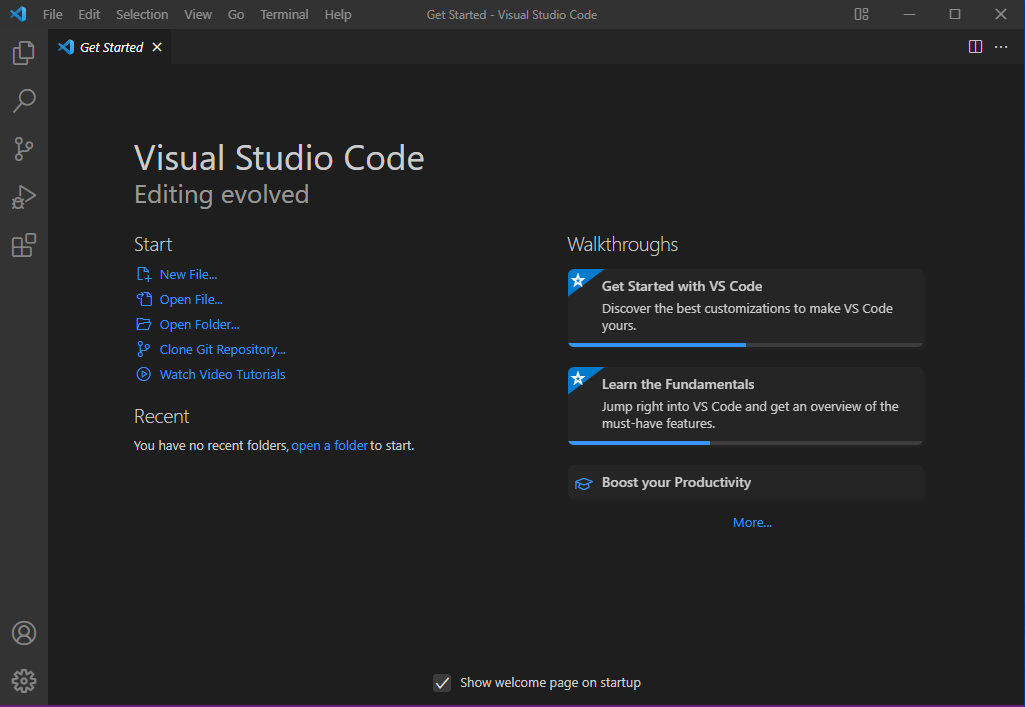


Fig 1.15 VS Code Interface

# SYSTEM ANALYSIS

## 2.1 Existing System

The existing systems for image steganography typically rely on a combination of traditional techniques and software tools tailored to conceal data within digital images. These systems often require specialized software packages or libraries, which may have limited functionalities and may not be easily customizable to suit specific requirements. Furthermore, many existing solutions may lack robustness and security features, leaving them vulnerable to detection or data loss. The complexity involved in implementing steganographic algorithms manually can be a barrier for users without a deep understanding of image processing and cryptography.

### 2.1.1 Drawbacks

1. **Limited Functionality:** Many existing systems for image steganography offer only basic functionalities, such as embedding data using simple LSB (Least Significant Bit) techniques. This limitation restricts the versatility and effectiveness of the system, especially when dealing with more complex steganographic methods or larger volumes of data.
2. **Lack of Robustness:** Some existing systems may lack robustness in terms of hiding data securely within images. Vulnerabilities such as poor encryption methods or inadequate data embedding techniques can lead to the detection or corruption of hidden information, compromising the confidentiality of the communication.
3. **Dependency on Third-party Software:** Many image steganography systems rely on third-party software tools or libraries, which may introduce compatibility issues or dependencies on external resources. This reliance can hinder the portability and flexibility of the system, making it challenging to deploy in different environments or integrate into existing workflows.
4. **Limited Customization:** Existing systems may lack flexibility in terms of customization options, making it difficult for users to tailor the steganographic process to their specific requirements or security needs. This limitation can impede the adoption of the system in diverse use cases.
5. **Complexity for Users**: Implementing steganographic algorithms manually can be complex and daunting for users without a deep understanding of image processing, cryptography, and steganography principles.

## 2.2 Proposed System

The proposed system aims to overcome the limitations of existing image steganography solutions by offering a comprehensive and user-friendly platform for embedding and extracting hidden information within images. It will provide enhanced functionality through a variety of steganographic techniques, including advanced encryption algorithms and frequency domain-based methods, ensuring robust data concealment. Prioritizing robustness and security, the system will implement state-of-the-art encryption techniques to safeguard the confidentiality and integrity of hidden data against detection or loss. Unlike existing systems relying on third-party software, this self-contained solution will enhance portability and compatibility across different environments. Extensive customization options will enable users to tailor the steganographic process to their specific needs, while a user-friendly interface will streamline operations for users of all skill levels. Overall, the proposed system represents a significant advancement in image steganography, offering enhanced functionality, robust security, and user-friendly features to meet the diverse needs of users in today's digital landscape.

### 2.2.1 Advantages

1. **Enhanced Security:** The proposed system implements state-of-the-art encryption techniques and advanced steganographic algorithms, ensuring robust data concealment and safeguarding the hidden information against detection or unauthorized access.
2. **Versatility:** By offering a variety of steganographic techniques beyond simple LSB embedding, the proposed system provides users with a wide range of options to choose from based on their specific requirements and security considerations, thereby enhancing versatility and adaptability.
3. **Improved Robustness:** The system's emphasis on robustness and security features mitigates the risk of data loss or corruption, even in the face of sophisticated attacks or forensic analysis, ensuring the integrity and confidentiality of the hidden information.
4. **Self-contained Solution:** Unlike existing systems reliant on third-party software or libraries, the proposed system is self-contained, minimizing dependencies and enhancing portability and compatibility across different environments, thereby simplifying deployment and usage.
5. **Extensive Customization Options:** The proposed system offers users extensive customization options, allowing them to tailor the steganographic process to their specific needs and preferences, including adjusting embedding parameters, selecting encryption algorithms, and fine-tuning security settings.
6. **User-friendly Interface:** With an intuitive and user-friendly interface, the proposed system streamlines the steganographic process and makes it accessible to users of all skill levels, enhancing usability and reducing the learning curve associated with image steganography.
7. **Comprehensive Solution:** The proposed system provides a comprehensive platform for image steganography, offering functionalities for embedding, extracting hidden information within images, as well as options for monitoring the progress of steganographic operations in real-time, thereby meeting the diverse needs of users in various digital communication scenarios.

## 2.3 Modules

1. **Image Loading and Manipulation Module:**

This module is responsible for loading digital images into the system and performing necessary manipulations for steganographic operations. It involves reading image files from storage, converting them into a suitable data format for processing (such as arrays or matrices), and performing any required preprocessing steps, such as resizing or adjusting colour channels.

1. **Data Embedding Module:**

The Data Embedding module handles the process of hiding information within the loaded images. It involves selecting an appropriate steganographic technique (such as Least Significant Bit (LSB) embedding or its variants) and embedding the data into the image pixels while ensuring minimal perceptual changes to the image. This module may also involve techniques to encode the hidden data to enhance robustness and security against detection or tampering.

1. **Data Extraction Module:**

The Data Extraction module is responsible for retrieving hidden information from steganographic images. It involves analysing the image pixels using appropriate extraction algorithms to detect and extract the embedded data accurately. This module may include error checking and correction mechanisms to ensure the integrity of the extracted data, especially in the presence of noise or distortions.

1. **Encryption and Decryption Module:**

This module handles the encryption and decryption of the data before embedding and after extraction, respectively. It ensures the confidentiality and security of the hidden information by encrypting it using cryptographic algorithms before embedding it into the image. Upon extraction, the encrypted data is decrypted using the corresponding decryption key to retrieve the original information.

1. **User Interface Module:**

The User Interface module provides an interactive interface for users to interact with the system. It includes graphical elements such as windows, menus, buttons, and input fields to facilitate user input and display output. This module allows users to load images, specify data to be embedded, select steganographic parameters, perform embedding and extraction operations, and visualize the results in a user-friendly manner.

## 2.4 Feasibility Study

A feasibility study for the Image Steganography project involves evaluating its technical, economic, and operational viability. Here is a breakdown of the feasibility study:

### 2.4.1 Technical Feasibility

1. **Software Requirements:**

Assess the availability of necessary software tools and libraries for implementing image processing, cryptography, and user interface development in the chosen programming language (e.g., Python).

1. **Hardware Requirements:**

Evaluate the hardware resources required for running the system, including computing power, memory, and storage capacity. Ensure that the hardware infrastructure meets the system's demands.

1. **Skill Requirements:**

Determine if the project team possesses the required skills in image processing, cryptography, and software development to successfully implement and maintain the system.

### 2.4.2 Economic Feasibility

1. **Cost Analysis:**

Estimate the costs associated with software licenses, hardware acquisition, development tools, and personnel expenses (e.g., salaries, training).

1. **Benefit Analysis:**

Assess the potential benefits of the project, such as increased security, improved data confidentiality, and operational efficiency. Compare the anticipated benefits with the projected costs to determine the project's economic viability.

### 2.4.3 Operational Feasibility

1. **User Acceptance:**

Gauge the willingness of end-users (e.g., individuals, organizations) to adopt and use the steganography system. Identify any potential resistance to change and develop strategies to address user concerns.

1. **Integration with Existing Systems**:

Evaluate the compatibility of the steganography system with existing software and infrastructure within the organization. Ensure smooth integration and interoperability to minimize disruptions to existing workflows.

1. **Maintenance and Support:**

Assess the feasibility of providing ongoing maintenance, updates, and technical support for the system post-deployment. Determine the resources required to address software bugs, security vulnerabilities, and user feedback effectively.

### 2.4.4 Legal and Ethical Feasibility

1. **Regulatory Compliance:**

Ensure that the steganography system complies with relevant laws, regulations, and industry standards related to data privacy, security, and intellectual property rights.

1. **Ethical Considerations**:

Evaluate the ethical implications of using steganography for hiding information, including potential misuse for illegal activities or circumvention of security measures. Develop policies and guidelines to promote responsible usage and ethical behavior.

# REQUIREMENT ANALYSIS

## 3.1 Software Requirements

* For developing the application, the following are the

1. **Software Requirements:** Python, Visual Studio Code
2. **Operating Systems supported:** Windows 10 and above

## 3.2 Functional Requirements

1. Encoding and Decoding Algorithms
2. User Interface Design

## 3.3 Hardware Requirements

1. **Processor:** Intel i5 or Above
2. **Ram:** Minimum 4GB
3. **Secondary storage:** Minimum 40 GB

## 3.4 Libraries Required

1. **tkinter:** This is the standard GUI (Graphical User Interface) toolkit for Python. It provides classes and functions for creating and manipulating windows, buttons, labels, text boxes, and other GUI components.
2. **PIL (Python Imaging Library):** This library is used for opening, manipulating, and saving many different image file formats. In the code, it is used for tasks such as opening images, resizing them, and converting them to formats suitable for display in the GUI.
3. **os:** This is a module in Python's standard library for interacting with the operating system. In the code, it is used for getting information about files, such as their size and file path.

# 

# SOFTWARE DESIGN

## 4.1 Methodology

### User Interface Design

Develop an intuitive user interface with two main tabs: "Encrypt" and "Decrypt". Within the "Encrypt" tab, provide options for the user to select an image file and the information/message they want to hide. Implement a file chooser dialog for selecting the image and information files. Include option for user to specify the destination path to save resulting image file after encryption. In the "Decrypt" tab, allow the user to select only the encrypted image file. Provide a destination path input for the user to specify where they want to save the extracted secret information file.

### 4.1.2 Encoding Process

Utilize an algorithm that hides information within multiple layers of the image, starting from the least significant bit (LSB) layer. Before embedding the information, save metadata such as the filename and size of the information file in specific locations within the image. Ensure that the algorithm maintains the quality and integrity of image while hiding the information. Implement a method to encode the information into the image seamlessly, ensuring that the encoded image appears no different from the original to the human eye. Provide feedback to the user upon successful encryption, indicating the location of the encrypted image file.

### 4.1.3 Decoding Process

Develop a decryption algorithm that retrieves the hidden information from the encrypted image. Extract the metadata stored within the image, including the filename and size of the hidden information file. Use this metadata to accurately extract the hidden information from the image. Upon successful extraction, save the extracted information file to the specified destination path. Provide feedback to the user upon successful decryption, indicating the location of the extracted secret information file.

### Testing and Validation

Test the application thoroughly to ensure proper functionality and reliability. Validate the encryption and decryption processes with various types of image files and information/messages. Verify that the application maintains the quality and integrity of the image during encryption and accurately retrieves the hidden information during decryption.

### Documentation and User Guide

Create comprehensive documentation detailing the functionality, usage, and implementation details of the steganography system. Develop a user guide to assist users in navigating and utilizing the application effectively. Include instructions on how to encrypt and decrypt information using the application, along with troubleshooting tips and best practices.

### Deployment and Maintenance

Deploy the application for use by end-users, ensuring compatibility with various operating systems and environments. Provide ongoing maintenance and support to address any issues or updates required post-deployment. Gather user feedback to continuously improve the application and incorporate new features or enhancements as needed.

## 4.2 Data Flow Diagram

* **User Input:** Users provide input in the form of image files, information files, and selecting the operation (Encrypt or Decrypt).
* **Main Menu:** The main menu interface displays options for the user to select the desired operation (Encrypt or Decrypt).
* **Controller:** This component manages the flow of operations based on the user's selection.
* **Encryption:** When the user selects "Encrypt," the encryption module hides the information within the image file.
* **Decryption:** When the user selects "Decrypt," the decryption module extracts the hidden information from the encrypted image file.
* **Save Encrypted Image:** After encryption, the encrypted image file is saved to the specified destination path.
* **Save Decrypted Information:** After decryption, the extracted information is saved to the specified destination path.

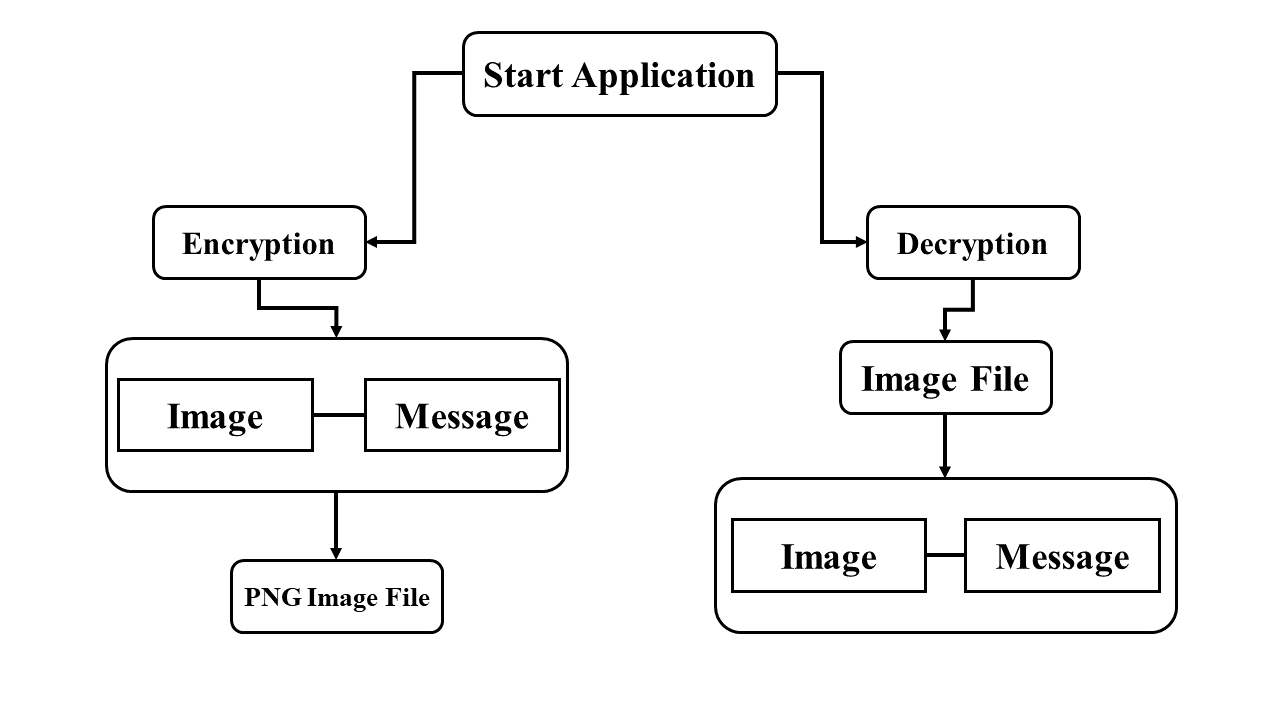


Fig 4.1 Data Flow Diagram

## 4.3 Use Case diagram

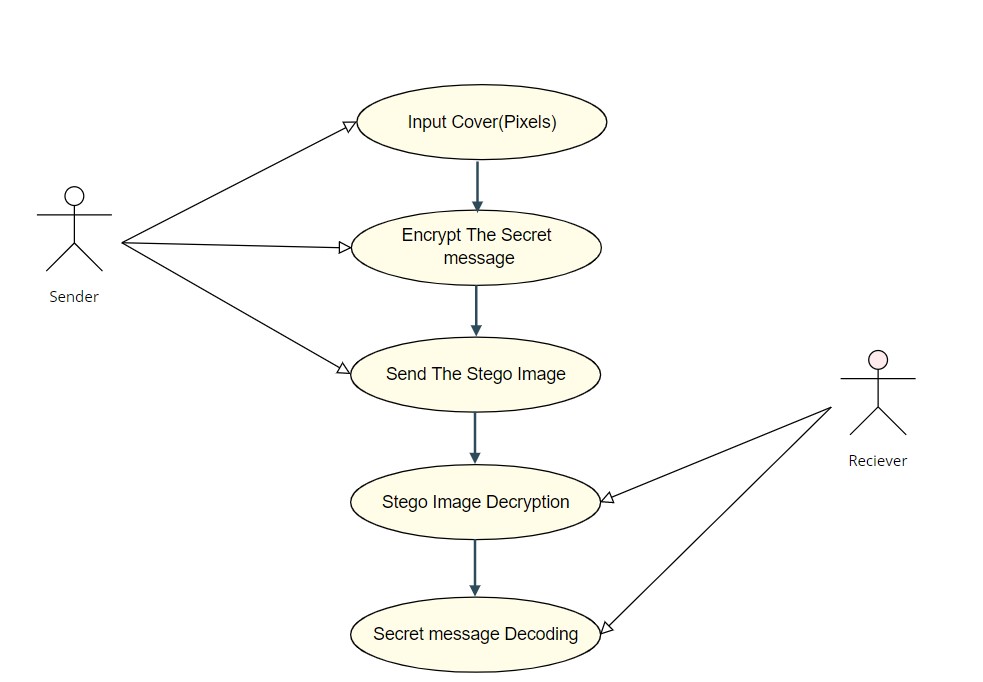


Fig 4.2 Use Case Diagram

In the above user case diagram, the user or the sender provides the input cover image

(pixels) and then encrypts the secret message for sending to the receiver on the other side. After the successful encryption of the secret message and cover image, Stego image is forwarded. The Receiver will receive the stego image that has the secret message to be decoded. The receiver will decrypt the image pixels; extract out the last significant bits and then Decode the Secret message successfully.

## 4.4 Sequence Diagram

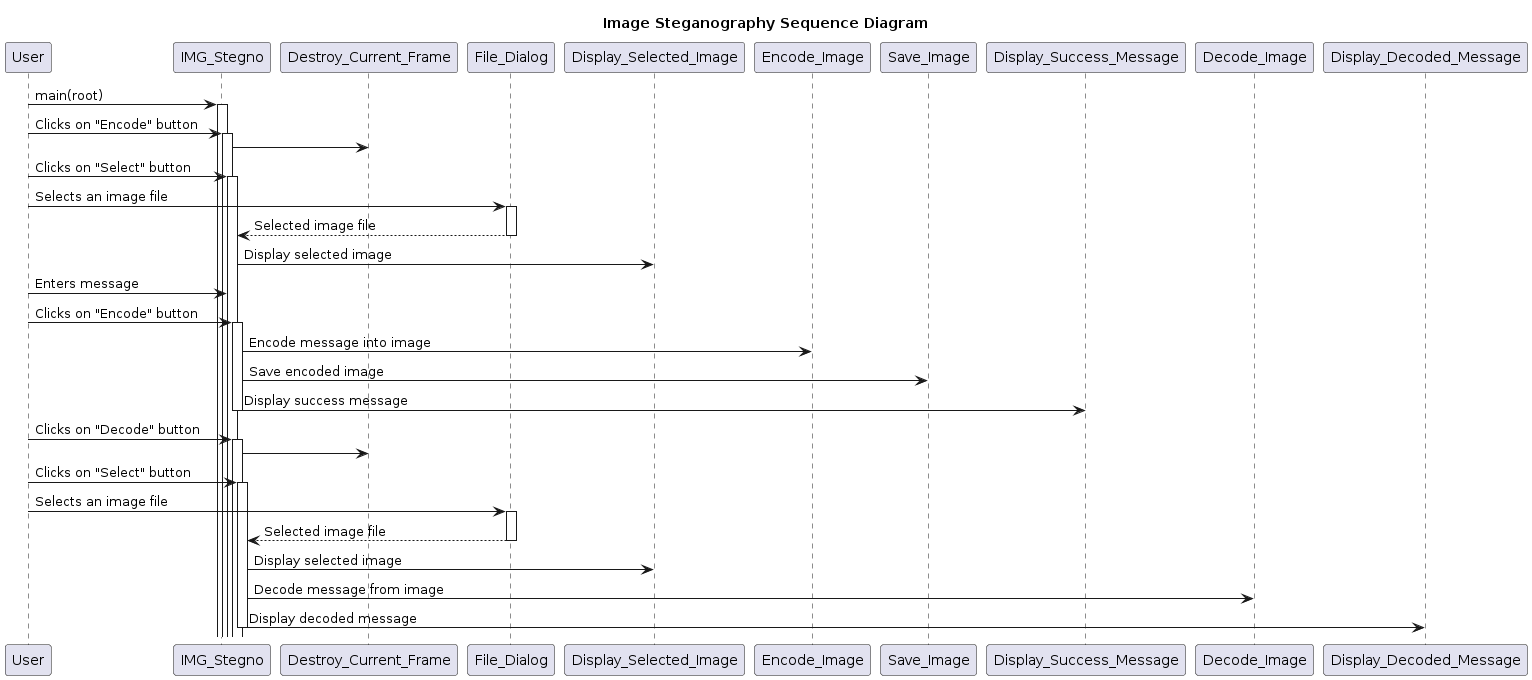


Fig 4.3 Sequence Diagram

The sequence diagram illustrates the flow of interactions between the end user and the Image Steganography application.

Here's a description of the diagram:

**Initialization:** The user initiates the application by executing the main function, which sets up the graphical user interface (GUI) represented by IMG\_Stegno.

**Encode Process:**

1. The user clicks on the "Encode" button, triggering the encode\_frame1 method.
2. The current frame is destroyed to prepare for the next step.
3. Upon clicking the "Select" button, a file dialog opens for the user to choose an image file.
4. The user selects an image file, and the application displays the selected image.
5. The user enters a message into the text field provided.
6. Clicking the "Encode" button initiates the encoding process, where the message is embedded into the selected image.
7. The encoded image is then saved, and a success message is displayed to the user.

**Decode Process:**

1. The user decides to decode a message by clicking on the "Decode" button.
2. Similar to the encoding process, the current frame is destroyed, and the user selects an image file.
3. The selected image is displayed, and the application proceeds to extract and decode any hidden message within the image.
4. Finally, the decoded message is displayed to the user.

## 4.5 State Diagram

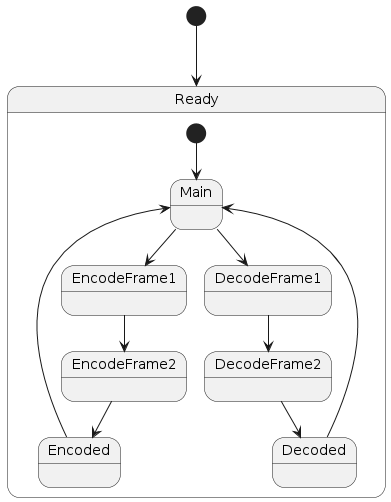


Fig 4.4 State Diagram

Here are the states in state diagram

**Ready State:**

* This is the initial state when the program starts.
* Transitions to the Main State when the program begins.

**Main State:**

* Represents the main interface where the user can choose between encoding or decoding.
* **Transitions to:**
* EncodeFrame1 when the "Encode" button is clicked.
* DecodeFrame1 when the "Decode" button is clicked.

**EncodeFrame1 State:**

* Represents the interface for selecting the image to hide text.
* Transitions to EncodeFrame2 after an image is selected.

**EncodeFrame2 State:**

* Represents the interface for entering the text to hide.
* Transitions to Encoded State after the "Encode" button is clicked.

**DecodeFrame1 State:**

* Represents the interface for selecting the image with hidden text.
* Transitions to DecodeFrame2 after an image is selected.

**DecodeFrame2 State:**

* Represents the interface for displaying the hidden text.
* Transitions to Decoded State after decoding the text.

**Encoded State:**

* Indicates successful encoding of the text into the image.
* Transitions back to Main State after encoding is completed.

**Decoded State:**

* Indicates successful decoding of the text from the image.
* Transitions back to Main State after decoding is completed.

## 4.6 Package Diagram

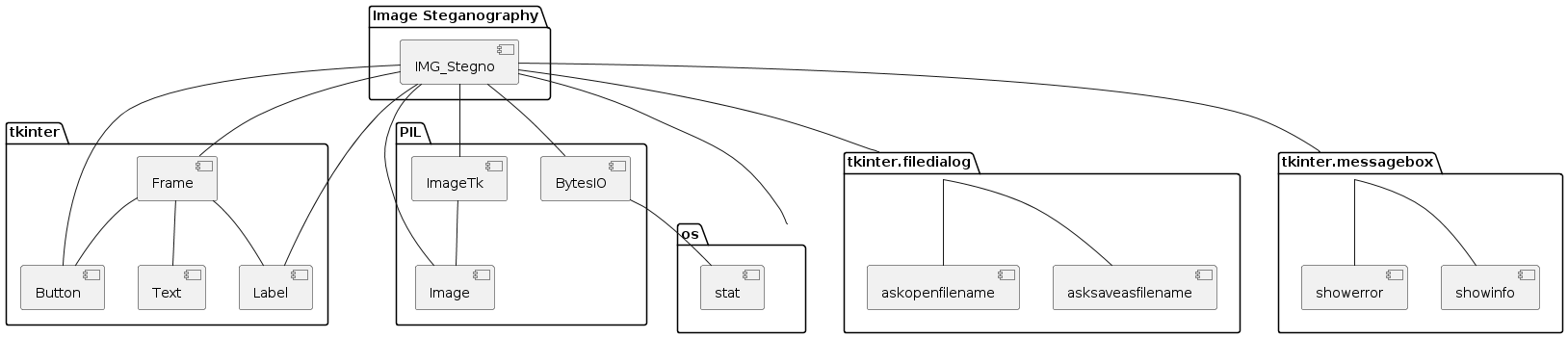


Fig 4.5 Package Diagram

**Image Steganography Package:** Represents the main class IMG\_Stegno responsible for the image steganography functionality.

**tkinter Package:** Contains the graphical user interface (GUI) components used for building the application's interface, including frames, labels, buttons, and text fields.

**PIL Package:** Contains components related to image processing, such as ImageTk, Image, and BytesIO, used for handling image data.

**os Package:** Provides functionality for interacting with the operating system, particularly the stat function, used for retrieving information about files.

**tkinter.filedialog Package:** Includes functions for opening and saving files (askopenfilename and asksaveasfilename), which are utilized for selecting image files.

**tkinter.messagebox Package:** Contains functions for displaying different types of message boxes (showerror and showinfo), used for showing error and informational messages to the user.

These components represent the key elements of the Image Steganography project and its dependencies on external libraries for GUI development, image processing, file handling, and user interaction.

# IMPLEMENTATION

## 5.1 Steganography Technique

Over the past few years, numerous steganography techniques that embed hidden messages in multimedia objects have been proposed. There have been many techniques for hiding information or messages in images in such a manner that alteration made to the image is perceptually indiscernible. Commonly approaches are including LSB, Masking and filtering and Transform techniques.

Least significant bit (LSB) insertion is a simple approach to embedding information in image file. The simplest steganography techniques embed the bits of the message directly into least significant bit plane of the cover-image in a deterministic sequence. Modulating the least significant bit does not result in human perceptible difference because the amplitude of the change is small. In this technique, the embedding capacity can be increased by using two or more least significant bits. At the same time, not only the risk of making the embedded message statistically detectable increase but also the image fidelity degrades. Hence a variable size LSB embedding schema is presented, in which the number of LSBs used for message embedding/extracting depends on the local characteristics of the pixel.

The advantage of LSB-based method is easy to implement and high message pay-load. Although LSB hides the message in such way that the humans do not perceive it, it is still possible for the opponent to retrieve the message due to the simplicity of the technique. Therefore, malicious people can easily try to extract the message from the beginning of the image if they are suspicious that there exists secret information that was embedded in the image. Therefore, a system named Secure Information Hiding System (SIHS) is proposed to improve the LSB scheme. It overcomes the sequence-mapping problem by embedding the massage into a set of random pixels, which are scattered on the cover-image. Masking and filtering techniques, usually restricted to 24 bits and gray scale image, hide information by marking an image, in a manner similar to paper watermarks. The technique performs analysis of the image, thus embed the information in significant areas so that the hidden message is more integral to cover image than just hiding it in the noise level. Transform techniques embed the message by modulating coefficient in a transform domain, such as the Discrete Fourier Transform, or Wavelet Transform. These methods hide messages in significant areas of the cover image, which make them more robust to attack. Transformations can be applied over the entire image, to block throughout the image, or other variant.

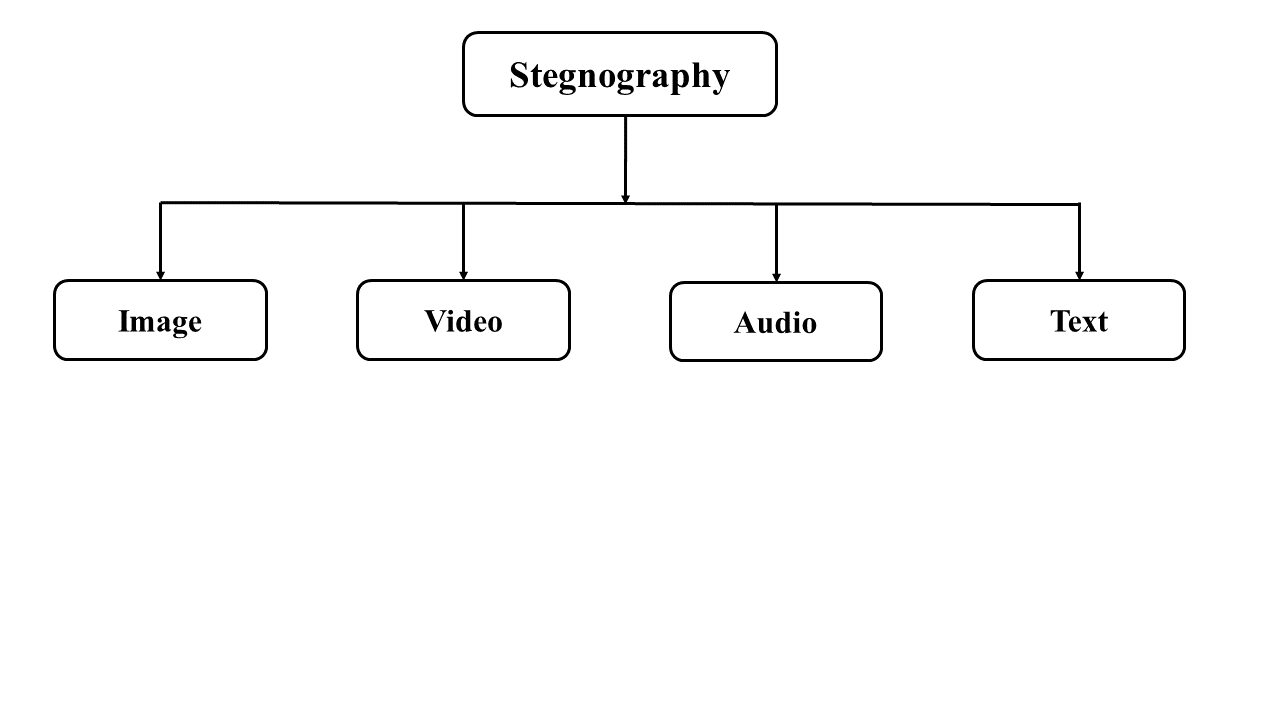


Fig 5.1 Different Mediums to Achieve Steganography

### 5.1.1 Image Steganography

To hide the information use of pixel intensities is done, if the cover object taken is image, then it is known as image steganography. A watermark diagram changes the cover of the object (for example, the identity of the owner). In other words, adding a watermark improves the cover source using only additional data.

### 5.1.2 Video Steganography:

Digital video format is used to hide any type of information in video steganography. In this technique for hiding information in images in the video the discrete cosine transform (DCT) adjusts the value which is not conspicuous through human eye. AVI, Mp4, etc video formats are used by video steganography.

### 5.1.3 Audio Steganography:

Audio steganography is one of the most significant medium due to demand of Voice Over Internet Protocol (VOIP). As in this technique audio is chosen for information hiding, that’s why it is called Audio Steganography. WAVE, MPEG, etc. digital audio formats are used in audio steganography.

### 5.1.4 Text Steganography:

In this technique, for information hiding, capital letters, white spaces, number of tabs and many others are used.

## 5.2 Phases of Steganography

For every hidden message exchanging process from sender to receiver, every Steganography algorithm must come through various stages

1. **Sender:** The prime objective of the sender is to embed the hidden message in the stego- medium and transmit it through the channel of communication.
2. **Communication channel:** A physical or wireless medium that holds an encoded cover picture across the network or some other distribution medium with a hidden.
3. **Receiver:** In this steganography process, it is the last stage where the cover medium is retrieved and extracted to see if the hidden text that was sent over the communication channel

The framework of image steganography refers to the general steps involved in embedding secret information into an image while maintaining the image’s perceptual integrity.

The framework typically includes the following steps:

1. **Image selection:** The first step in the framework is to select an appropriate cover image that will be used to hide the secret information. The cover image should be large enough to accommodate the secret message and should not have any noticeable visual changes after the message is embedded.
2. **Message selection:** The second step is to select the message to be hidden in the cover image. The message can be any form of digital data, including text, audio, or video.
3. **Embedding:** The next step is to embed the message into the cover image using a steganography algorithm. The steganography algorithm should ensure that the message is hidden securely and that the image's perceptual integrity is not compromised. proposing solutions for optimizing their performance and efficiency

The above steps provide a high-level overview of the image steganography framework. The specific details of each step will depend on the steganography algorithm used and the requirements of the specific application.

## 5.3 Least significant bit (LSB)

LSB is very simple method to embed data in digital image, audio, or video file. The technique works by altering the least significant bits of the pixels in an image, which have little impact on the overall appearance of the image.

LSB substitution has 2 stages: ENCODING and DECODING

### 5.3.1 LSB Encoding

This phase of the algorithm involves the sender’s part. The main aim of the encoding is to alter the least significant bit and adding the secret message bits to it.

Let us consider an example:

1. Suppose there is the secret Message “H” that is to be encoded in the random RGB pixels given below:

R = 1001101**1**

G = 1011001**0**

B = 0100011**0**

1. ASCII value of the secret message which here is the letter “H” is 72.
2. The 8-bit binary representation of 72 will be 01001000
3. The task is to encode the 8 bit binary digits of 72 in the given RGB pixels.
4. Below will be the basic idea of the bit alteration. And then the altered bit pixels will be sent to the receiver.

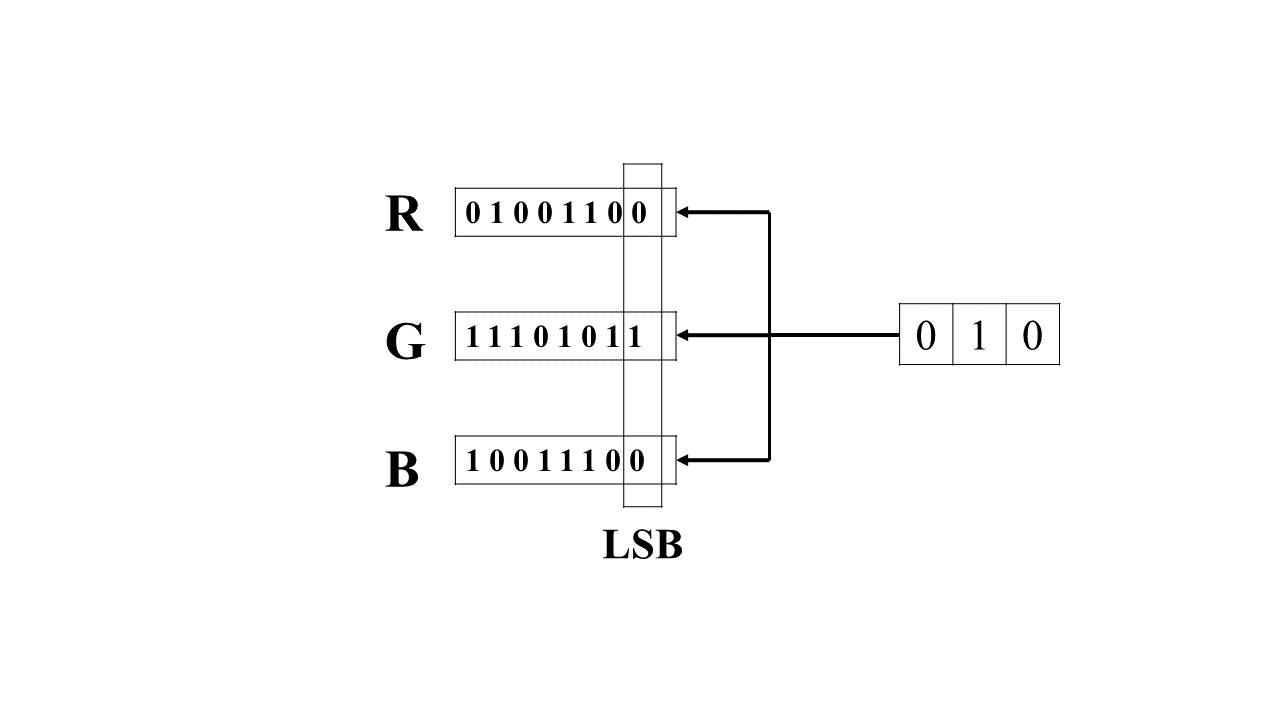


Fig 5.2 Process of Embedding 3-Bit into Pixel

### 5.3.2 LSB Decoding

This is the algorithm that concerns the receiver. After the successful sending of the stegno image containing the altered RGB pixel bits, the task of the receiver is to extract the least significant bits of the stegno image pixels and combine them in the pairs of 8 and then get the desired 8 bits.

Taking the above example:

1. The receiver will be receiving the stegno image pixels (Secret message + original image). Suppose the following RGB pixels will be received:

**R G B**

**10010100, 10010101, 10010100**

**10010100, 10010101, 10010100**

**10010100, 10010100, 10010100**

1. So the red colored bits in the above picture are least significant bits of the stegno image pixels. The aim of the decoding is to take these red colored bits individually, combine them and then find out the decimal number of the that 8 bit representation.
2. The decimal number calculated will be the ASCII code of the secret message that is embedded in the original cover image.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **0** | **0** | **1** | **0** | **0** | **0** |

The red bits combined will look like:

1. Finding the Decimal number of the above bits will give the value 72 which is the ASCII code of the letter “H”. Finally, the decoded word will be the letter H.
2. So, the LSB substitution Image Steganography without majorly changing the image quality, successfully allows the data hiding process.
3. It is mainly used for the Lossless Cover image format such as PNG.

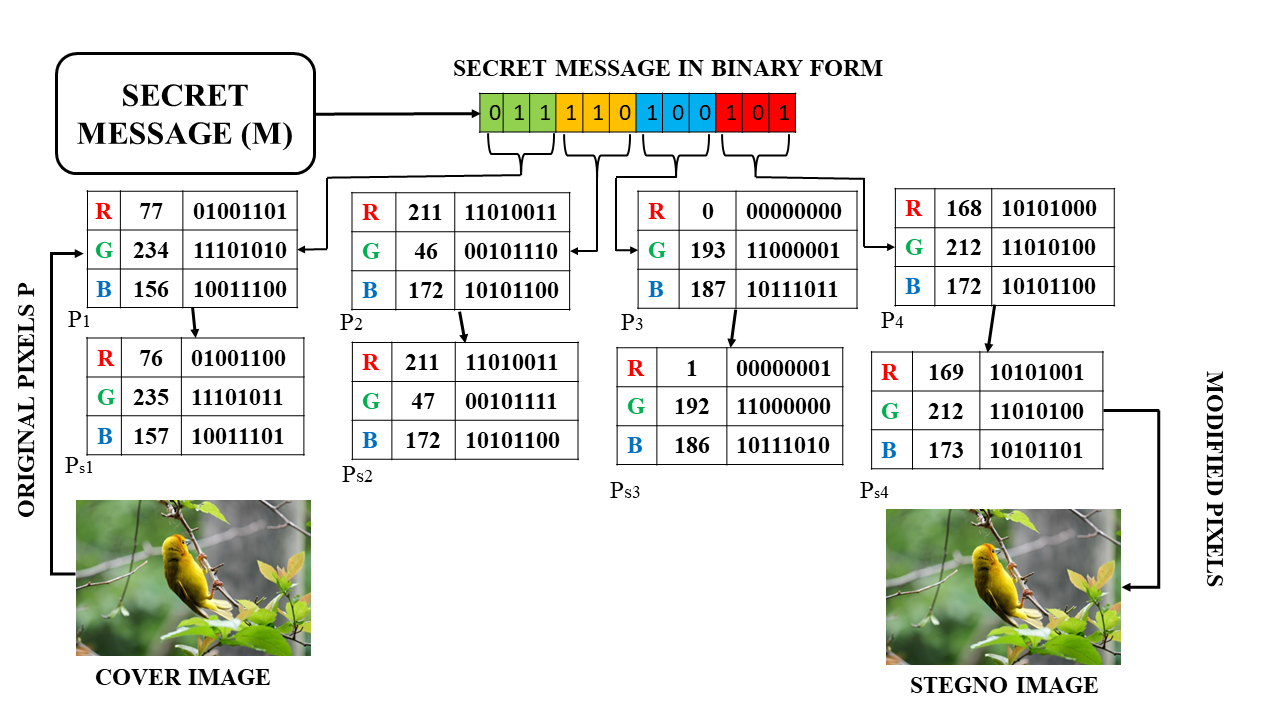


Fig 5.3 LSB Substitution Architecture

## 5.4 Image Steganography using Python

Image Steganography is a python project in which we hide the secret message inside any image using Tkinter and the PIL module. Let us start the development. Image steganography is a GUI-based project in which we are hiding a secret message within the image using encoding and decoding functions. We are creating a window in which there are two buttons: encoding and decoding. For encoding, select any image, this image will be converted into PNG format. Type message in the message box then it will convert into base64, merge this encoded string into image and the user can save the image where he/she wants. For decoding, select the image which is encoded, the base 64 string will get separated by decoding, and by Tkinter module hidden text is shown in the textbox.

### 5.4.1 Project Prerequisite

This project requires good knowledge of python and the Tkinter library. Tkinter is the python binding to the Tk toolkit which is used across many programming languages for building the Graphical user interface which is a GUI. Also, we require a PIL module. This is the images module from the pillow. The PIL module helps to open, manipulate and save many different forms of images.

### 5.4.2 Steps to Build a Python Image Steganography Project

1. Import Necessary Libraries:

from tkinter import \*

import tkinter.filedialog

from tkinter import messagebox

from PIL import ImageTk, Image

import os

1. Create the IMG\_Stegno Class:

* Define class variables.
* Define methods for encoding, decoding, and other functionalities.

1. Create the GUI using Tkinter:

* Define the main method to create the main window and GUI components.
* Use Frame, Label, Button, Text, etc., to design the GUI.
* Use geometry and resizable methods to set window size and resize options.

1. Handle User Interactions:

* Use command attribute to link GUI components with corresponding methods.
* Implement methods like encode\_frame1, decode\_frame1, etc., to handle user actions.
* Implement Encoding and Decoding Functions:
* Implement methods like encode, decode, etc., to perform image steganography operations.
* Utilize PIL library to manipulate images (open, resize, save, etc.).

1. Handle File Operations:

* Use tkinter.filedialog to prompt the user to select files for encoding or decoding.
* Utilize askopenfilename and asksaveasfilename methods to handle file selection and saving.

1. Display Messages and Errors:

* Use tkinter.messagebox to display messages and errors to the user.
* Utilize showerror and showinfo methods to show error and informational messages.

1. Run the Application:

* Create a Tkinter window using Tk class.
* Create an instance of the IMG\_Stegno class.
* Call the main method to build the GUI.
* Run the Tkinter event loop using mainloop method.

## 5.5 Applications of The Project

1. **Securing data while Breaches**: Image steganography is an important technique used in digital forensics area during the occurrences of the data breaches or the data leak incidents. Basically, whenever a data breach occurs, crucial data can be hidden using the steganography technique.
2. **End-to-End Private communication:** If a client and a server want to share some peer-to-peer confidential information over a network, then they can use the data hiding and steganography technique to avoid any outer unauthorized interference.
3. **Protection of Intellectual property Rights and Ownership:** Data hiding through image steganography can be used to hide the owner’s designations or signatures secretly in any original cover image, audio, video, or text so that the ownership rights remain with the owner and others cannot steal the owner’s work.
4. **Multimedia Fingerprinting:** This is like the ownership rights. The owner of the media can embed the secret copyrights on the covers so that the originality of the owner’s work remains unaltered.
5. **Smart Identity Cards:** Various Intelligence organizations make use of the image steganography to hide the details of their employees in their ID card images.

## 5.6 Source Code

from tkinter import \*

import tkinter.filedialog

from tkinter import messagebox

from PIL import ImageTk

from PIL import Image

from io import BytesIO

import  os

class IMG\_Stegno:

    output\_image\_size = 0

    def main(self, root):

        root.title('Image Steganography Project')

        root.geometry('600x600')

        root.resizable(width =False, height=False)

        frame = Frame(root)

        frame.grid()

        frame.place(relx=.5, rely=.5, anchor="center")

        title = Label(frame,text='Image Steganography')

        title.config(font=('Times new roman',18, 'bold'))

        title.grid(pady=10)

        title.grid(row=1)

        encode = Button(frame,text="Encode",command= lambda :self.encode\_frame1(frame), padx=14,bg = '#e3f4f1' )

        encode.config(font=('Arial Black',14,'bold'), bg='#586EE8',fg='#FFFFFF')

        encode.grid(row=2)

        decode = Button(frame, text="Decode",command=lambda :self.decode\_frame1(frame), padx=14,bg = '#e3f4f1')

        decode.config(font=('Arial Black',14,'bold'), bg='#586EE8',fg='#FFFFFF')

        decode.grid(pady = 12)

        decode.grid(row=3)

        root.grid\_rowconfigure(1, weight=1)

        root.grid\_columnconfigure(0, weight=1)

    def back(self,frame):

        frame.destroy()

        self.main(root)

    def encode\_frame1(self,F):

        F.destroy()

        F2 = Frame(root)

        F2.grid()

        F2.place(relx=.5, rely=.5, anchor="center")

        label1= Label(F2,text='Select The Image In Which You Want To Hide Text :')

        label1.config(font=('Times new roman',18, 'bold'))

        label1.grid()

        button\_bws = Button(F2,text='Select',command=lambda : self.encode\_frame2(F2))

        button\_bws.config(font=('Arial Black',14,'bold'), bg='#586EE8',fg='#FFFFFF')

        button\_bws.grid()

        button\_back = Button(F2, text='Cancel', command=lambda : IMG\_Stegno.back(self,F2))

        button\_back.config(font=('Arial Black',14,'bold'),bg='#586EE8',fg='#FFFFFF')

        button\_back.grid(pady=15)

        button\_back.grid()

    def decode\_frame1(self,F):

        F.destroy()

        d\_f2 = Frame(root)

        d\_f2.grid()

        d\_f2.place(relx=.5, rely=.5, anchor="center")

        label1 = Label(d\_f2, text='Select Image with Hidden text:')

        label1.config(font=('Times new roman',18,'bold'))

        label1.grid()

        button\_bws = Button(d\_f2, text='Select', command=lambda :self.decode\_frame2(d\_f2))

        button\_bws.config(font=('Arial Black',14,'bold'), bg='#586EE8',fg='#FFFFFF')

        button\_bws.grid()

        button\_back = Button(d\_f2, text='Cancel', command=lambda : IMG\_Stegno.back(self,d\_f2))

        button\_back.config(font=('Arial Black',14,'bold'), bg='#586EE8',fg='#FFFFFF')

        button\_back.grid(pady=15)

        button\_back.grid()

    def encode\_frame2(self,e\_F2):

        e\_pg= Frame(root)

        e\_pg.grid()

        e\_pg.place(relx=.5, rely=.5, anchor="center")

        myfile = tkinter.filedialog.askopenfilename(filetypes = ([('png', '\*.png'),('jpeg', '\*.jpeg'),('jpg', '\*.jpg'),('All Files', '\*.\*')]))

        if not myfile:

            messagebox.showerror("Error","You have selected nothing !")

        else:

            my\_img = Image.open(myfile)

            new\_image = my\_img.resize((300,200))

            img = ImageTk.PhotoImage(new\_image)

            label3= Label(e\_pg,text='Selected Image')

            label3.config(font=('Times new roman',14,'bold'))

            label3.grid()

            board = Label(e\_pg, image=img)

            board.image = img

            self.output\_image\_size = os.stat(myfile)

            self.o\_image\_w, self.o\_image\_h = my\_img.size

            board.grid()

            label2 = Label(e\_pg, text='Enter the message')

            label2.config(font=('Times New Roman',18,'bold'))

            label2.grid(pady=15)

            text\_a = Text(e\_pg, width=50, height=10)

            text\_a.grid()

            encode\_button = Button(e\_pg, text='Cancel', command=lambda : IMG\_Stegno.back(self,e\_pg))

            encode\_button.config(font=('Arial Black',14,'bold'), bg='#586EE8',fg='#FFFFFF')

            data = text\_a.get("1.0", "end-1c")

            button\_back = Button(e\_pg, text='Encode', command=lambda : [self.enc\_fun(text\_a,my\_img),IMG\_Stegno.back(self,e\_pg)])

            button\_back.config(font=('Arial Black',14,'bold'), bg='#586EE8',fg='#FFFFFF')

            button\_back.grid(pady=15)

            encode\_button.grid()

            e\_pg.grid(row=1)

            e\_F2.destroy()

    def decode\_frame2(self,d\_F2):

        d\_F3 = Frame(root)

        d\_F3.grid(row=1)

        d\_F3.place(relx=.5, rely=.5, anchor="center")

        myfiles = tkinter.filedialog.askopenfilename(filetypes = ([('png', '\*.png'),('jpeg', '\*.jpeg'),('jpg', '\*.jpg'),('All Files', '\*.\*')]))

        if not myfiles:

            messagebox.showerror("Error","You have selected nothing! ")

        else:

            my\_img = Image.open(myfiles, 'r')

            my\_image = my\_img.resize((300, 200))

            img = ImageTk.PhotoImage(my\_image)

            label4= Label(d\_F3,text='Selected Image :')

            label4.config(font=('Times New Roman',14,'bold'))

            label4.grid()

            board = Label(d\_F3, image=img)

            board.image = img

            board.grid()

            hidden\_data = self.decode(my\_img)

            label2 = Label(d\_F3, text='Hidden data is :')

            label2.config(font=('Times New Roman',14,'bold'))

            label2.grid(pady=10)

            text\_a = Text(d\_F3, width=50, height=10)

            text\_a.insert(INSERT, hidden\_data)

            text\_a.configure(state='disabled')

            text\_a.grid()

            button\_back = Button(d\_F3, text='Main Menu', command= lambda :self.frame\_3(d\_F3))

            button\_back.config(font=('Arial Black',14,'bold'),bg='#586EE8',fg='#FFFFFF')

            button\_back.grid(pady=15)

            button\_back.grid()

            d\_F2.destroy()

    def decode(self, image):

        image\_data = iter(image.getdata())

        data = ''

        while (True):

            pixels = [value for value in image\_data.\_\_next\_\_()[:3] +

                      image\_data.\_\_next\_\_()[:3] +

                      image\_data.\_\_next\_\_()[:3]]

            binary\_str = ''

            for i in pixels[:8]:

                if i % 2 == 0:

                    binary\_str += '0'

                else:

                    binary\_str += '1'

            data += chr(int(binary\_str, 2))

            if pixels[-1] % 2 != 0:

                return data

    def generate\_Data(self,data):

        new\_data = []

        for i in data:

            new\_data.append(format(ord(i), '08b'))

        return new\_data

    def modify\_Pix(self,pix, data):

        dataList = self.generate\_Data(data)

        dataLen = len(dataList)

        imgData = iter(pix)

        for i in range(dataLen):

            # Extracting 3 pixels at a time

            pix = [value for value in imgData.\_\_next\_\_()[:3] +

                   imgData.\_\_next\_\_()[:3] +

                   imgData.\_\_next\_\_()[:3]]

                      for j in range(0, 8):

                if (dataList[i][j] == '0') and (pix[j] % 2 != 0):

                    if (pix[j] % 2 != 0):

                        pix[j] -= 1

                elif (dataList[i][j] == '1') and (pix[j] % 2 == 0):

                    pix[j] -= 1

               if (i == dataLen - 1):

                if (pix[-1] % 2 == 0):

                    pix[-1] -= 1

            else:

                if (pix[-1] % 2 != 0):

                    pix[-1] -= 1

            pix = tuple(pix)

            yield pix[0:3]

            yield pix[3:6]

            yield pix[6:9]

    def encode\_enc(self,newImg, data):

        w = newImg.size[0]

        (x, y) = (0, 0)

        for pixel in self.modify\_Pix(newImg.getdata(), data):

            # Putting modified pixels in the new image

            newImg.putpixel((x, y), pixel)

            if (x == w - 1):

                x = 0

                y += 1

            else:

                x += 1

    def enc\_fun(self,text\_a,myImg):

        data = text\_a.get("1.0", "end-1c")

        if (len(data) == 0):

            messagebox.showinfo("Alert","Kindly enter text in TextBox")

        else:

            newImg = myImg.copy()

            self.encode\_enc(newImg, data)

            my\_file = BytesIO()

            temp=os.path.splitext(os.path.basename(myImg.filename))[0]

            newImg.save(tkinter.filedialog.asksaveasfilename(initialfile=temp,filetypes = ([('png', '\*.png')]),defaultextension=".png"))

            self.d\_image\_size = my\_file.tell()

            self.d\_image\_w,self.d\_image\_h = newImg.size

            messagebox.showinfo("Success","Encoding Successful")

    def frame\_3(self,frame):

        frame.destroy()

        self.main(root)

root = Tk() #GUILOOP

o = IMG\_Stegno()

o.main(root)

root.mainloop()

# TESTING

## 6.1 System Test

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

## Types of Tests

### 6.2.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### 6.2.2 Integrated Testing

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

### 6.2.3 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

**Functional testing is Centered on the following items:**

1. **Valid Input:** Identified classes of valid input must be accepted.
2. **Invalid Input:** Identified classes of invalid input must be rejected
3. **Functions:** Identified functions must be exercised.
4. **Output:** Identified classes of application outputs must be exercised.
5. **Systems/Procedures:** Interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

### 6.2.4 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure, and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

### 6.2.5 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

## 6.3 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

### 6.3.1 Test objectives

1. All field entries must work properly.
2. Pages must be activated from the identified link.
3. The entry screen, messages and responses must not be delayed.

### 6.3.2 Features to be tested

1. Verify that the entries are of the correct format
2. No duplicate entries should be allowed
3. All links should take the user to the correct page.

## 6.4 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:**

All the test cases mentioned above passed successfully. No defects encountered.

# OUTPUT

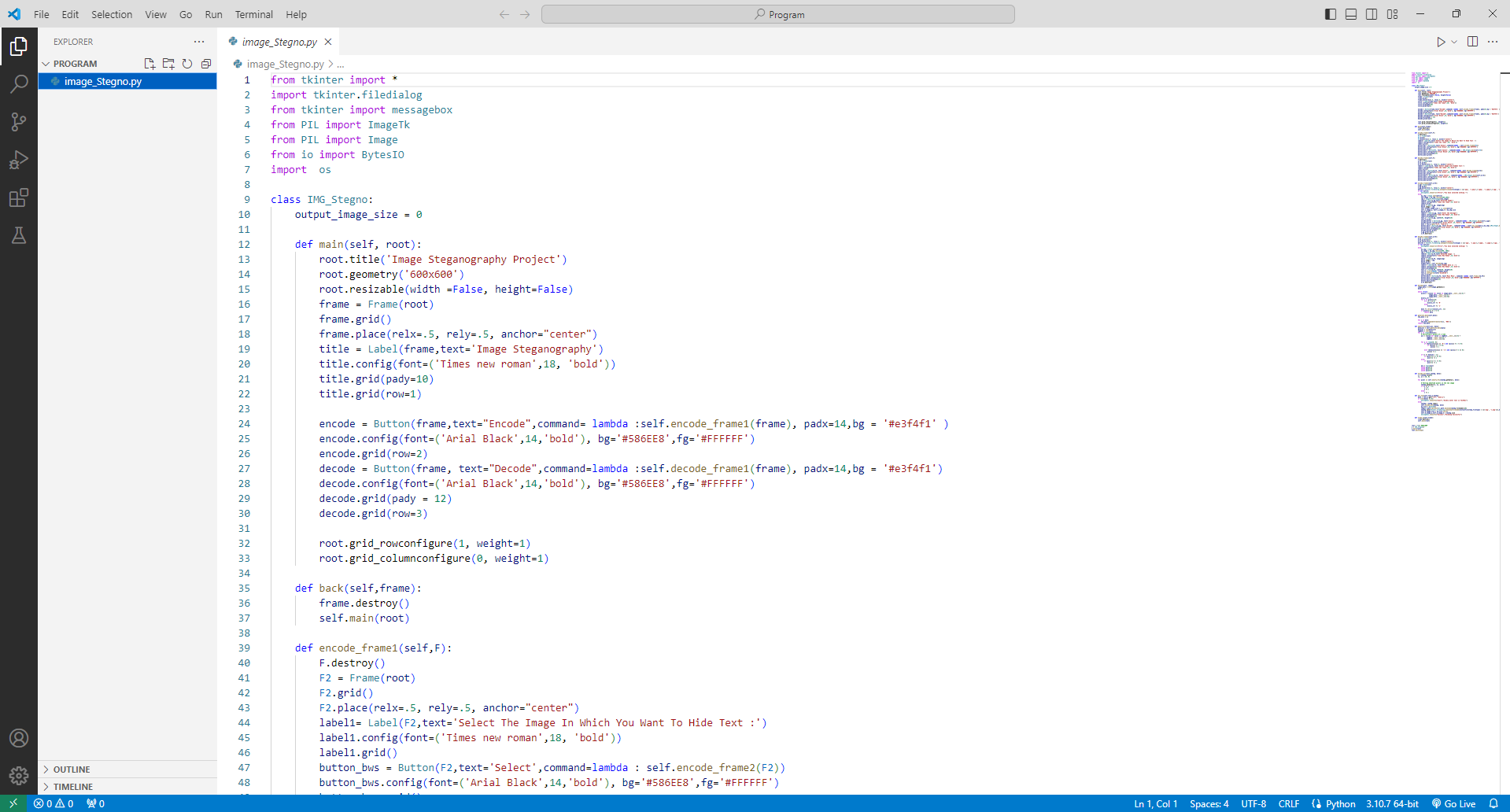
****

Fig 7.1 Open the file in the VSCODE

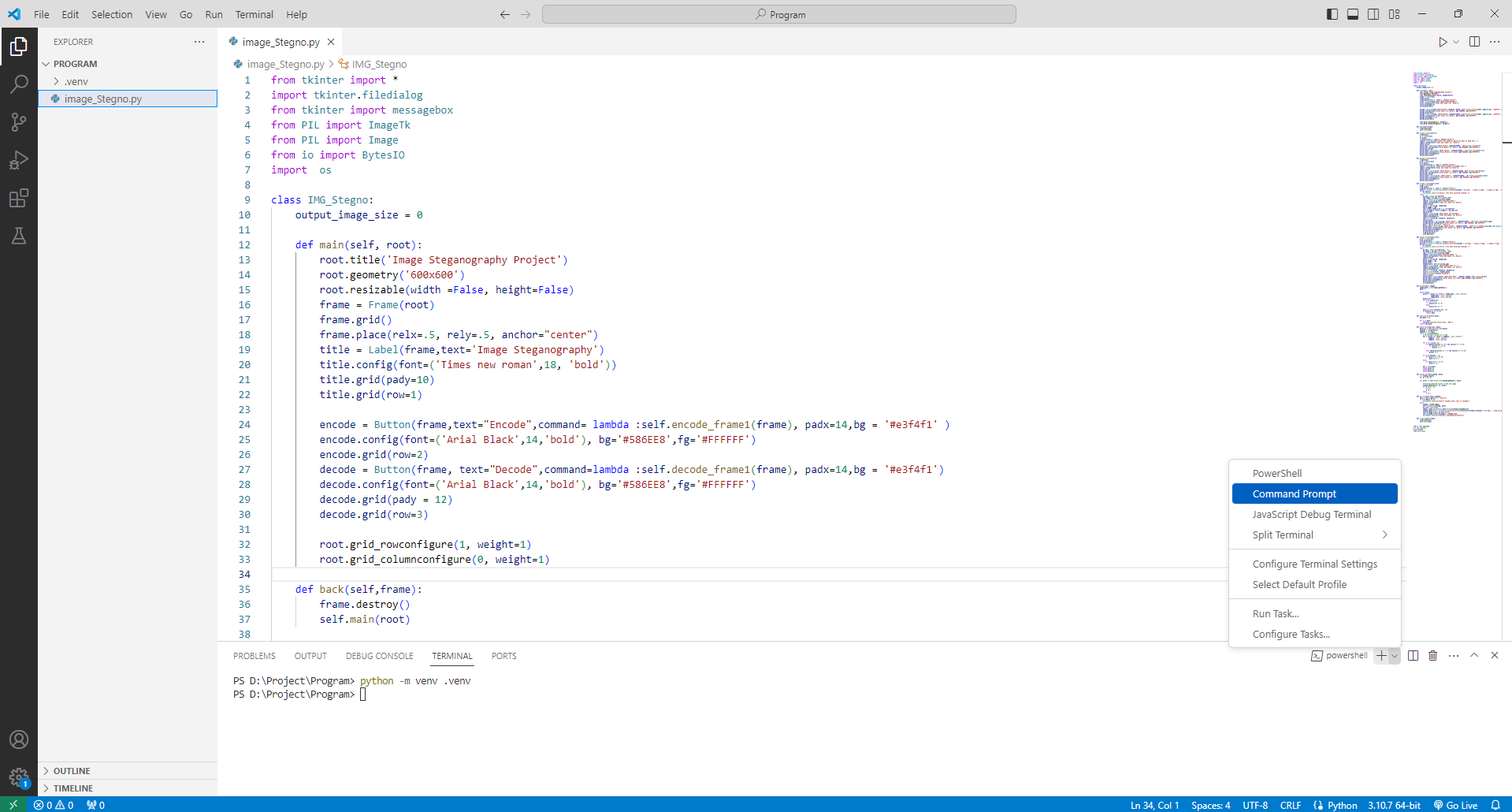
****

Fig 7.2 Create an Virtual Environment By using the syntax python -m venv .venv

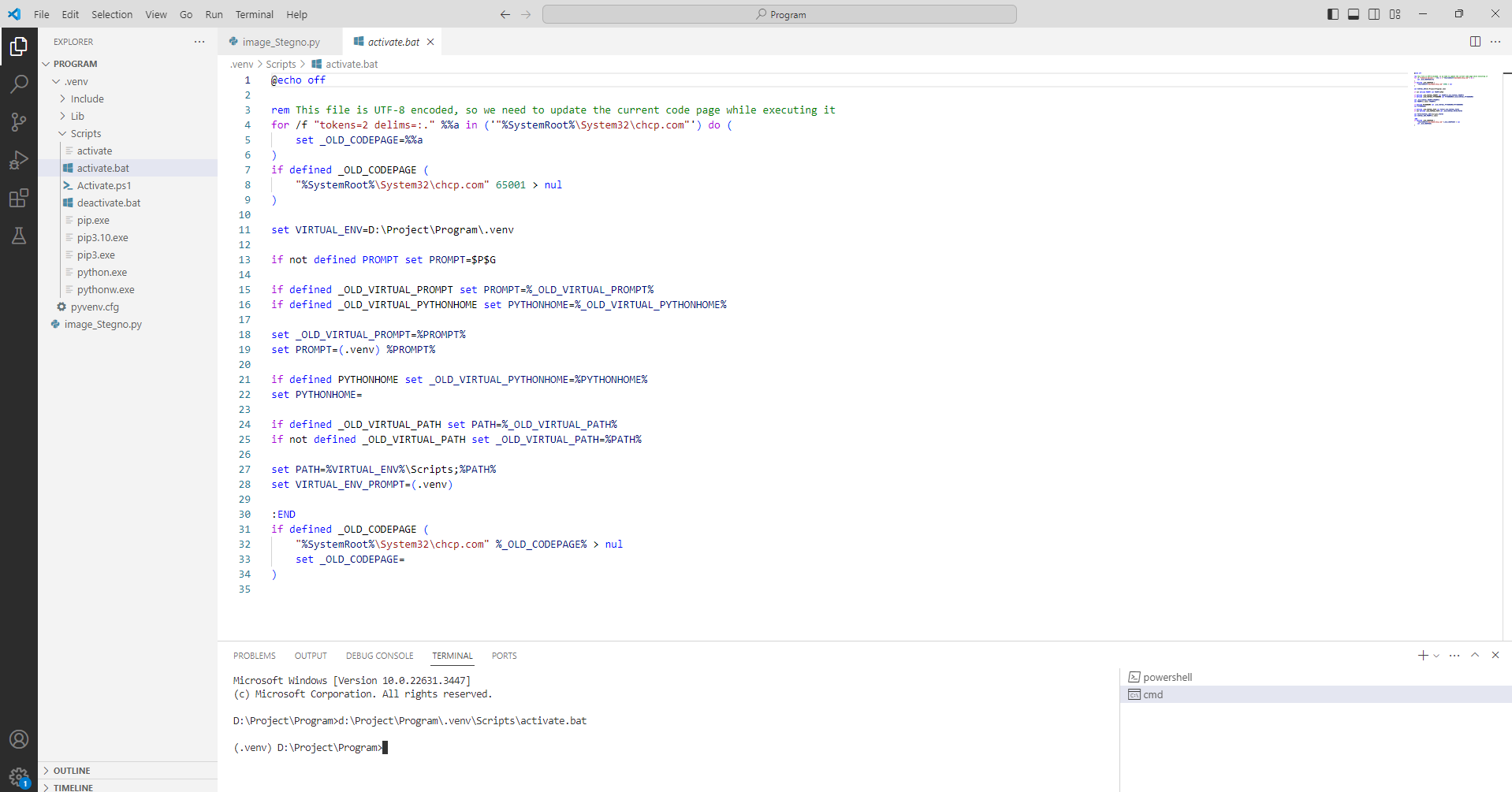
****

Fig 7.3 Open Command Prompt and Activate Virtual environment using activate.bat

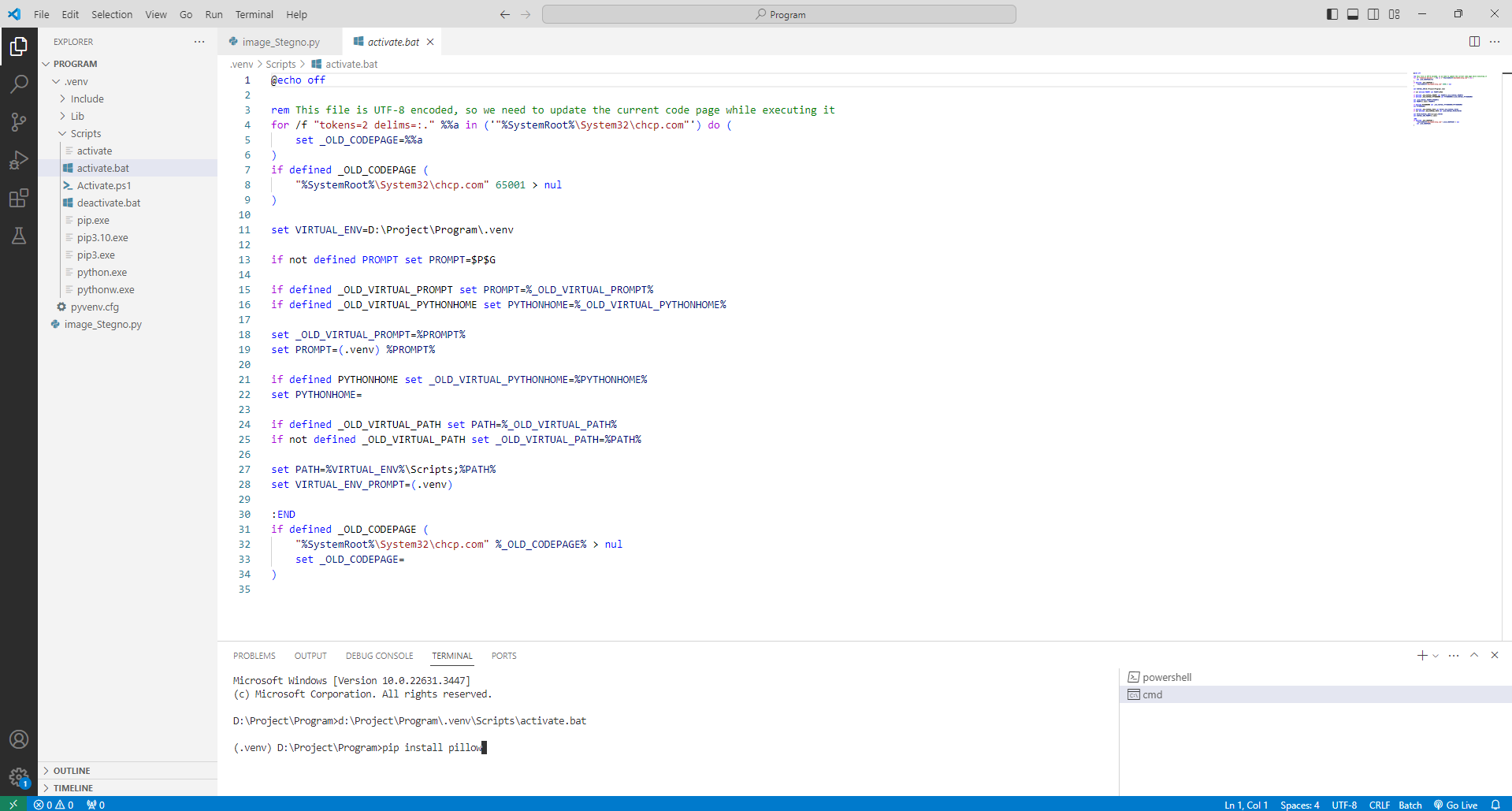
****

Fig 7.4 Install the Requirements (Pillow) using pip install pillow

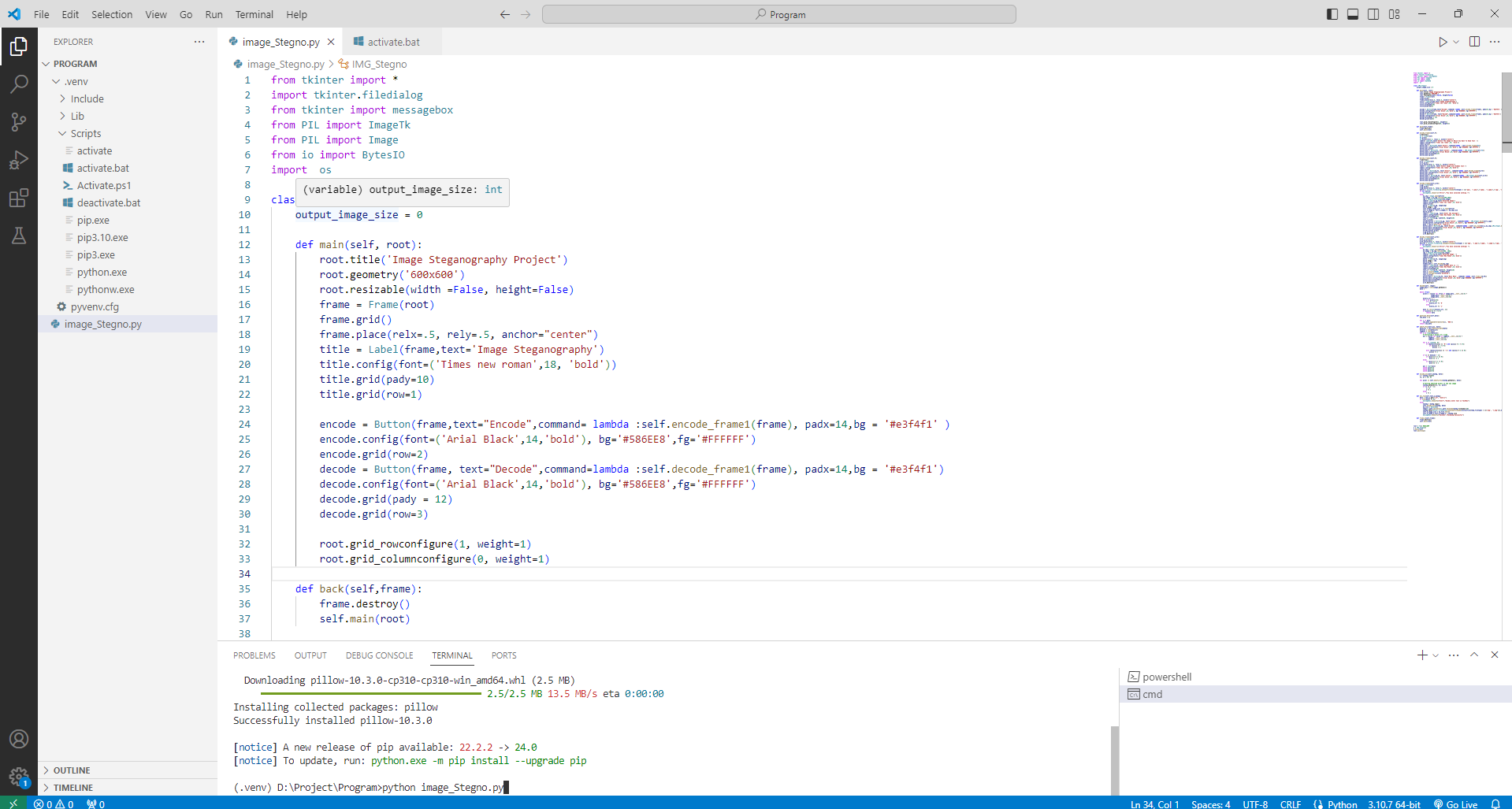
****

Fig 7.5 Run the main file python image\_Stegno.py

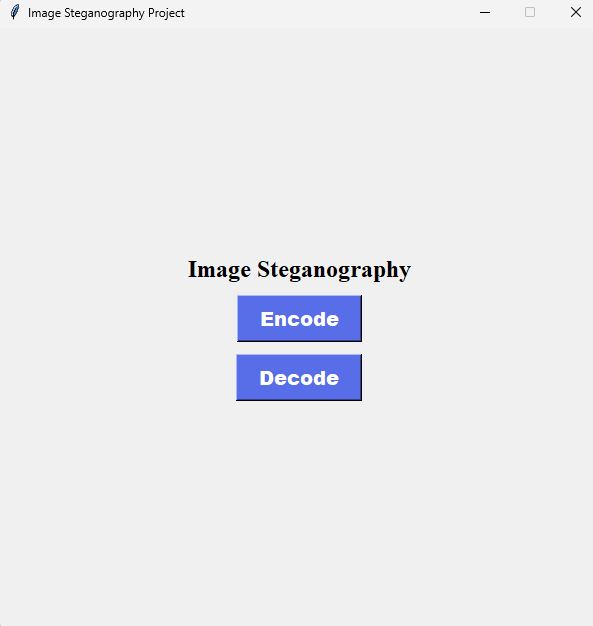
****

Fig 7.6 Interface of the Project

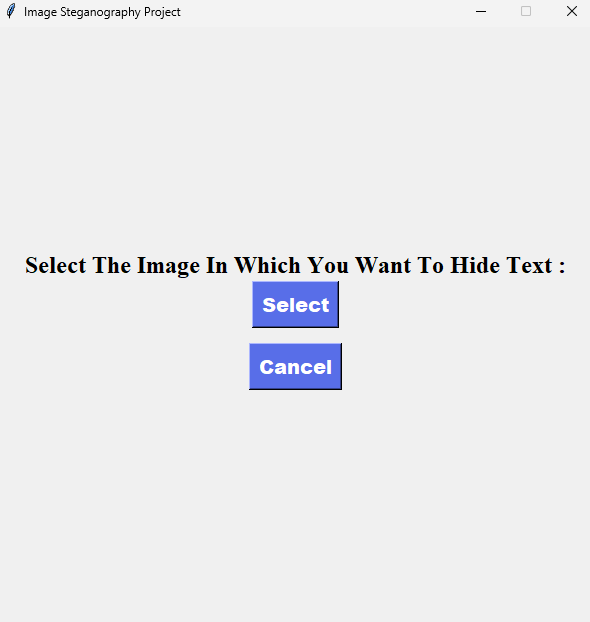
****

Fig 7.7 Frame for selecting the image to encode

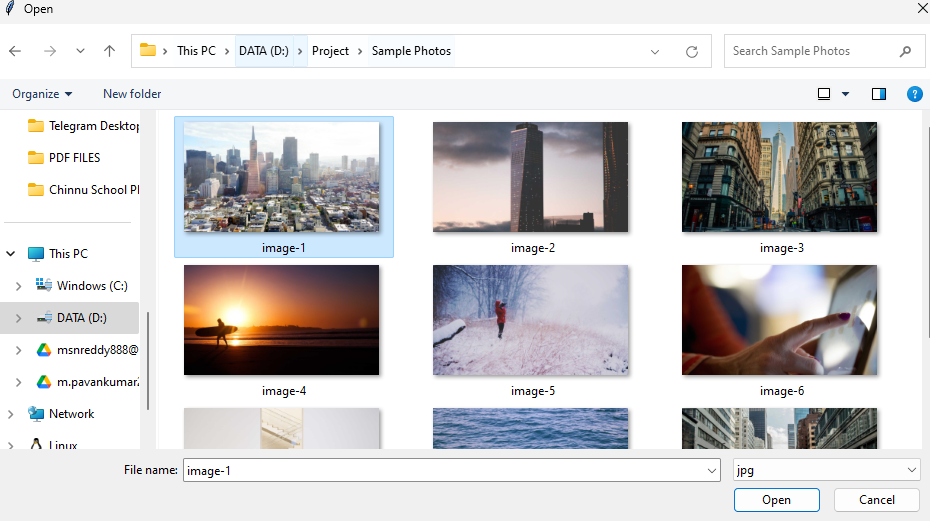
****

Fig 7.8 Select an image to Encode the message

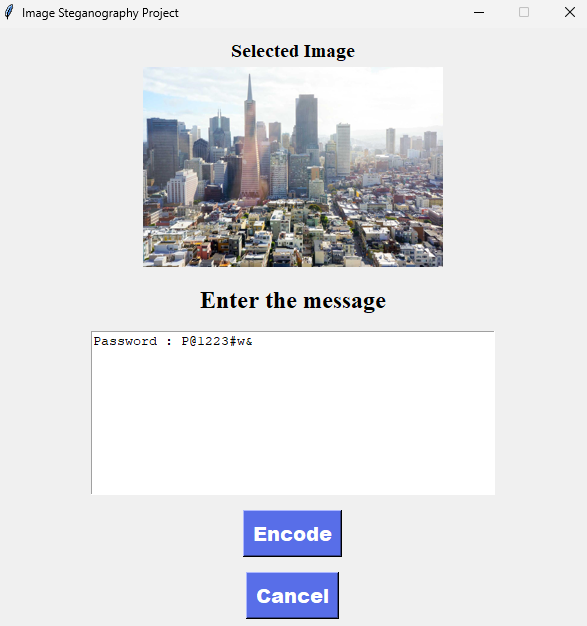
****

Fig 7.9 Enter the Message to Encode to Image

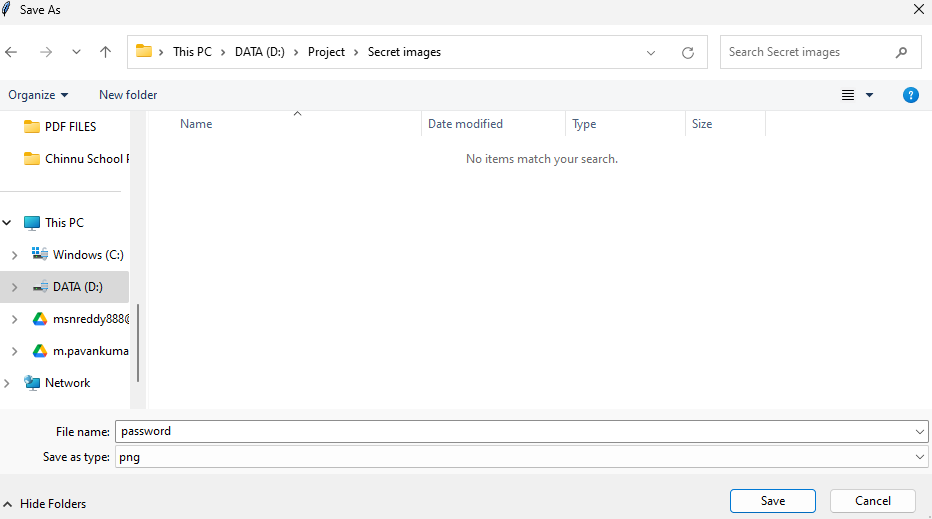
****

Fig 7.10 Choose the Location and File Name to save the Encoded Image

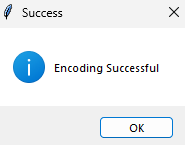
****

Fig 7.11 Success message represent the successful encoding

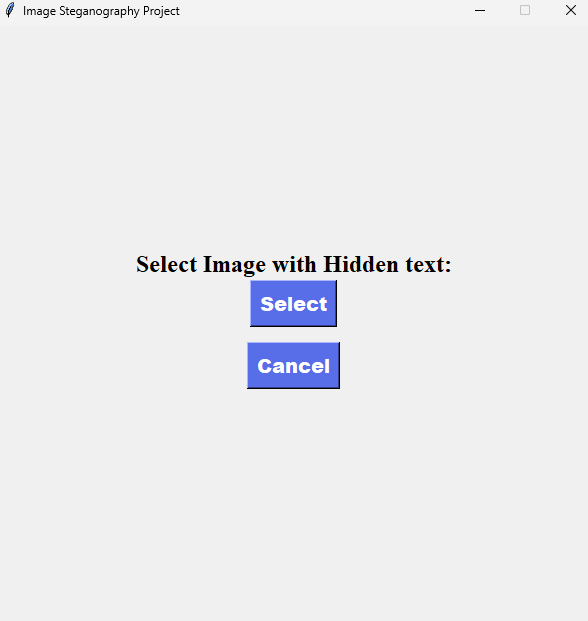
****

Fig 7.12 Frame for selecting the image to decode

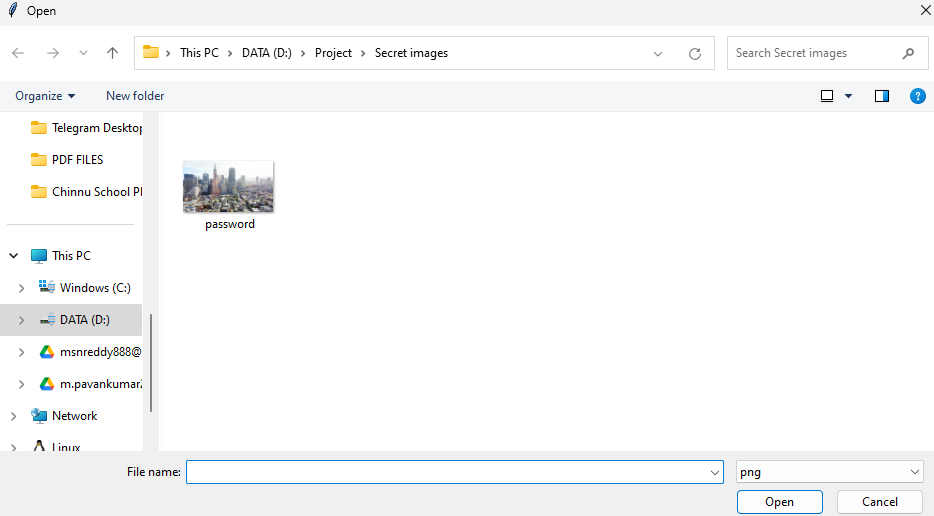
****

Fig 7.13 Select an image to Decode the message

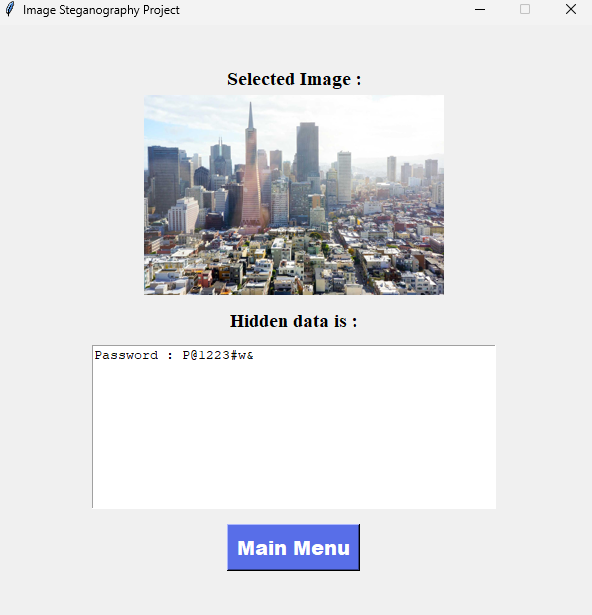
****

Fig 7.14 Message is decoded in image

# CONCLUSION & FUTURE ENHANCEMENT

In conclusion, the steganography project has successfully developed a functional system for securely hiding information within image files. Through the implementation of efficient encryption and decryption algorithms, alongside a user-friendly interface, users can seamlessly encrypt sensitive data into images and decrypt it without loss or alteration. Robust testing and validation have ensured the reliability and functionality of the system, while future enhancements aim to further enhance security measures, integrate advanced algorithms, support various image formats, improve user experience, and explore integration with cloud services. Overall, the project represents a significant advancement in secure information hiding techniques, providing users with a reliable and efficient solution for protecting sensitive data within images, with ongoing efforts to meet evolving security needs and expectations.

In future enhancements, the steganography system aims to bolster security measures by integrating additional encryption techniques, such as AES or RSA, and exploring advanced algorithms like adaptive steganography or deep learning-based methods to fortify data hiding capabilities. Support for a wider array of image formats beyond the common ones, such as TIFF or BMP, will be incorporated to cater to diverse user needs. Improvements in user experience will include a more intuitive interface with features like drag-and-drop functionality and real-time previews, alongside comprehensive documentation for seamless usage. Integration with cloud services will enable secure storage and sharing of encrypted images, while performance optimization efforts will focus on reducing encryption and decryption times through parallel processing and algorithm optimizations. These enhancements collectively aim to ensure the steganography system remains at the forefront of securely concealing information within images, meeting evolving security demands and user expectations.

# BIBLIOGRAPHY

1. P. S. Ritu Sindhu, “Information Hiding using Steganography,” Vols. Volume-9, 2020.
2. S. W, “Cryptography and Network Security, ”2007.
3. “Steganography,” https ://en.wikipedia.org /wiki/ Steganography.
4. P. Chandarana and a. P. Ahirao, “"Advanced Image Steganography.",” International Journal of Innovative Research in Information Security (IJIRIS), 2018
5. N. Singh, “"Survey Paper on Steganography.",” vol. Volume 6, pp. 68-71,2017
6. J.R.Jayapandiyan and CK.a.K.Sakthivel,Multi Image Steganography Using Distributed Lsb Algorithm and Secret Text Recovery On Stego Image Corruption,2020.
7. K.Sandhyarani & G.Prashanti, “A NewApproach for Data hiding with LSB Steganography,” 2015.
8. H. K. Masoud Nosrati “Steganography in Image Using Genetic Algorithm,” 2015.
9. Shrestha and A. Timalsina, “Color image steganography technique using daubechies discrete wavelet transform,” 2015.
10. Emam, “A Modified Image Steganography Method based on LSBTechnique, 2015.
11. S.A.El Rahman A Comprehensive ImageSteganography Tool using LSB Scheme 2015.
12. R. S. Indu Nehra, “Review Paper On Image Based,” International Journal of Scientific &Engineering Research, vol. Volume 6, no. Issue 6, 2015.
13. V.S.A Madhusudan, Two New Approaches, vol. IEEE XPLORE Digital Library, 2015.
14. R. Jain and J. Boaddh, “Advances in digital image steganography,” 2016
15. E. H. Houssein, M. A. S. Ali and A. E. Hassanien, “Essam H. Houssein; Mona A. S. Ali; Aboul Ella Hassanien,” 2016 Federated Conference on Computer Science and Information Systems (FedCSIS), 2016.
16. M. S. A. M. A. a. Sadaf Bukhari, “Enhancing security of images by Steganography and Cryptography techniques,” 2016.
17. B. A. M. &. A. M. Zeki, “PSW statistical LSB image steganalysis,” 2017.
18. R. Forgáč and R. Krakovský, “Contribution to image steganography using pulse coupled neural networks,” 16 November 2017.
19. N. M. Surse and P. Vinayakray-Jani, “A comparative study on recent image steganography techniques based on DWT.,”2017.
20. J.-J. S.-S. H. Y.-L. Wang, “An Improved Dual Image based Reversible Hiding Technique Using LSB Matching, International Journal of Network Security,” International Journal of Network Security, 2017