A PROJECT

On

INVISIBLE IMAGE WATERMARKING

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

Bachelor of Technology

In

Computer science and engineering

By

R. V. N. Swetha Sri - O180143

J. Ahalya Varshitha - O180144

V. Swathi - O180155

T. Deepika - O180156

Under the Guidance

P.Archana(Ph.D), Assistant Professor

Computer Science and Engineering

Department of Computer Science and Engineering

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

(Established through Government of A.P Act of 18 of 2008)

Ongole, Prakasam (Dt.) AP-523225

CERTIFICATE

This is to certify that the project entitled "**INVISIBLE IMAGE WATERMARKING"** being submitted by **R. V. N. Swetha Sri** of ID Number **O180143, J. Ahalya Varshitha of** ID Number **O180144**, **V. Swathi** of ID Number **O180155** and **T. Deepika** of ID Number **O180156** and in partial fulfillment of the requirements for the award of the degree of the Bachelor of Technology in Computer Science and Engineering in **RGUKT-AP, IIIT Ongole** is a record of bona fide work carried out by them under my guidance and supervision from September 2022 to January 2023.

The results presented in this project have been verified and found to be satisfactory. The results embodied in this project report have not been submitted to any other University for the award of any other degree or diploma.

Supervisor	**Mr. B. Samapth Babu**
Ms.P.Archana(Ph.D),	Head of Department, CSE,
Assistant Professor,	RGUKT, Ongole.
Department of CSE.	

APPROVAL SHEET

The report entitled **INVISIBLE IMAGE WATERMARKING** by R. V. N. Swetha Sri (O180143), J. Ahalya Varshitha(O180144), V. Swathi(O180155) and T. Deepika(O180156) is Ms. P. Archana approved for the degree of the Bachelor of Technology in Computer Science & Engineering.

Examiners: ________________________

____

____

Supervisor(s):

____

____

____

Date: __________________

Place: _________________

ACKNOWLEDGEMENT

It is our privilege to express a profound sense of respect, gratitude and indebtedness to our guide **Ms. P. Archana(Ph.D), Assistant Professor**, Dept. of Computer Science and Engineering RGUKT-AP, IIIT Ongole, for her indefatigable inspiration, guidance, cogent, discussion, constructive criticisms and encouragement throughout the dissertation work.

We express our sincere gratitude to **B. Sampath Babu, Assistant Professor & Head** of Department of Computer Science and Engineering, RGUKT-AP, IIIT Ongole for his suggestion motivations and co-operation for the successful completion of the work.

We extend our sincere thanks **Dr B Jaya Rami Reddy, Director**, RGUKT-AP, IIIT Ongole for his encouragement.

R. V. N. Swetha Sri - O180143

J. Ahalya Varshitha- O180144

V. Swathi - O180155

T. Deepika - O180156

DECLARATION

We hereby declare that the project work entitles **"INVISIBLE IMAGE WATERMARKING"** submitted to the **RGUKT-AP, IIIT Ongole** in partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology (B Tech)** in Computer Science and Engineering is a record of an original work done by us under the guidance of **Ms. P. Archana(Ph. D), Assistant Professor** and this project work have not been submitted to any university for the award of any other degree or diploma.

R. V. N. Swetha Sri- O180143

J. Ahalya Varshitha- O180144

V. Swathi- O180155

T. Deepika- O180156

Date:12 th July ,2023

ABSTRACT

A watermark is a pattern of bits included into a digital image that identifies the copyright information (author rights, etc.) of the file. Watermarks are typically designed to be prominent but undetectable. The watermark is placed inconspicuously in the most important areas of the host image so that interfering with it to remove it or destroy it would reduce the image's artistic value. Copyright information is covertly inserted into audiovisual data through digital watermarking. Therefore, copyright protection, finger protection, fingerprinting, copy protection, and broadcast monitoring have all used digital (invisible) watermarking.

The user-defined characters are employed as a zone of interest for the water marking procedure by the algorithm, eliminating the possibility of watermark removal. We are particularly interested in developing this project to prevent misuse by careless parties; there are numerous ways to include copyright information in an image.

CONTENTS

1.INTRODUCTION

The website with the name "Invisible Image Watermarking" allows users to hide the watermark when adding it to a photo. Thanks to the internet's and networks' quick development, sharing of multimedia data transformations has become widespread. Since multimedia material can be quickly changed and copied, copyright protection is becoming more and more important. It is the covert identification of multimedia data as belonging to a certain brand. The use of covert picture watermarking to copyright-protect multimedia content has been proposed. Invisible watermarking requires resilience and imperceptibility.

Copyright information is covertly embedded in multimedia data using invisible image watermarking. Thus, copyright protection, finger protection, fingerprinting, copy protection, and broadcast monitoring have all been achieved through the use of invisible image watermarking. Digital video, music clips, and image watermarks are popular sorts of signals. Here, the emphasis is on using still photographs to apply invisible image watermarking. The main technical problem is to develop an extremely reliable invisible image watermarking method that deters copyright infringement by making the removal of the watermark time-consuming and expensive.

The undetectable image in digital media, the data can be a number, text, image, or video, and watermarking explains the processes and procedures to disguise it. A message called a "watermark" can be added to digital data, including text, images, and video, and the embedded data can then be removed. Another type of watermarking is steganography, in which the messages are buried within the content without drawing attention to their presence. A notable example of watermarking is the Indian rupee. In the typical watermarking process, the original image is embedded with the watermark, and the finished product is an image with the watermark.

1.1 MOTIVATION:

More regularly than ever before, digital data and information are sent via the internet. Digital media's popularity has increased because to the availability and efficiency of global computer networks for the transmission of digital information and data. As a result, information security is becoming increasingly crucial for the exchange and transmission of information between individuals. Information security measures include using symmetric and asymmetric encryption technologies to prevent unauthorized access.

Historically, a number of encryption techniques have been used to safeguard digital data.
However, encryption by itself does not offer a sufficient solution because it merely guarantees reliable

content delivery. Once the content has been decrypted, it is no longer secure and may be illegally copied without any safeguards. In light of the internet and computers, piracy is a serious issue. The invisible picture watermarking technique offers an advantage over the other methods for preventing the theft and duplication of multimedia data. As a result, watermarking techniques have gained popularity in recent years.

1.2 PROBLEM DEFINITION

The drawback of the watermarking technique is that it can be used as a tracking tool to determine where biometric data came from. In order to watermark, the data must be embedded into the host data. A copyright protection technique called invisible watermarking is used to incorporate particular data in a cover file to stop unauthorized use. Our understanding of watermarking has been enlarged by the digitization of our world to include immaterial digital impressions for use in establishing ownership claims and safeguarding proprietary interests. However, invisible watermarks are conceptually similar to their paper forebears. They denote some aspect of the document or file's token that they inherit. Whether the product of paper press or discrete cosine transformations, watermarks of varying degree of visibility are added to presentation media as a guarantee of authenticity, quality ownership and source.

1.3 OBJECTIVE OF THE PROJECT:

The project's primary purpose is to implant a durable hidden watermark into an image, which is the ultimate goal of invisible watermarking technology. A new area in computer science, cryptology, signal processing, and communications is invisible watermarking technology. There are applications for both types of watermarks, visible and invisible. Here, we favor image watermarking that is invisible. The watermark should be inserted into randomly chosen areas in a particular domain of the watermark signal in order to ensure its security. This makes it challenging to erase the watermark. The main technical problem is to develop an extremely reliable invisible image watermarking method that deters copyright infringement by making the removal of the watermark time-consuming and expensive.

2.LITERATURE REVIEW

[1] Manjit Thapa (2011)” Digital Image Watermarking Technique Based on Different Attacks”, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 4, Pp 14 -19

[2] Puneet Kr. Sharma (2012), “Analysis of image watermarking using least significant bit algorithm “International Journal of Information Sciences and Techniques (IJIST) Vol.2, No.4, Pp 95 -101

[3] Radhika. V (2013), “Comparative Analysis of Watermarking in Digital Images Using DCT & DWT” International Journal of Scientific and Research Publications, Volume 3, Issue 2 Pp 1-4

[4] Mohan Durvey (2014), “A Review Paper on Digital Watermarking “International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) Volume 3, Issue 4, Pp 99- 105

[5] Zhu Yuefeng (2015), “Digital image watermarking algorithms based on dual transform domain and self-recovery” international journal on smart sensing and intelligent systems vol. 8, no. 1, pp 199- 219

[6] Compressive optical image watermarking using joint Fresnel transform correlator architecture – Optics and Lasers in Engineering, Volume 89, 2017, pp 29-33

[7] Digital watermarking: Applicability for developing trust in medical imaging workflows state of the art review – Computer Science Review, Volume 27, 2018, pp. 45-60

[8] Optical image watermarking based on singular value decomposition ghost imaging and lifting wavelet transform – Optics and Lasers in Engineering, Volume 114, 2019, pp. 76-82

[9] Optical ghost cryptography and steganography – Optics and Lasers in Engineering, Volume 130, 2020, Article 106094

[10] Cryptanalysis on the image encryption scheme based on interference and an amplitude mask – Optics Communications, Volume 517, 2022, Article 128272

[11] Enhanced Invisible image watermarking technique has been designed to overcome the problem of existing system and reduce the piracy issues.

3.ANALYSIS

3.1 EXISTED SYSTEMS

To prevent image copying without authorization, there are numerous approaches available today, including watermarking, digital watermarking, and many cryptographic methods. These techniques all result in compressed images with decreased quality under specific circumstances. The methods now in use cut down on copy theft and piracy. However, there are just a few shortcomings that can be fixed and improved upon beyond what is now possible. The watermarking approaches shield the image from a variety of situations, but they don't improve its qualities, instead lowering its size, quality, and resolution. Thus, the improved version of the current watermarking methods are now available.

3.2 PROPOSED SYSTEM

The method for invisible image watermarking that is being suggested seeks to offer a complete solution for embedding and extracting undetectable watermarks, ensuring authenticity, ownership verification, and protection of digital images. The suggested system has numerous features, including watermark acquisition, picture processing, and watermark embedding. Here, two images are used as input: one is a normal image and the other is a watermark image. The output includes a normal image that has an invisible watermark image placed in it. The watermark is extracted from the output image and added to the input image by decoding the output image. The main goal is to conceal the watermark, make the image appear normal, and safeguard it as well.

3.3 SOFTWARE REQUIREMENT SPECIFICATION

Our project was created using the Python programming language. Image processing and image acquisition technologies are employed. We finished our project by utilizing the aforementioned programming languages and approaches. We employ several hardware requirements as well as software languages in this project. They are the following: keyboard, monitor, hard disk, RAM, and ethernet connection. Editors like Jupyter Notebook are used to build all of these through coding, and "chrome/Firefox" is the preferred browser program for visualizing our project.

3.3.1 PURPOSE:

Our project's primary goals are to forge, invalidate, and remove watermarks, which have numerous advantages. Watermarking is a pattern of bits added to a digital image, video, or audio file that reveals the file's copyright information. It is simple to identify and challenging to erase. Watermarking serves as a protective measure by allowing an owner's property right to be asserted for photos, video, audio, and software through a concealed watermark. The main objective of the invisible image watermarking project is to achieve a strong level that makes it impossible to remove a watermark without degrading the quality of the data objects.

3.3.2. SCOPE:

The creation, application, and use of the watermarking system are all included in the scope of an invisible picture watermarking project. Create and create sophisticated algorithms for including and removing undetectable watermarks from digital photos. Think about things like watermark invisibility, attack resistance, information embedding capability, and image processing operations resistance. incorporating the watermarking techniques into a system or library of software that may be incorporated into applications or workflows already in use for image processing. Assure that it is compatible with widely used image formats, and offer batch processing and scalability support. All of them are taken into account, and the suggested system is put into place.

3.3.3 OVERALL DESCRIPTION:

The project's goal is to create a system for invisible image watermarking, a method for adding undetectable data to digital photographs for a variety of reasons, including copyright protection, authentication, and content ownership verification. Without sacrificing the aesthetic quality of digital photos, the goal is to increase their security and integrity. A reliable and effective algorithm will be used by the system to embed the watermark into the image. The image's visual content won't be changed because the watermark won't be visible to the human eye. The embedded watermark will act as a covert signature or identify that authorized parties can extract and validate.

This project seeks to make a contribution to the field of digital picture security and intellectual property protection by establishing an invisible image watermarking system. The approach can be used to prevent unlawful usage, uphold ownership, and safeguard the integrity of digital images across a variety of industries, including photography, digital art, publishing, and online media.

4.DESIGN

4.1 UML DIAGRAMS:

The abbreviation UML stands for Unified Modelling Language. In the area of object-oriented software engineering, UML is a general purpose modeling language that has been standardized. The Object Management Group oversees and developed the standard.

The creation of a common modeling language for object-oriented software engineering is a key objective of UML. UML now consists of two main parts: a notation and a meta-model. In the future, UML might also be coupled with or added to in the form of a method or process. The Unified Modeling Language is a standard language used for business modeling, non-software systems, and specifying, building, and documenting the artifacts of software systems.

UML, or Unified Modeling Language, is the acronym. UML is a general purpose modeling language that has been standardized in the field of object-oriented software engineering. The standard was created and is supervised by the Object Management Group.

Establishing a common modeling language for object-oriented software engineering is one of UML's key objectives. The two core elements of UML today are the notation and the meta-model. A technique or procedure may be added to or improved upon in the future in addition to UML. For business modeling, non-software systems, as well as for describing, producing, and documenting the artifacts of software systems, the Unified Modeling Language is used as a standard language.

The following are the main objectives in the UML's design:

1. Offer users an expressive visual modeling language that is ready to use so they can create and trade meaningful models.
2. Offer methods for specialization and extendibility to expand the fundamental ideas.
3. Be unreliant on specific development methodologies and programming languages.
4. Promote the commercial expansion of OO tools.
5. Encourage the integration of best practices and support higher level development ideas like collaboration, frameworks, patterns, and components.

The goal of a UML Diagram, which is based on the UML (Unified Modeling Language), is to graphically represent a system together with its primary players, roles, actions, objects, or classes in order to better understand, edit, maintain, or document system-related information. Structural and behavioral UML diagrams make up the UML diagrams.

STRUCTURAL UML DIAGRAMS:

Static representations of a system's structure are shown in structural diagrams. It is frequently employed in software architecture documentation. Developers and stakeholders can better understand and convey the system architecture, linkages, and interactions between various classes thanks to these diagrams, which provide a visual depiction of a system's static structure. They are:

Class Diagram
Object Diagram
Component Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Profile Diagram

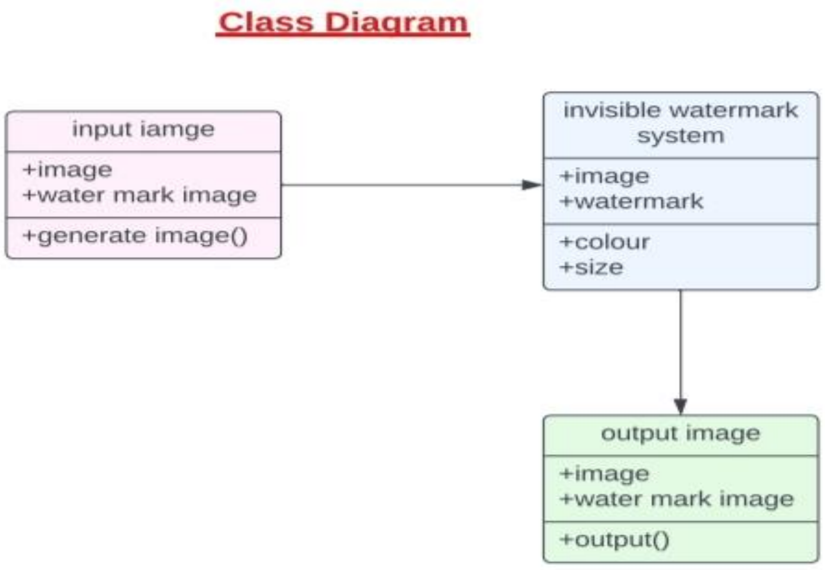
BEHAVIOURAL UML DIAGRAM:

A dynamic picture of a system or the behavior of a system, which explains the operation of the system, is provided by behavioral diagrams. These illustrations aid in explaining how objects interact, react to circumstances, and change states. They support the design, analysis, and validation of the system, ensuring that it operates as intended and satisfies the required specifications. There are seven diagrams. They are:

Use case Diagram
Sequence Diagram
Activity Diagram
State Machine Diagram
Interaction Overview Diagram
Communication Diagram
Timing Diagram

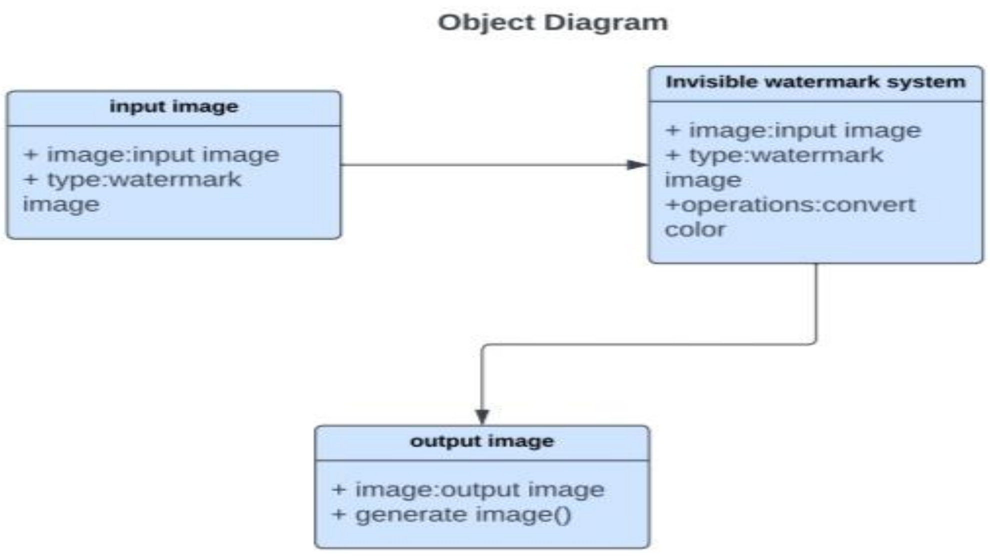
CLASS DIAGRAM

One of the most often used diagrams is the class diagram. All object-oriented software systems are built on it. It shows the system's static structure. It shows the class, properties, and operations of the system. It aids in understanding the relationships between various classes and objects.



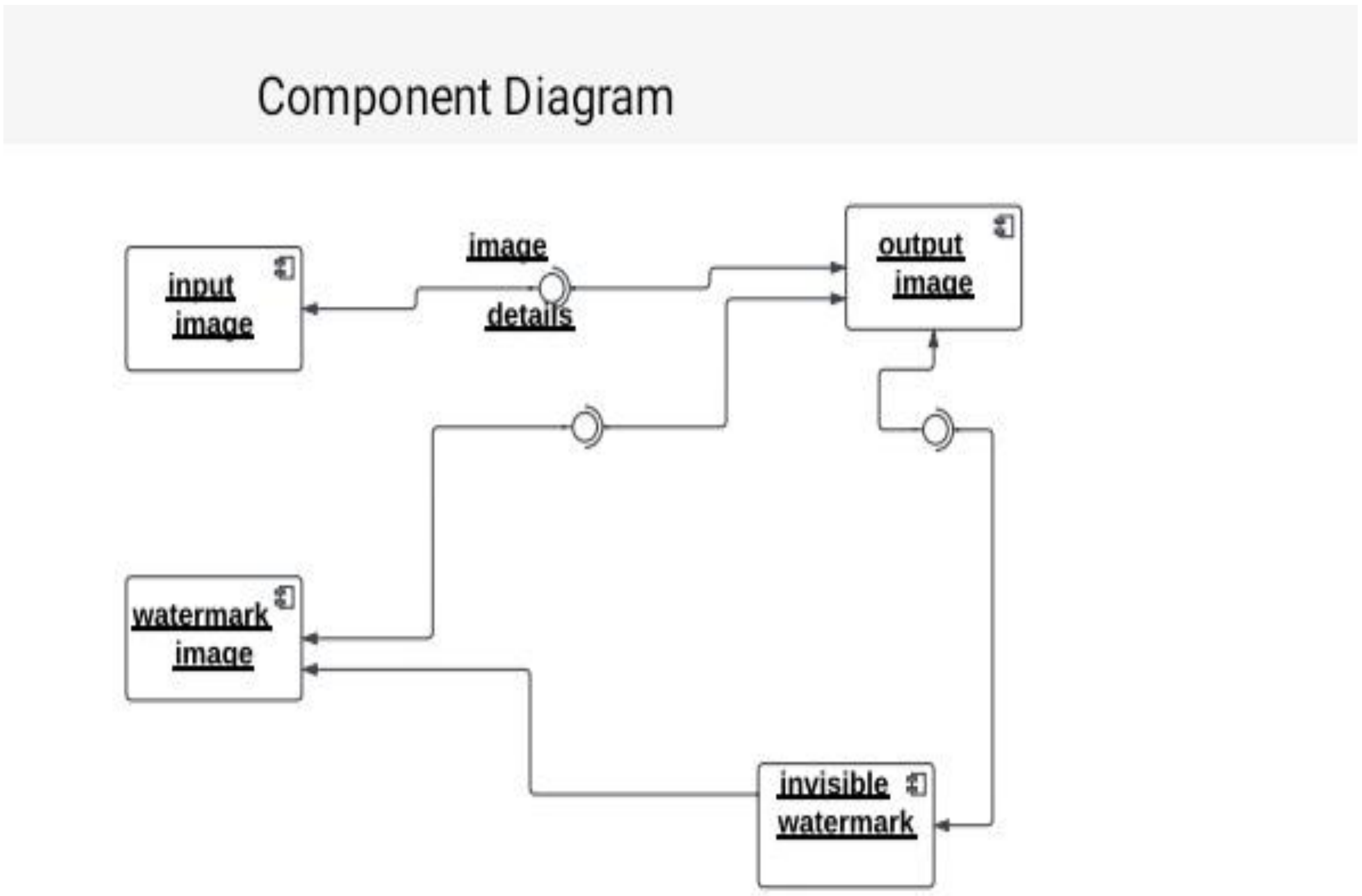
OBJECT DIAGRAM

Class diagrams are exemplified by object diagrams. Both class diagrams and object diagrams have the same fundamental ideas. It describes the system's static structure at a specific period. It can be utilized to gauge how accurate class diagrams are. It displays various instances of classes at once along with their relationships.



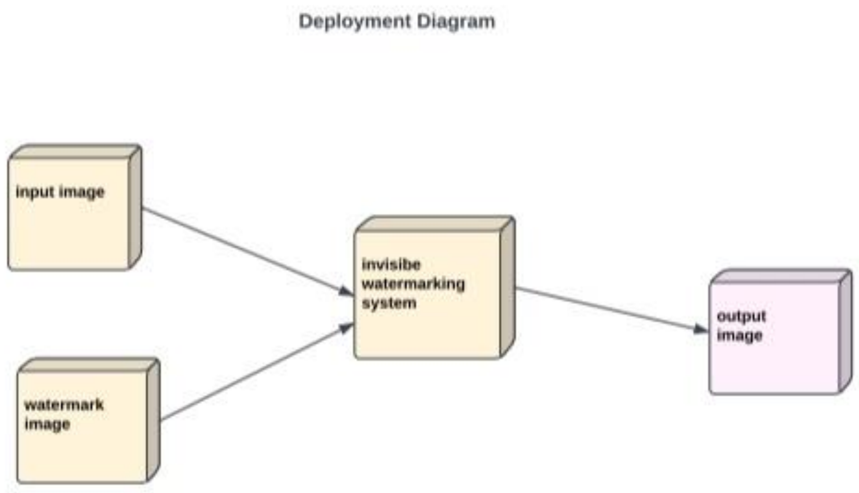
COMPONENT DIAGRAM

It illustrates how the system's physical components are arranged. It is employed for modeling specifics of execution. Because it contains structural linkages between the components of a software system, it can tell whether the planned development has taken into account the desired functional requirements or not.



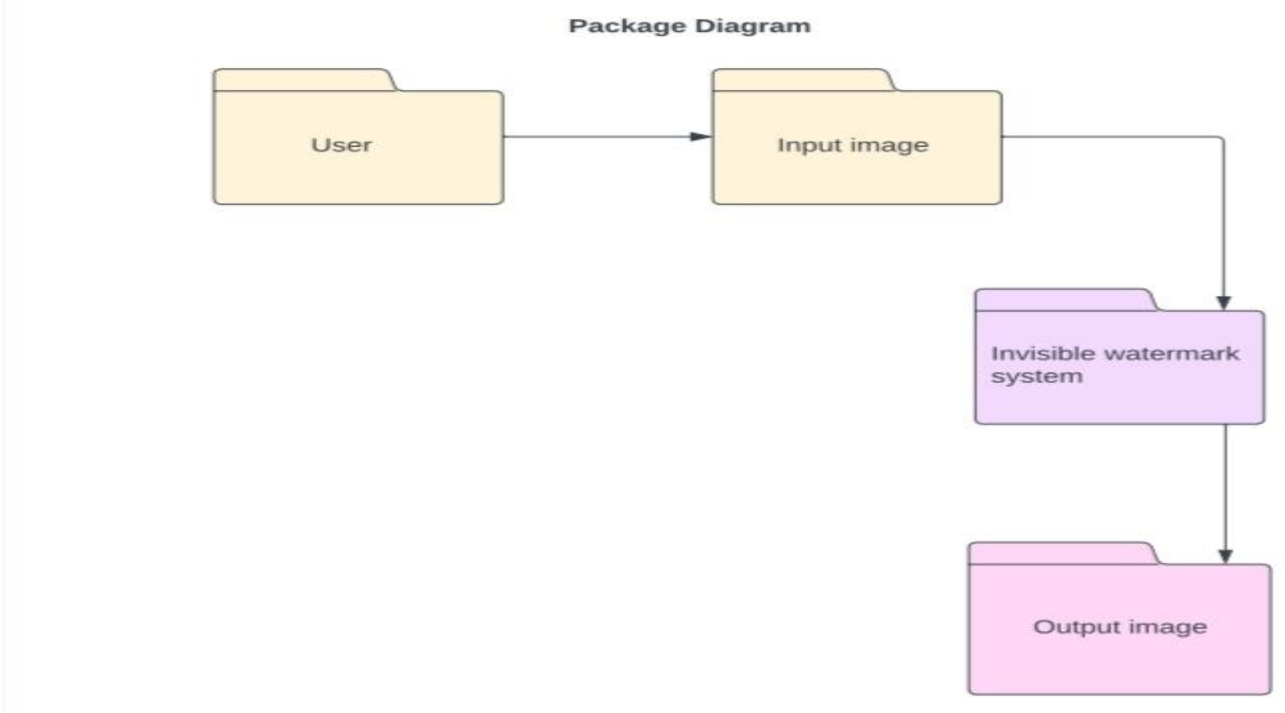
DEPLOYMENT DIAGRAM

By describing the physical components that are now in place and the software components that are executing on them, it shows both the hardware and software of the system. It generates data on system software. Every time software is utilized, disseminated, or installed across a number of machines with various configurations, it is included.



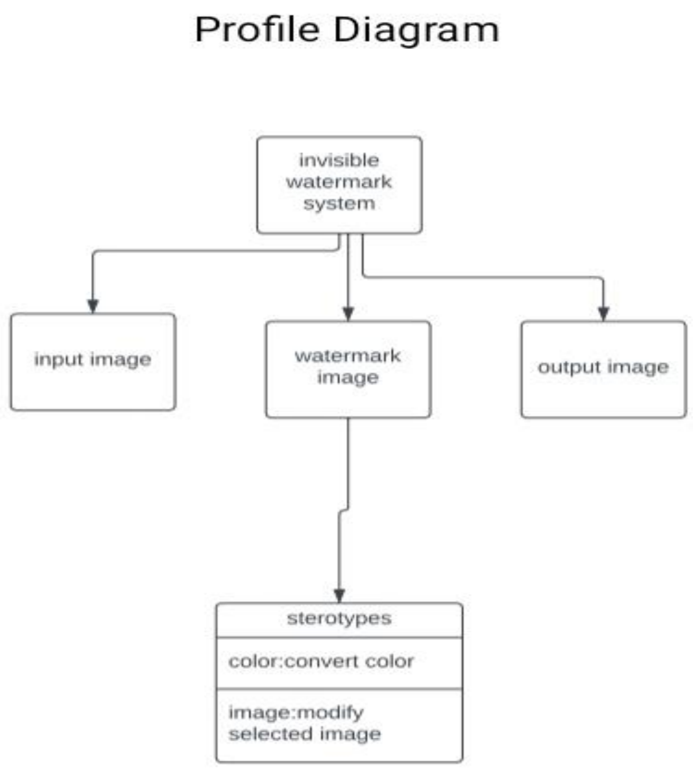
PACKAGE DIAGRAM

It serves as an example of how the components of the packages are arranged. It displays the connections between different packages. It controls UML diagrams by making them simple to comprehend. Use case diagrams and class organization are accomplished using it.



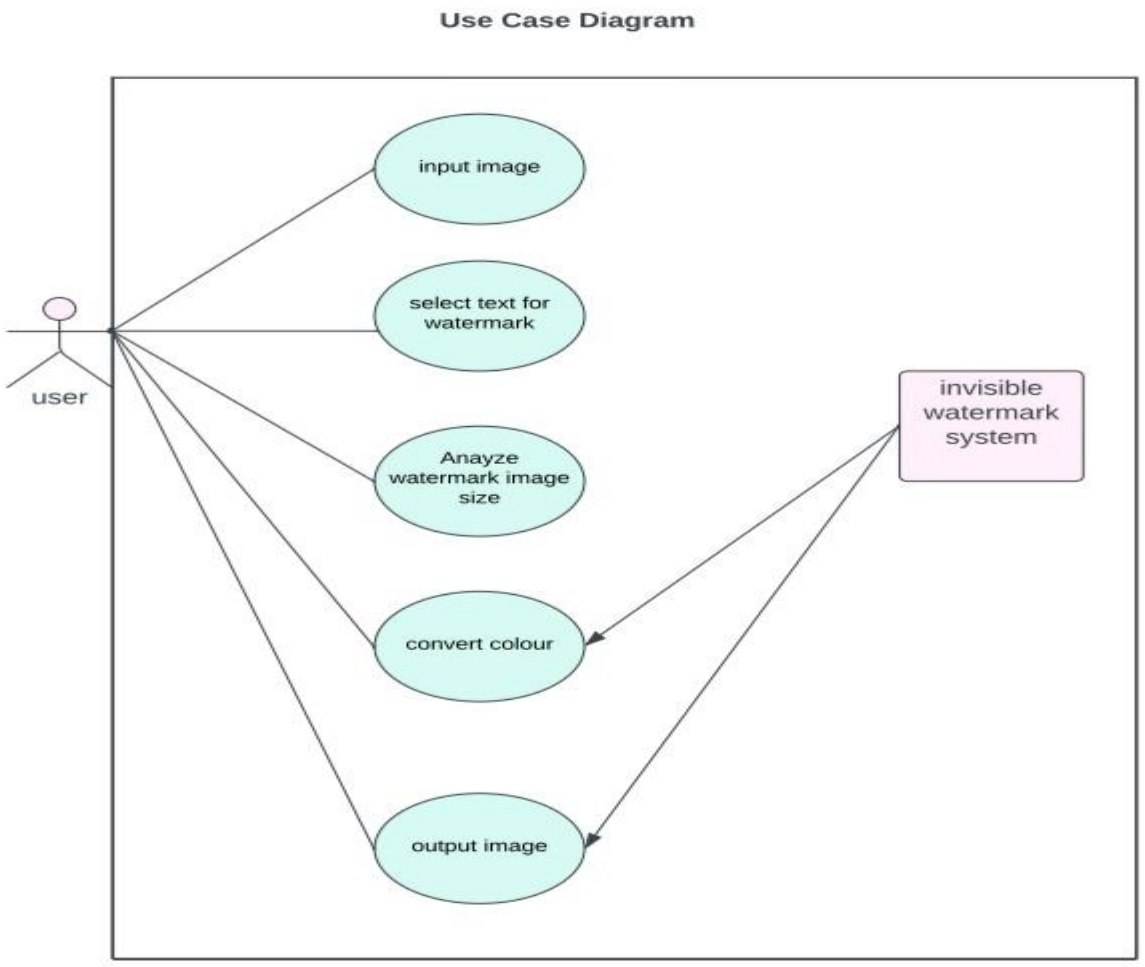
PROFILE DIAGRAM

The Unified Modeling Language's (UML) profile diagram, a type of structural diagram, offers a general extension mechanism for tailoring UML models for specific domains and platforms. Extension mechanisms enable the stringent additive refinement of standard semantics, preventing them from conflicting with standard semantics. Stereotypes, tagged value definitions, and constraints that are applied to particular model elements, such as Classes, Attributes, Operations, and Activities, are used to define profiles.



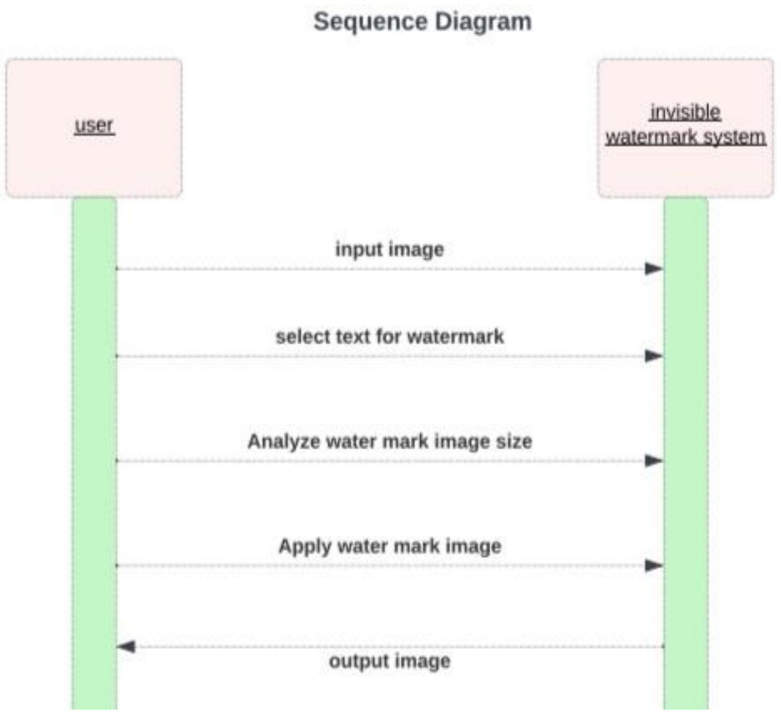
USECASE DIAGRAM

Use-case diagrams aid in capturing system requirements and depict a system's behavior in UML. The scope and high-level functions of a system are described in use-case diagrams. The interactions between the system and its actors are also depicted in these diagrams. utilize-case diagrams show what the system does and how the actors utilize it, but they do not show how the system works within.


Use Case Diagram
user
input image
select text for watermark
Anayze watermark image size
convert colour
output image
invisible watermark system

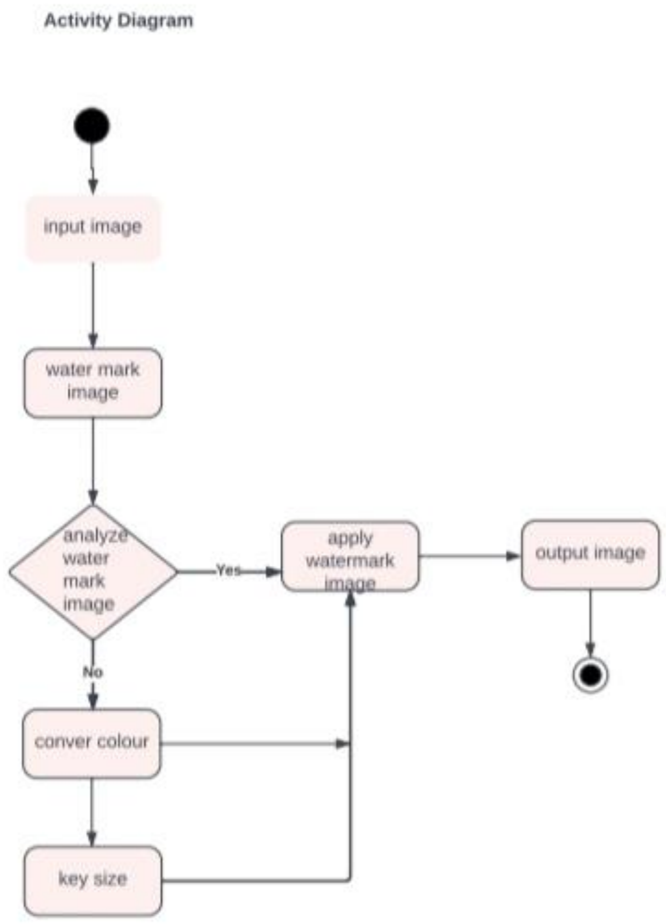
SEQUENCE DIAGRAM

The sequence diagram, which is also known as an event diagram, shows how messages move through the system. It aids in creating a variety of dynamic settings. It depicts communication between any two lifelines as a chronologically ordered series of activities, implying that these lifelines were active at the moment of communication. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.


Sequence Diagram
user
invisible watermark system
input image
select text for watermark
Analyze water mark image size
Apply water mark image
output image

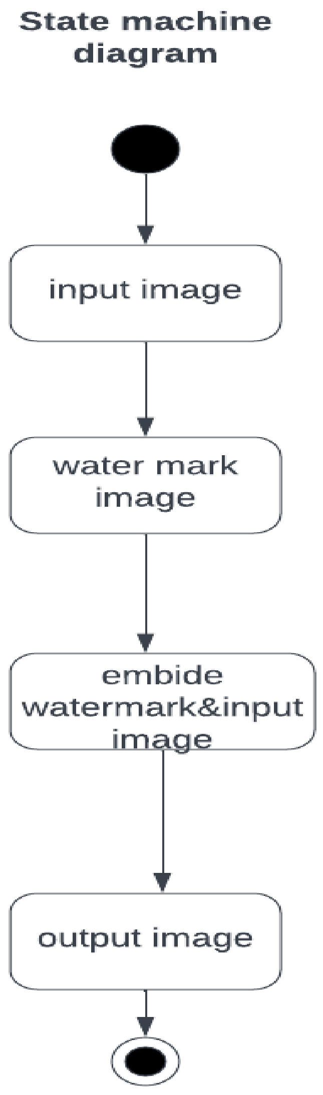
ACTIVITY DIAGRAM

The activity diagram in UML is used to show the system's control flow rather than its implementation. Both concurrent and sequential activities are modeled. The workflow from one action to the next can be seen using the activity diagram. It placed emphasis on the existence of flow and the sequence in which it takes place. The flow may be sequential, branching, or concurrent, and the activity diagram has been designed with a fork, join, etc. to deal with these many types of flows. A flowchart that is object-oriented is another name for it. It includes tasks that include applying a series of actions or processes to simulate the behavioral diagram.



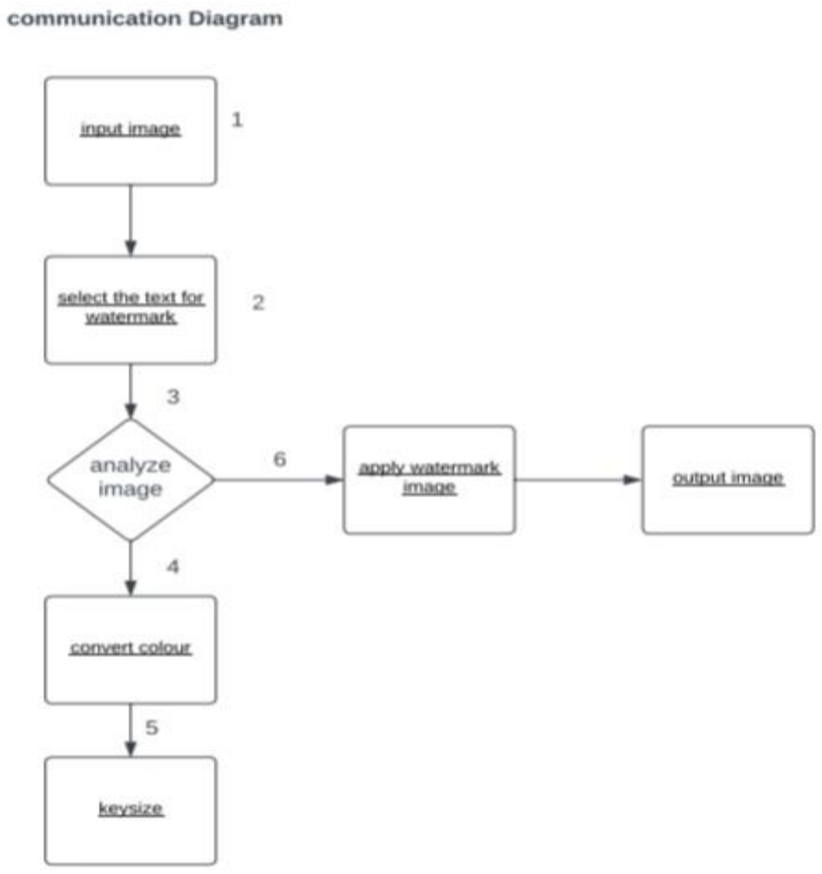
STATE MACHINE DIAGRAM

The state machine diagram, also known as a state chart or a state transition diagram, displays the sequence of states that an object in the system goes through. It records the behavior of the software system. It simulates how a class, a subsystem, a package, and a whole system behave. It often proves to be a useful technique for modeling how the system and external entities interact and collaborate. To manage an object's state, it models event-based systems. Additionally, it specifies a number of unique states in which a system component may exist. Every item or component is in a particular state.



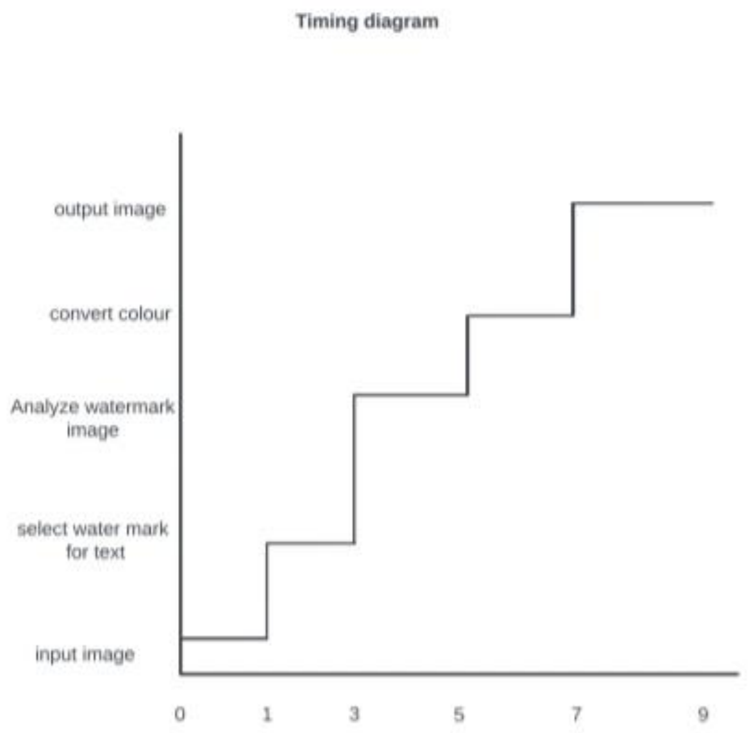
COMMUNICATION DIAGRAM

It demonstrates how the objects exchange sequence messages. It emphasizes the relationships between objects. It depicts a system's static and dynamic behavior. In terms of sequential messages, a communication diagram represents the interactions between objects or parts. Communication diagrams combine data from the class, the sequence, and the use case. diagrams that show a system's static structure and dynamic behavior.



TIMING DIAGRAM

Timing diagrams in UML are a subset of Interaction diagrams, which do not use the same notations as sequence and collaboration diagrams. It is made up of a graph or wave form that shows the condition of a lifeline at a particular moment in time. It illustrates how conditions are altered both inside and between lifelines alongside linear time axis.



5. IMPLEMENTATION

5.1 MODULES

We divide our project into different modules and methodologies. The modules are:

- opencv
- math
- argparse
- numpy
- encoding
- decoding

5.2 MODULE DESCRIPTION

Opencv module:

A Python package called OpenCV makes it possible to carry out image processing and computer vision tasks. It offers a wide range of features, including tracking, object identification, facial recognition, object analysis, and video capture. A cross-platform package called OpenCV allows us to create real-time computer vision applications. It offers a wide variety of algorithms and functions for undertaking various tasks, including feature extraction, object tracking, and more. In order to process images, OpenCV offers a number of functions, including image filtering, thresholding, morphological operations, picture scaling, cropping, and more. Images can be changed by changing their colors, improving their quality, or extracting particular elements. Algorithms for extracting and matching features in images are included in OpenCV.

Math Module:

Python's math module offers a collection of mathematical constants and functions. Since it is a component of the Python Standard Library, using it does not require the installation of any other packages. Python has a math module that can handle these calculations. Both simple operations like addition (+), subtraction (-), multiplication (*), and division (/) and advanced operations like trigonometric, logarithmic, and exponential functions are covered by the functions in the math module. The value of numerous constants, including pi and tau, is provided via the math module. The usage of such constants eliminates the need to precisely and repeatedly write down the value of each constant.

Argparse Module:

It is simple to create user-friendly command-line interfaces thanks to the argparse module. The program specifies the arguments it needs, then argparse works out how to extract those arguments from sys.argv. Additionally, the argparse module automatically creates use and help messages. When users supply the application with erroneous parameters, the module will also produce errors. By allowing user input values to be parsed and used, argparse can be used to offer flexibility and reuse to your code in place of manually setting variables inside of the code. If you wish to let other developers run your code from the command line, you might find this to be helpful.

Numpy Module:

The Python package Numpy is used to manipulate arrays. The equivalent of arrays in Python are lists, although they take a long time to execute. The goal of NumPy is to offer array objects that are up to 50 times faster than conventional Python lists. The NumPy array object is referred to as ndarray, and it has a number of supporting methods that make using ndarray relatively simple. In contrast to lists, NumPy arrays are stored in a single continuous location in memory, making it very easy for programs to access and manipulate them. It provides a vast library of high-level mathematical functions that work on these arrays and matrices, as well as strong data structures that ensure efficient calculations with arrays and matrices.

Encoding Module

The technique of embedding a watermark into the content (such as photographs or videos) so that it is invisible to the human eye or other senses is known as encoding in the context of invisible watermarking. The intention is to include a watermark that is challenging to find or erase but can be retrieved later to confirm the content's legitimacy or owner. The watermark that will be incorporated into the content must first be created. A logo, phrase, or other distinguishing feature that shows the content's owner or validity in a particular way might serve as a watermark.

Decoding Module

The decoding module in a watermarking project is in charge of removing the watermark from the watermarked content. The decoding module examines the watermarked data after the watermark has been integrated into the content in order to extract the watermark. It's important to remember that the quality of the watermarked content, the reliability of the encoding method, and potential content attacks or updates all affect how effective the decoding module is.

METHODOLOGIES:

Image acquisition

Image acquisition is the process of adding a watermark on the input photos. The program takes the image and statically evaluates its color, intensity, and resolution.

Image processing

For projects involving undetectable picture watermarking, image processing is essential. In order to embed and remove invisible watermarks while preserving the visual quality of the original image, photos must be modified and analyzed. It is significant to note that the specific techniques and algorithms used in image processing for projects involving invisible picture watermarking can vary based on the needs, security factors, and the watermarking strategy selected. In order to obtain the appropriate amount of concealment, sturdiness, and security for the watermarking system, researchers and practitioners frequently create, test, and change fresh approaches or alter already existing ones.

5.3 INTRODUCTION OF TECHNOLOGIES USED:

PYTHON:

Python is a popular and flexible programming language that may be utilized in many different parts of a project to create undetectable watermarks. It has a number of benefits, including simplicity, a vast ecosystem of libraries, and robust support for image processing tasks. Powerful libraries for image processing and manipulation are available in Python, such as OpenCV. These libraries provide the functions and resources needed to read and write images, execute transformations, filter images, and implement the different image processing methods needed for embedding and extracting watermarks.

The implementation of machine learning and deep learning algorithms is increasingly being done in Python. These methods can be used for tasks like feature extraction, watermark embedding or extraction, or improving the watermark's robustness in watermarking projects. Python's interactive features and speedy prototyping capabilities let researchers and developers easily test out various strategies and change methods. Iterative development is made easier by Python's simplicity of writing and editing code, which enables quick iterations and tweaks based on experimental findings or research discoveries.

VISUAL STUDIO CODE

Microsoft created the well-known source code editor known as Visual Studio Code (VS Code). It is incredibly flexible, lightweight, and supports a huge selection of programming languages and frameworks. A comprehensive and effective code editing experience is offered by VS Code. It has capabilities like syntax highlighting, code navigation, code restructuring, and IntelliSense (code completion). It comes with support for a large number of programming languages, and you can install more language extensions if you require them for certain tasks. You may run scripts, perform commands, and interact with the command-line interface of your project using VS Code's integrated terminal without leaving the editor. There is no longer a requirement to alternate between the editor and a third-party terminal window.

The popularity of Visual Studio Code is due to its adaptability, performance, and numerous customizability choices. It has become a popular alternative for many coding projects thanks to its extensive adoption among developers working in a variety of disciplines and programming languages.

5.4 SAMPLE CODE

```
#Y = R *  0.29900 + G *  0.58700 + B *  0.11400
#  Cb = R * -0.16874 + G * -0.33126 + B *  0.50000 + 128
#  Cr = R *  0.50000 + G * -0.41869 + B * -0.08131 + 128

import numpy as np
# import matplotlib.pyplot as plt
import cv2
# from imwatermark import WatermarkEncoder, WatermarkDecoder
import math
import argparse

def bgr2ycbcr(img):
    Y = 0.29900 * img[:,:,2] + 0.58700 * img[:,:,1] + 0.11400 * img[:,:,0]
    Cb = -0.16874 * img[:,:,2] + -0.33126 * img[:,:,1] + 0.50000 * img[:,:,0] + 128
    Cr = 0.50000 * img[:,:,2] + -0.41869 * img[:,:,1] + -0.08131 * img[:,:,0] + 128

    Y = Y.astype('int')
    Cb = Cb.astype('int')
    Cr = Cr.astype('int')

    return Y, Cb, Cr

def ycbcr2rgb(im):
    xform = np.array([[1, 0, 1.402], [1, -0.34414, -.71414], [1, 1.772, 0]])
    rgb = im.astype('float')
    rgb[:,:,[1,2]] -= 128
    rgb = rgb.dot(xform.T)
    np.putmask(rgb, rgb > 255, 255)
    np.putmask(rgb, rgb < 0, 0)
    return np.uint8(rgb)

def create_wm_img(w, h, wm_text = 'Do not copy this!'):
    wm_img = np.zeros((h, w))
    font = cv2.FONT_HERSHEY_SIMPLEX
    color = 1
    fontScale = 1.5
    org = (0, 100)
    thickness = 5

    size, _ = cv2.getTextSize(wm_text, font, fontScale, thickness)

    wm_w, wm_h = size
    space = 20
    for i in range(w // wm_w + 1):
        for j in range(h // wm_h + 1):
            org = ((j % 2) * space + (wm_w + space) * i, (wm_h + space) * j)
            wm_img = cv2.putText(wm_img, wm_text, org, font, fontScale, color, thickness,
cv2.LINE_AA)
    return wm_img, wm_w, wm_h

def calc_diff(img):
    return np.std(img).astype('int')

def sigmoid(x):
    return 1.0 / (1.0 + math.exp(-x))

def encode_img(img, wm):
    w, h = img.shape[1], img.shape[0]
    wm_img, wm_w, wm_h = create_wm_img(w, h, wm)

    Y, Cb, Cr = bgr2ycbcr(img)

    # Y, Cb, Cr = img[:,:,2], img[:,:,1], img[:,:,0]

    # plt.imshow(Y, cmap='gray')
    # plt.show()
    diff_size = 5
    diff = 0
    for i in range(h):
        for j in range(1, w):
            if i + diff_size < h and j + diff_size < w and i - diff_size > -1 and j - diff_size > -1:
                diff = calc_diff(Y[i-diff_size:i+diff_size, j-diff_size:j+diff_size])

                depth = (sigmoid(diff) - 0.5) * 15 + 1
                if wm_img[i, j] > 0:
                    Y[i, j] = Y[i, j] + (depth / 2 - 1) / 2 + depth / 2 - Y[i, j] % depth
                else:
                    Y[i, j] = Y[i, j] + (depth / 2 - 1) / 2 - Y[i, j] % depth

    Y = np.expand_dims(Y, axis=2)
    Cr = np.expand_dims(Cr, axis=2)
    Cb = np.expand_dims(Cb, axis=2)

    YCbCr = np.concatenate((Y, Cb, Cr), axis=2)

    rgb = ycbcr2rgb(YCbCr)

    bgr = cv2.cvtColor(rgb, cv2.COLOR_RGB2BGR)

    # encoder = WatermarkEncoder()
    # encoder.set_watermark('bytes', wm.encode('utf-8'))
    # bgr_encoded = encoder.encode(bgr, 'dwtDctSvd')

    return bgr

def decode_img(img, wm_len):
    w, h = img.shape[1], img.shape[0]

    wm_text = []
    # if wm_len == -1:
```

```
#     for i in range(30):
#         try:
#             decoder = WatermarkDecoder('bytes', i * 8)
#             watermark = decoder.decode(img, 'dwtDctSvd')
#             wm_text.append(watermark.decode('utf-8'))
#         except:
#             pass
# else:
#     try:
#         decoder = WatermarkDecoder('bytes', wm_len * 8)
#         watermark = decoder.decode(img, 'dwtDctSvd')
#         wm_text.append(watermark.decode('utf-8'))
#     except:
#         pass

Y, _, _ = bgr2ycbcr(img)

wm_img = np.zeros((h, w))

diff = 0
diff_size = 3
for i in range(h):
    for j in range(1, w):
        if i + diff_size < h and j + diff_size < w and i - diff_size > -1 and j - diff_size > -1:
            diff = calc_diff(Y[i-diff_size:i+diff_size, j-diff_size:j+diff_size])
            cnt = 0
            depth = (sigmoid(diff) - 0.5) * 15 + 1
            if Y[i, j] % depth > (depth / 2 - 1) / 2:
                cnt += 1
            if cnt >= 1:
                wm_img[i, j] = 255

return wm_img, wm_text
```

```
def main(wm="WATERMARK", image="input.jpg", mode="encode", output="output.jpg"):
    if mode == "encode":
        img = cv2.imread(image)
        bgr = encode_img(img, wm)
        cv2.imwrite(output, bgr)
        print("OKAY")
        # img = cv2.imread(output)
        # cv2.imshow("encoded", img)
        # cv2.waitKey(0)

    else:
        bgr = cv2.imread(image)
        res_img, res_txt = decode_img(bgr, -1)
        cv2.imwrite(output, res_img)
        # print(res_txt)
        print("OKAY")
        # cv2.imshow("encoded", bgr)
        # cv2.imshow("watermark", res_img)
        # cv2.waitKey(0)

def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--wm', type=str, default='WATERMARK')
    parser.add_argument('--image', type=str, default='input.jpg')
    parser.add_argument('--mode', type=str, default='encode')
    parser.add_argument('--output', type=str, default='output.jpg')
    opt = parser.parse_args()
    return opt

if __name__ == '__main__':
    opt = parse_opt()
    main(**vars(opt))
```

6.TEST CASES

INPUT IMAGE:

A user should first place the watermark in the input image area when they want to add one to any image. The user should verify whether the function accepts the input image or not. Only images with the extensions jpg, jpeg, and png should be used as input. The code changes the picture to one that matches the watermark image's size and resolution if it is not of a specific size and resolution.

WATERMARK IMAGE:

Along with the input image, the user needs also add the watermark image. Only jpg, jpeg, and png extensions should be used for the watermark image. The code modifies the image to a resolution and size that match the input image if the image is not of the required size and resolution.

ENCODED IMAGE:

The code runs and an encoded image is received as output when the user enters both the watermark and the input image. The image that has an invisible watermark placed in it is the encoded image.

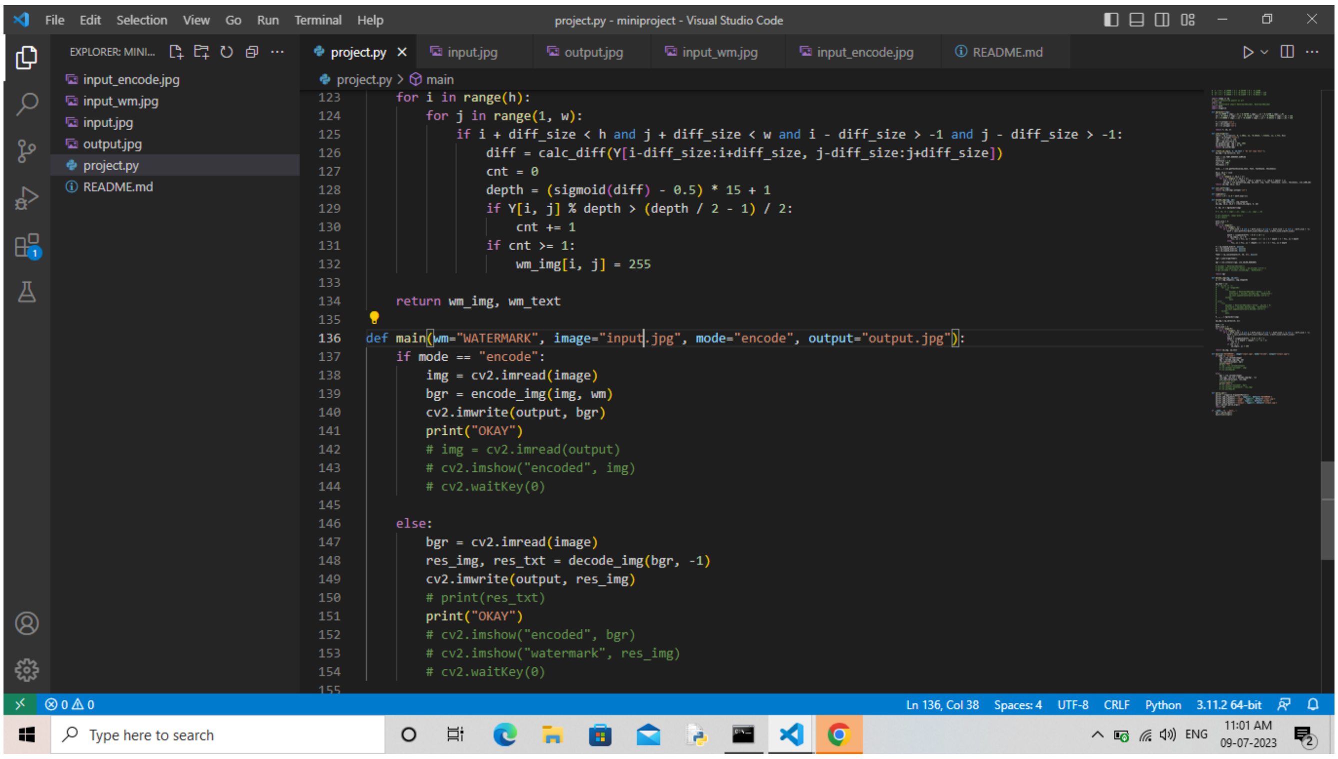
DECODED IMAGE:

The decoded image is produced when the user executes the decoding code after encoding the image. The watermark is the output that is taken from the encoded image in the decoding procedure, demonstrating that the watermark is invisibly incorporated into the image.

7.SCREENSHOTS

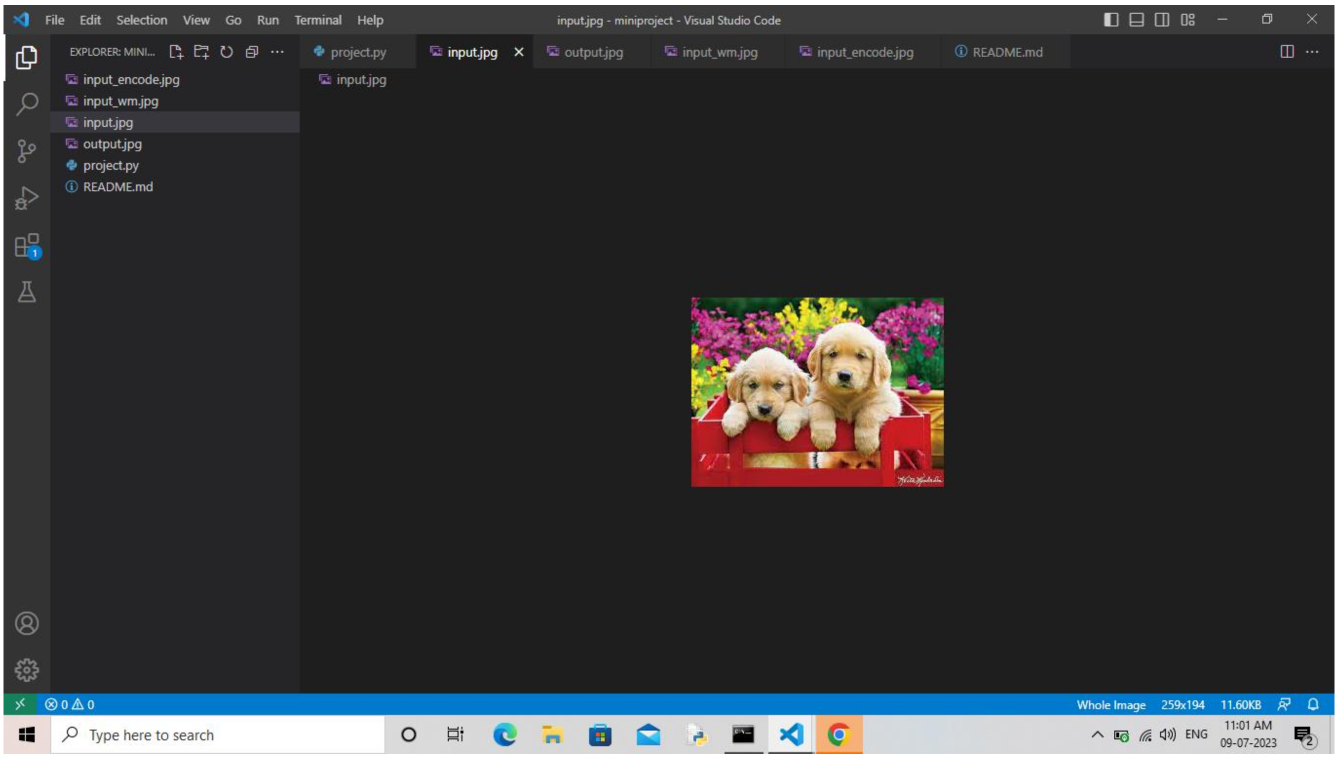
EXAMPLE-1:

CODE WITH IMAGES PATHS:

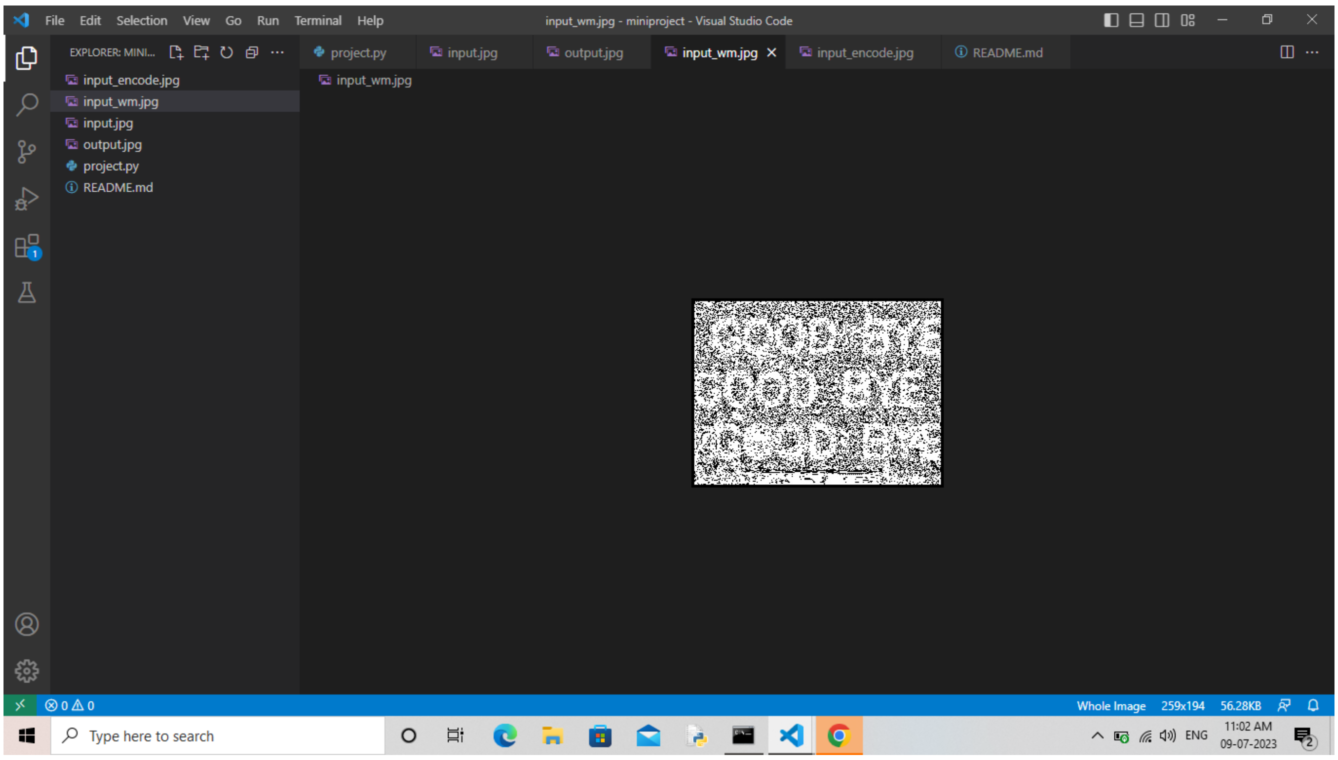


```python
        for i in range(h):
            for j in range(1, w):
                if i + diff_size < h and j + diff_size < w and i - diff_size > -1 and j - diff_size > -1:
                    diff = calc_diff(Y[i-diff_size:i+diff_size, j-diff_size:j+diff_size])
                    cnt = 0
                    depth = (sigmoid(diff) - 0.5) * 15 + 1
                    if Y[i, j] % depth > (depth / 2 - 1) / 2:
                        cnt += 1
                    if cnt >= 1:
                        wm_img[i, j] = 255

        return wm_img, wm_text

def main(wm="WATERMARK", image="input.jpg", mode="encode", output="output.jpg"):
    if mode == "encode":
        img = cv2.imread(image)
        bgr = encode_img(img, wm)
        cv2.imwrite(output, bgr)
        print("OKAY")
        # img = cv2.imread(output)
        # cv2.imshow("encoded", img)
        # cv2.waitKey(0)

    else:
        bgr = cv2.imread(image)
        res_img, res_txt = decode_img(bgr, -1)
        cv2.imwrite(output, res_img)
        # print(res_txt)
        print("OKAY")
        # cv2.imshow("encoded", bgr)
        # cv2.imshow("watermark", res_img)
        # cv2.waitKey(0)
```

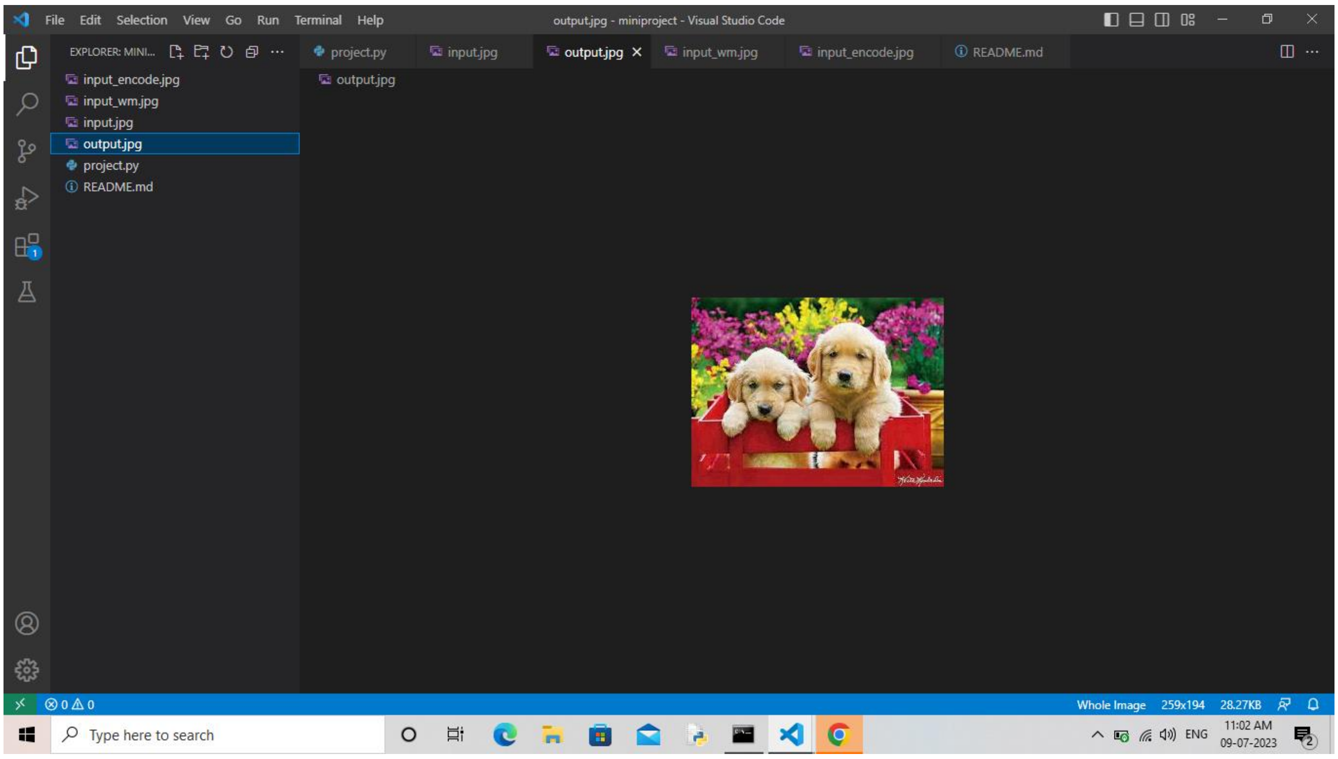
INPUT IMAGE:


File Edit Selection View Go Run Terminal Help
input.jpg - miniproject - Visual Studio Code
EXPLORER: MINI...
project.py
input.jpg
output.jpg
input_wm.jpg
input_encode.jpg
README.md
input_encode.jpg
input_wm.jpg
input.jpg
output.jpg
project.py
README.md
input.jpg
Whole Image 259x194 11.60KB
Type here to search
ENG
11:01 AM
09-07-2023

WATERMARK IMAGE (DECODED IMAGE):

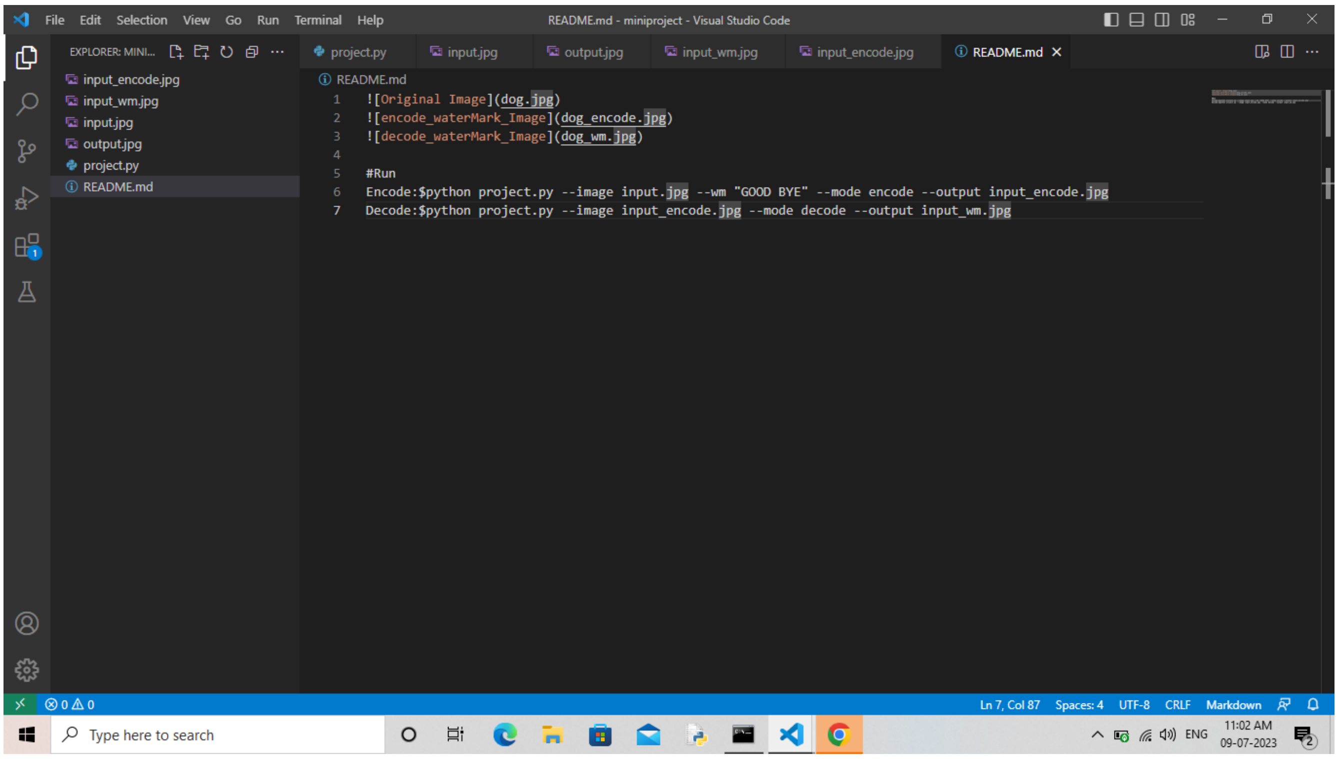


OUTPUT IMAGE:


File Edit Selection View Go Run Terminal Help
output.jpg - miniproject - Visual Studio Code
EXPLORER: MINI...
input_encode.jpg
input_wm.jpg
input.jpg
output.jpg
project.py
README.md
project.py
input.jpg
output.jpg
input_wm.jpg
input_encode.jpg
README.md
output.jpg
Whole Image 259x194 28.27KB
Type here to search
ENG
11:02 AM
09-07-2023

Encode:$ python project.py –image input.jpg –wm “GOODBYE” –mode encode –output input_encode.jpg

Decode:$ python project.py –image input_encode.jpg –mode decode –output input_wm.jpg

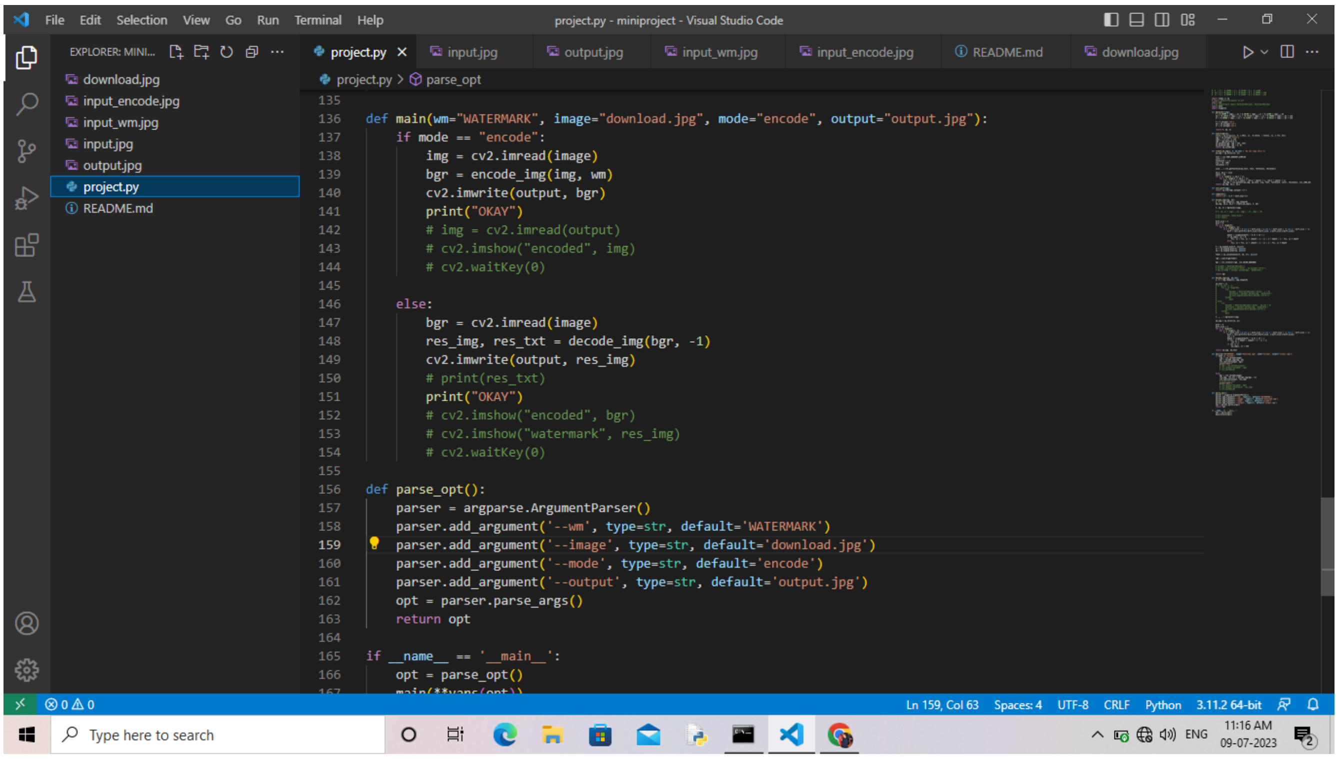


```
![Original Image](dog.jpg)
![encode_waterMark_Image](dog_encode.jpg)
![decode_waterMark_Image](dog_wm.jpg)

#Run
Encode:$python project.py --image input.jpg --wm "GOOD BYE" --mode encode --output input_encode.jpg
Decode:$python project.py --image input_encode.jpg --mode decode --output input_wm.jpg
```

README.md - miniproject - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER: MINI...
input_encode.jpg
input_wm.jpg
input.jpg
output.jpg
project.py
README.md
Ln 7, Col 87 Spaces: 4 UTF-8 CRLF Markdown
Type here to search
11:02 AM
09-07-2023

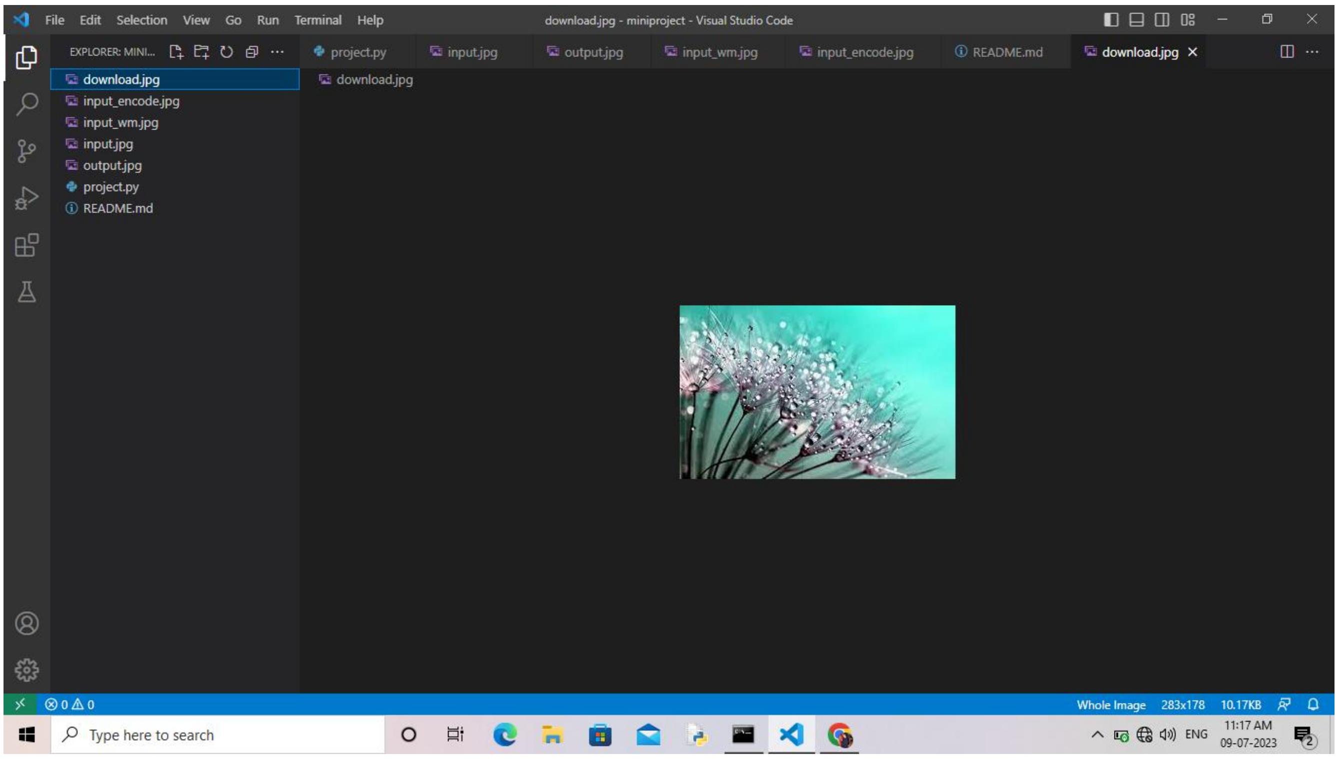
EXAMPLE-2:

CODE WITH IMAGES AS INPUT:

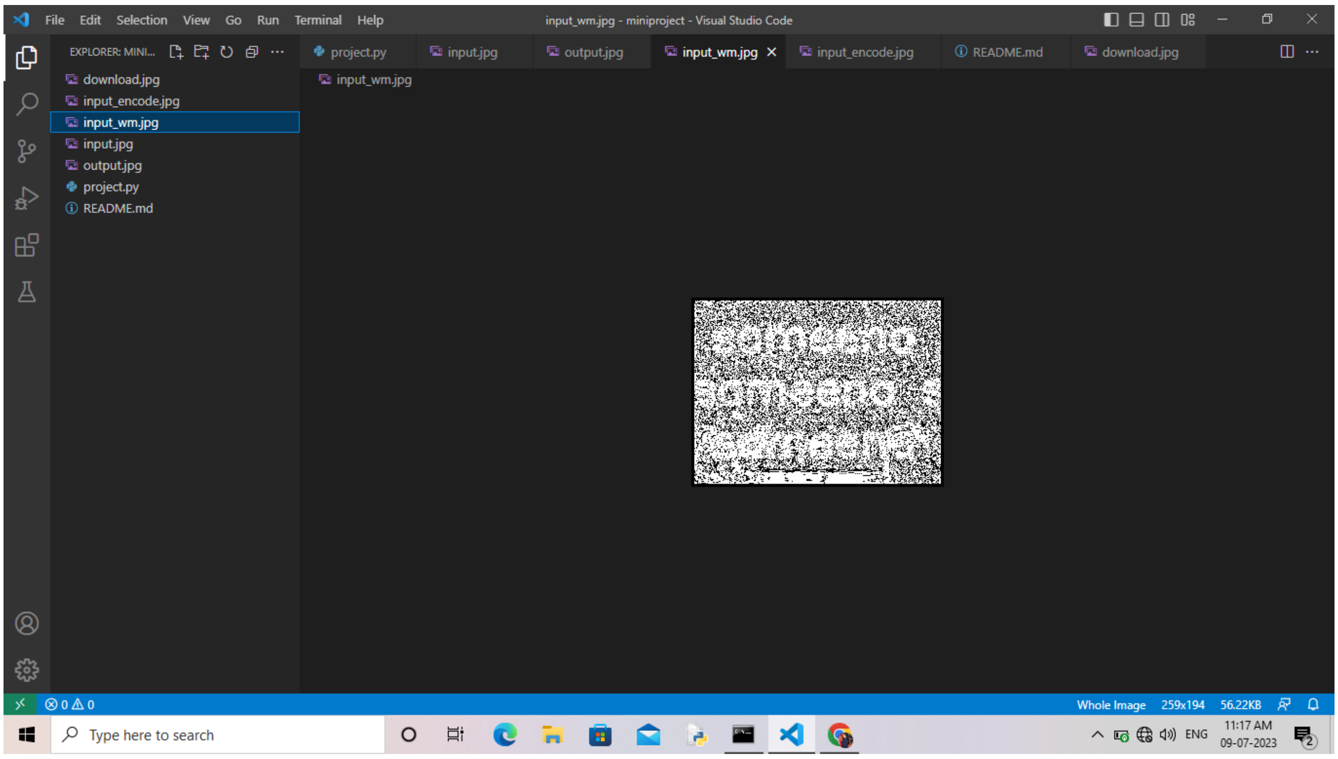


```python

def main(wm="WATERMARK", image="download.jpg", mode="encode", output="output.jpg"):
    if mode == "encode":
        img = cv2.imread(image)
        bgr = encode_img(img, wm)
        cv2.imwrite(output, bgr)
        print("OKAY")
        # img = cv2.imread(output)
        # cv2.imshow("encoded", img)
        # cv2.waitKey(0)

    else:
        bgr = cv2.imread(image)
        res_img, res_txt = decode_img(bgr, -1)
        cv2.imwrite(output, res_img)
        # print(res_txt)
        print("OKAY")
        # cv2.imshow("encoded", bgr)
        # cv2.imshow("watermark", res_img)
        # cv2.waitKey(0)

def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--wm', type=str, default='WATERMARK')
    parser.add_argument('--image', type=str, default='download.jpg')
    parser.add_argument('--mode', type=str, default='encode')
    parser.add_argument('--output', type=str, default='output.jpg')
    opt = parser.parse_args()
    return opt

if __name__ == '__main__':
    opt = parse_opt()
    main(**vars(opt))
```

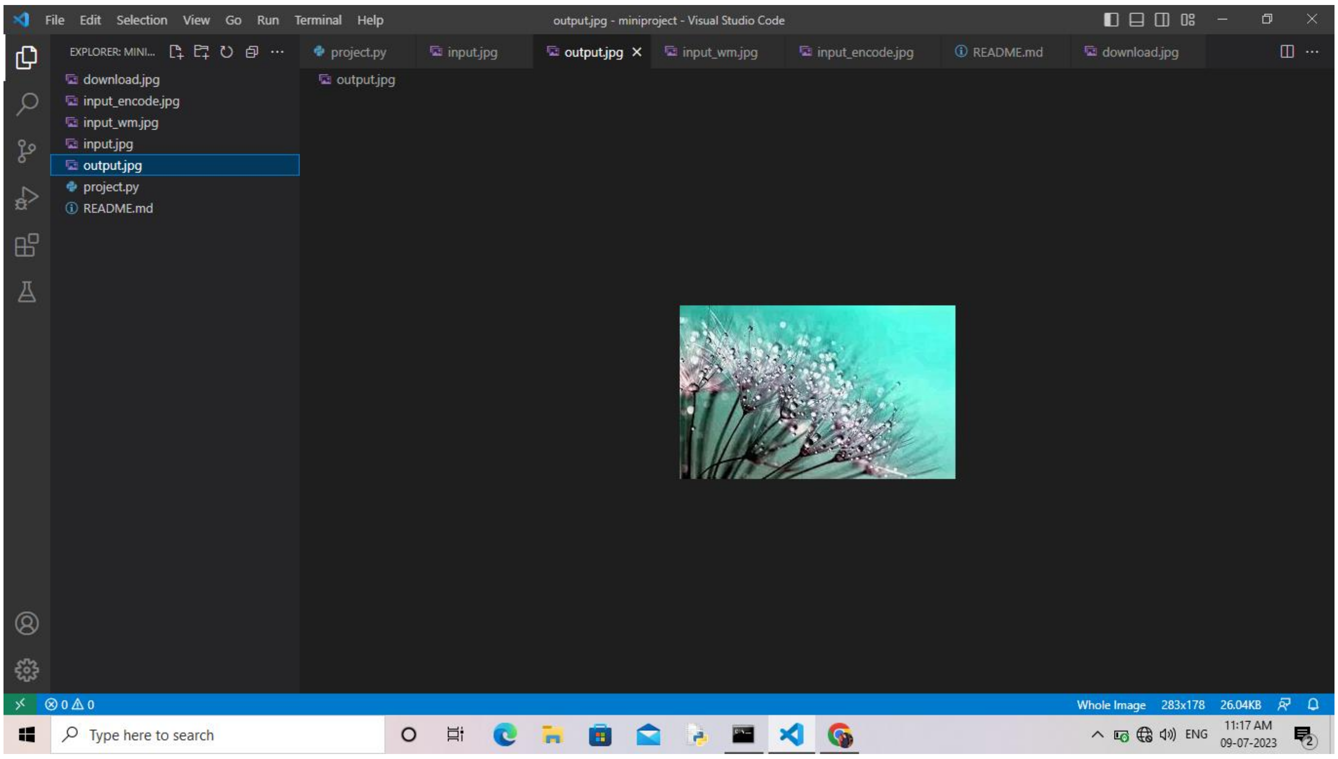
INPUT IMAGE:



WATERMARK IMAGE:



OUTPUT IMAGE:



8.CONCLUSION

A watermarking system's main goal is to add a covert, long-lasting watermark on digital files. Invisible picture watermarking technology is a growing discipline in the study of communications, signal processing, computer science, cryptology, and signal analysis. We have discussed the algorithms for watermarking and dewatermarking photos as part of the project. The study of watermarking is more fascinating since it necessitates the application of interdisciplinary concepts from other fields, such as computer graphics, multimedia, and human psychovisual analysis. The watermark has its own uses because it cannot be seen. To ensure the watermark's security, it should be put into randomly selected regions within a specific domain of the watermark signal. This makes removing the watermark difficult. By adding an invisible watermark on the image, it helps to decrease copying and piracy.

9.FUTURE SCOPE

Each application has advantages and disadvantages of its own. The application can yet be improved, making the website run in a far more appealing and practical way than it does now. The necessity for safe and tamper-proof picture security grows more critical as technology develops and digital content becomes more common. Researchers and programmers can work on improving the resilience, attack resistance, and invisibility of current watermarking techniques or creating new ones. In order to produce more effective and secure watermarking, this may include investigating cutting-edge techniques like deep learning, artificial intelligence, or machine learning, using watermarking methods to accommodate multimedia content outside than just traditional image formats, such as films, audio files, 3D models, and augmented reality (AR) applications. The demand for security and authentication of numerous digital media formats will be satisfied by this growth.

10. BIBILOGRAPHY

1.Methodologies in Digital Watermarking: Robust and Reversible Watermarking Techniques for Authentication, Security and Privacy Protection by Xin Cindy Guo

2. Digital Watermarking: A Tutorial, Dr. Vipula Singh

3. Selective Region Based Invisible Watermarking Using Asymmetric Key Encryption, Somenath Nag Choudhury CSE Department

4. Security and Robustness Enhancement of Digital Image Watermarking by Chittaranjan Pradhan

5. Watermarking of Digital Images by Saraju Prasad Mohanty

Reference links:

https://www.researchgate.net/publication/228450058_DIGITAL_WATERMARKING_A_TECHNOLOGY_OVERVIEW

https://www.ijsdr.org/papers/IJSDR1903002.pdf

https://rcciit.org/students_projects/projects/ece/2018/GR5.pdf

https://www.researchgate.net/publication/327546956_Application_of_invisible_image_watermarking

https://www.researchgate.net/publication/221231679_An_Invisible_Text_Watermarking_Algorithm_using_Image_Watermark