**The Project entitled**

**DIGITAL IMAGE WATERMARKING**

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

Bachelor of Engineering (Computer Science and Engineering)

**By**

**V. SAIKRISHNA (1608-14-733-099)**

**Y.ANEESH JAGPREET (2451-14-733-179)**

**N.KISHAN NAIK (2451-14-733-174)**

****

**Department of Computer Science and Engineering**

**M.V.S.R. ENGINEERING COLLEGE**

**(Affiliated to Osmania University & Recognized by AICTE)**

**Nadergul, Saroor Nagar Mandal, Hyderabad – 501 510**

**2016-17.**

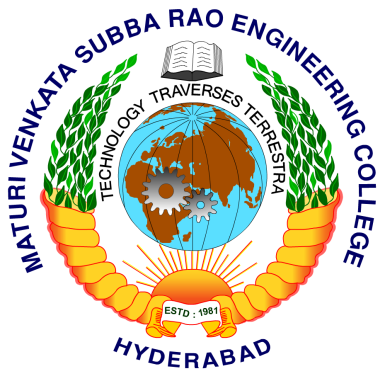
**Department of Computer Science and Engineering**

**M.V.S.R. ENGINEERING COLLEGE**

**(Sponsored by Matrusri Education Society, Estd. 1980)**

**(Affiliated to Osmania University & Recognized by AICTE)**

**Nadergul, Saroor Nagar Mandal, Hyderabad – 501 510**

****

**CERTIFICATE**

This is to certify that the project entitled **“DIGITAL IMAGE WATERMARKING”,** is being submitted by Mr.**V.SAIKRISHNA** bearing H.T No **1608-14-733-099,** Mr.**Y.ANEESH JAGPREET** bearing H.T No **2451-14-733-179** &Mr.**N.KISHAN NAIK** bearing H.T No **2451-14-733-174** in partial fulfillment of academic requirements for the award of the degree of BACHELOR OF ENGINEERING in COMPUTER SCIENCE AND ENGINEERING from MVSR Engineering College, affiliated to OSMANIA UNIVERSITY, is a record of bonafide work carried out by us under the guidance and supervision of the faculty (CSED). The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of my knowledge and belief.

|  |  |  |
| --- | --- | --- |
| Guide |  | Project Coordinator |
| **P.PHANI PRASAD** |  | **B.SANDHYA** |
| Assistant Professor, |  | Associate Professor |
| Dept. of CSE. |  | Dept. of CSE. |
| MVSR Engineering College |  | MVSR Engineering College |

**ACKNOWLEDGMENT**

I take this opportunity to express my profound and sincere gratitude to all those who helped us in carrying out this project successfully.

At the very outset, I am thankful to our principal Sri **Dr. V.CHANDRA SEKHAR** and Sri **Dr. AKHIL KHARE**, Professor and Head, Department of Computer Science and Engineering, MVSR Engineering College, Hyderabad for their consent to do the project work as a part of our B.E Degree (CSE). We thank them for their valuable suggestions and advice throughout our stay at the college, during our project work.

I would like to thank our Internal Project Guide Mr. **P.PHANI PRASAD**, Assistant Professor, Department of Computer Science and Engineering, MVSR Engineering College for her useful suggestions, guidance and encouragement.

We thank the teaching and non-teaching staff of CSE for extending their support.

Finally we are thankful to our parents for their cooperation and support throughout all endeavors in our life.

V.SAIKRISHNA (1608-14-733-099)

Y.ANEESH JAGPREET (2451-14-733-179)

N.KISHAN NAIK (2451-14-733-174)

**ABSTRACT**

**DIGITAL IMAGE WATERMARKING APPLICATION** is a java based app developed based on YOUNG’S WATERMARKING ALGORITHM. This app is developed for the purpose of adding watermarks into images. The image for which the watermark is added i.e., watermarked image will get authorized to the particular organization which has the

Watermarked logo.

This app is designed based on the principles of java programming language of the ORACLE Company. The app is developed on ECLIPSE IDE and NETBEANS IDE. PAINT, MICROSOFT PICTURE MANAGER, ADOBE PHOTOSHOP were found to be the only system for inserting and accessing the watermark into image regarding authorization. But, these methods are not so secure. Because, the watermark can easily be inserted and removed

Just by using select, cut and delete tools.

For this purpose we have come up with a new idea of developing a watermarking app for the Images. This app helps to not only insert and displaying watermarked image but also to secure it from removing and changes as the watermark is inserted directly at the pixel level. Any ways to remove the watermark will even damage the entire Image which would prevent the Image resource from attacks and insecurity. The logo is filtered to remove unnecessary components, and then it’s compressed to match with pixel colours of the background Image thus making it a secured watermarked image.

**List of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Chapter** | **Pg.no.** |
| **1** | Chapter 1 – **Introduction** | **1** |
| **2** | * 1. – Motivation | **1** |
| **3** | * 1. – Problem Definition | **2** |
| **4** | 1.3 – Objectives | **4** |
| **5** | Chapter 2 **- System Analysis** | **5** |
| **6** | 2.1 – Digital Image Watermarking System | **5** |
| **7** | 2.2 – Object Oriented Programming System | **7** |
| **8** | 2.3 - Requirements | **12** |
| **9** | Chapter 3 - **System Design** | **13** |
| **10** | 3.1 - System Architecture | **13** |
| **11** | 3.2 – Class Diagram | **14** |
| **12** | 3.3 – Use Case Diagram | **15** |
| **13** | 3.4 – Activity Diagram | **16** |
| **14** | Chapter 4 – **Implementation** | **17** |
| **15** | 4.1 – Project Sample Code | **17** |
| **16** | 4.2 - Installation Procedure | **19** |
| **17** | 4.3 - Test Cases | **21** |
| **18** | 4.4 - Output Screens | **22** |
| **19** | Chapter 5 – **Conclusion** | **25** |
| **20** | Future Enchancement | **25** |
| **21** | REFERENCES | **26** |

1. **INTRODUCTION**

**1.1 MOTIVATION**

In recent years, digitization plays a big role in human life as numerous applications in field of engineering, healthcare, communication, documentation and many more. Here, multimedia content like image and video is major content. Therefore, authentication, information security and other various issues are raised with multimedia sources and content. Digital data can be stored efficiently and with a very high quality, and it can be manipulated very easily using Computers. Furthermore, digital data can be transmitted in a fast and inexpensive way through data communication networks without losing quality. Digital media offer several distinct advantages over analogue media. The quality of digital audio, images and video signals are better than that of their analogue counterparts. Editing is easy because one can access the exact discrete locations that need to be changed. Copying is simple with no loss of fidelity and a copy of a digital media is identical to the original.

The above problem can be solved using digital Image watermarking. Digital Image watermarking is the act of hiding a message related to a digital image (i.e. an image, song, and video) within the signal itself. It is a concept closely related to steganography, in that they both hide a message inside a digital Image. However, what separates them is their goal. Watermarking tries to hide a message/logo related to the actual content of the digital Image, while in steganography the digital signal has no relation to the message, and it is merely used as a cover to hide its existence.

Watermarking has been around for several centuries, in the form of watermarks found initially in plain paper and subsequently in paper bills. However, the field of digital watermarking was only developed during the last 15 years and it is now being used for many different applications.

**1.2 PROBLEM DEFINITION**

**Classification of watermarking**

**Visible watermark**

Visible watermarking technique generate a visible logo or symbol that clearly seen on watermarked image. This type of watermark used for show the ownership of content like TV channel.

**Invisible Watermark**

This type of watermark is used to find the ownership as well as prevention from authorized application of image or content. Here, a watermark can insert information into an image which cannot be seen, but can be interrogated with the watermark extraction algorithm.

**Robust Watermark**

Robustness watermarking scheme is used for sign copyright information of the digital works, the embedded watermark can resist the common edit processing and various attacks.

**Fragile Watermark**

Fragile watermarking is mainly used for integrity protection, which must be very sensitive to the changes of signal. It can be determined whether the data has been tampered according to the state of fragile watermarking.

**Semi fragile Watermark**

Semi fragile watermarking is capable of tolerating some degree of the change to a watermarked image, such as the addition of quantization noise compression attacks.

**Invisible-Robust Watermark**:

The invisible-robust watermark is embedding in such a way that processes made to the pixel level; which are perceptually not determine and it can be recovered only with appropriate decoding process.

**Invisible-Fragile Watermark**

The invisible-fragile watermark is embedded in such a way that any attacks of the image would alter or destroy the watermark [1].

**Stages in watermarking**

Digital Watermarking is a technique which is used in the digital signal processing of embedding hidden information into multimedia data. This information is not usually visible, only dedicated detector or extractor can seen and extracts that information. Digital Image Watermarking use digital image for embedding the hidden information, after embedding the watermarked image is generated and the watermarked image is more robust against attacks. Figure 1 shows the stages of digital watermarking. Basically working of digital image watermarking can be divided in three stages:

**Embedding Stage**

The embedding stage is the first stage in which the watermark is embedded in the original image by using the embedding algorithm and the secret key. Then the watermarked image is generated. So the watermarked image is transmitted over the network.

**Distortion/Attack Stage**

In this stage, when the data is transmitted over the network. Either some noise is added with the watermarked image or some attacks are performed on the watermarked image. So, our watermarked data is either modified or destroyed.

**Detection/Retrieval Stage**

In the detection stage, the watermark is detected or extracted by the dedicated detector from the watermarked image by applying some detection algorithm and by using secret key. In addition to this, noise is also detected.

**1.3 OBJECTIVES**

**Watermarking Approaches**

There are various algorithms present in the today scenario that are used to hide the information. Those algorithms come into two domains, Spatial and Frequency domain.

**Spatial Domain**

Spatial domain digital watermarking algorithms directly load the raw data into the original image. Spatial watermarking can also be applied using color separation. In this way, the watermark appears in only one of the colour bands. This renders the watermark visibly subtle such that it is difficult to detect under regular viewing. Spatial domain is manipulating or changing an image representing an object in space to enhance the image for a given application. Techniques are based on direct manipulation of pixels in an image. Some of its main algorithms are as discussed below:

**Additive Watermarking**

The most straightforward method for embedding the watermark in spatial domain is to add pseudo random noise pattern to the intensity of image pixels. The noise signal is usually integers like (-1, 0, 1) or sometimes floating point numbers. To ensure that the watermark can be detected, the noise is generated by a key, such that the correlation between the numbers of different keys will be very low.

**Least Significant Bit**

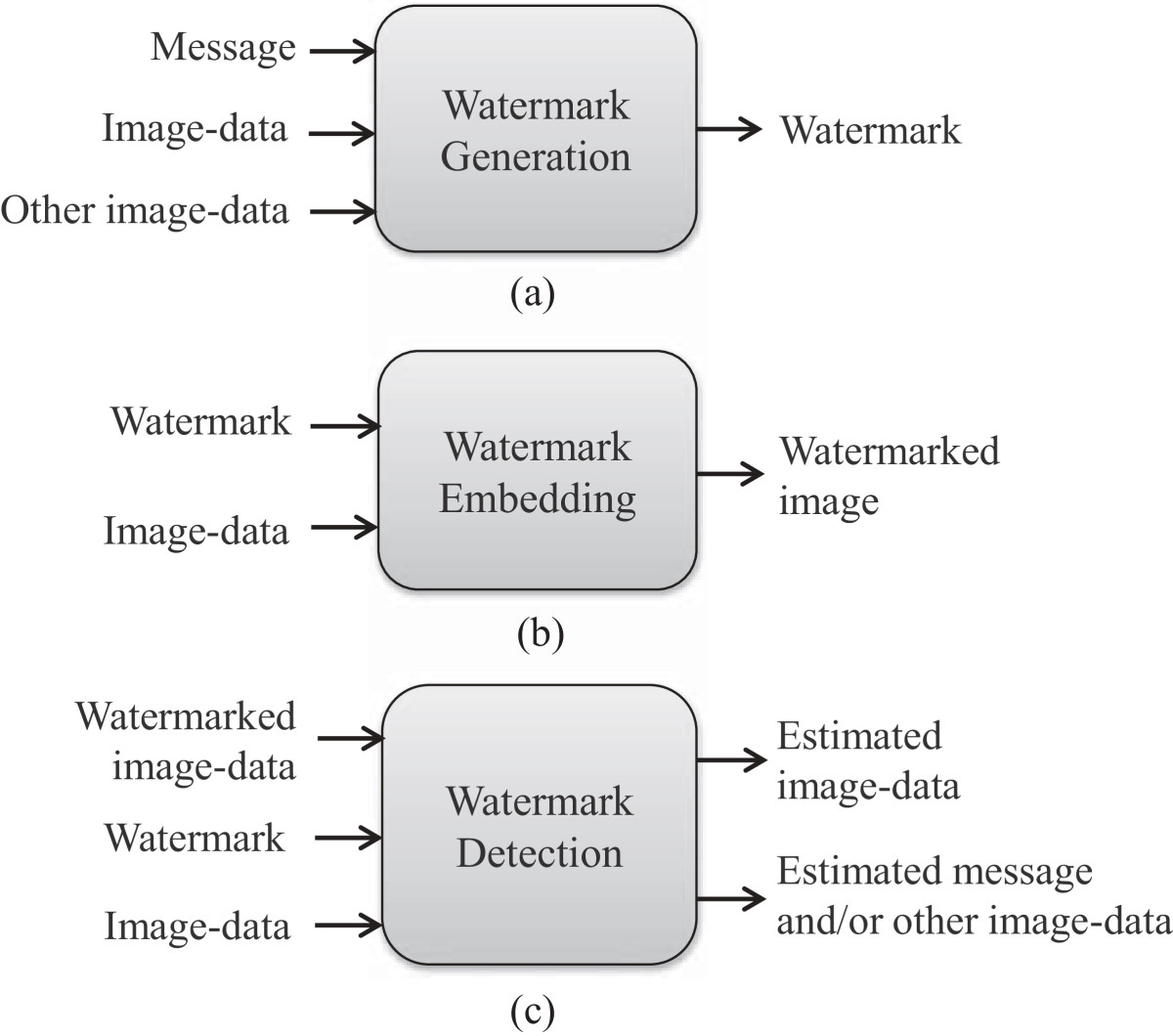
Old popular technique embeds the watermark in the LSB of pixels. This method is easy to implement and does not generate serious distortion to the image; however, it is not very robust against attacks. The embedding of the watermark is performed choosing a subset of image pixels and substituting the least significant bit of each of the chosen pixels with watermark bits. The watermark may be spread throughout the image or may be in the select locations of the image. But these primitive techniques are vulnerable to attacks and the watermark can be easily destroyed. Such an approach is very sensitive to noise and common signal processing and cannot be used in practical applications.

SSM Modulation Based Technique: Spread-spectrum techniques are methods in which energy generated at one or more discrete frequencies is deliberately spread or distributed in time. SSM based watermarking algorithms embed information by linearly combining the host image with a small pseudo noise signal that is modulated by the embedded watermark.

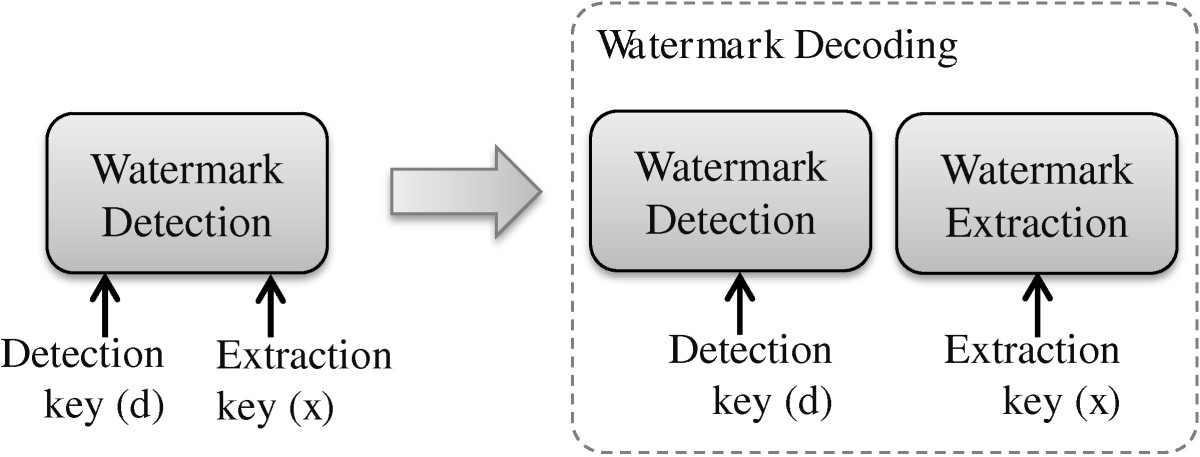
**2. SYSTEM ANALYSIS**

**2.1 DIGITAL IMAGEWATERMARKING SYSTEM**

A basic model, as it implies, is expected to represent a basic scenario for the image watermarking applications. We firstly identify the fundamental components and their possible inputs and outputs of a watermarking scheme. Irrespective of the system and security requirements, a watermarking scheme can have three fundamental components as shown in Figure . In order for their systematic definition, we consider three functions: watermark generation, *G* (·), embedding, *E* (·), and detection, *D* (·), and define their possible inputs and outputs as shown in Table . The primary roles of these functions in an image watermarking application are described below. To denote different data (e.g., inputs and outputs) within this context, in what follows, plain letters indicate the original versions, and respective single-bar letters and tilde letters indicate their watermarked and estimated versions accordingly.



To adopt and generalize the use of keys, we extend the basic scenario to a key-based scenario. We assume two individual keys, generation key, *g* and embedding key, *e* for *G* and *E*, respectively. Although in our basic construction, for simplicity, *D*(·) is considered to perform the *detection* and *extraction* tasks inherently, this should naturally be split into separate functions for security reasons. We, therefore, separate the computation of extraction from *D*(·) using an additional function *X*(·), which we call the extraction function. Thus, an individual detection key, *d* and extraction key, *x* can be used as shown in Figure [3](http://asp.eurasipjournals.springeropen.com/articles/10.1186/1687-6180-2014-135#Fig3). These two functions, *D*(·) and *X*(·) can be further defined as sub-functions of watermark *decoding* (to resemble our earlier construction) as shown in Table [3](http://asp.eurasipjournals.springeropen.com/articles/10.1186/1687-6180-2014-135#Tab3). The other two functions, *G*(·) and *E*(·) can similarly be the sub-functions of watermark *encoding*. Figure [2](http://asp.eurasipjournals.springeropen.com/articles/10.1186/1687-6180-2014-135#Fig2) illustrates the watermark encoding and decoding processes.



**2.2 OBJECT ORIENTED PROGRAMMING SYSTEM USING JAVA**

**Object** means a real word entity such as pen, chair, table etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

* Object
* Class
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

**Object**

Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

**Class**

**Collection of objects** is called class. It is a logical entity.

**Inheritance**

**When one object acquires all the properties and behaviours of parent object** i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

**Polymorphism**

When **one task is performed by different ways** i.e., known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.

In java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something e.g. cat speaks meow, dog barks woof etc.

**Abstraction**

**Hiding internal details and showing functionality** is known as abstraction. For example: phone call, we don't know the internal processing.

In java, we use abstract class and interface to achieve abstraction.

**Encapsulation**

**Binding (or wrapping) code and data together into a single unit is known as encapsulation**. For example: capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

**Advantage of OOPs over Procedure-oriented programming language**

|  |
| --- |
| 1) OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows. |
| 2) OOPs provides data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere. |
| 3) OOPs provides ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language. |

|  |  |
| --- | --- |
| Global Data | Object Data |

**What is difference between object-oriented programming language and object-based programming language?**

Object based programming language follows all the features of OOPs except Inheritance. JavaScript and VBScript are examples of object based programming languages.

**Java I/O** (Input and Output) is used *to process the input* and *produce the output*.

Java uses the concept of stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

We can perform **file handling in java** by Java I/O API.

**Stream**

A stream is a sequence of data.In Java a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.

In java, 3 streams are created for us automatically. All these streams are attached with console.

**1) System.out:** standard output stream

**2) System.in:** standard input stream

**3) System.err:** standard error stream

**Java BufferedInputStream Class**

Java BufferedInputStream class is used to read information from stream. It internally uses buffer mechanism to make the performance fast.

The important points about BufferedInputStream are:

* When the bytes from the stream are skipped or read, the internal buffer automatically refilled from the contained input stream, many bytes at a time.
* When a BufferedInputStream is created, an internal buffer array is created.

**Java BufferedInputStream class declaration**

Let's see the declaration for Java.io.BufferedInputStream class:

1. public class BufferedInputStream extends FilterInputStream

**Java BufferedInputStream class constructors**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **BufferedInputStream(InputStream IS)** | It creates the BufferedInputStream and saves it argument, the input stream IS, for later use. |
| **BufferedInputStream(InputStream IS, int size)** | It creates the BufferedInputStream with a specified buffer size and saves it argument, the input stream IS, for later use. |

**Java BufferedInputStream class methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| **int available()** | It returns an estimate number of bytes that can be read from the input stream without blocking by the next invocation method for the input stream. |
| **int read()** | It read the next byte of data from the input stream. |
| **int read(byte[] b, int off, int ln)** | It read the bytes from the specified byte-input stream into a specified byte array, starting with the given offset. |
| **void close()** | It closes the input stream and releases any of the system resources associated with the stream. |
| **void reset()** | It repositions the stream at a position the mark method was last called on this input stream. |
| **void mark(int readlimit)** | It sees the general contract of the mark method for the input stream. |
| **long skip(long x)** | It skips over and discards x bytes of data from the input stream. |
| **markSupport()** | It tests for the input stream to support the mark and reset methods. |

**Java BufferedOutputStream Class**

Java BufferedOutputStream class is used for buffering an output stream. It internally uses buffer to store data. It adds more efficiency than to write data directly into a stream. So, it makes the performance fast.

For adding the buffer in an OutputStream, use the BufferedOutputStream class. Let's see the syntax for adding the buffer in an OutputStream:

**OutputStream os= new BufferedOutputStream(new FileOutputStream("D:\\IO Package\\testout.txt"));**

**Java BufferedOutputStream class declaration**

Let's see the declaration for Java.io.BufferedOutputStream class:

**public class BufferedOutputStream extends FilterOutputStream**

**Java BufferedOutputStream class constructors**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **BufferedOutputStream(OutputStream os)** | It creates the new buffered output stream which is used for writing the data to the specified output stream. |
| **BufferedOutputStream(OutputStream os, int size)** | It creates the new buffered output stream which is used for writing the data to the specified output stream with a specified buffer size. |

**Java BufferedOutputStream class methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| **void write(int b)** | It writes the specified byte to the buffered output stream. |
| **void write(byte[] b, int off, int len)** | It write the bytes from the specified byte-input stream into a specified byte array, starting with the given offset |
| **void flush()** | It flushes the buffered output stream. |

**2.3 REQUIREMENTS**

Table 2.3.1 **Hardware Requirements and Tools**

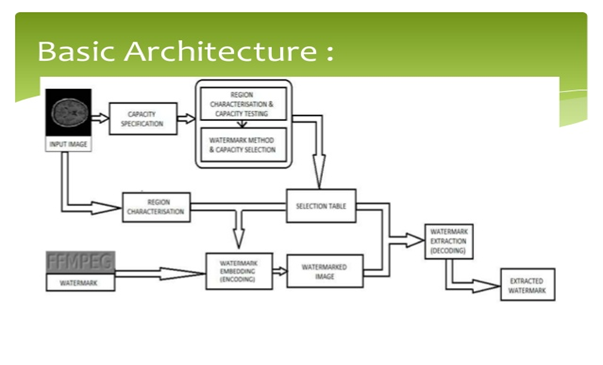
|  |  |
| --- | --- |
| Development tools | NOTEPAD++ or Edit++ |
| Operating system | Windows XP Profession Edition, Windows Vista Enterprise Edition, Windows 7 and latest versions. |
| Web browser | Internet Explorer 6.0 and above |
| Hardware | intel, 1 GB RAM, 80 GB HDD |

Table 2.3.2 **Software Requirements**

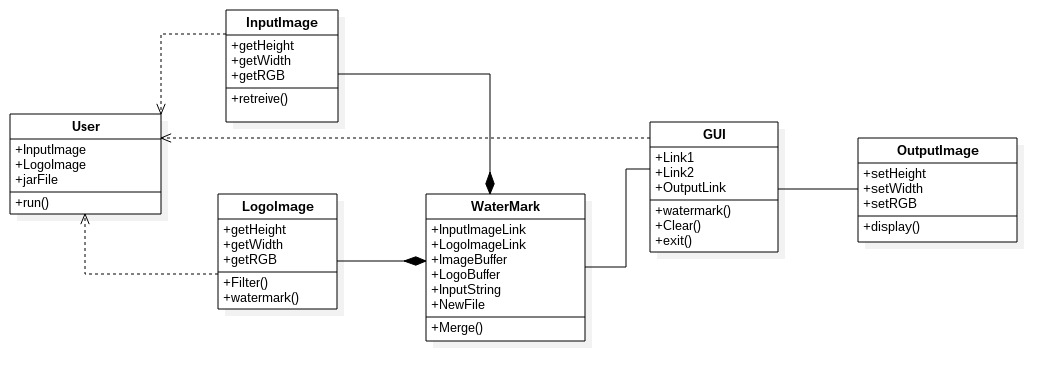
|  |  |
| --- | --- |
| Integrated Development Environment(IDE) | Net beans ,eclipse |
| For G.U.I Development | Net Beans GUI development package |

**3. SYSTEM DESIGN**

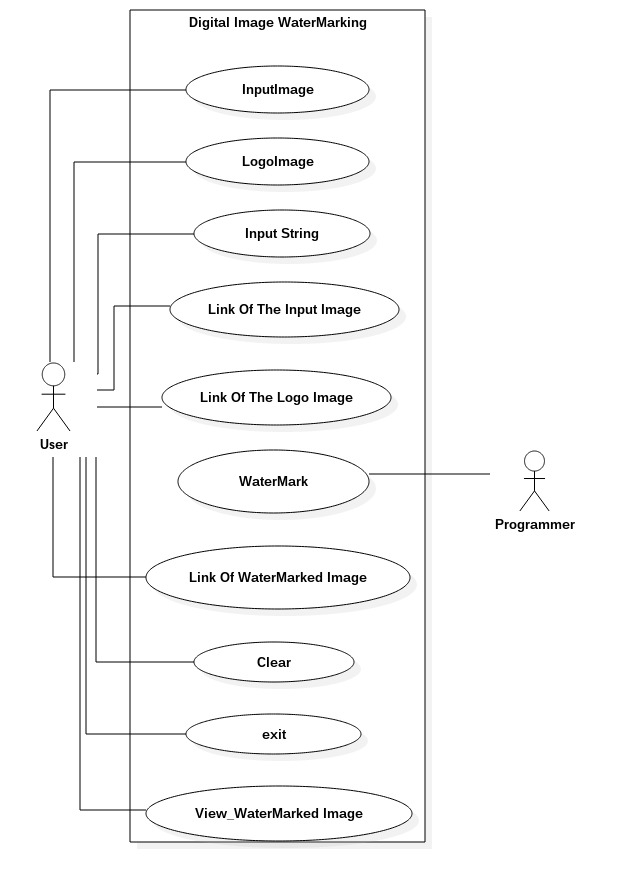
**3.1 SYSTEM ARCHITECHTURE**

****

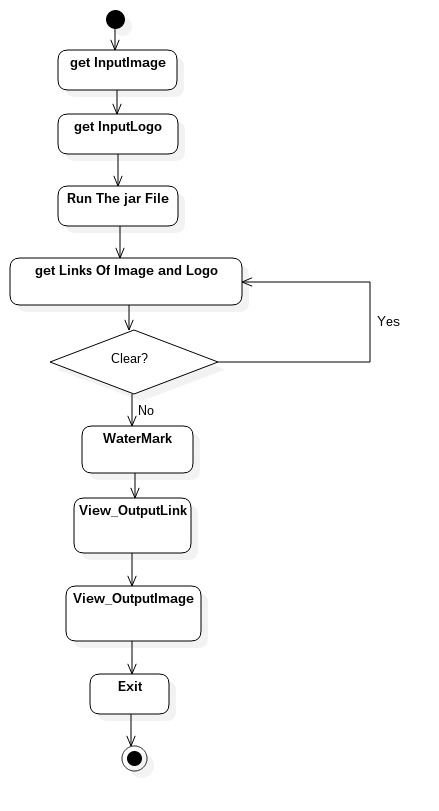
**3.2 CLASS DIAGRAM**

****

**3.3 USE CASE DIAGRAM**

****

**3.4 ACTIVITY DIAGRAM**



**4. IMPLEMENTATION**

**4.1: PROJECT SAMPLE CODE:**

***BACKEND:***

import java.awt.Font;

import java.awt.Graphics;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;

public class WatermarkLogo {

public static void main(String[] args) {

File img = new File("C:/OrignalImage.jpg");

ImageIcon icon = new ImageIcon(img.getPath());

File logo = new File("D:/logo.jpg");

ImageIcon logo\_icon = new ImageIcon(logo.getPath());

BufferedImage bufferedImage = new BufferedImage(icon.getIconWidth(),icon.getIconHeight(), BufferedImage.TYPE\_INT\_RGB);

Graphics graphics = bufferedImage.getGraphics();

graphics.drawImage(icon.getImage(), 0, 0, null);

graphics.setFont(new Font("Arial", Font.BOLD, 30));

String title = "MVSR ENGINEERING WATERMARKED IMAGE";

graphics.drawString(title, 0, icon.getIconHeight() / 2);

BufferedImage bufferedLogo = new BufferedImage(logo\_icon.getIconWidth(),logo\_icon.getIconHeight(),BufferedImage.TYPE\_INT\_RGB);

Graphics logo\_graphics = bufferedLogo.getGraphics();

graphics.drawImage(logo\_icon.getImage(),100,100,null);

graphics.dispose();

File newFile = new File("C:/output\_WatermarkedImage.jpg");

try {

ImageIO.write(bufferedImage, "jpg", newFile);

}

catch (IOException e)

{

e.printStackTrace();

}

System.out.println(newFile.getPath() + " WATERMARKED IMAGE created successfully!");

}

**4.2 INSTALLATION PROCEDURE:**

In Installation procedure first we have to know what technologies we are using in project, in Digital Image Watermarking project the technologies are backend-Java and frontend- swings

from java.

First we need to install jdk 1.4 or higher; in this we can execute java programs like applets and swings...Etc.

After installing java, set environment variables.

**Steps to set environment variables**

1. Select Start, select Control Panel. Double click System, and select the advanced tab.
2. Click Environment Variables. ...
3. In the Edit System Variable (or New System Variable) window, specify the value of the PATH environment variable. ...
4. Reopen Command prompt window, and run your java code.

**To create a NETBEANS IDE project:**

1. Start Net Beans IDE.
2. In the IDE, choose File > New Project, as shown in the figure below.
3. Net Beans IDE with the File > New Project menu item selected.
4. In the New Project wizard, expand the Java category and select Java Application as shown in the figure below. Then click next.
5. New Project wizard: Choose Project
6. In the Name and Location page of the wizard, do the following (as shown in the figure below):
7. In the Project Name field, type Digital Image Watermarking APP.
8. Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.
9. In the Create Main Class field, type helloworldapp.HelloWorldApp.

10. New Project wizard: Name and Location

11. Click Finish.

**Compiling and Running the Program**

Because of the IDE's Compile on save feature, you do not have to manually compile your project in order to run it in the IDE. When you save a Java source file, the IDE automatically compiles it.

The Compile on save feature can be turned off in the Project Properties window. Right-click your project, select Properties. In the Properties window, choose the Compiling tab. The Compile on save checkbox is right at the top. Note that in the Project Properties window you can configure numerous settings for your project: project libraries, packaging, building, running, etc.

**To run the program:**

* Choose Run > Run Project.

**Building and Deploying the Application**

Once you have written and test runs your application, you can use the Clean and Build command to build your application for deployment. When you use the Clean and Build command, the IDE runs a build script that performs the following tasks:

* Deletes any previously compiled files and other build outputs.
* Recompiles the application and builds a JAR file containing the compiled files.

**To build your application:**

* Choose Run > Clean and Build Project.

**To run the complete Project:**

1. You can view the build outputs by opening the Files window and expanding the Digital Image Watermarking App node.
2. The compiled byte code file ImageDisplayApp.class is within the build/classes/ Digital Image Watermarking app sub node.
3. A deployable JAR file that contains the Digital Image Watermarking App.class is within the dist. node.

**4.3 TEST CASES**

The following test-cases being mentioned here are done from starting of the application to its ending on ECLIPSE IDE.

The functioning of the app and the data information being entered into the app as well as the output obtained are thoroughly tested. Following are the outcomes of the tests done:

|  |  |  |
| --- | --- | --- |
| **DESCRIPTION** | **EXPECTED OUTCOMES** | **ORIGINAL OUTCOME** |
| Clicking on the ImageDisplay.jar file of the application | The application opens normally as any other java app. | The app opened as usual as other applications. |
| Presence of labels and text fields which will display the links of IMAGE, LOGO AND OUTPUT. | The text fields will be visible on the screen with names and small descriptions | The text fields will be visible on the screen with names and small descriptions which will display the links of IMAGE, LOGO AND OUTPUT. |
| Presence of buttons like clear  And other default buttons like watermark. | Clicking the buttons will enable you to clear the text fields and clicking watermark button shows the link of the output | As expected buttons helped in the convenience of app function |
| Pressing the exit button on the app GUI | The app runs in the background. But, GUI gets stopped. | As expected app runs in background and GUI gets stopped. |

**4.4 SCREENSHOTS**

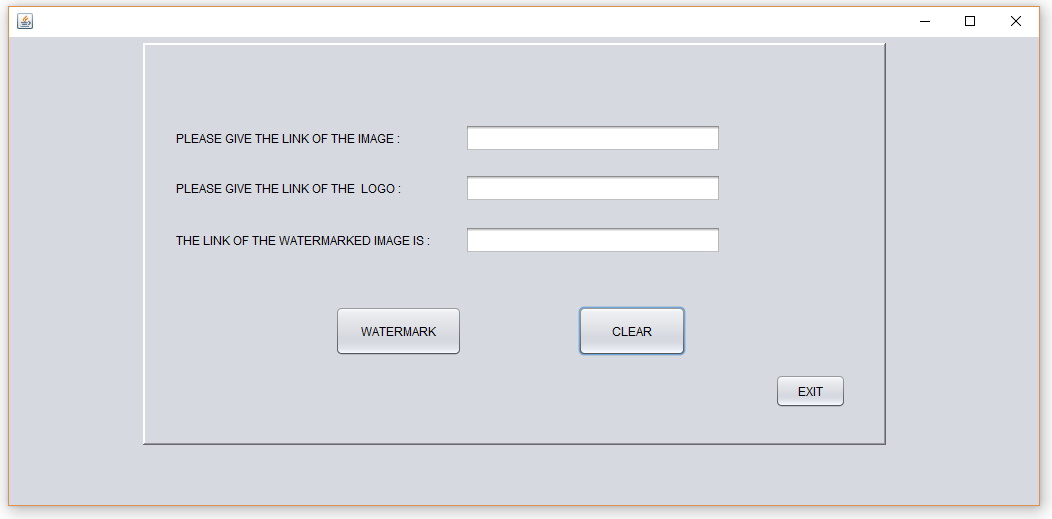


Fig 4.4.1: Image Display.jar window (displays buttons and text fields).

Screenshot and test case of a layout of the app “ImageDisplay.jar”. This screen shows the labels asking for links of image, logo and a label which will provide the link of watermarked image.

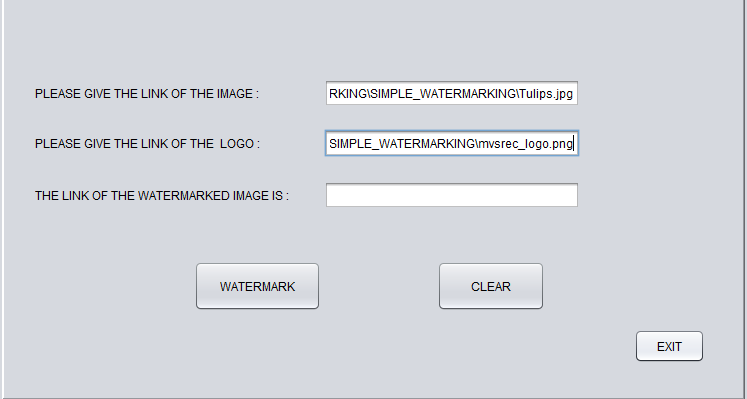
****

Fig 4.4.2: Sample layout of links of image and logo being entered into the app.

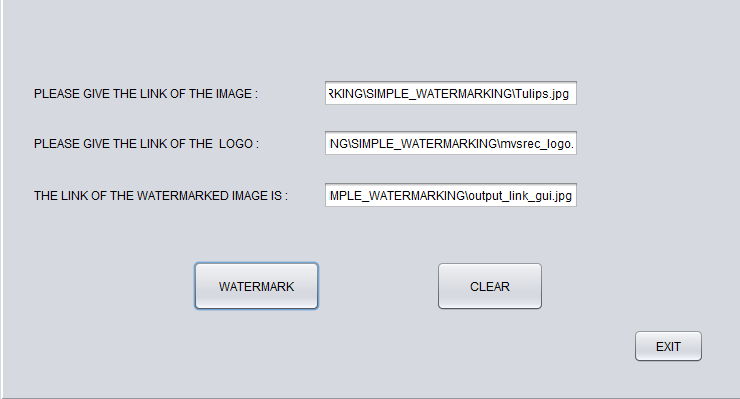
****

Fig 4.4.3: This image shows the link of the output image displayed after watermark button is pressed.



Fig 4.4.4: This is the input image “Tulips.jpg”. This image has no watermark .The link of it C:\Users\RDRL\Desktop\DIGITAL\_WATER\_MARKING\SIMPLE\_WATERMARKING\Tulips.jpg



Figure 4.4.5: MVSR ENGINEERING COLLEGE LOGO

C:\Users\RDRL\Desktop\DIGITAL\_WATER\_MARKING\SIMPLE\_WATERMARKING\mvsrec\_logo.jpg



Figure 4.4.6: The output Watermarked Image.

C:\Users\RDRL\Desktop\DIGITAL\_WATER\_MARKING\SIMPLE\_WATERMARKING\output\_link\_gui.png

**5. CONCLUSION**

In last few years, Digital watermarks have thus helped us to protect the ownership of digital data. In this paper, classification of watermarking, different stages in watermarking, several techniques based on spatial domain and frequency domain and its applications were discussed.

Digital watermarking scheme is widely utilized for authentication of data, copyright protection and communication process. It provides a consistent robust performance on different original image and watermarked image in various analyses.

**Watermarking applications**

**Ownership Assertion**

Protects the ownership rights.

**Fingerprinting**

To avoid unauthorized duplication of copies and its distribution.

**Authentication and Integrity Verification**

A unique key is used to embed and extract, this verifies the integrity of the system.

**Content Labelling**

Extra information like date, place etc can be added.

**Usage Control**

Only a limited number of copies can be created.

**Content Protection**

Visible watermark makes it very difficult to modify the contents.

**FUTURE ENHANCEMENT**

This app can be developed further into adding more security aspects like inserting invisible watermarks, passwords, linking layouts with one another, developing into multi window java application file so that it can be accessed for many image processing works like grey scale conversion, logo compression, logo filtering, image compression etc.

.

**REFERENCES**

1. <http://www.tutorialspoint.com/java_dip/> for learning DIGITAL IMAGE PROCESSING USING JAVA.
2. Java The Complete Reference, Seventh Edition for programming in JAVA.
3. Software engineering principles-roger s pressman.
4. <https://netbeans.org/kb/docs/java/quickstart.html> for learning how to work on NETBEANS IDE.
5. <https://en.wikipedia.org/wiki/Digital_watermarking>
6. <https://netbeans.org/kb/docs/java/gui-functionality.html>
7. <https://netbeans.org/kb/docs/java/quickstart-gui.html>
8. [www.mvsrec.edu.in](http://www.mvsrec.edu.in) for logo.
9. <https://tjkelly.com/wp-content/uploads/windows-xp-desktop-background-wallpaper-tulips-800x600.jpg> for Tulips image.
10. <http://www.javatpoint.com/java-oops-concepts> for documentation on OOPS.