CHAPTER 1

INTRODUCTION

The modern world of computation revolves around the word "Data". However, why is data so fascinating? In this modern world, people are starting to realize the significance of the data in order to expand the scope of their businesses. Business owners used data to potentially predict customer trends, increase sales, and push the organization to newer heights. With the help of the rapid advancement in technology and data for nonstop revolution, it has become the chief importance to secure data. Data sharing has also increased as thousands of messages and data are being transmitted on the internet day-to-day from one place to another. Data protection is the key concern of the sender, and, significantly, we encrypt the message in a secret method that only the receiver can understand.

In this following project, we will understand the concept of Steganography along with its practical implementation using the Python programming language.

1.1 About The Project

Steganography refers to the process of hiding a secret message within a larger one in such a manner that someone cannot know the presence or contents of the hidden message. The objective of Steganography is to maintain secret communication between two parties. Unlike Cryptography, where we can conceal the contents of a secret message, Steganography conceals the fact that a message is transmitted. Even though Steganography differs from Cryptography, there are various analogies between the two. Some authors classify Steganography as a form of Cryptography as hidden communication seems like a secret message.

1.2 Scope of the Project

Information Hiding -Introduction:

Purpose:

- Hiding sensitive information vital for security purposes.
- Securing private files and documents.

Hiding information - 3methods - Cryptography, Steganography, Watermarking.

Cryptography - Data to be hidden is coded

Cryptography is the practice of securing communication from unauthorized access and modification by converting plain text into a coded or encrypted message. It involves using mathematical algorithms and methods to convert plain text into an unreadable form that can only be read by the intended recipient who has the key to decrypt the message.

Cryptography is used for a wide range of applications, including secure communication over the internet, protecting sensitive information such as passwords and credit card details, and securing electronic transactions. It is also used by governments, military organizations, and intelligence agencies for securing classified information and protecting national security.

There are various types of cryptographic algorithms, including symmetric-key encryption, asymmetric-key encryption, hashing, and digital signatures. Each of these types has its own strengths and weaknesses, and the choice of algorithm depends on the specific application and security requirements.

Watermarking - a technique with similarities to steganography

Watermarking is a technique used to embed information or a digital signal into multimedia content such as images, audio, or video, in a way that is imperceptible to the human eye or ear. The watermark serves as a unique identifier or a proof of ownership and is often used to prevent unauthorized copying or distribution of digital content.

There are two types of watermarks: visible and invisible. Visible watermarks are usually text or logos overlaid on the digital content and are intended to be seen by the viewer. They are commonly used by photographers and content creators to brand their work and discourage unauthorized use.

Invisible watermarks, on the other hand, are embedded into the digital content and cannot be seen by the viewer. They are usually created by altering the digital signal in a way that is imperceptible to the human senses, such as by adding noise or modifying the least significant bits

of the data. Invisible watermarks are commonly used for copyright protection and digital rights management (DRM) purposes.

Watermarking techniques are constantly evolving to keep up with advances in digital technology and to stay ahead of potential attacks by hackers and unauthorized users.

Steganography -hiding information by using another information as a cover.

Steganography is the practice of concealing a message or information within another object, such as an image, audio file, or video, in a way that is not detectable by the human senses or by traditional methods of analysis. Unlike cryptography, which hides the meaning of a message by scrambling its content, steganography hides the very existence of the message by embedding it in a seemingly innocent carrier file.

Steganography has been used for centuries, dating back to ancient Greece, where secret messages were written on wax tablets and covered with a layer of wax. In modern times, steganography has become more sophisticated, with digital techniques being used to hide information in image files, audio files, and other types of digital media.

There are several different techniques for steganography, including hiding information in the least significant bits of an image or audio file, embedding information in the frequency domain of an audio file, and using digital watermarking techniques to embed information in a digital signal.

Steganography is often used in combination with cryptography to provide an additional layer of security, with the steganographically hidden message serving as a backup or redundancy in case the encrypted message is intercepted or deciphered. It is used in a variety of applications, including digital forensics, military and intelligence operations, and corporate data protection.

Types:

- Text steganography
- Image steganography

Image steganography is widely used.

a) Images contain redundant information.
b) Easy to manipulate few pixels without affecting visual data observably
c) Slight changes in RGB values undetected by human eye (HVS) - Psych visual redundancy.
d) Types: Text in image; image in image

Image steganography:

Image steganography is a technique of hiding secret information within an image file in a way that is undetectable to the human eye. The process involves embedding the secret information within the image by modifying its pixels in a manner that does not significantly alter the overall appearance of the image.

One of the most common techniques used in image steganography is Least Significant Bit (LSB) insertion, in which the least significant bits of the image pixels are replaced with the secret message bits. Since the least significant bits are typically not as important for image quality as the more significant bits, this technique allows for a large amount of data to be hidden within an image without causing noticeable changes to its appearance.

Other techniques for image steganography include frequency domain techniques, where the secret message is embedded in the frequency domain of the image using the Discrete Fourier Transform (DFT) or the Discrete Cosine Transform (DCT). These techniques allow for higher data capacity and better robustness against attacks than LSB insertion, but they are more complex and require more computational resources.

Image steganography is commonly used for various applications such as copyright protection, data hiding, and secret communication. However, it can also be used maliciously to hide malware or to conduct illegal activities, which highlights the need for strong security measures to prevent unauthorized access to hidden information.

CHAPTER 2

SYSTEM OVERVIEW

In any web development, bugs are inevitable. Let it be in any kind of product bugs arise at any phase of development. In the Existing system, the bugs are not properly maintained and they are simply relied on shared lists and email to monitor the bugs. In this type of system, it becomes difficult to track, a bug if a bug is over looked then it may because tremendous errors in the next phase and can improve the cost of project whatever necessary effort spent on the bug maintenance may not be worthy. Therefore, bug history has to be maintained properly. In addition, there is no efficient search technique. One has to search the whole database for the details of particular bug, which might have occurred sometime earlier. It is both time consuming and error prone. In addition, it is very difficult to share the bug among several users, as there is no proper maintenance of the bugs. System study is classified into two types.

- Existing system
- Proposed system

2.1 Existing System

Text Steganography:

Text steganography is an approach of hiding secret text message within another text as a covering message or creating a cover message associated to the initial secret message.

Text steganography can include anything from transforming the formatting of an existing text, to changing words within a text, to producing random character sequences or utilizing context-free grammars to make readable texts.

Text steganography is held to be the trickiest because of deficiency of redundant data which is present in image, audio or a video file. The mechanism of text documents is identical with what it can identify, while in another types of documents including in picture, the structure of document is different from what it can identify.

Thus, in such documents, it can hide information by learning changes in the structure of the document without creating a famous change in the concerned output.

2.1.1 A Spectrum of Error Types

Text steganography is the practice of hiding secret information within a text message or document without arousing suspicion. Here are some potential error types that can occur during the process of text steganography:

a) **Insertion errors:** Insertion errors occur when the hidden message is not properly inserted into the cover text, resulting in either the failure to hide the message or the distortion of the original message.
b) **Deletion errors:** Deletion errors occur when the hidden message is accidentally deleted or removed from the cover text during the encoding process, resulting in loss of the secret message.
c) **Substitution errors:** Substitution errors occur when the wrong character or word is used to replace the original text in order to embed the secret message. This can result in the alteration of the original text, which may be noticeable to the receiver.
d) **Encoding errors:** Encoding errors occur when the chosen encoding method is not compatible with the cover text, resulting in the corruption of the hidden message.
e) **Decoding errors:** Decoding errors occur when the receiver attempts to extract the hidden message from the cover text but encounters errors in the decoding process. This can result in the failure to retrieve the secret message or the retrieval of a corrupted message.
f) **Detection errors:** Detection errors occur when an adversary attempts to detect the presence of a hidden message but fails to do so, allowing the secret message to be successfully transmitted. Alternatively, detection errors can occur when an adversary incorrectly identifies a message as containing a hidden message when it does not.

2.1.2 Drawbacks of the Existing System

There are several drawbacks of the existing system for text steganography:

a) **Limited Capacity:** One of the major drawbacks of existing text steganography systems is their limited capacity. It is difficult to embed large amounts of data in a text document without raising suspicion or significantly altering the original document.
b) **Vulnerability to Detection:** Existing text steganography techniques are often vulnerable to detection by adversaries who may be able to detect the presence of hidden information. This can occur through statistical analysis or by using specialized software designed to detect steganography.
c) **Increased File Size:** The process of hiding information within a text document can result in an increase in the size of the document. This can be problematic in situations where file size is a critical factor, such as when transmitting files over a limited bandwidth network.
d) **Dependence on Text Format:** Many existing text steganography techniques rely on specific text formats, such as ASCII or Unicode. This can limit their compatibility with other text formats and make them less useful in certain contexts.
e) **Lack of Security:** Existing text steganography techniques may not provide sufficient security for sensitive data. Adversaries who are able to detect the presence of hidden information may be able to extract that information and compromise the security of the data.
f) **Difficulty in Extraction:** In some cases, it can be difficult to extract the hidden information from a text document. This can occur if the encoding technique used to hide the information is complex or if the document has been altered in some way.

2.1.3 Need for New System

The need for a new system for text steganography arises from the limitations and drawbacks of the existing systems. Here are some reasons why a new system is needed:

a) **Increased Capacity:** A new system for text steganography could potentially increase the capacity for embedding larger amounts of data within text documents, without significantly altering the original document or raising suspicion.
b) **Enhanced Security:** A new system could provide better security for sensitive data by utilizing stronger encryption algorithms and techniques that are less vulnerable to detection by adversaries.
c) **Greater Compatibility:** A new system could be designed to be compatible with a wider range of text formats, making it more versatile and useful in different contexts.
d) **Improved Extraction Techniques:** A new system could incorporate improved techniques for extracting hidden information from text documents, making it easier to access the embedded data.
e) **Reduced File Size:** A new system could potentially reduce the size of text files that have been modified to include hidden information, making it easier to transmit and store these files.
f) **Resistance to Attacks:** A new system could be designed to resist attacks by adversaries who attempt to detect or modify the hidden information in order to compromise its security.

2.3 Proposed System

Image Steganography:

Image steganography is the practice of hiding secret or confidential information within an image file in such a way that it is not easily detectable by someone who is not intended to know about it. The process of steganography involves embedding the secret information within the pixels of an image in a manner that does not significantly alter the appearance of the image.

Steganography techniques can be used to hide various types of information, such as text messages, images, audio files, or even video files, within the pixels of an image. This is achieved by altering the least significant bits of the color values of the pixels in the image. Since the least significant bits are not typically used to convey important information in an image, they can be replaced with the secret data without noticeably affecting the visual quality of the image.

The hidden information can only be retrieved by someone who knows the secret key or password used to encrypt the data before it was embedded in the image. This makes steganography a useful technique for securely transmitting confidential information over public networks or storing sensitive data on unsecured storage devices.

2.3.1 Features of proposed system

The Features of image steganography are:

a) **Increased security:** Image steganography provides a high level of security because the message is hidden within the image and is not easily visible to anyone who is not intended to see it. This makes it difficult for hackers to detect and steal the message.

b) **Unobtrusive:** Since the message is hidden within an image, it does not raise any suspicion or attract attention to itself. This means that the communication can be conducted discreetly without any unwanted attention.

c) **Large data capacity:** Images can contain a large amount of data, and with steganography, this capacity can be utilized to hide a significant amount of information. This makes it ideal for transmitting large amounts of sensitive information.

d) **Robustness:** Image steganography can be made more robust by embedding the message in multiple parts of the image, making it difficult for the message to be destroyed or lost.

e) **Easy transmission:** Images can be easily transmitted over the internet or other communication channels. This makes it easy to send and receive messages without the need for specialized software or hardware.

f) **Low probability of detection:** Image steganography can be used to hide messages in a way that is difficult to detect, making it an effective tool for covert communication.

CHAPTER 3

SYSTEM ANALYSIS

System analysis for image steganography involves analysing the various components of an image steganography system to understand how it works and how effective it is.

3.1. Feasibility Study

Python is a popular programming language for implementing steganography algorithms due to its simplicity and rich library support. Here is a feasibility study for implementing image steganography in Python:

a) Research on available steganography algorithms:

There are several steganography algorithms available for hiding data within images, such as Least Significant Bit (LSB) algorithm, Spread Spectrum (SS) algorithm, and others. Researching these algorithms will help you choose the best one for your implementation.

b) Choose a suitable image format:

Images can be stored in various file formats, such as JPEG, PNG, BMP, and others. Each format has its advantages and disadvantages in terms of compression, quality, and support for steganography algorithms. Therefore, it is essential to choose a suitable format that supports steganography algorithms and does not degrade image quality.

c) Study the Python libraries for image processing:

There are several Python libraries available for image processing, such as PIL (Python Imaging Library), OpenCV, and Scikit-image. These libraries provide various functions for image manipulation, such as reading and writing images, image enhancement, and pixel manipulation.

d) Implement steganography algorithm in Python:

Once you have chosen a suitable algorithm, you can start implementing it in Python. You can use the Python libraries for image processing to read the cover image, embed the secret message, and

save the stego image. You can also use the same libraries to extract the secret message from the stego image.

e) Test and evaluate the implementation:

After implementing the steganography algorithm, you should test it thoroughly to ensure its correctness and efficiency. You can use different cover images and secret messages to evaluate the algorithm's performance in terms of capacity, security, and visual quality.

3.1.1 Technical Feasibility

A technical feasibility study for image steganography in Python would involve assessing the technical requirements and limitations of the project. Here are some key areas to consider:

a) Hardware requirements:

The hardware requirements for implementing image steganography in Python are relatively low. A typical desktop or laptop computer with a reasonable processor and memory should be sufficient for most projects. However, large image files may require more storage space.

b) Software requirements:

The software requirements for image steganography in Python include the Python programming language and various image processing libraries, such as PIL, OpenCV, and Scikit-image. These libraries provide the necessary functions for image manipulation and steganography.

c) Image format compatibility:

Image steganography algorithms may not be compatible with all image formats. Therefore, it is essential to choose a suitable image format that supports steganography algorithms and does not degrade image quality. Common formats include JPEG, PNG, and BMP.

d) Security considerations:

It is important to consider the security implications of steganography, such as the potential for data leakage or detection. Therefore, encryption of the secret message before embedding it in the image can enhance security.

e) Algorithm complexity:

Some steganography algorithms are more complex than others and may require more processing power and memory. Therefore, it is important to choose an algorithm that balances security and efficiency.

f) Performance and scalability:

The performance of image steganography in Python depends on the size of the image and the secret message, as well as the chosen algorithm. Therefore, it is important to test the implementation's performance and scalability to ensure that it can handle large files and meet the project's requirements.

g) User interface:

If the implementation involves a graphical user interface (GUI), it is important to consider the user's experience and ensure that the interface is intuitive and easy to use.

A technical feasibility study for image steganography in Python requires consideration of hardware and software requirements, image format compatibility, security considerations, algorithm complexity, performance, scalability, and user interface. By carefully assessing these factors, you can ensure that your implementation is feasible and effective.

3.1.2 Economic Feasibility

An economic feasibility study for image steganography in Python involves assessing the economic viability of the project. Here are some key areas to consider:

a) **Cost of development:**

The cost of developing image steganography in Python includes the cost of hardware, software, and personnel. This cost can vary depending on the complexity of the project and the skills of the developers.

b) **Potential revenue:**

The potential revenue from image steganography in Python can come from selling the software, licensing the software, or offering a paid service. The revenue potential depends on the target market, competition, and pricing strategy.

c) **Market demand:**

The market demand for image steganography in Python depends on the target audience and the competition. It is important to conduct market research to determine the potential market size and competition.

d) **Return on investment (ROI):**

The ROI of image steganography in Python depends on the revenue generated and the cost of development. The ROI can be calculated by dividing the revenue generated by the cost of development.

e) **Maintenance and support:**

The cost of maintenance and support should also be considered in the economic feasibility study. This includes the cost of bug fixes, updates, and technical support.

f) **Legal and regulatory requirements:**

It is important to consider the legal and regulatory requirements for image steganography in Python, such as intellectual property rights and data protection regulations.

g) **Risks and uncertainties:**

The economic feasibility study should also consider the risks and uncertainties associated with the project, such as market volatility, competition, and technical issues.

An economic feasibility study for image steganography in Python requires consideration of the cost of development, potential revenue, market demand, ROI, maintenance and support, legal and regulatory requirements, and risks and uncertainties. By carefully assessing these factors, you can determine the economic viability of the project and make informed decisions about its development and marketing.

3.1.3 Operational Feasibility

Operational feasibility is an important aspect to consider when developing a project such as image steganography in Python. In order to assess the operational feasibility of this project, we need to consider several factors:

a) **Technical expertise:** The development of image steganography in Python requires a certain level of technical expertise in both Python programming and image processing. It is important to assess whether the development team has the required skills to successfully complete the project.

b) **Hardware and software requirements:** The project may require specific hardware and software requirements, such as powerful processors, high-resolution displays, and specialized image processing libraries. It is important to assess whether the hardware and software requirements are readily available and can be easily accessed.

c) **Cost:** The development of image steganography in Python may require a significant amount of resources, such as developer salaries, hardware and software costs, and testing and deployment expenses. It is important to assess whether the project is financially feasible and whether the benefits of the project outweigh the costs.

d) **User adoption:** Image steganography may not be widely known or used by users, so it is important to assess whether the project will be adopted by users and whether there is a market demand for such a tool.

e) **Legal and ethical considerations:** The use of steganography may raise legal and ethical concerns, such as privacy violations, intellectual property theft, and national security risks. It is important to assess whether the project complies with legal and ethical standards.

An operational feasibility study for image steganography in Python would need to consider the technical expertise of the development team, hardware and software requirements, cost, user adoption, and legal and ethical considerations. By carefully assessing these factors, the

development team can determine whether the project is feasible and whether it should be pursued further.

CHAPTER 4

SYSTEM PLANNING

System planning is an important phase of software development that involves defining project goals, requirements, resources, and timelines.

4.1. System Configuration

System configuration refers to the process of setting up and configuring the software and hardware components of a system to work together to meet the system requirements.

4.1.1 Hardware Requirement

CPU TYPE	2.0 GHz OR MORE
RAM SIZE	4GB OR MORE
HARD DISK CAPACITY	50.5GB OR MORE
KEYBOARD TYPE	INTERNET KEYBOARD
MONITOR TYPE	15 INCH COLOUR MONITOR
CD -DRIVE TYPE	52xmax

4.1.2 Software Requirement

OPERATING SYSTEM	WINDOWS 8.1 OR LATER
SIMULATION TOOL	• PYTHON • VISUAL STUDIO CODE
DOCUMENTATION	MS-OFFICE

4.2. Software Description

Software is a term used to describe a collection of programs, data, and instructions that tell a computer how to perform specific tasks. It can refer to any type of program that runs on a computer, including applications, operating systems, utilities, and games.

4.2.1 Microsoft windows

Microsoft Windows OS is a popular family of operating systems developed by Microsoft Corporation. It is one of the most widely used operating systems in the world and is designed to provide a graphical user interface (GUI) for easy navigation.

a) Microsoft Windows is a popular operating system (OS) developed by Microsoft Corporation. It was first released in 1985 and has since undergone many upgrades and versions. Windows is a graphical user interface (GUI) based OS that allows users to interact with their computer through icons, menus, and windows.

b) Windows has become one of the most widely used operating systems worldwide due to its user-friendly interface and its compatibility with a wide range of software and hardware. Windows is used in personal computers, laptops, and servers, and is available in many different versions, including Windows 10, Windows 8, Windows 7, and Windows Vista.

c) Some of the key features of Windows include its taskbar, which allows quick access to frequently used applications and files, and its file explorer, which provides easy access to files and folders on the computer. Windows also includes built-in security features such as Windows Defender, which protects against malware and viruses, and BitLocker, which provides encryption for sensitive data.

d) Windows has become a staple in the computer industry and continues to be a popular choice for users due to its user-friendly interface, compatibility with software and hardware, and built-in security features.

e) Microsoft Windows is a popular operating system (OS) developed by Microsoft Corporation. Windows OS has evolved over the years, with different versions released since its initial launch in 1985. Some of the most popular Windows

versions include Windows 3.0, Windows 95, Windows 98, Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1, and Windows 10.

f) Windows OS is designed to be user-friendly and provide a graphical user interface (GUI) for easy navigation. It supports a wide range of software applications and hardware devices, making it suitable for both personal and professional use.

g) Windows OS is widely used in businesses, homes, and schools around the world. It supports multiple languages and is available in different editions, such as Home, Professional, Enterprise, and Education.

h) Windows OS is also known for its security features, including built-in firewalls, antivirus software, and regular updates to fix security vulnerabilities. Microsoft provides support and updates for Windows OS to ensure that users have the latest features and security patches.

i) Microsoft Windows OS has been a dominant player in the operating system market for many years, and its popularity is due to its user-friendly interface, compatibility with a wide range of software and hardware, and robust security features.

4.2.2 Python

Python is a high-level, interpreted programming language that is used for a wide variety of purposes, including web development, data analysis, machine learning, scientific computing, and more. Python was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages in the world.

a) Python is a popular high-level programming language that was created in the late 1980s by Guido van Rossum. It is an interpreted language, which means that the code is executed line by line rather than compiled into an executable file like other languages. Python is known for its simplicity, ease of use, and readability, making it a great choice for beginners and experienced programmers alike.

b) Python has a wide range of applications, including web development, data analysis, machine learning, artificial intelligence, scientific computing, and more. Its standard library includes modules for many common tasks, such as working with files, connecting to databases, and handling network protocols. There are also many third-party libraries available for Python, which can be easily installed using package managers like pip.
c) One of the key features of Python is its emphasis on code readability. Python code is written using indentation, rather than using curly braces or other delimiters, which makes it easy to read and understand. This also makes it easier to collaborate on projects with other developers.
d) Python supports multiple programming paradigms, including object-oriented, functional, and procedural programming. It also has a dynamic type system, which means that variables can change types during runtime. This flexibility allows for quick development and testing of code.
e) Python has a large and active community of developers who contribute to its development and create a wide range of libraries and tools. It is available on many platforms, including Windows, macOS, and Linux, and can be used with a variety of integrated development environments (IDEs) or text editors.
f) Overall, Python is a versatile and powerful programming language that is widely used in many industries and applications. Its simplicity and ease of use make it a great choice for beginners, while its flexibility and extensive libraries make it a popular choice for experienced programmers as well.

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language.

For example, x=10 Here, x can be anything such as String, int, etc.

Python is an interpreted, object-oriented programming language similar to PERL, that has gained popularity because of its clear syntax and readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted

in a number of operating systems, including UNIX-based systems, Mac OS, MS-DOS, OS/2, and various versions of Microsoft Windows 98. Python was created by Guido van Rossum, a former resident of the Netherlands, whose favourite comedy group at the time was Monty Python's Flying Circus. 11

The source code is freely available and open for modification and reuse. Python has a significant number of users.

Features in Python

There are many features in Python, some of which are discussed below

a) Easy to code
b) Free and Open Source
c) Object-Oriented Language
d) GUI Programming Support
e) High-Level Language
f) Extensible feature
g) Python is Portable language
h) Python is Integrated language
i) Interpreted Language

4.2.3 Microsoft Visual Studio Code:

Visual Studio Code (VS Code) is a free and open-source code editor developed by Microsoft for Windows, Linux, and MacOS. It provides developers with a range of features such as debugging, syntax highlighting, intelligent code completion, code refactoring, and version control. VS Code is designed to be highly extensible through its vast library of extensions and customizations, making it a popular choice among developers of various programming languages and frameworks. Additionally, it also supports a wide range of file types and programming languages, making it a versatile tool for any developer.

a) Microsoft Visual Studio Code is a free and open-source code editor developed by Microsoft. It is widely used by developers for a variety of programming tasks, including web development, mobile app development, and cloud computing.
b) Visual Studio Code is known for its rich set of features, including built-in support for debugging, syntax highlighting, code completion, and Git integration. It also supports many programming languages, including Python, Java, JavaScript, Typescript, and many others, making it a versatile choice for developers.
c) One of the key benefits of Visual Studio Code is its ability to be customized with extensions. There are many extensions available for Visual Studio Code, which can add features and functionality to the editor, such as support for new programming languages or frameworks.
d) Visual Studio Code also has a strong community of developers who contribute to its development and create new extensions and tools. It is available on many platforms, including Windows, macOS, and Linux, and can be easily installed using package managers or downloaded from the official website.
e) Microsoft Visual Studio Code is a powerful and versatile code editor that is widely used by developers around the world. Its rich set of features, customization options, and community support make it a popular choice for many programming tasks.
f) VS Code supports a wide range of programming languages and file types and includes features such as syntax highlighting, code completion, debugging, and version control integration. It also has a built-in terminal for executing commands and running scripts.
g) One of the key features of VS Code is its extensibility. It has a large library of extensions that can be installed to add additional functionality, such as support for specific programming languages, code snippets, and more. The extensions are created by the community and can be easily installed through the integrated extension marketplace.

h) VS Code also supports collaboration and team development through features such as Live Share, which allows multiple developers to work on the same codebase in real-time. It also integrates with popular version control systems such as Git and GitHub.

i) VS Code is a powerful and flexible code editor that is popular among developers due to its extensive features and customization options. It is a great choice for a wide range of programming projects, from simple scripts to large-scale applications.

PYTHON IN VISUAL STUDIO CODE:

Visual Studio Code (VS Code) has excellent support for Python, making it a popular choice among Python developers. Here are some of the ways you can use Python in VS Code:

a) Create a Python Project: You can create a Python project directly in VS Code by using the "Python: Create New Project" command. This command will create a new Python environment and a project structure with a sample script and a virtual environment.

b) Install Python Extensions: VS Code has a range of extensions for Python development, including the Python extension, which adds features like IntelliSense, debugging, and code formatting to VS Code.

c) Run Python Code: You can run Python code in VS Code using the integrated terminal or the Python Interactive window. The Python Interactive window allows you to run code and see the results immediately, making it a great tool for prototyping and testing code.

d) Debug Python Code: VS Code has a built-in debugger for Python that allows you to step through your code and see how it executes. You can set breakpoints, inspect variables, and navigate through your code to find and fix errors.

e) Manage Dependencies: VS Code makes it easy to manage Python dependencies using the integrated terminal and the Python package manager (pip). You can install, update, and remove packages directly within VS Code.
f) Use Jupyter Notebooks: VS Code also supports Jupyter notebooks, which are interactive documents that combine code, text, and visualizations. You can create, edit, and run Jupyter notebooks directly within VS Code.

VS Code provides a comprehensive and powerful environment for Python development, with many features that help make Python development faster, easier, and more efficient.

CHAPTER 5

SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It is a critical step in software engineering and involves determining the structure, behaviour, and functionality of a software system. System design considers various aspects of the system, such as scalability, performance, reliability, security, maintainability, and usability. It involves identifying the major components of the system, their interactions, and the data flows between them. The goal of system design is to create a system that meets the requirements of the stakeholders and can be implemented efficiently and effectively. System design is an iterative process and involves feedback from stakeholders, architects, developers, and other members of the development team.

5.1 System Design

Steganography is the practice of hiding information within other information. In image steganography, the information to be hidden is embedded within an image file. In this design, we will use Python to implement a simple image steganography system.

Here is a system design for implementing image steganography in Python:

a) Load the cover image and the secret message.
b) Convert the secret message into binary format.
c) Check if the cover image is large enough to hide the secret message. If not, raise an error or prompt the user to select a larger cover image.
d) Modify the least significant bits (LSBs) of the cover image pixels to hide the secret message bits. One common approach is to replace the LSBs of the red, green, and blue channels of each pixel with the bits of the secret message. This can be done using the following steps:

I. Loop through each pixel in the cover image.
II. Get the pixel RGB values.
III. Convert the RGB values to binary format.
IV. Modify the LSBs of the RGB values to hide the secret message bits.
V. Convert the modified RGB values back to decimal format.

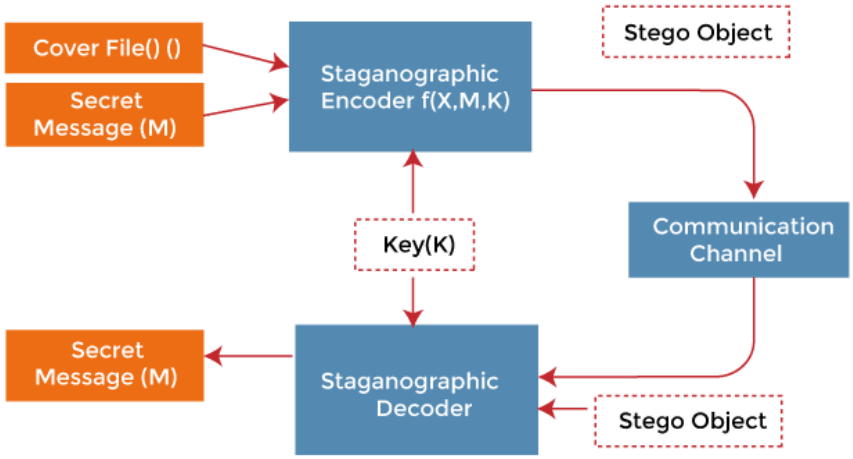
f. Update the pixel RGB values in the cover image.

I. Save the steganography image.
II. To extract the secret message from the steganography image, follow these steps:

- Load the steganography image.
- Loop through each pixel in the steganography image.
- Get the pixel RGB values.
- Convert the RGB values to binary format.
- Extract the LSBs of the RGB values and concatenate them into a binary string.
- Once you have the binary string, convert it back into ASCII characters using the chr() function.

III. Display or save the extracted secret message.

5.2 Data Flow Diagram



5.3 Layout Design

Here is a high-level overview of the layout design:

a) Importing Required Libraries.
b) Loading the Image.
c) Encoding the Message.
d) Decoding the Message.
e) Testing the System.

Step 1: Importing Required Libraries

We will use the Python Imaging Library (PIL) to work with images. To install PIL, we can use the command **'pip install pillow'** in the command prompt.

"from PIL import Image"

Step 2: Loading the Image

We can load the image using the **'Image.open()'** method provided by PIL.

"image = Image.open("image.png")"

Step 3: Encoding the Message

We can encode the message into the image by manipulating the RGB values of the pixels. The easiest way to encode the message is by changing the least significant bit (LSB) of the RGB values of the pixels.

```
def encode_image(image, message):
    # Convert the message into binary format
    binary_message = ''.join([format(ord(i), "08b") for i in message])
    # Get the pixel map of the image
    pixel_map = image.load()
    # Get the size of the image
    width, height = image.size
    # Counter variable to keep track of the index of the binary message
    index = 0
    # Loop through all the pixels of the image
    for i in range(width):
        for j in range(height):
            # Get the RGB values of the pixel
            r, g, b = pixel_map[i, j]
            # Change the LSB of each of the RGB values to the corresponding bit of the binary message
            if index < len(binary_message):
                pixel_map[i, j] = (r - r % 2 + int(binary_message[index]), g - g % 2 + int(binary_message[index + 1]), b - b % 2 + int(binary_message[index + 2]))
```

```
                index += 3
                # If the message has been completely encoded, return the image
                if index >= len(binary_message):
                    return image
        return None
```

Step 4: Decoding the Message

We can decode the message from the image by extracting the LSB of the RGB values of the pixels.

```
    def decode_image(image):
        # Get the pixel map of the image
        pixel_map = image.load()
        # Get the size of the image
        width, height = image.size
        # Variable to store the binary message
        binary_message = ""
        # Loop through all the pixels of the image
        for i in range(width):
            for j in range(height):
                # Get the LSB of each of the RGB values of the pixel and append it to the binary message
                r, g, b = pixel_map[i, j]
                binary_message += str(r % 2)
```

```
            binary_message += str(g % 2)
            binary_message += str(b % 2)
        # Convert the binary message into ASCII characters and return the message
        message = ""
        for i in range(0, len(binary_message), 8):
            message += chr(int(binary_message[i:i+8], 2))
        return message
```

Step 5: Testing the system

To test the system, we can encode a message into an image and then decode the message from the encoded image.

```
# Encode the message into the image
image = Image.open("image.png")
message = "Hello, World!"
encoded_image = encode_image(image, message)

# Save the encoded image
encoded_image.save("encoded_image.png")

# Decode the message from the encoded image
encoded_image = Image.open("encoded_image.png")
    decoded_message = decode_image
```

5.4 Database Design

Designing a database for image steganography in Python will depend on the specific requirements and features of the application. However, a basic design could include the following tables:

a) **Users table:** This table will store information about the users of the application such as username, password, email, etc.
b) **Images table:** This table will store the images uploaded by the users along with their metadata such as image name, format, size, and path.
c) **Steganography table:** This table will store the details about the steganographic images such asthe original image ID, steganographic image path, steganographic algorithm used, and key.
d) **Log table:** This table will store the logs of all the user activities such as image upload, steganography performed, and user authentication.
e) **Statistics table**: This table will store the statistics of the application such as the number of images uploaded, the number of steganographic images created, and the number of users.

Here is an example of how you can implement the above database design using Python and SQLite:

```
import sqlite3

# Connect to the database

conn = sqlite3.connect('steganography.db')

# Create a cursor object

c = conn.cursor()

# Create Users table

c.execute('''CREATE TABLE IF NOT EXISTS Users (

        user_id INTEGER PRIMARY KEY,
```

```
            username TEXT NOT NULL,
            password TEXT NOT NULL,
            email TEXT NOT NULL
        )''')
# Create Images table
c.execute('''CREATE TABLE IF NOT EXISTS Images (
            image_id INTEGER PRIMARY KEY,
            image_name TEXT NOT NULL,
            image_format TEXT NOT NULL,
            image_size INTEGER NOT NULL,
            image_path TEXT NOT NULL
        )''')
# Create Steganography table
c.execute('''CREATE TABLE IF NOT EXISTS Steganography (
            stego_id INTEGER PRIMARY KEY,
            original_image_id INTEGER NOT NULL,
            stego_path TEXT NOT NULL,
            stego_algorithm TEXT NOT NULL,
            stego_key TEXT NOT NULL,
            FOREIGN KEY(original_image_id) REFERENCES Images(image_id)
        )''')
# Create Log table
c.execute('''CREATE TABLE IF NOT EXISTS Log (
```

```
        log_id INTEGER PRIMARY KEY,
        user_id INTEGER NOT NULL,
        activity TEXT NOT NULL,
        activity_time TEXT NOT NULL,
        FOREIGN KEY(user_id) REFERENCES Users(user_id)
    )''')
# Create Statistics table
c.execute('''CREATE TABLE IF NOT EXISTS Statistics (
        stat_id INTEGER PRIMARY KEY,
        images_uploaded INTEGER NOT NULL,
        steganographic_images_created INTEGER NOT NULL,
        users INTEGER NOT NULL
    )''')
# Commit changes and close the connection
conn.commit()
conn.close()
```

Note that this is just a basic implementation of the database design, and you may need to modify it according to your specific needs

5.5 Table Design

To design a table for image steganography in Python, you need to consider the various components involved in the process of steganography. Here is an example of a table design that you can use for image steganography:

a) **Images Table:** This table will store the original images and their metadata, including the image ID, image name, format, size, and path.

```
CREATE TABLE Images (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    format TEXT NOT NULL,
    size INTEGER NOT NULL,
    path TEXT NOT NULL
);
```

b) **Users Table:** This table will store the information about the users of the application, including the user ID, username, password, and email.

```
CREATE TABLE Users (
    id INTEGER PRIMARY KEY,
    username TEXT NOT NULL,
    password TEXT NOT NULL,
    email TEXT NOT NULL
);
```

c) **Steganography Table:** This table will store the information about the steganographic images created from the original images, including the stego ID, original image ID, stego path, stego algorithm used, and key.

```
CREATE TABLE Steganography (
    id INTEGER PRIMARY KEY,
    original_image_id INTEGER NOT NULL,
    stego_path TEXT NOT NULL,
```

stego_algorithm TEXT NOT NULL,

stego_key TEXT NOT NULL,

FOREIGN KEY (original_image_id) REFERENCES Images(id)

);

d) Log Table: This table will store the logs of all the user activities, including the log ID, user ID, activity performed, and activity timestamp.

CREATE TABLE Log (

id INTEGER PRIMARY KEY,

user_id INTEGER NOT NULL,

activity TEXT NOT NULL,

timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,

FOREIGN KEY (user_id) REFERENCES Users(id)

);

e) Statistics Table: This table will store the statistics of the application, including the statistics ID, number of images uploaded, number of steganographic images created, and number of users.

CREATE TABLE Statistics (

id INTEGER PRIMARY KEY,

images_uploaded INTEGER NOT NULL,

stego_images_created INTEGER NOT NULL,

users INTEGER NOT NULL

);

These are just example tables and you may need to add more tables or modify these tables based on your specific requirements.

5.6 Report Generation

To generate a report for image steganography in Python using Visual Studio Code, you can use the same libraries and approach as described in the previous answer. Here are the steps:

a) Open Visual Studio Code and create a new Python file.
b) Import the required libraries:

```
import pandas as pd
import matplotlib.pyplot as plt
from reportlab.pdfgen import canvas
from reportlab.lib.units import inch
```

c) Retrieve the data from your database and store it in a pandas dataframe:

```
# Retrieve data from database
df = pd.read_sql_query('SELECT * FROM Steganography', conn)
# Print the dataframe to check if the data has been retrieved correctly
print(df.head())
```

d) You can then use the pandas dataframe to create visualizations using matplotlib. For example, to create a bar chart showing the number of steganographic images created by each algorithm:

```
# Group the data by algorithm and count the number of steganographic images created by each algorithm
algorithm_counts = df.groupby('stego_algorithm')['id'].count()

# Create a bar chart

plt.bar(algorithm_counts.index, algorithm_counts.values)
```

 # Set the chart title and labels

plt.title('Steganographic Images Created by Algorithm')
plt.xlabel('Algorithm')
plt.ylabel('Number of Images')
 # Show the chart
plt.show()

e) Finally, you can use ReportLab to generate a PDF report containing the visualizations and other information:
 # Create a new PDF file
pdf = canvas.Canvas('steganography_report.pdf')

 # Set the title of the report

pdf.setTitle('Steganography Report')

 # Add the bar chart to the report

plt.savefig('steganography_chart.png')
pdf.drawImage('steganography_chart.png', inch, inch, width=6*inch, height=4*inch)

 # Add some text to the report

pdf.drawString(inch, 7*inch, 'Steganography Report')
pdf.drawString(inch, 6.5*inch, 'Number of Steganographic Images Created by Algorithm')

 # Save the PDF file

pdf.save()

f) Save the Python file with an appropriate name and run it in Visual Studio Code.
g) The report will be generated in the same directory as the Python file. You can open the PDF file to view the report.

This is just an example of how to generate a report for image steganography in Python using Visual Studio Code.

CHAPTER 6

SYSTEM TESTING & IMPLEMENTATION

System testing and implementation are important phases in the software development lifecycle. In this answer, we will discuss what these phases are and their importance in the development process.

6.1 System Testing

System testing for image steganography in Python involves testing the software system as a whole to ensure that it meets the specified requirements and works as expected. The system testing for image steganography can include the following steps:

a) **Test planning:** Define the test strategy, test scope, test objectives, and test criteria for image steganography.

- **Define the test objectives:** Determine what the main goals of your tests are. For example, you might want to test if your implementation of steganography in Python can successfully hide a message within an image without significantly altering the appearance of the image.
- **Identify test scenarios:** Based on the test objectives, identify different scenarios that you want to test. For example, you might want to test the performance of the steganography algorithm when encoding a large message or when hiding a message in a high-resolution image.
- **Choose test cases:** Once you have identified the scenarios, create specific test cases for each scenario. Each test case should have a specific input and expected output. For example, a test case might involve hiding a specific message in a specific image and then verifying that the message can be successfully extracted.

- **Define test data:** Create a set of test data that you will use for testing. This should include a range of different images and messages that you will use to test the algorithm.
- **Plan test execution:** Decide how you will execute the tests. You might choose to write test scripts or use a testing framework such as Pytest. You should also decide on the criteria for passing or failing each test.
- **Record and report results:** Once you have executed the tests, record the results and report them to the appropriate stakeholders. This might involve creating a test report or presenting the results in a meeting.
- **Perform regression testing:** As you make changes to the steganography algorithm, it is important to perform regression testing to ensure that existing functionality has not been broken. Repeat steps 2-6 as necessary.

b) **Test case development:** Develop test cases and test scenarios based on the requirements and test plan for image steganography. These test cases should cover various scenarios like testing the different steganography algorithms, testing the encoding and decoding process, testing the capacity of the steganographic image, etc.

- **Test case for hiding and extracting a message:** This test case involves hiding a message within an image and then extracting it to ensure that the message is not corrupted. The test should cover various scenarios such as hiding a message in different types of images (e.g., JPEG, PNG), with different message lengths, and using different steganography techniques.
- **Test case for detecting hidden messages:** This test case involves detecting whether an image contains a hidden message or not. The test should cover various scenarios such as detecting messages hidden using different steganography techniques and with different message lengths.
- **Test case for data loss:** This test case involves testing for data loss during the steganography process. The test should cover different scenarios such as hiding a message in a low-quality image, encoding a long message in a small image, and using different steganography techniques.
- **Test case for image quality:** This test case involves verifying that the steganography process does not degrade the quality of the image. The test

should cover different scenarios such as hiding a message in different types of images (e.g., high-resolution images, low-resolution images), using different steganography techniques, and extracting the image to ensure that there is no visible distortion or degradation.

- **Test case for performance:** This test case involves measuring the performance of the steganography algorithm. The test should cover different scenarios such as encoding and decoding large messages, using different steganography techniques, and processing images of different sizes and types.
- **Test case for security:** This test case involves testing the security of the steganography algorithm against various attacks such as brute force, statistical analysis, and frequency analysis. The test should cover different scenarios such as hiding a message in different types of images and using different steganography techniques.

c) **Test execution:** Execute the test cases and record the test results. For example, test that the steganographic image is correctly created and can be decoded to retrieve the original image.

- **Import the necessary libraries:** To perform image steganography in Python, you will need to import certain libraries such as OpenCV, NumPy, and Matplotlib. These libraries can be installed using pip or Anaconda.
- **Load the image:** Load the image that you want to hide the message in using OpenCV.
- **Convert the image to an array:** Convert the image to a NumPy array so that you can manipulate its pixels.
- **Convert the message to binary:** Convert the message that you want to hide into binary format so that it can be embedded in the image.
- **Embed the message:** Embed the message into the least significant bit (LSB) of the pixels in the image array. This can be done by iterating through the array and modifying the LSB of each pixel based on the corresponding bit in the message.
- **Save the image:** Save the modified image as a new file.

- **Extract the message:** To extract the message from the image, load the modified image and extract the LSB of each pixel. Reconstruct the binary message by concatenating the LSBs, and then convert the binary message back to its original format.
- **Verify the message:** Verify that the extracted message is the same as the original message that was embedded in the image.
- **Perform additional testing:** Perform additional testing to ensure that the steganography algorithm works as expected. This can include testing the algorithm with different types of images, different message lengths, and different steganography techniques.
- **Optimize the code:** Optimize the code for performance and security by using more advanced steganography techniques or encryption algorithms.

d) **Defect tracking:** Report and track defects found during testing. If any issues are discovered during testing, they should be logged and tracked until they are resolved. Identify and document the defect: Identify and document the defect that has been found in the steganography algorithm. This can include issues such as data loss, image quality degradation, message corruption, and performance issues.

- **Assign a severity level:** Assign a severity level to the defect based on its impact on the steganography algorithm and its importance to the overall system. The severity level can range from low to critical.
- **Assign a priority level:** Assign a priority level to the defect based on its urgency and importance to the project. The priority level can range from low to critical.
- **Create a defect report:** Create a defect report that includes a description of the defect, the steps to reproduce the issue, the expected behaviour, and the actual behaviour. The report should also include the severity level, priority level, and any additional information that may be relevant to the defect.
- **Assign the defect to a developer:** Assign the defect to a developer who is responsible for fixing the issue. The developer should be provided with the defect report and any additional information that may be needed to reproduce the issue.

- **Fix the defect:** The developer should fix the defect and test the fix to ensure that the issue has been resolved.
- **Verify the fix:** The tester should verify the fix by retesting the steganography algorithm and ensuring that the defect has been resolved.
- **Close the defect:** Once the defect has been fixed and verified, it can be closed. The defect report should be updated with the resolution details and any other relevant information.
- **Monitor the system:** Continue to monitor the steganography algorithm for any new defects that may arise and repeat the defect tracking process as needed.

e) **Test reporting:** Document the test results and provide a summary of the test coverage, test results, and defect status. The test report should also include the details of the testing environment used and any issues found during testing.

- **Test Case Execution:** Execute the test cases that were developed for image steganography in Python. These test cases should cover all the critical scenarios and edge cases for the steganography algorithm.
- **Record Test Results:** Record the results of the executed test cases in a test management tool or a test report template. The results should include the status of each test case, the test execution date, and any defects that were identified during the testing process.
- **Identify Defects:** Identify any defects that were found during the testing process. These defects should be documented in the test report along with the severity and priority level.
- **Analyze Test Results:** Analyze the test results to identify any patterns or trends that could indicate issues with the steganography algorithm. This analysis can help in identifying areas that require additional testing or optimization.
- **Summarize Test Results:** Summarize the test results in a concise manner to provide a high-level overview of the steganography algorithm's performance. This summary should include the total number of test cases executed, the pass/fail percentage, and any major issues that were identified.
- **Provide Recommendations**: Provide recommendations based on the test results and analysis. These recommendations can include suggestions for

optimizing the steganography algorithm or improving the testing process.

- **Share the Test Report:** Share the test report with the project stakeholders, including the development team, project manager, and client. The report should be presented in a clear and concise manner to ensure that stakeholders can easily understand the results.
- **Follow-up Actions:** Based on the feedback from the stakeholders, follow-up actions may be required, such as retesting the steganography algorithm, addressing any defects that were identified, or optimizing the algorithm.

During system testing, it is important to verify that the steganographic image is properly created and that the hidden information is not detectable to the naked eye. Additionally, it is important to ensure that the steganographic image does not degrade the quality of the original image significantly.

It is also important to test the steganography algorithm's robustness against various attacks like steganalysis, which is the process of detecting the presence of hidden data in a steganographic image.

6.2 System Implementation

Image steganography is the technique of hiding secret information within an image without altering the visual appearance of the image. In order to implement image steganography, you can follow these steps:

a) **Choose an image:** Select the image in which you want to hide the secret information. The image should be in a common format like PNG, JPEG, or BMP.

b) **Choose the secret message:** Decide what message you want to hide in the image. This can be a text message, an image, or any other file.

c) **Convert the message to binary:** Convert the secret message to binary format. This is necessary as we will be hiding the message in the image by altering the binary values of the pixels in the image.

d) **Determine the number of bits to alter:** Decide how many bits you want to alter in each pixel of the image. This will determine the maximum amount of data that can be hidden in the image.

e) **Embed the message:** Alter the binary values of the pixels in the image to embed the secret message. To do this, you will need to determine which pixels to modify and how to modify them. One common approach is to modify the least significant bits of each pixel in the image.
f) **Save the modified image:** Save the modified image with the hidden message types of implementation of image steganography.

Types of Implementation

a) **Pixel Value Differencing (PVD) method:** In this method, the difference between the adjacent pixels in the image is calculated, and the secret message is embedded in these differences. This method can hide more data than the LSB method, but it is more complex to implement.
b) **Spread Spectrum (SS) method:** This method uses a mathematical algorithm to spread the secret message over the entire image. This makes it difficult to detect the presence of the hidden message. However, this method requires a large amount of computational resources.
c) **Transform Domain Techniques:** Transform domain techniques involve transforming the image into a different domain (such as the frequency domain), and then embedding the secret message in the transformed image. Examples of transform domain techniques include Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), and Singular Value Decomposition (SVD).
d) **Phase Encoding Techniques:** Phase encoding techniques use the phase information of an image to hide the secret message. This method is more secure than other methods as it can resist various attacks like steganalysis, but it is computationally expensive.

CONCLUSION

In conclusion, image steganography in Python is a technique that enables users to hide secret messages in images without altering the image's appearance. This technique has various applications, including secure communication and data encryption.

Testing is an essential aspect of image steganography in Python, and it involves various activities, such as test planning, test case development, test execution, defect tracking, and test reporting. The testing process helps to ensure that the steganography algorithm functions as intended, and any defects are identified, resolved, and documented.

Furthermore, Python provides various libraries such as OpenCV and Pillow, which enable developers to implement image steganography easily. Developers should ensure that the steganography algorithm is optimized to avoid performance issues that may affect its functionality.

Image steganography in Python is an effective technique for secure communication, and developers must implement robust testing practices to ensure its effectiveness and reliability.

Image steganography is a useful technique for hiding secret information within an image while maintaining the image's appearance. In Python, there are several libraries available, such as Pillow and Stegano, that can be used to implement steganography algorithms.

To ensure the accuracy and reliability of the steganography algorithm, it is essential to develop a comprehensive test plan that covers all critical scenarios and edge cases. The test plan should include test case development, test execution, defect tracking, and test reporting.

By following a structured approach to testing, any defects in the steganography algorithm can be identified, tracked, and resolved, ensuring that the algorithm meets the desired requirements and performs as expected. Additionally, test reporting provides valuable insights into the algorithm's performance and can help identify areas for optimization.

Overall, image steganography in Python is a powerful technique that can be used for various applications, such as secure data transmission and digital watermarking. Proper testing and optimization can help ensure its effectiveness and reliability.

FUTURE ENHANCEMENTS

Steganography is the practice of hiding information within an image or another form of media. There are several ways to enhance image steganography in the future:

a) **Advanced encryption techniques:** The encryption techniques used in steganography can be enhanced by using advanced algorithms like elliptic curve cryptography (ECC) or homomorphic encryption. These techniques will make it much harder for hackers to break the encryption and extract the hidden information.

b) **Artificial intelligence and machine learning**: AI and ML can be used to improve steganography by automatically detecting and removing hidden information from images. This can be useful in identifying any malicious intent or suspicious activity.

c) **Improved steganographic methods:** New steganographic methods can be developed that are more secure and can store larger amounts of data. These methods can be designed to be resistant to known steganalysis techniques and can be tailored for specific applications.

d) **Integration with block chain:** Steganography can be integrated with block chain technology to create a more secure and decentralized way of storing and transmitting sensitive data. This can ensure that the data remains secure and tamper-proof.

e) **Hardware-based steganography:** Hardware-based steganography can be developed that is more secure and reliable. This can include developing specialized chips or devices that are specifically designed for steganography, making it more difficult for hackers to breach the security of the system.

BIBLIOGRAPHY

a) "Image Steganography Using Python" by Sushant Singh - This article on the GeeksforGeeks website provides an introduction to steganography and step-by-step instructions on how to implement image steganography using Python.
b) "Image Steganography with Python" by Divyansh Prakash - This tutorial on Towards Data Science provides a detailed guide on how to use Python to hide text within an image using LSB (Least Significant Bit) steganography.
c) "Steganography: Hiding Images within Images using Python" by Harshita Sharma - This article on the Analytics Vidhya website explains how to use Python and the OpenCV library to perform image steganography using LSB steganography.
d) "Python Image Steganography Library" by Sudarshan Sharma - This library on GitHub provides a simple interface for hiding and extracting messages within images using various steganography techniques, including LSB, DCT, and F5.
e) "Image Steganography using Python and Pillow" by Ismail Uddin - This tutorial on Medium explains how to use the Pillow library in Python to perform image steganography using LSB steganography.
f) "Steganography: A Review of Techniques and Applications" by Anand Singh Jalal, published in the Journal of Emerging Technologies and Innovative Research: This paper provides a comprehensive review of steganography techniques, including image steganography, and discusses their applications and limitations. Link: https://www.jetir.org/papers/JETIR2003J21.pdf
g) "Steganography: A Digital Image Hiding Technique using LSB Technique with Python" by Anju Rani and Sumit Chhabra, published in the International Journal of Computer Applications: This paper provides a detailed description of how to implement LSB steganography using Python and the Pillow module, and includes examples of how to hide and retrieve messages and images.
h) "Steganography in Python with Examples" by Tadas Baltrusaitis, published on his personal blog: This tutorial provides an overview of steganography, including various

techniques and tools, and includes examples of how to implement LSB and DCT-based steganography using Python and the Pillow module. Linkhttps://tadas

Websites referred

a) Cryptography and Steganography: https://www.geeksforgeeks.org/cryptography-and-steganography-introduction-and-difference/
b) Steganography Tools: https://www.jjtc.com/Steganography/
c) Digital Invisible Ink Toolkit: https://diit.sourceforge.io/
d) OpenStego: https://www.openstego.com/
e) Steghide: http://steghide.sourceforge.net/
f) Steganography Online: https://stylesuxx.github.io/steganography/
g) Image Steganography: https://www.educba.com/image-steganography/

APPENDICES

10.1 Sample Source Codes

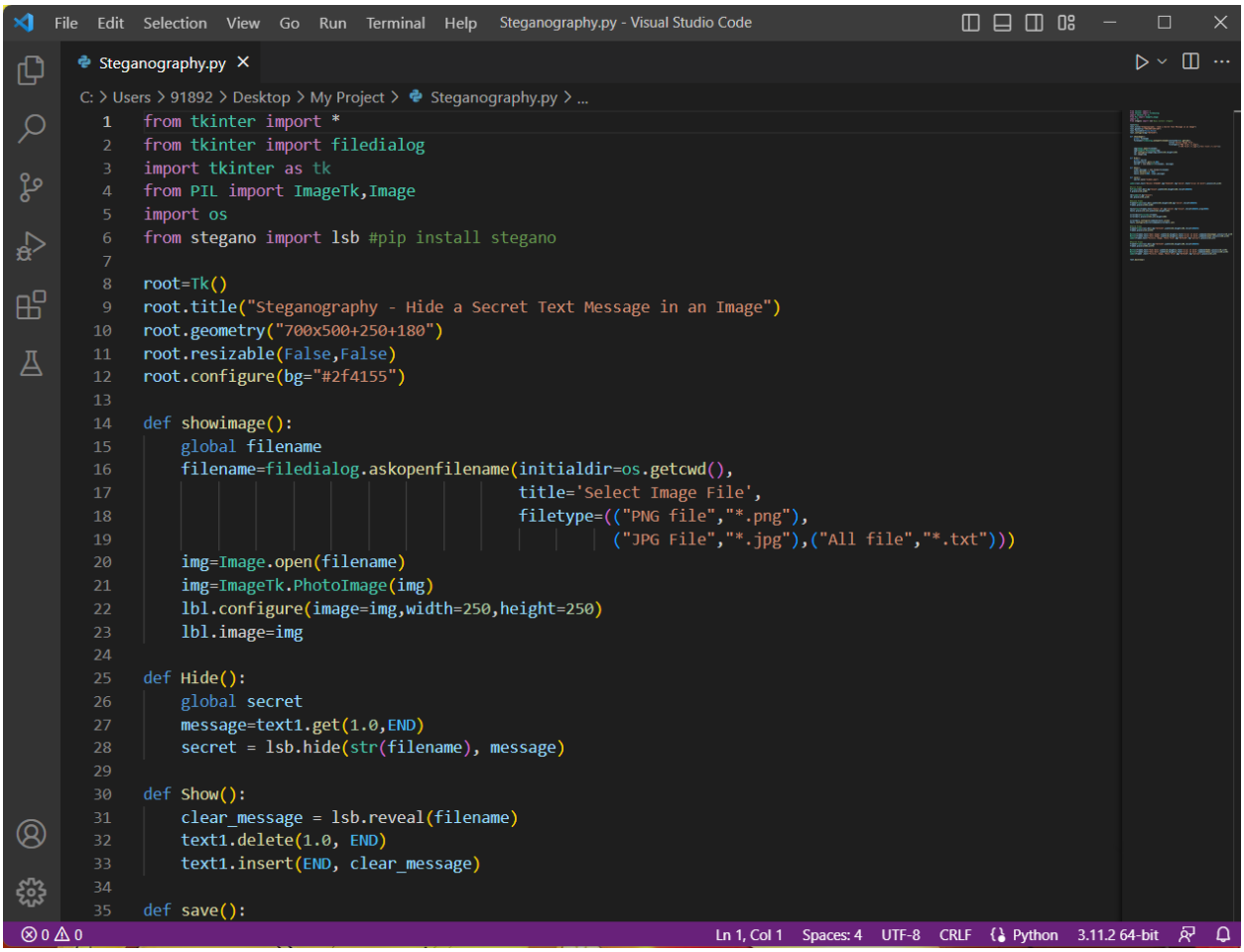
CODING:

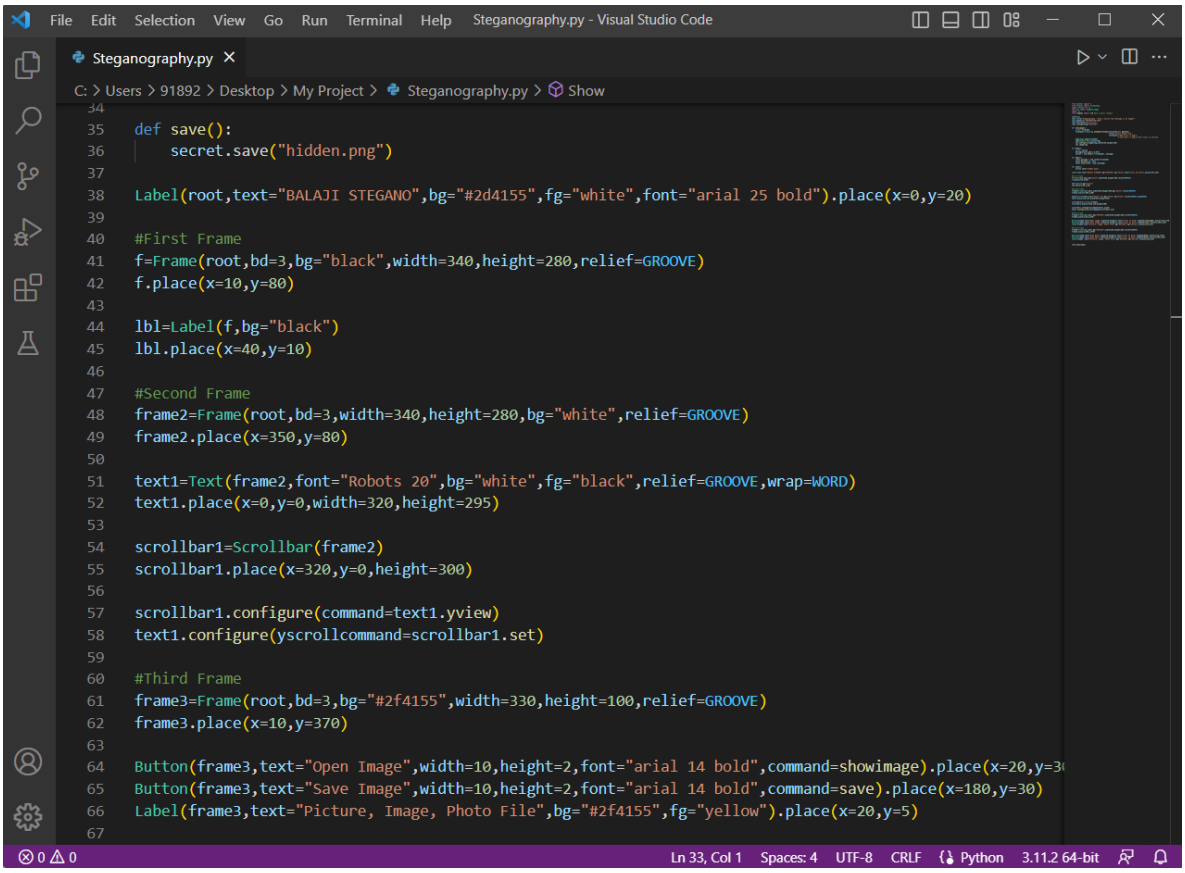
```
from tkinter import *
from tkinter import filedialog
import tkinter as tk
from PIL import ImageTk,Image
import os
from stegano import lsb #pip install stegano
root=Tk()
root.title("Steganography - Hide a Secret Text Message in an Image")
roo
t.geometry("700x500+250+180")
root.resizable(False,False)
root.configure(bg="#2f4155")
def showimage():
global filename
filename=filedialog.askopenfilename(initialdir=os.getcwd(),
title='Select Image File',
filetype=(("PNG file","*.png"), ("JPG File","*.jpg"),("All file","*.txt")))
img=Image.open(filename)
img=ImageTk.PhotoImage(img)
lbl.configure(image=img,width=250,height=250)
lbl.image=img
def Hide():
global secret
message=text1.get(1.0,END)
secret = lsb.hide(str(filename), message)

def Show():
clear_message = lsb.reveal(filename)
```

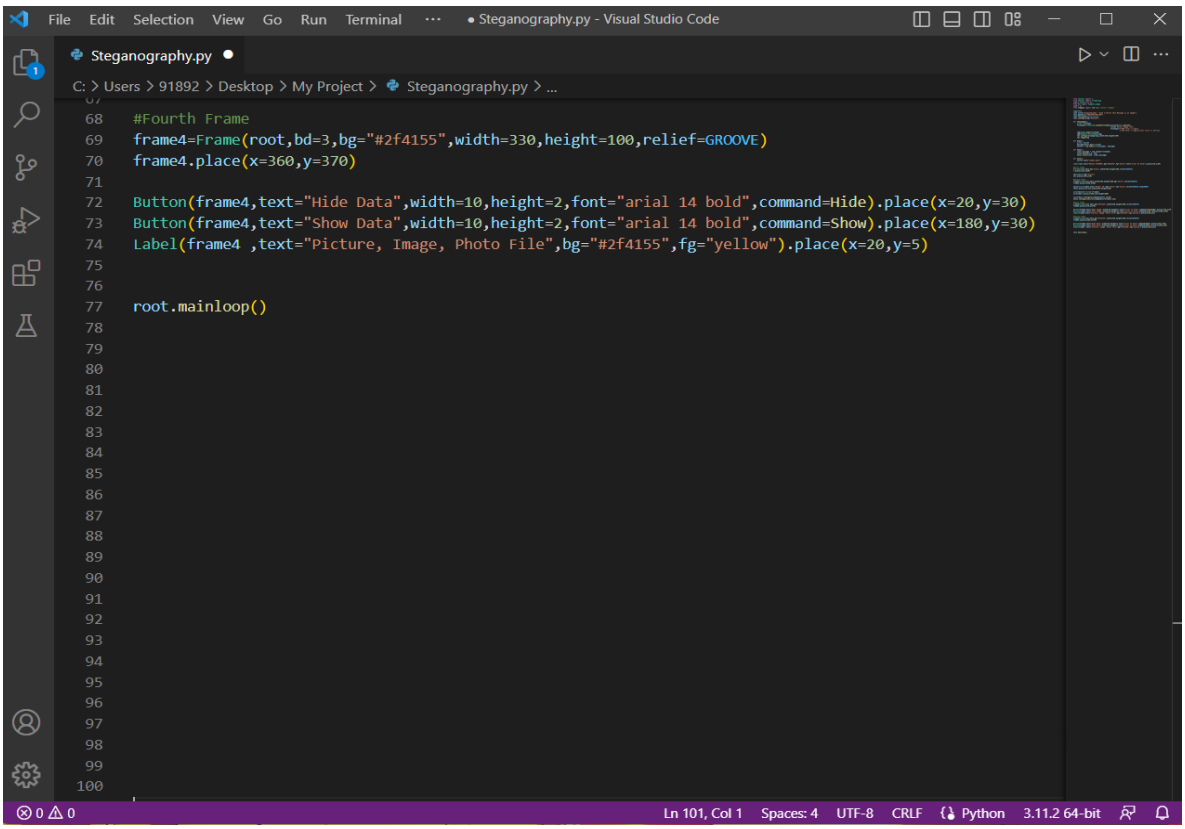
```
    text1.delete(1.0, END)
    text1.insert(END, clear_message)
    def save():
    secret.save("hidden.png")
    Label(root,text="KA.GV STEGANO",bg="#2d4155",fg="white",font="arial 25
bold").place(x=10,y=20)

    #First Frame
    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=40,y=10)

    #Second Frame
    frame2=Frame(root,bd=3,width=340,height=280,bg="white",relief=GROOVE)
    frame2.place(x=350,y=80)
    text1=Text(frame2,font="Robots 20",bg="white",fg="black",relief=GROOVE,wrap=WORD)
    text1.place(x=0,y=0,width=320,height=295)
    scrollbar1=Scrollbar(frame2)
    scrollbar1.place(x=320,y=0,height=300)
    scrollbar1.configure(command=text1.yview)
    text1.configure(yscrollcommand=scrollbar1.set)

    #Third Frame

    frame3=Frame(root,bd=3,bg="#2f4155",width=330,height=100,relief=GROOVE)
    frame3.place(x=10,y=370)
    Button(frame3,text="Open Image",width=10,height=2,font="arial 14
bold",command=showimage).place(x=20,y=30)
    Button(frame3,text="Save Image",width=10,height=2,font="arial 14
bold",command=save).place(x=180,y=30)
    Label(frame3,text="Picture, Image, Photo File",bg="#2f4155",fg="yellow").place(x=20,y=5)
```

```
    #Fourth Frame
    frame4=Frame(root,bd=3,bg="#2f4155",width=330,height=100,relief=GROOVE)
    frame4.place(x=360,y=370)
    Button(frame4,text="Hide Data",width=10,height=2,font="arial 14
bold",command=Hide).place(x=20,y=30)
    Button(frame4,text="Show Data",width=10,height=2,font="arial 14
bold",command=Show).place(x=180,y=30)
    Label(frame4 ,text="Picture, Image, Photo File",bg="#2f4155",fg="yellow").place(x=20,y=5)
    root.mainloop()
```

10.2 Screen Shots




File Edit Selection View Go Run Terminal Help Steganography.py - Visual Studio Code
Steganography.py
C: > Users > 91892 > Desktop > My Project > Steganography.py > Show
def save():
secret.save("hidden.png")
Label(root,text="BALAJI STEGANO",bg="#2d4155",fg="white",font="arial 25 bold").place(x=0,y=20)
#First Frame
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=40,y=10)
#Second Frame
frame2=Frame(root,bd=3,width=340,height=280,bg="white",relief=GROOVE)
frame2.place(x=350,y=80)
text1=Text(frame2,font="Robots 20",bg="white",fg="black",relief=GROOVE,wrap=WORD)
text1.place(x=0,y=0,width=320,height=295)
scrollbar1=Scrollbar(frame2)
scrollbar1.place(x=320,y=0,height=300)
scrollbar1.configure(command=text1.yview)
text1.configure(yscrollcommand=scrollbar1.set)
#Third Frame
frame3=Frame(root,bd=3,bg="#2f4155",width=330,height=100,relief=GROOVE)
frame3.place(x=10,y=370)
Button(frame3,text="Open Image",width=10,height=2,font="arial 14 bold",command=showimage).place(x=20,y=3
Button(frame3,text="Save Image",width=10,height=2,font="arial 14 bold",command=save).place(x=180,y=30)
Label(frame3,text="Picture, Image, Photo File",bg="#2f4155",fg="yellow").place(x=20,y=5)
Ln 33, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.2 64-bit


File Edit Selection View Go Run Terminal • Steganography.py - Visual Studio Code
Steganography.py
C: > Users > 91892 > Desktop > My Project > Steganography.py > ...
#Fourth Frame
frame4=Frame(root,bd=3,bg="#2f4155",width=330,height=100,relief=GROOVE)
frame4.place(x=360,y=370)
Button(frame4,text="Hide Data",width=10,height=2,font="arial 14 bold",command=Hide).place(x=20,y=30)
Button(frame4,text="Show Data",width=10,height=2,font="arial 14 bold",command=Show).place(x=180,y=30)
Label(frame4 ,text="Picture, Image, Photo File",bg="#2f4155",fg="yellow").place(x=20,y=5)
root.mainloop()
Ln 101, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.2 64-bit

RESULT:

User documentation is an essential component of any software system, as it helps user understand how to use the software and what features it offers.

STEP 1: Open the Python Code (Image Steganography).

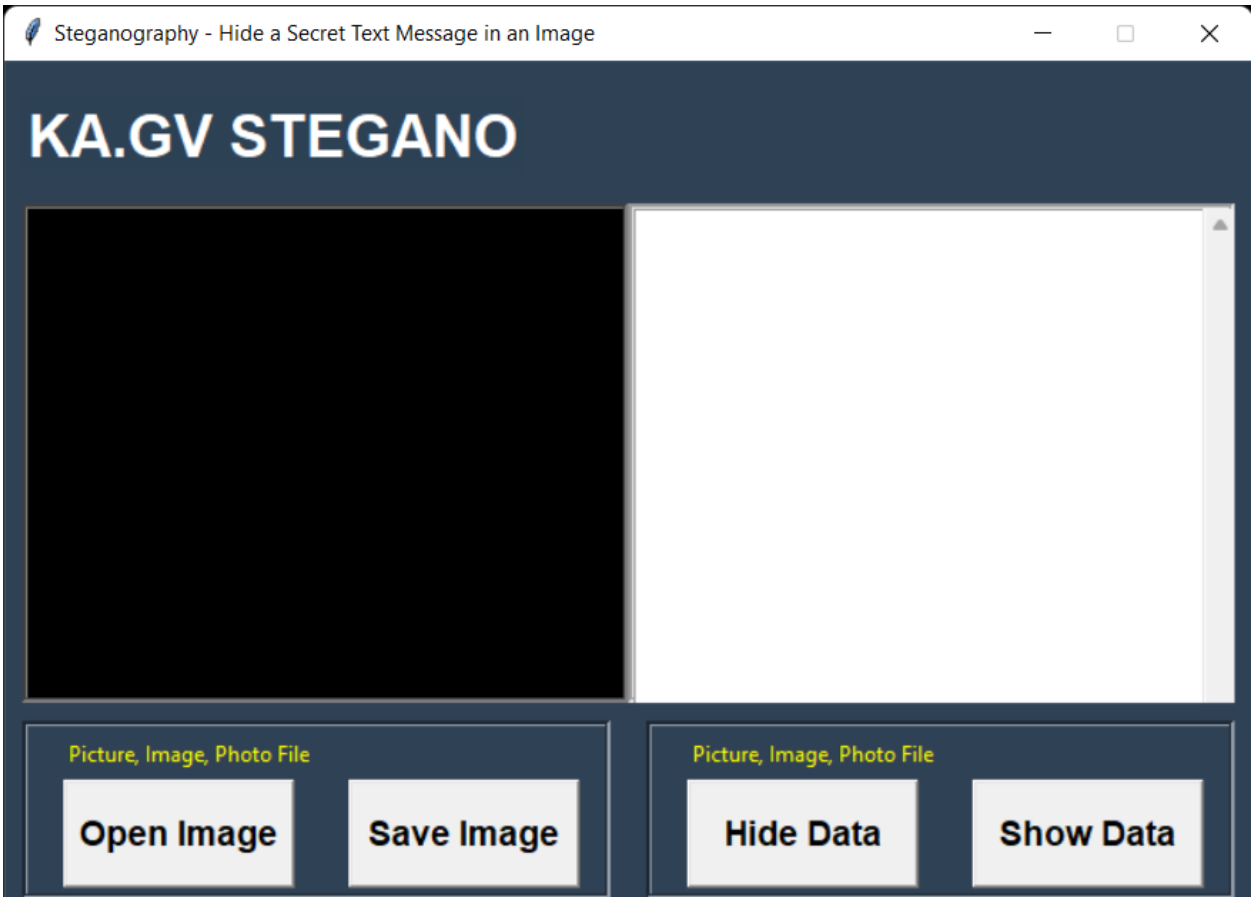
STEP 2: Run the Python Code (Image Steganography).

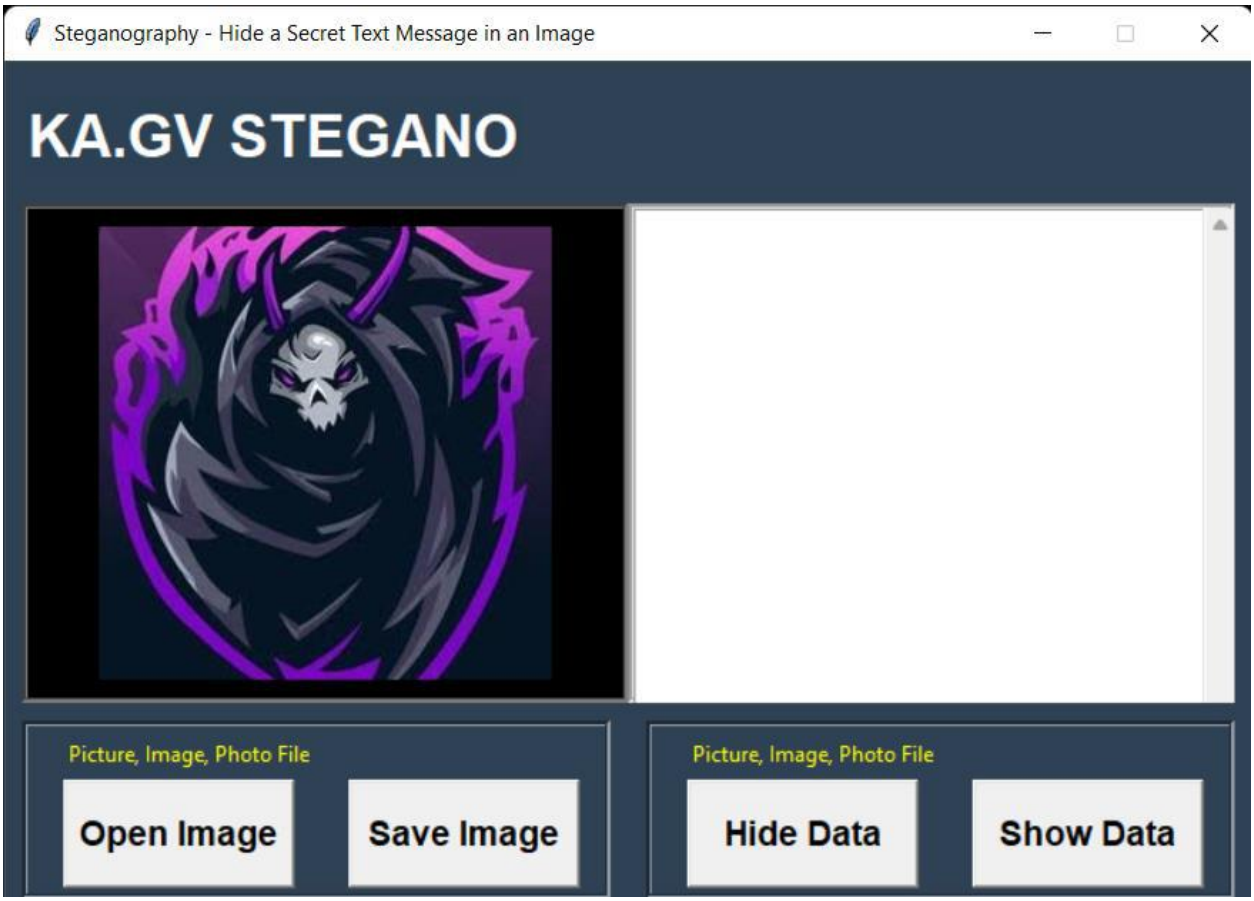
STEP 3: Now our Image Steganography tool appears. Select any Image from your Computer by clicking Open Image button.

STEP 4: Then type the Data that you want to Hide. Next click Hide Date button.

STEP 5: Now click Save Image button to save the Encrypted Image.

STEP 6: Now Open the Encrypted Image and Click Show Data button To see the Decrypted Data in the Image.


Steganography - Hide a Secret Text Message in an Image
KA.GV STEGANO
Picture, Image, Photo File
Open Image
Save Image
Picture, Image, Photo File
Hide Data
Show Data


Steganography - Hide a Secret Text Message in an Image
KA.GV STEGANO
Picture, Image, Photo File
Open Image
Save Image
Picture, Image, Photo File
Hide Data
Show Data


Steganography - Hide a Secret Text Message in an Image
KA.GV STEGANO
Hi! Python is a Easy Programing Laguage
Picture, Image, Photo File
Open Image
Save Image
Picture, Image, Photo File
Hide Data
Show Data