VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi- 590 018

PROJECT REPORT

ON

"IMAGE STEGANOGRAPHY USING QUANTUM TECHNOLOGY"

Submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Under the Guidance of

Mr. MANIKANTA PRASAD J.B.E., M.Tech.
Assistant Professor,
Department of Computer Science & Engineering
Adichunchanagiri Institute of Technology
Chikkamagaluru

Submitted by

DIWAKAR N **(4AI21CS029)**

RATHAN M E **(4AI21CS076)**

A N KARANJITH **(4AI21CS001)**

MOHAMMED ASIM FARHAN **(4AI21CS057)**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY
(Affiliated to V.T.U., Accredited by NAAC)
CHIKKAMAGALURU-577102, KARNATAKA
2024- 2025

ADICHUNCHANAGIRI INSTITUTEOF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi)

Chikkamagaluru, Karnataka, India-577102.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project work (21CSP76) entitled **"IMAGE STEGANOGRAPHY USING QUANTUM TECHNOLOGY"**is a bonafide work carried out by **DIWAKAR N (4AI21CS029), RATHAN M E (4AI21CS076), A N KARANJITH (4AI21CS001),MOHAMMED ASIM FARHAN (4AI21CS057).** students of 7th semester B.E, in partial fulfilment for the award of Degree of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the academic year **2024-2025**. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The project report has been approved, as it satisfies the academic requirements in respect of Project Work prescribed for the said Degree.

Signature of the Guide
Mr. Manikanta Prasad.J **B.E., M.Tech.**
Assistant Professor
Dept. of CS&E
A.I.T, Chikkamagaluru

Signature of the Project Coordinator
Mrs. Shruthi G K **B.E., M.Tech**
Assistant Professor
Dept. of CS&E
A.I.T, Chikkamagaluru

Signature of the Project Coordinator
Mrs. Mithuna B N **B.E., M.Tech**
Assistant Professor
Dept. of CS&E
A.I.T, Chikkamagaluru

Signature of the HOD
Dr.PushpaRavikumar **B.E., M.Tech., Ph.D.,LMISTE**
Professor & Head
Dept. of CS&E
A.I.T, Chikkamagaluru

Signature of the Principal
Dr. C.T Jayadeva **B.E., M.Tech., Ph.D**
Principal
A.I.T, Chikkamagaluru

External examiner	**Signature with Date**
1. ______________________	______________________
2. ______________________	______________________

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi)
Chikkamagaluru, Karnataka, India – 577102.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

APPROVAL

The **Project Report (21CSP76)** entitled **"IMAGE STEGANOGRAPHY USING QUANTUM TECHNOLOGY"** is here by approved as a credible study of Engineering subject carried out and presented in a satisfactory manner for acceptance as a pre-requestee to the Degree of **BACHELOR OF ENGINEERING** in **COMPUTER SCIENCE AND ENGINEERING** during the academic year 2024- 25.

Submitted By:

Diwakar N	**(4AI21CS029)**
Rathan M E	**(4AI21CS076)**
A N Karanjith	**(4AI21CS001)**
Mohammed Asim Farhan	**(4AI21CS057)**

Signature of the Guide
Mr. Manikanta Prasad.J B.E., M.Tech.
Assistant Professor
Dept. of CS&E
A.I.T, Chikkamagaluru

Signature of the Project Coordinator
Mrs. Shruthi G K B.E., M.Tech
Assistant Professor
Dept. of CS&E
A.I.T, Chikkamagaluru

Signature of the Project Coordinator
Mrs. Mithuna B N B.E., M.Tech
Assistant Professor
Dept. of CS&E
A.I.T, Chikkamagaluru

Signature of the HOD
Dr.PushpaRavikumar B.E., M.Tech., Ph.D.,LMISTE
Professor & Head
Dept. of CS&E
A.I.T, Chikkamagaluru

ABSTRACT

Traditional image steganography techniques, such as Least Significant Bit (LSB) encoding, while simple and effective, suffer from several disadvantages. These methods are vulnerable to detection through steganalysis techniques, which can identify the altered pixels and expose the hidden data. Additionally, the limited capacity for data embedding and the predictable structure of these methods make them susceptible to attacks. As digital communication grows, the need for more advanced, secure, and complex steganography methods is critical to protect sensitive information from unauthorized access and interception.

This project addresses these limitations by introducing a quantum-inspired image steganography approach that leverages principles of quantum randomness and superposition to enhance the security of hidden data. Using a Flask-based web application, users can upload images and embed secret messages using quantum-like transformations, which randomize pixel alterations in a more unpredictable manner. The code uses Python's Pillow library for image processing and applies randomized bit manipulation to encode data securely, making it difficult for attackers to detect or extract the hidden message. This approach significantly improves the robustness of image steganography against modern steganalysis techniques.

ACKNOWLEDGEMENTS

We express our humble Pranamas to his holiness **Parama Poojya Jagadguru Padmabushana Sri Sri Sri Dr.Balagangadharanatha Mahaswamiji** and **Parama Poojya Jagadguru Sri Sri Sri Dr. Nirmalanandanatha Mahaswamiji** and also to **Sri Sri Gunanatha Swamiji** Sringeri Branch Chikkamagaluru who have showered their blessings on us for framing our career successfully.

We are deeply indebted to our honourable Director **Dr. C K Subbaraya** for creating the right kind of care and ambience.

We are Thankful to our beloved Principal **Dr. C T Jayadeva** for inspiring us to achieve greater endeavours in all aspects of Learning.

We express our deepest gratitude to **Dr. Pushpa Ravikumar**, Professor & Head of the, Department of Computer Science & Engineering for her valuable guidance, suggestions and constant encouragement without which success of our Project work would have been difficult.

We are Grateful to our project coordinators **Mrs.Shruthi G K and Mrs.Mithuna B N** Asst Professors, Dept. of Computer Science & Engineering for their excellent guidance, constant encouragement, support and constructive suggestions. We are thankful to our guide **Mr.Manikanta Prasad J** Asst. Professor, Dept. of Computer Science & Engineering AIT, Chikkamagaluru for his inspiration and lively correspondence for carrying our project work.

We would like to thank our beloved parents for their support, encouragement and blessings. And last but not least, I would be pleased to express our heartfelt thanks to all teaching and non-teaching staff of CS&E Department and our friends who have rendered their help, motivation and support.

Diwakar N (4AI21CS029)
Rathan M E (4AI21CS076)
A N Karanjith (4AI21CS001)
Mohammed Asim Farhan (4AI21CS057)

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

The Quantum-inspired steganography introduces a revolutionary approach to secure communication by leveraging principles from quantum mechanics, such as randomness and superposition. These concepts enable the development of unpredictable and highly secure data embedding techniques that significantly reduce the likelihood of detection by attackers. Unlike traditional methods, quantum-inspired techniques mimic quantum randomness to randomize pixel alterations, making patterns nearly impossible to identify through steganalysis tools. This project implements these principles using Python’s Pillow library for image manipulation and a Flask-based web application, offering users a seamless way to encode and decode messages. The combination of quantum-like transformations with practical implementation ensures higher levels of security while maintaining the quality and integrity of the original images.

Traditional image steganography methods, such as Least Significant Bit (LSB) encoding, have been widely used due to their simplicity and effectiveness. These techniques embed data into the least significant bits of an image's pixel values, ensuring minimal visual distortion. However, LSB encoding suffers from several limitations, including vulnerability to modern steganalysis tools that can detect predictable patterns in altered pixels. This makes such methods increasingly insecure in the face of advancing computational power and detection algorithms. Additionally, LSB techniques often have limited data capacity, restricting their applicability in scenarios requiring robust and covert data transmission. As a result, the growing sophistication of cyber threats has exposed the need for innovative alternatives that address the weaknesses of traditional methods.

1.1.1 Overview of the Project

The primary objective of this work is to develop a quantum-inspired image steganography technique that enhances the security and robustness of hidden data. Traditional methods like Least Significant Bit (LSB) encoding suffer from vulnerabilities to steganalysis, as their predictable embedding patterns can be easily detected. This project addresses these limitations by integrating quantum-inspired randomness and superposition into the data embedding process, ensuring that the altered pixel patterns are significantly harder to detect. The proposed method uses Python’s Pillow library for image manipulation and a Flask-based web application to create a user-friendly system for encoding and decoding messages. By introducing a quantum-inspired approach, this project ensures higher security while maintaining the visual integrity of the original image. This innovation aims to address the increasing demand for secure data transmission in fields like national security, personal privacy, and financial systems. Additionally, the approach bridges the gap between theoretical advancements in quantum-inspired techniques and their practical implementation in real-world scenarios.

Another key aspect of this work is the emphasis on accessibility and usability. While quantum principles enhance security, the system’s user interface ensures it remains practical for non-technical users. The Flask-based web application allows users to upload images, embed secret messages, and retrieve encoded outputs seamlessly. To achieve this, the project supports multiple image formats such as PNG and JPEG and ensures that the embedded data does not compromise the quality of the image. Additionally, the system incorporates error-checking mechanisms to ensure data integrity, making it reliable for sensitive communication needs. This dual focus on advanced security mechanisms and ease of use makes the proposed system suitable for a variety of applications, including secure communication, covert data transmission, and safeguarding personal privacy against malicious actors. Beyond individual use cases, the system could be extended to corporate or government-level applications, where maintaining the confidentiality of sensitive information is critical.

The methodology involves several stages, starting with the implementation of quantum-inspired randomness to guide the pixel modification process. Images are preprocessed to optimize compatibility and ensure efficient encoding, while Python's Pillow library facilitates secure pixel manipulation. The Flask framework provides a seamless interface for encoding and decoding, ensuring that the system is both robust and user-friendly. Rigorous testing is conducted to evaluate the method’s performance against traditional techniques, with metrics such as embedding capacity, Peak Signal-to-Noise Ratio (PSNR), and resistance to steganalysis used to validate the system’s effectiveness. The inclusion of testing ensures that the solution remains scalable and adaptable to diverse scenarios, providing a reliable benchmark for real-world performance. These stages collectively ensure that the system aligns with the demands of modern-day secure communication while addressing the limitations of existing methods.

Finally, the proposed method was compared with traditional LSB-based techniques to demonstrate its superior performance. The results consistently showed that the quantum-inspired approach offers significantly better resistance to detection, higher data capacity, and improved security. These advantages highlight the robustness and reliability of the method, positioning it as a cutting-edge solution for secure data transmission. Furthermore, the system's adaptability ensures its relevance across diverse domains, from personal communication to highly sensitive military operations. This versatility and performance establish the proposed approach as a significant contribution to the ongoing advancement of steganographic methods, creating new possibilities for secure communication in an increasingly connected world.

This project not only provides a secure and innovative steganographic technique but also ensures practical usability, bridging the gap between advanced cryptographic concepts and real-world applications. By integrating quantum-inspired principles with a user-friendly system, this work contributes to the ongoing development of secure communication technologies in the digital era. The combination of theoretical innovation with practical implementation makes this approach a model for future advancements in data security. In conclusion, the project offers a significant advancement in the field of steganography, addressing the limitations of traditional methods while laying the groundwork for future innovations in data security.

1.2 Motivation

1.2.1 Need for Quantum-Inspired Security

As steganography techniques evolve, attackers continue to develop sophisticated steganalysis methods capable of detecting hidden data in digital media. Traditional approaches, such as Least Significant Bit (LSB) encoding, are vulnerable due to their predictable patterns, making them easily identifiable through modern detection tools. With the rise of interconnected systems and the growing reliance on digital communication, there is an urgent need for more advanced methods that go beyond conventional techniques. Quantum-inspired security leverages principles like randomness and superposition to introduce an unpredictable embedding process, creating a robust defense against detection. Unlike deterministic methods, quantum-inspired approaches randomize pixel modifications in a way that is extremely difficult to predict, even with advanced computational power. This randomness ensures that embedded data remains hidden while maintaining the quality of the original media. By addressing the inherent weaknesses of traditional methods, quantum-inspired steganography provides a next-generation solution for safeguarding sensitive information in an increasingly adversarial digital environment.

1.2.2 Importance of Explainability

The growing complexity of automated systems raises concerns about their lack of transparency and accountability, often referred to as their "black-box" nature. This issue is particularly significant in quantum-inspired steganography, where decisions related to data embedding can have critical consequences. Explainable Artificial Intelligence (XAI) offers a solution by making these processes interpretable and understandable to users. By integrating explainability, this project ensures that users can gain insights into how randomness and quantum principles influence pixel manipulation and data protection. This transparency fosters trust, as users are better equipped to understand the rationale behind the system's security mechanisms. Additionally, explainability facilitates compliance with ethical and regulatory standards, particularly in applications where accountability is essential. It also empowers users to identify and address potential flaws, contributing to the continuous improvement of robust and fair steganographic techniques.

1.2.3 Social Impact

The increasing prevalence of cyber threats, data breaches, and unauthorized surveillance poses significant risks to individual privacy, organizational security, and even national defense. The inability to securely transmit sensitive information can lead to severe consequences, such as identity theft, corporate espionage, and the compromise of classified information. Traditional steganographic techniques, with their susceptibility to detection, are no longer sufficient to address these challenges. A robust quantum-inspired steganography system can play a pivotal role in mitigating these risks by providing a secure and covert means of communication. Such systems not only help protect sensitive information but also restore confidence in digital communication platforms, fostering trust in a world where privacy is increasingly under threat. Furthermore, by enabling the secure transmission of critical data, these systems contribute to the broader goals of societal safety, organizational resilience, and technological innovation in data security.

1.3 Problem Statement

"The challenge of securely embedding and concealing sensitive data within digital images requires advanced techniques that leverage quantum-inspired randomness to enhance robustness and security. These systems aim to resist detection by modern steganalysis tools while ensuring transparency through explainability."

Input: The project takes digital images as input, allowing users to embed secret messages securely within image files.

Process:

- **Quantum-Inspired Encoding**: A quantum-inspired algorithm is used to randomize pixel modifications, ensuring the embedded data remains undetectable.
- **Image Processing**: Python's Pillow library is used to manipulate images for encoding and decoding processes.
- **Explainability:** The system integrates explainability techniques to provide insights into how pixel modifications and randomness contribute to security, fostering user trust.

Output:

- The system outputs an encoded image with the embedded secret message, maintaining the original visual quality.
- Explainability techniques provide a detailed report of the encoding process, highlighting the factors that ensure data security and robustness against detection**.**

1.4 Scope of the Project

The scope of the project are as follows:

- **Practical Applications:**

  The proposed quantum-inspired steganography technique provides significant practical applications for secure and covert communication in an era of increasing cyber threats. Government agencies can use the system to safeguard classified data, ensuring secure communication even under adversarial conditions. In corporate environments, it can protect sensitive business information, intellectual property, and client data from unauthorized access. For individuals, the system offers a reliable method to transmit personal or financial information covertly. Additionally, the technique can be adapted for secure transmission in fields like healthcare, where protecting patient data is critical, or in journalism, where maintaining confidentiality of sources is paramount. By addressing the limitations of traditional steganography, this system makes a substantial contribution to data security across diverse sectors. Furthermore, its ease of integration with existing digital workflows enhances its practical utility. This adaptability makes the system suitable for applications requiring both flexibility and high-security standards.

- **Enhanced Security:**

  The quantum-inspired approach introduces randomness and unpredictability in the data embedding process, making it resistant to modern steganalysis techniques. Unlike conventional methods such as Least Significant Bit (LSB) encoding, which can be easily detected due to predictable patterns, this technique ensures robustness by randomizing pixel modifications. This security enhancement is vital in combating advanced cyber threats and ensures that sensitive data remains undetectable even under extensive analysis. Such a high level of security makes the system suitable for applications requiring strict confidentiality, including military and intelligence operations. The additional randomness introduced mimics quantum mechanics principles, significantly increasing the difficulty for attackers attempting to reverse-engineer the hidden data. By integrating these advanced techniques, the system establishes a new standard for secure and covert communication.

- **Generalizability:**

The system’s ability to generalize across various image formats, such as PNG and JPEG, enhances its applicability in real-world scenarios. By ensuring compatibility with multiple formats, the system caters to diverse user needs and environments. Furthermore, the quantum-inspired technique can be adapted for different types of media, such as videos or audio files, expanding its scope beyond static images. This flexibility makes the system a versatile solution for addressing secure communication needs in a range of industries, from finance to media. Its modular design also allows future expansion to support emerging media formats and communication protocols. This generalizability ensures that the system remains relevant and scalable as the digital landscape continues to evolve. Additionally, the ability to handle various levels of data complexity makes the system adaptable to a range of use cases, from low-security personal use to high-security governmental applications. By being media-agnostic, the system ensures comprehensive coverage across platforms, making it a robust solution for global communication challenges.

- **Transparency:**

Transparency is a critical feature of the proposed system, achieved through the integration of Explainable Artificial Intelligence (XAI) techniques. These explainability methods provide users with insights into how randomness and pixel modifications enhance security, ensuring trust and accountability. For example, XAI tools such as SHAP and LIME can offer detailed explanations of the encoding process, allowing users to understand how specific features contribute to the system's robustness. Transparency fosters trust among users, regulators, and stakeholders, ensuring the widespread adoption and acceptance of the proposed system. Additionally, explainability enables users to identify and address potential weaknesses, improving the system's reliability and overall performance. By ensuring that users can understand and verify the security mechanisms, transparency also aids in aligning with ethical and regulatory standards for secure communication technologies. Ultimately, this approach supports a balance between advanced security and user trust, driving adoption across different sectors.

1.5 Objectives

The Objectives are as follows:

- To Develop a quantum-inspired image steganography technique that utilizes randomness and superposition to improve the unpredictability and security of the embedding process against modern steganalysis techniques.
- To Design and deploy a Flask-based web application that allows users to encode and decode hidden messages within images, ensuring usability for individuals with varying levels of technical expertise.
- To Evaluate the system's performance by comparing it with traditional LSB-based techniques using metrics such as embedding capacity, Peak Signal-to-Noise Ratio (PSNR), and resistance to steganalysis attacks to validate its robustness and practicality.

1.6 Review of Literature

The literature survey has been carried out to study the projects and researches previously performed on this same topic. We have found many approaches implemented on various datasets which have motivated us to do this project.

[1] Jehn-Ruey Jiang "A Quick Overview of Quantum Machine Learning" (QML) (Published in the Year 2023)

This work was proposed by Jehn-Ruey Jiang. The paper presents a concise introduction to

Quantum Machine Learning (QML), focusing on quantum-inspired techniques and their applications. It provides an overview of how quantum mechanics can be applied to enhance machine learning models, improving computational power and efficiency for various tasks, including data analysis and encryption.

[2] Manjunath T D, Biswajit Bhowmik "Quantum Machine Learning and Recent Advancements" (Published in the Year 2023)
This piece of work was proposed by Manjunath T D and Biswajit Bhowmik. The paper explores the evolution of quantum machine learning, focusing on recent advancements in quantum algorithms and their performance benefits over classical methods. It highlights the potential of quantum-inspired models to handle complex datasets more effectively, offering significant improvements in the speed and accuracy of data processing tasks.

[3] Shahid Rahman, Jamal Uddin, MuhammHilal Ahmad Bhat, Farooq Ahmad Khanday,brajesh Kumar Kaushik, Faisal Bashir,and Khurshed Ahmad Shah "Quantum Computing: Fundamentals, Implementations and Applications " (Published in the Year 2022)
This work was proposed by Hilal Ahmad Bhat, Farooq Ahmad Khanday, Brajesh Kumar Kaushik, Faisal Bashir, and Khurshed Ahmad Shah. The paper introduces the fundamentals of quantum computing, its various implementations, and applications in diverse fields. The authors discuss recent developments in quantum computing, emphasizing its potential to solve problems that are intractable for classical computers, including optimization, cryptography, and secure communications.

[4] Zakarya,hameed Hussain, Ayaz Ali Khan, Aftab Ahmed And Muhammad Haleem "A Comprehensive Study of Digital Image Steganographic Techniques" (Published in the Year 2023)
This research was proposed by Shahid Rahman, Jamal Uddin, Muhammad Zakarya, Hameed Hussain, Ayaz Ali Khan, Aftab Ahmed, and Muhammad Haleem. The paper provides an extensive review of digital image steganography techniques, analyzing the strengths and weaknesses of various methods used to conceal data in images. It explores the evolution of these techniques and their applications in secure communication, highlighting the challenges faced in preventing detection by steganalysis tools.

[5] S. Sravani, R. Ranjith "Image Steganography For Confidential Data Communication" (Published in the Year 2021)
This research was proposed by S. Sravani and R. Ranjith. The paper examines the use of Least Significant Bit (LSB)-based image steganography techniques to securely hide confidential data within digital images. The authors discuss how the method works and its effectiveness in concealing information, while also addressing the limitations of LSB encoding in terms of security and resistance to detection by advanced steganalysis methods.

1.7 Organization of the Report

The report is organized into the chapters as follows:

Chapter 1- Introduction: The chapter 1 highlights challenges in secure communication and the limitations of traditional steganography. It introduces quantum-inspired steganography, leveraging principles like randomness and superposition to enhance security while maintaining image quality. The chapter emphasizes the system's ability to address vulnerabilities in traditional methods and its relevance in securing data transmission for applications like national security, personal privacy, and financial systems. It also outlines the project's objectives and the significance of bridging theoretical advancements with practical implementation.

Chapter 2 -System requirement specification: The chapter 2 presents the specific requirements, software and hardware components, and a summary of the chapter. It details functional requirements like encoding and decoding secret data, QR code generation, and data integrity while addressing non-functional requirements like usability, reliability, and performance. The chapter ensures that the system is designed to meet user needs and operates efficiently across various platforms and environments.

Chapter 3 - High level Design: The chapter 3 explains system architecture, design considerations, use case diagrams, data flow diagrams, and state charts to outline system operations and interactions. The chapter focuses on quantum-inspired randomness to enhance data security and transparency through explainability. It ensures scalability by supporting multiple formats and adaptability to diverse media types, making the system robust and flexible for real-world applications.

Chapter 4 - Detail design: The chapter 4 Provides structural charts, detailed module functionalities, and flowcharts for tasks like image preprocessing, encoding, decoding, and explainability. The chapter emphasizes error handling and user-friendliness by detailing interactions between input, preprocessing, encoding, and decoding modules. Additionally, it ensures computational efficiency for high-resolution images and large datasets, supporting seamless integration with the web interface.

Chapter 5 – Conclusion: The chapter 5 describes the findings and implications of the project, highlighting the system's ability to achieve secure and covert communication using quantum-inspired techniques. It reflects on the successful validation of the system's functionalities and discusses its potential to address challenges in data security and steganography. The chapter also underscores the scalability of the system for diverse applications across industries.

Chapter 6 - System Testing: Evaluates system functionality, security, and performance. Core tests include input validation, encoding/decoding accuracy, and usability, ensuring the system is ready for real-world deployment.

Chapter 7 - Results and Discussion: Presents experimental results, including snapshots of system functionalities, discussing the advantages of quantum-inspired randomness and areas for improvement.

Chapter 8 - Future Scope and Applications: Explores potential advancements, scalability, and applications of the system in secure communications, healthcare, and other industries, emphasizing its adaptability for future developments.

1.8 Summary

The first chapter addresses the growing need for secure communication in the digital age, focusing on image steganography. It highlights the limitations of traditional methods like Least Significant Bit (LSB) encoding, which are vulnerable to modern steganalysis, and introduces quantum-inspired steganography. By leveraging quantum principles like randomness and superposition, the proposed system enhances security by randomizing pixel modifications, making hidden data harder to detect. The chapter outlines the project’s goal to develop a robust, user-friendly steganography system using Python’s Pillow library and a Flask-based web application. It emphasizes the system's relevance in securing data transmission for national security, personal privacy, and financial systems.

Chapter 2

SYSTEM REQUIREMENT SPECIFICATION

System requirement specifications are gathered by extracting the appropriate information necessary for implementing the proposed quantum-inspired image steganography system. These specifications define the conditions that the system must fulfill to achieve its objectives. The SRS provides a comprehensive understanding of the system's purpose and functionality without detailing the implementation strategies. It ensures that the system’s internal mechanisms and processes remain concealed while delivering clarity about its intended outcomes.

2.1 Specific Requirements

• **Python**

Programming language used to design the proposed method is Python. Python is a highlevel programming language with dynamic semantics. It is an interpreted language i.e. interpreter executes the code line by line at a time, thus makes debugging easy Python Imaging Library (PIL) is one of the popular libraries used for image processing. PIL can be used to display image, create thumbnails, resize, rotation, convert between file formats, contrast enhancement, filter and apply other digital image processing techniques etc.

Python is often used as a support language for software developers, for build control and management, testing, and in many other ways. Python is designed by Guido van Rossum. It is very easy for user to learn this language because of its simpler coding. It provides an easy environment to furnish computation, programming visualization. Python supports modules and packages, which encourages program modularity and code reuse. It has various built-in commands and functions which will allow the user to perform functional programming.

Apart from being an open-source programming language, developers use it extensively for application development and system development programming. It is a highly extensible language. Python contains many inbuilt functions which helps beginners to learn easily. Python is best-suited for the backend side of web development and is frequently combined with a frontend language such as JavaScript to handle the user interface.

• **Flask**

Flask is a lightweight and highly customizable Python web framework used for developing the user interface of the proposed system. Its micro-framework approach ensures that only essential components are included, providing developers with the flexibility to integrate additional libraries as needed. Flask's simplicity makes it beginner-friendly while maintaining robust functionality, making it ideal for creating a web application that enables seamless encoding and decoding of messages. Flask’s built-in server and debugging tools streamline testing and deployment processes. Furthermore, its compatibility with modern front-end technologies ensures that the system remains responsive and user-centric. Flask also supports secure data handling, ensuring that sensitive information is protected during the communication process. Its extensible architecture allows the integration of plugins for performance optimization and scalability. With a vast community and extensive documentation, Flask ensures a smooth development experience for both small-scale and large-scale projects.

• **HTML**

HTML (HyperText Markup Language) is used for structuring the content and interface of the web application. It organizes the layout with tags and elements for headings, images, and input forms, ensuring intuitive navigation and user interaction. HTML’s compatibility with multimedia elements like images ensures efficient communication between the user and the quantum-inspired steganography system. Its semantic structure improves readability, accessibility, and SEO compatibility, supporting a seamless and inclusive user experience. Modern HTML features enable embedding advanced media elements like video and audio for future scalability. It also supports custom data attributes, which can be utilized to pass information between the front end and back end efficiently. Combined with frameworks like Bootstrap, HTML simplifies responsive design and ensures cross-platform compatibility for diverse user environments.

• **CSS**

CSS (Cascading Style Sheets) is a stylesheet language used to control the presentation and layout of web pages. It separates content from design, allowing developers to style HTML elements with colors, fonts, spacing, and more. CSS provides powerful tools for defining layouts, such as grids and flexbox, which make designing responsive and dynamic web pages easier. It uses selectors to target HTML elements and apply styles, with rules organized in cascading order to prioritize specific declarations. CSS supports external stylesheets, enabling consistent design across multiple pages. It also includes features for animations, transitions, and effects, enhancing interactivity and user experience. With media queries, CSS enables responsive design, adapting content for various devices and screen sizes. It supports variables and custom properties for reusability and consistency in designs. Additionally, CSS frameworks like Bootstrap and Tailwind simplify styling with pre-built components. As a versatile and essential tool in web development, CSS allows for creative and functional web designs while maintaining flexibility and efficiency.

• Java Script

JavaScript is included in the web application to handle interactivity and logic on the client side. For example, a script is provided to close the browser window when necessary or alert the user if the action cannot be performed. JavaScript enhances the responsiveness of the user interface by enabling dynamic interactions without reloading the page. It is also capable of managing client-side form validations, ensuring that users enter the correct data format before submission. In the Steganography App, JavaScript can be extended to provide real-time feedback, such as previewing uploaded images or displaying progress indicators during encoding or decoding processes. Additionally, JavaScript supports seamless integration with modern libraries and frameworks, such as React or Angular, should the application require further scalability. Its versatility allows developers to add features like animations, tooltips, or error messages, creating a more interactive and user-centric experience. These capabilities make JavaScript an indispensable part of the application's functionality.

Backend:

• Flask

Flask, a Python-based microframework, is used to handle the backend of the Steganography App. It facilitates routing, enabling communication between the frontend and backend. Flask processes form submissions for QR code generation, text encoding into images, and decoding text from images. Its lightweight structure allows easy integration with external libraries like Pillow and QrCode. Flask also supports form validation and file handling, ensuring efficient interaction between the client and server.

• Python Libraries:

- **QRcode:** Used for generating QR codes from user-input text. It ensures high-quality QR codes with error correction to improve readability.

- **Numpy:** Supports pixel-level image manipulation by converting images into arrays for processing. It facilitates implementing quantum-inspired encoding techniques for steganography.

- **IO:** Handles in-memory storage and transmission of images, allowing users to download encoded images directly without saving them to a physical file system.
- **Pillow (PIL):** Used for image manipulation, including tasks like resizing, encoding, and decoding text into images. Pillow is essential for reading and processing uploaded image files.

Functional Features:

• QR Code Generation:

The `/generate_qr` endpoint takes text input from the user, generates a QR code using the `qrcode` library, and returns it as a downloadable image.

• QR Code Decoding:

The `/decode_qr` endpoint processes an uploaded image file, extracts QR code data using `pyzbar`, and displays the decoded text to the user.

• Image Steganography:

1. **Encoding:**
   The `/encode` endpoint accepts an image and secret text, encodes the text into the image using quantum-inspired randomness (simulated via rotation and normalization), and provides the user with a downloadable encoded image.
2. **Decoding:**
   The `/decode` endpoint processes an uploaded image, extracts the hidden data, and displays it to the user.

2.2 Hardware Requirements

- **System :** Intel Core i5/i7/i9.
- **Hard Disk :** Minimum 500GB SSD.
- **RAM:** Minimum 8 GB

2.3 Software Requirements

- **Operating System:** Windows 10/ 11.
- **Software Tools:** Visual Studio Code, Flask ,Python libraries (Pillow, NumPy, qrcode, pyzbar).
- **Coding Languages:** Python, HTML, CSS, JS

2.4 Functional Requirements

Functional requirements describe the core functionalities and operations that the system must perform to meet the user needs and achieve the project goals. These requirements specify the expected behavior of the system and define how it will interact with users and process data. Below are the key functional requirements for the proposed quantum-inspired steganography system.

- **Encoding Text into Images:** The system should allow users to securely encode secret text into images using quantum-inspired randomness techniques to ensure the hidden data remains undetectable.
- **Decoding Text from Images:** The system should accurately decode and retrieve the embedded text from the uploaded images without data loss or errors.
- **QR Code Generation:** The system should generate QR codes from user-provided text and allow users to download them as image files.
- **QR Code Decoding:** The system should read and decode QR codes from uploaded image files, extracting and displaying the encoded text.
- **Data Integrity:** The system should ensure that the quality of the original image is preserved after encoding the secret data.
- **File Handling:** The system should support file uploads and manage various image formats, such as PNG and JPEG, for seamless processing.

2.5 Non-Functional Requirements

- **Usability:** System should be user friendly.
- **Reliability:** The system should be reliable.
- **Performance:** The system should process tasks like image encoding and QR code generation efficiently without significant delays.

- **Supportability:** System should be easily updatable for future enhancement.

2.6 Summary

Chapter 2 outlines the system requirements for the proposed quantum-inspired image steganography system. Section 2.1 details the specific requirements, including programming languages and libraries like Python, Flask, HTML, CSS, and JavaScript. Section 2.2 discusses hardware requirements, specifying the need for an Intel Core i5/i7/i9 processor, 500GB SSD storage, and at least 8GB RAM. Sections 2.4 and 2.5 cover the functional and non-functional requirements, detailing system functionalities and performance criteria.In essence, the summary of the Software Requirements Specification encapsulates the core components of the document, providing a comprehensive understanding of the system's purpose, its operational requirements, and the performance criteria that will define the success of the project.

Chapter 3

HIGH LEVEL DESIGN

High-level design (HLD) outlines the architecture for developing the quantum-inspired image steganography system. It provides an overview of the system, identifying the primary modules, their interactions, and how they collectively achieve secure and covert communication. The HLD is presented in terms that are comprehensible to administrators and stakeholders, while the low-level design delves deeper into implementation specifics.

HLD ensures clarity in design by defining the flow of data between modules, sub-modules, and their integration. It visualizes the system's overall implementation process, highlighting areas for iterative refinement to address errors identified in subsequent phases.

3.1 Design Considerations

Design considerations ensure the system is secure, scalable, and robust while maintaining user accessibility and image quality. The following key aspects were considered:

3.1.1 Quantum-Inspired Security

The system employs quantum-inspired randomness for pixel modifications to enhance data security. Randomizing embedding patterns ensures resistance to modern steganalysis tools, making it difficult to detect hidden data.

3.1.2 Transparency in Encoding

Explainability is integrated to provide transparency in the data embedding process by offering insights into how the system modifies pixel values to securely embed data. It ensures users understand the randomness introduced during encoding, enhancing trust and reliability. Visual overlays highlight modified regions in the image, allowing users to see the impact of the embedding process. Additionally, descriptive feedback explains how the quantum-inspired randomness ensures data security while maintaining image quality.

- **Modified Pixel Visualization**: Visual overlays highlight the pixel regions modified during encoding.
- **Explainable Encoding Process**: Descriptive feedback explains how randomness secures data, enhancing trust among users.

3.1.3 Scalability and Generalization

To support diverse real-world applications and improve robustness, the system incorporates:

- **Support for Multiple Formats**: Compatible with various image formats like PNG, JPEG, and BMP.
- **Adaptability**: The quantum-inspired approach can be extended to other media types, such as videos and audio.
- **Dataset Augmentation**: Increases robustness by applying transformations like rotation, flipping, and zooming.
- **Dataset Augmentation**: Increases robustness by applying transformations like rotation, flipping, and zooming.
- These Techniques like augmentation ensure the model generalizes well to unseen data.

3.1.4 Efficiency

The system prioritizes computational efficiency to handle high-resolution images and large datasets. Techniques like in-memory processing and optimized pixel manipulation ensure minimal latency during encoding and decoding.

3.2 System Architecture

The architecture as figure 3.1 outlines the core components and data flow within the image steganography system. It is structured into several key modules to ensure efficient and secure communication.

3.2.1 Web Interface Module

The web interface module provides a user-friendly platform for interaction, allowing users to upload images, input text, and perform encoding or decoding operations seamlessly.

3.2.2 Encoding Component

This component handles the process of embedding secret text into an image through the following steps:

1.**Binary Conversion**: Converts the input text into binary format for embedding.

2.**Pixel Modification**: Modifies specific pixel values in the image to encode the binary data using systematic techniques.

3.2.3 Decoding Component

The decoding component extracts the hidden text from an encoded image:

1.**Pixel Analysis**: Analyzes pixel-level changes to retrieve encoded binary data.

2.**Text Reconstruction**: Converts the binary data back into its original textual form.

3.2.4 QR Code Generation Component

This module generates QR codes for the embedded data, providing an additional layer of secure data sharing.

3.2.5 PSNR Calculation Module

The PSNR module calculates the Peak Signal-to-Noise Ratio to measure image quality after encoding, ensuring the steganographic process maintains visual integrity.The system leverages libraries like Pillow for image processing, Numpy for numerical computations, QRCode for generating QR codes, and Math for advanced calculations, as shown in Figure 3.1.

The figure 3.1 shows the system architecture for the tabular dataset.

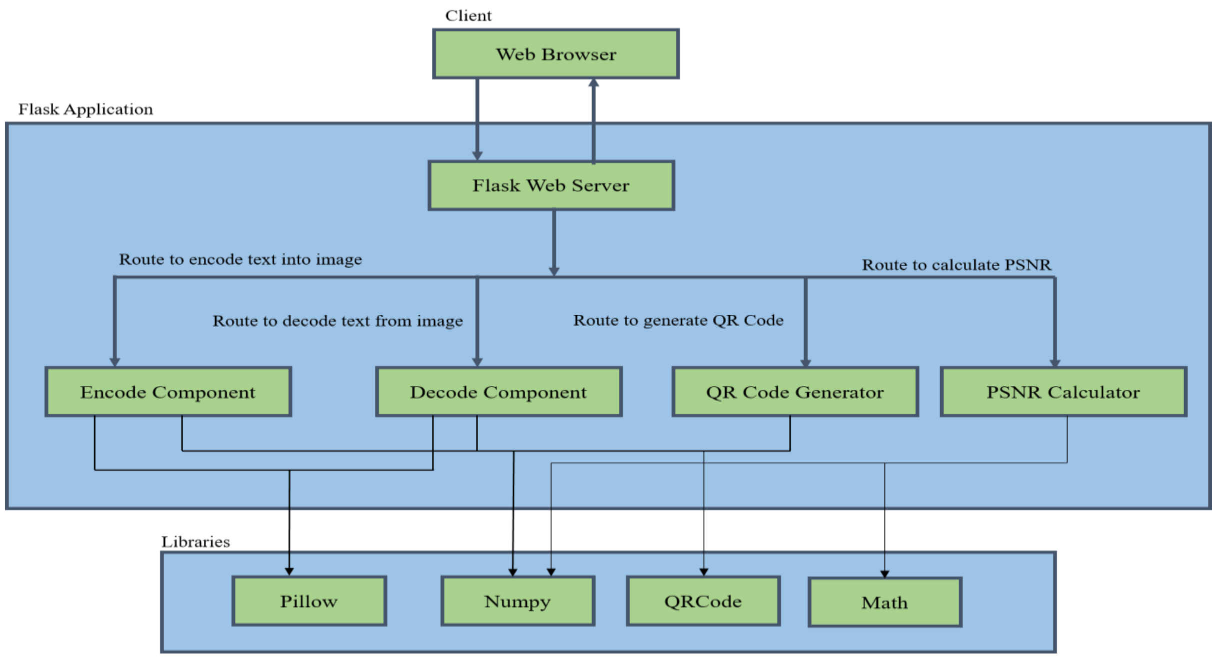

Figure 3.1: System Architecture

3.3 Specification using Use Case Diagram

A Use Case Diagram provides a graphical representation of the system's functionality, showing the interaction between the user and the system.

Actors:

1.**User**: Uploads an image for classification and views the results and explanations.

2.**System**: Processes the image, classifies it, and provides explanations.

3.3.1 Use Case Diagram for Preprocessing Module

This use case diagram in figure 3.2 illustrates a simplified workflow for a system that processes and analyzes user-provided content to extract and select relevant features. The process begins with the **user** providing the **input content**, which could include text, images, or other data types requiring analysis. Once the input is submitted, the system performs **feature extraction**, identifying significant patterns or characteristics within the content that are essential for analysis. This stage is crucial for transforming raw input into structured and meaningful information. The extracted features then proceed to the **feature selection** phase, where the most relevant and impactful features are chosen for further processing, ensuring efficiency and accuracy in subsequent tasks. The **system** interacts with the user by accepting the input and delivering the selected features as output, enabling the user to utilize these results for decision-making or further analysis. This diagram demonstrates a clear and sequential interaction between the user and the system, emphasizing the importance of organized data processing in achieving optimal outcomes.

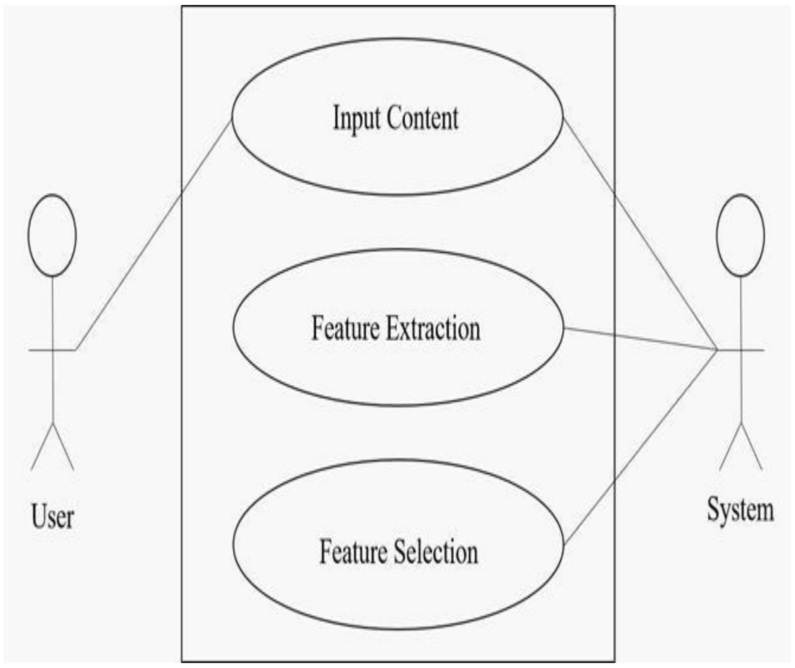

Figure 3.2: Use Case Diagram for Preprocessing Module

3.3.2 Use Case Diagram for Detection and Analysis Model

The use case diagram in figure 3.3 represents a structured workflow for an **Image Steganography System**, illustrating the interaction between the user and the system. The process begins with the user embedding a message into an image using steganographic techniques, followed by uploading the image as a carrier for the concealed message. The system then allows the user to view results and analytics, providing feedback on the embedding process, such as success metrics and image quality. It also supports extracting hidden messages, enabling the user to retrieve concealed content from a steganographic image. Additionally, the system analyzes the quality of the image to ensure that the embedding process does not degrade its visual or structural integrity. Finally, it manages the entire steganography process, ensuring secure handling of data, maintaining workflow consistency, and enabling seamless user interaction. This workflow emphasizes the efficient and transparent transformation of user input into actionable results.

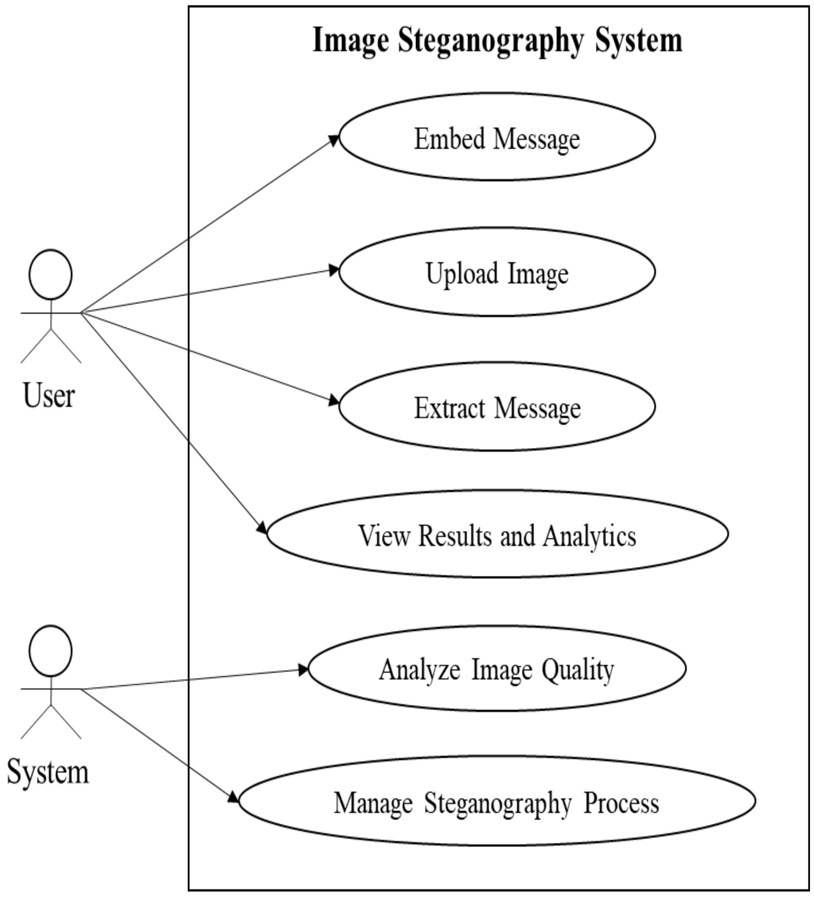
Image Steganography System
Embed Message
Upload Image
Extract Message
View Results and Analytics
Analyze Image Quality
Manage Steganography Process
User
System

Figure 3.3: Use Case Diagram for Detection and Analysis

3.4 Data Flow Diagram

As the name suggests, a Data Flow Diagram (DFD) visually depicts how data flows and is transformed within a system (from figure 3.4,3.5,3.6,3.7 and 3.8). It serves as a fundamental tool in systems analysis and design, providing a high-level overview of the system's processes and data movement. DFDs illustrate the flow of data from its source to its destination, highlighting the various processes involved in transforming the data. This graphical representation helps in understanding the system's functionality, identifying potential bottlenecks, and making informed decisions during the system development process.

3.4.1 DFD for the Module (POST /encode with image and text):

1.**Input**: The process starts with a **Client request**.
2.**Process**: The image is processed through the /encode endpoint, converted to RGBA, a quantum rotation is applied, and secret data is embedded.
3.**Output**: The modified image containing the embedded secret data is returned to the client.

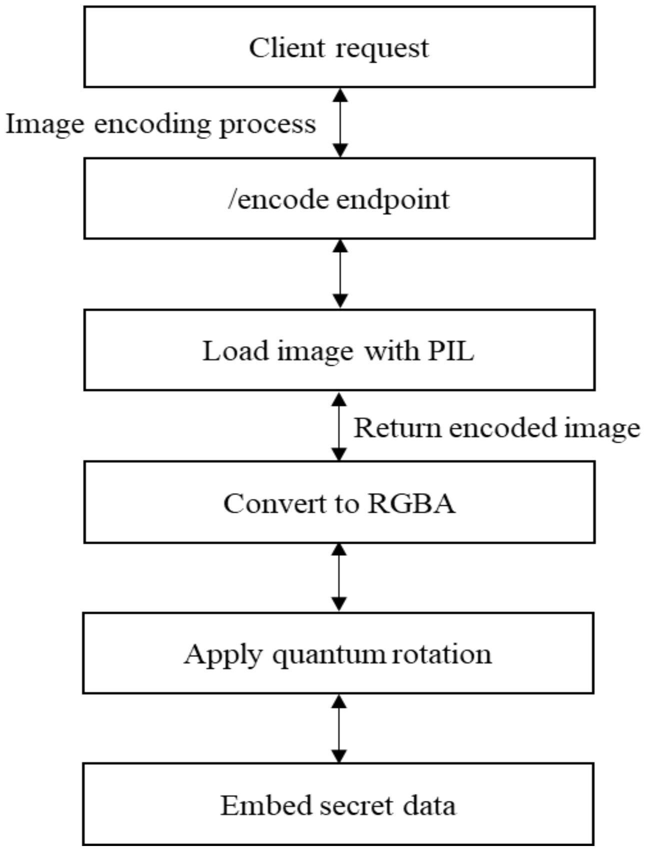

Figure 3.4: DFD for the Module (POST /encode with image and text)

3.4.2 DFD for the Module (POST decode with image):

1. **Input**: Client request to decode an image.
2. **Process**: The image is decoded using the /decode endpoint.
3. **Output**: The decoded image is passed to the "Load encoded image" process.

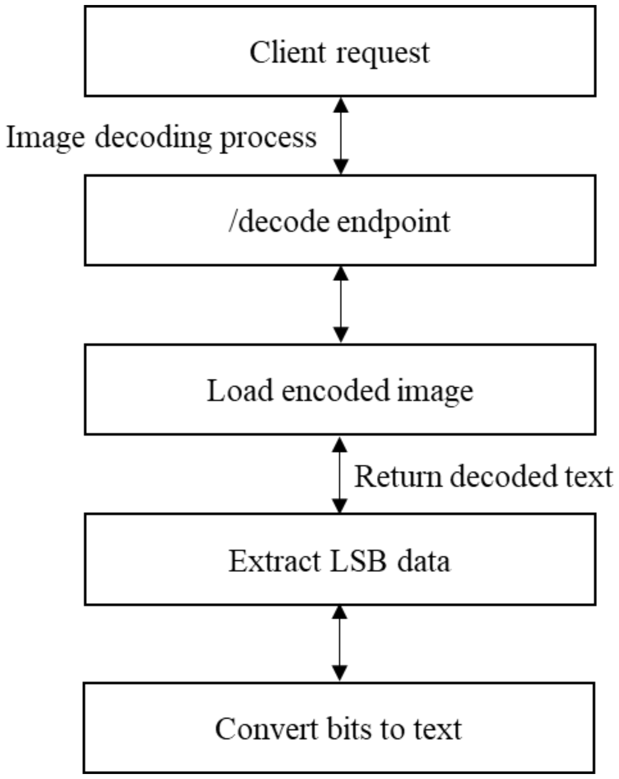

Figure 3.5: DFD for the Module (POST decode with image)

3.4.3 DFD for the Module (POST /generate_qr with text):

1. **Input**: Client request to generate a QR code.
2. **Process**: The QR code is generated using the /generate_qr endpoint.
3. **Output**: The generated QR code is passed to the "Convert to base64" process.

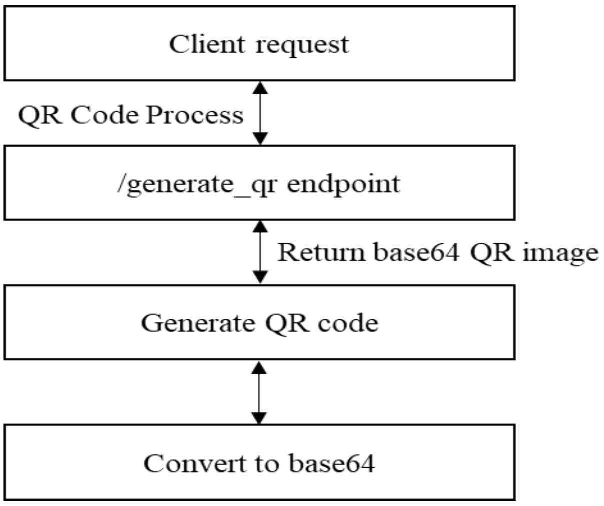

Figure 3.6: DFD for the Module (POST /generate_qr with text)

3.4.4 DFD for the PSNR calculation Module:

1.**Input**: Grayscale images.

2.**Process**: Mean Squared Error (MSE) is calculated between the grayscale images.

3.**Output**: Calculated MSE is passed to the "Compute PSNR" process.

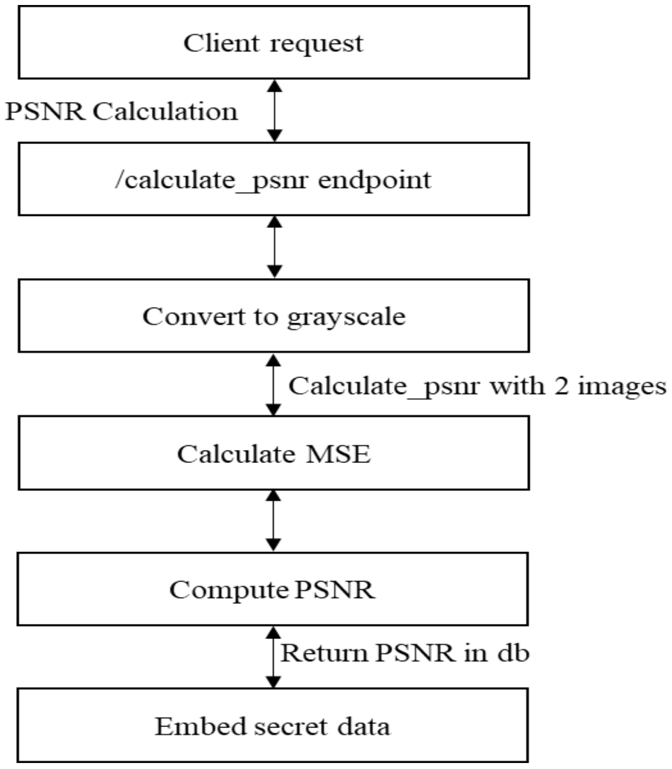

Figure 3.7: DFD for the PSNR calculation Module

3.4.5 DFD for the Steganographic Processing and Analysis Module:

1.**Input**: The process in figure 3.8 begins with the **User** selecting an operation from the options available.

2.**Process**: The user selects an operation, which determines the workflow. Encoding generates and post-processes an encoded image, decoding extracts and verifies hidden text, PSNR calculation computes and validates the metric, and QR code generation designs and styles the code. Each operation returns the respective result to the user.

3.**Output**: The results (encoded image, decoded text, PSNR value, or QR code) are delivered to the **User** based on their selected operation.

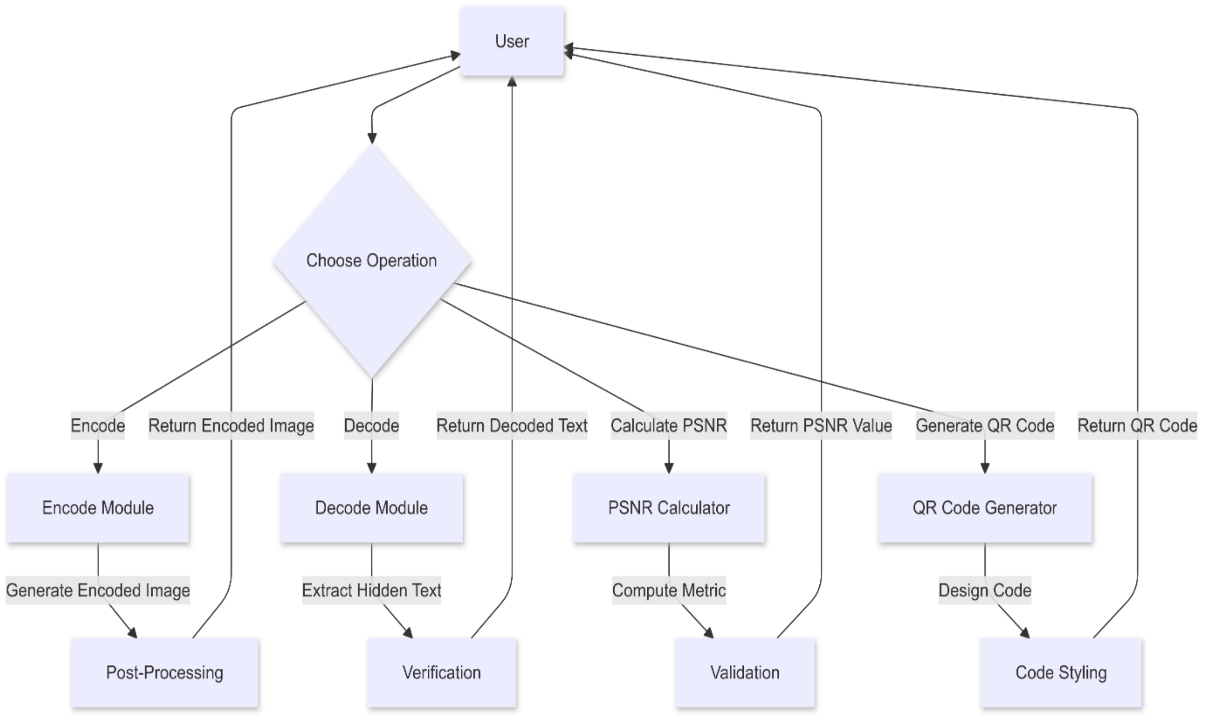

Figure 3.8: DFD for the Steganographic Processing and Analysis Module

3.5 Sequence Diagram

A **Sequence Diagram** illustrates the interactions and flow between different components of a system. It shows how objects or components communicate in a time-ordered sequence. Below is the description of the system's sequence diagram for encoding and decoding a secret message within an image:

3.5.1 States:

1. **Upload Image:** The user uploads an image to the WebApp.

2. **Enter Secret Message:** The user inputs the secret message they wish to encode.

3. **Encode Secret Message:** The WebApp sends the image and secret message to an image-processing module for encoding.

4. **Return Encoded Image:** The image-processing module returns the encoded image to the WebApp.

5. **Provide Encoded Image:** The WebApp sends the encoded image back to the user.

6. **Upload Encoded Image:** The user uploads the encoded image to the WebApp when they want to decode the hidden message.

7. **Decode Secret Message:** The WebApp sends the encoded image to the image-processing module for decoding.

8. **Extracted Message Returned:** The image-processing module extracts and sends the hidden secret message back to the WebApp.

9. **Show Secret Message:** The WebApp displays the extracted secret message to the user.

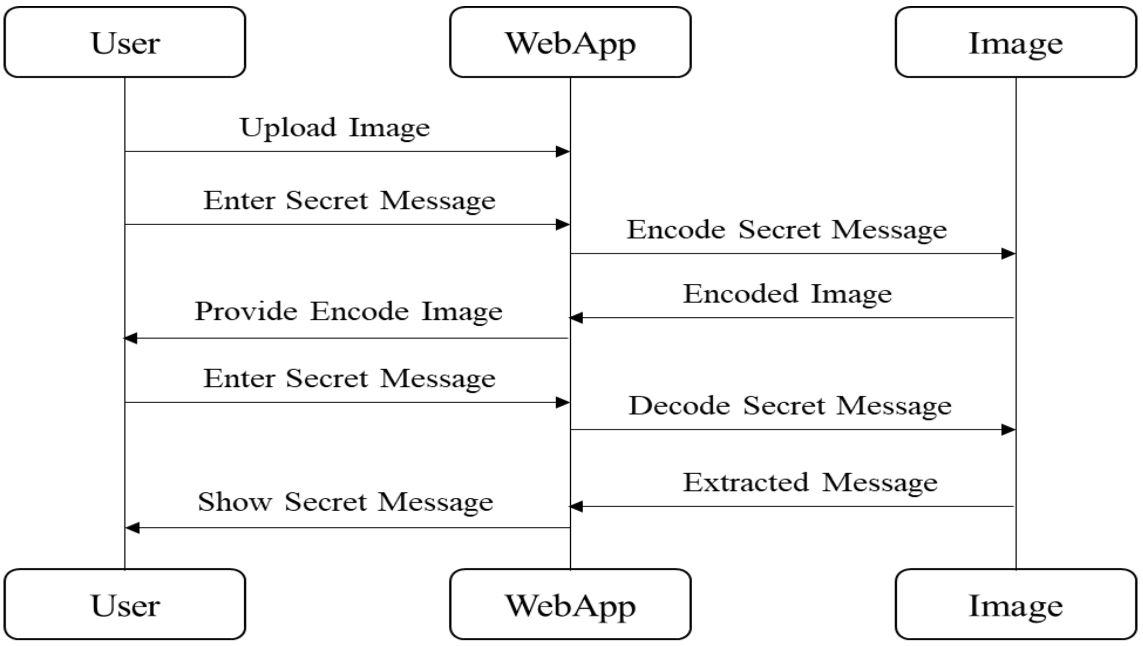

Figure 3.9: Sequence Diagram of the Model

The above figure 3.9 illustrates the process of encoding a secret message into an image and decoding it back. The interactions between the user, WebApp, and image-processing module are shown sequentially, ensuring a smooth workflow. The user provides the input (image and secret message), while the WebApp processes these inputs through the image module to encode or decode the message. This system is particularly useful for secure and private communication, where messages are hidden within images using steganography techniques.

3.6 Summary

The High-Level Design (HLD) outlines the architecture for the quantum-inspired image steganography system, focusing on secure and efficient communication. It identifies key modules, including input, preprocessing, encoding, decoding, and explainability modules, and explains their interactions to achieve seamless functionality. Quantum-inspired randomness ensures robust security by making hidden data resistant to detection, while explainability features, such as visual overlays and descriptive feedback, enhance transparency. The system supports scalability with multi-format compatibility and adaptability to diverse media types. Preprocessing includes resizing, normalization, and validation to ensure data readiness, while encoding and decoding involve binary conversion and pixel modification for secure data embedding and retrieval. Computational efficiency is prioritized to handle high-resolution images and large datasets. By integrating explainable features and optimizing for performance, the system ensures reliable, secure, and user-friendly steganographic operations. A detailed architectural diagram complements this design overview.

Chapter 4

DETAILED DESIGN

4.1 Structural Chart

A Structural Chart represents the hierarchical relationships between the various components of the quantum-inspired image steganography system. It breaks down the system into its core modules, illustrating how each part interacts and contributes to the overall functionality.

4.1.1 Overview of Modules

1. **Input Module**:
    - Handles user interaction by accepting uploaded images and secret text.
    - Validates the image format and checks if the text length is within the allowable range.
2. **Preprocessing Module**:
    - Resizes images to standard dimensions (256x256 pixels) to ensure compatibility with the encoding process.
    - Normalizes pixel values to the [0, 1] range for consistent processing during embedding.
3. **Encoding Module**:
    - Converts the secret text into binary form and embeds it into the image pixels using quantum-inspired randomness techniques.
    - Ensures the data is securely hidden while preserving the image’s visual integrity.
4. **Decoding Module**:
    - Extracts the hidden data from the encoded image by reading the least significant bits of pixel values.
    - Reconstruct the original text from the binary data without errors.
5. **Explainability Module**:
    - Highlights pixel regions modified during the encoding process to provide transparency.

- Generates visual overlays to explain how quantum-inspired randomness secures the hidden data.

6. **Output Module**:
   - Displays the encoded image for download after the embedding process.
   - Shows the decoded secret text along with visual explanations for the encoding process.

4.1.2 Relationships between Modules

Each module performs a specific function and sends the processed data to the next module in a sequential pipeline. From below figure 4.1 here's how the modules interact:

- **Input Module** Receives user-provided images and text, then forwards them for processing.
- **Preprocessing Module** processes the image and forwards it to the Encoding Module for embedding the data.
- **Encoding Module** embeds the secret data and provides the encoded image to the Output Module.
- **Decoding Module** retrieves the hidden data and sends it to the Output Module for display.
- **Explainability Module** interacts with both the Encoding and Output Modules to generate and display visual explanations.

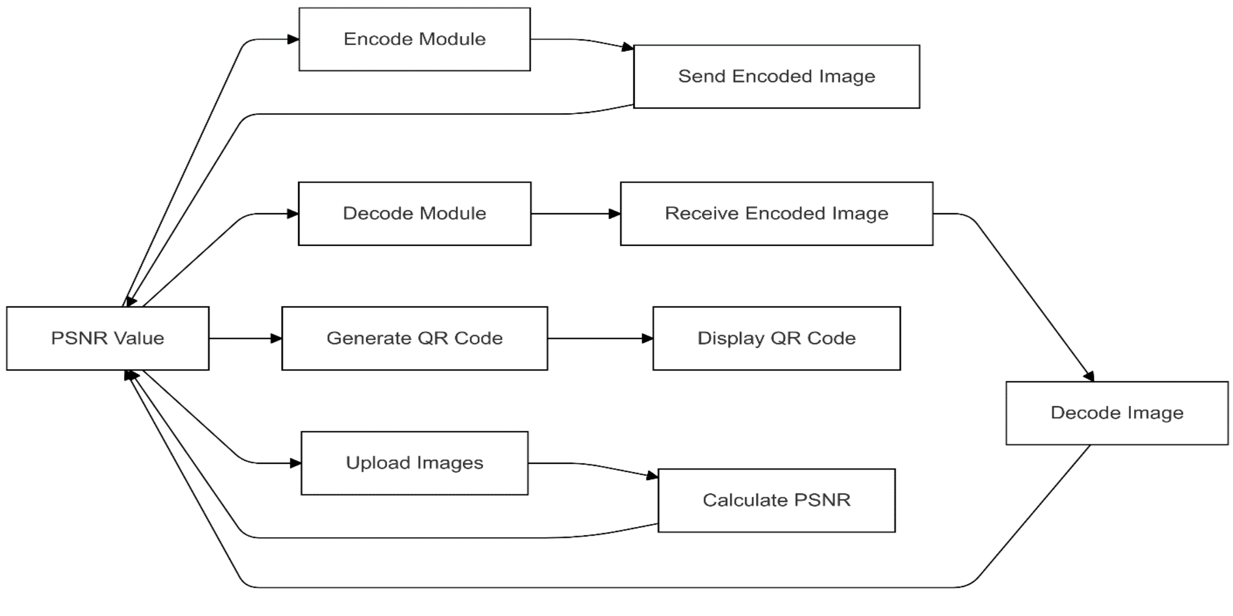

Figure 4.1: A Structural Chart

4.2 Detailed Description of Each Module Using Flowcharts

Each module in the system is described below, supported by flowcharts that outline the operations performed within each.

4.2.1 Data Gather Module

The flowchart as figure 4.2 outlines a process for gathering, processing, and storing data. It starts by defining a **Data Gathering Module** that collects data from sensors. The module then cleans and preprocesses the collected data to ensure it is ready for storage. Finally, the cleaned data is stored in a database, making it available for further use. This iterative process ensures that data is collected, refined, and stored efficiently, enabling consistent availability of high-quality data.

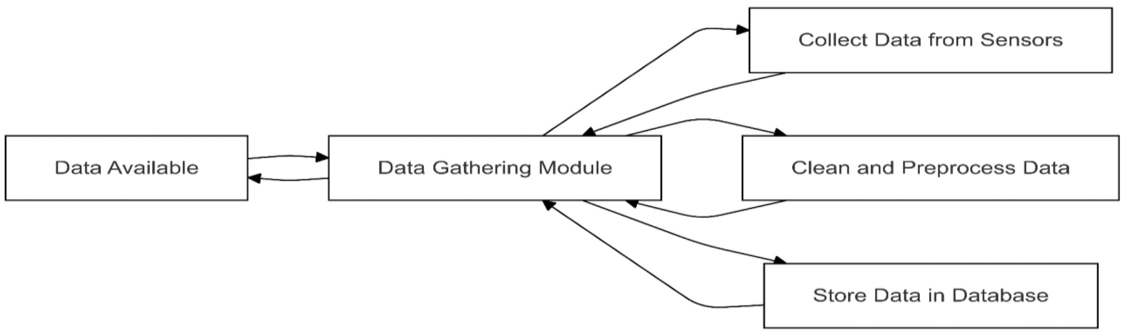

Figure 4.2: Flow Diagram for Data Gather Module

4.2.2 Preprocessing Module

The flowchart as figure 4.3 depicts a series of image preprocessing steps crucial for preparing data for a deep learning model. It begins by decoding the input image and resizing it to a standardized dimension to ensure consistency across the dataset. Subsequently, the image undergoes normalization to enhance model stability and training speed. This typically involves scaling pixel values to a specific range, such as between 0 and 1. To improve processing efficiency, a batch dimension is added to the image data, allowing the model to process multiple images simultaneously. The preprocessed images are then iteratively processed in batches, with the extraction of image paths and corresponding labels for supervised learning tasks. Within each batch, individual images undergo further preprocessing steps, such as data augmentation techniques, to improve model robustness. Finally, the preprocessed images and their associated labels are yielded as arrays, ready to be fed into the deep learning model for training or inference.

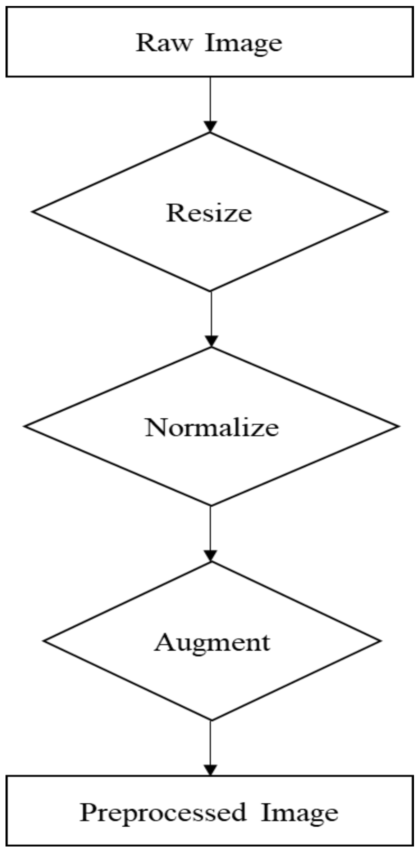

Figure 4.3: Flow Diagram for Preprocessing Module

4.2.3 Training Module

The flowchart as figure 4.4 illustrates the step-by-step process for developing a quantum-inspired image steganography system. The process begins by gathering the necessary input data, such as the image and secret message. This is followed by preprocessing the image, which includes resizing and normalizing it for consistency. Once preprocessed, the system encodes the secret message into the image using quantum-inspired randomness techniques. After encoding, the model is evaluated to ensure the security and quality of the steganographic image, using metrics like PSNR and MSE. Finally, the trained and validated system is saved for deployment, providing a robust and efficient solution for secure communication.

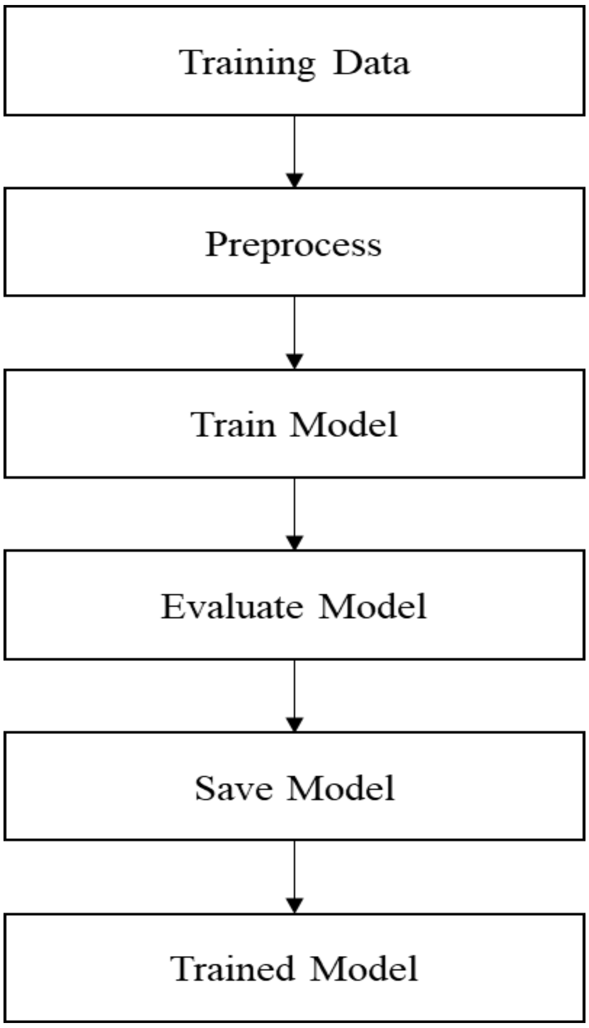

Figure 4.4: Flow Diagram for Training Module

4.2.4 Evaluating Model

The flowchart as figure 4.5 depicts a series of steps involved in evaluating a trained deep learning model using a validation dataset. Initially, a data generator is prepared for the validation set, ensuring the data is appropriately formatted for model evaluation. The number of validation steps is then determined, defining the number of batches of data to be used for evaluation. Subsequently, the model generates predictions for the validation data. For binary classification tasks, these predictions might require thresholding to obtain discrete classifications. The predicted labels are then aligned with the corresponding ground truth labels from the validation set. Following this, various evaluation metrics, such as accuracy, precision, recall, and F1-score, are calculated to assess the model's performance. Finally, these calculated metrics are displayed, providing valuable insights into the model's effectiveness on the validation data.

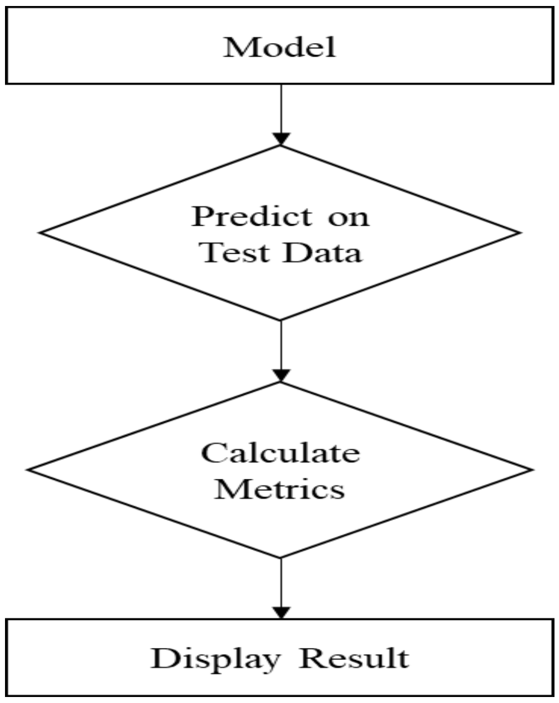

Figure 4.5: Flow Diagram for Evaluating Module

4.2.5 Flow Model

The flowchart as figure 4.6 outlines the functionalities of a Flask application designed for image processing, steganography, QR code generation, and image quality assessment. Here's a breakdown of the key components and their interactions.

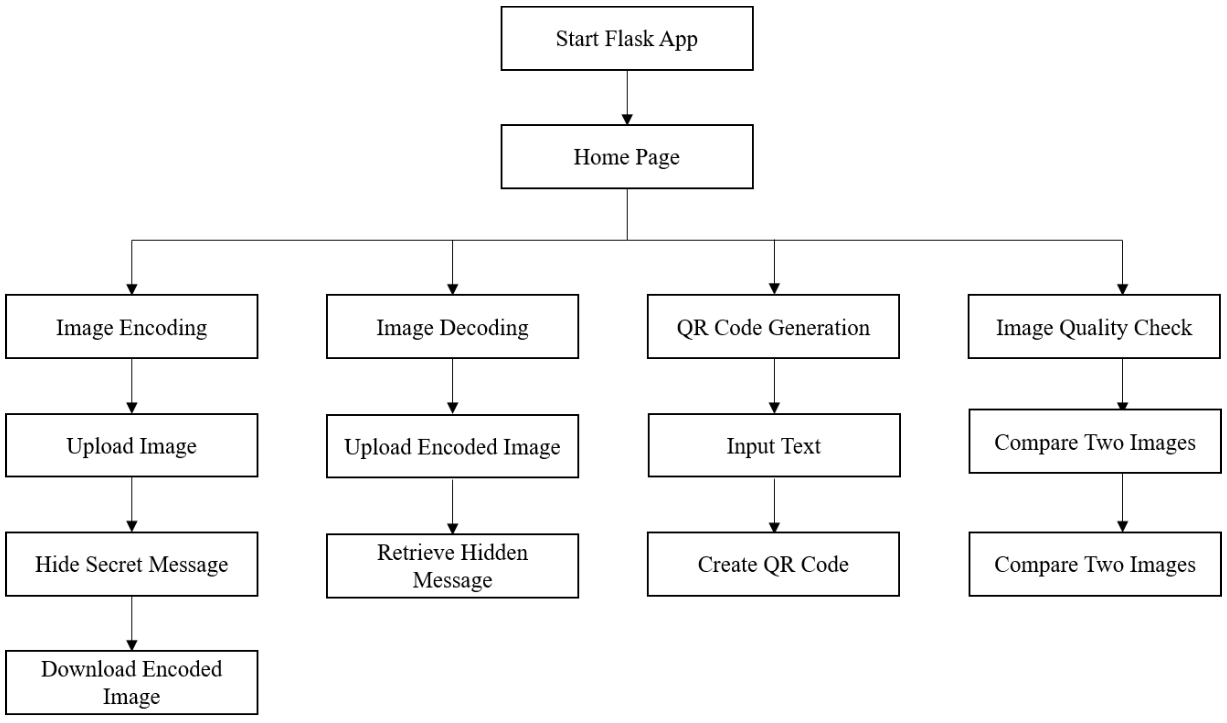

Figure 4.6: Flow Diagram

4.2.6 Functionality

Each module performs the functionalities of a Flask-based application with four main modules: Image Encoding, Image Decoding, QR Code Generation, and Image Quality Check:

1. Start Flask App:

- This is the initial step, initiating the Flask application.

2. Home Page:

- The application starts with a Home Page, which serves as the entry point for users.

3. Image Encoding Module:

- **Upload Image:** Users can upload an image to this module.
- **Hide Secret Message:** The uploaded image is processed to hide a secret message within it.
- **Download Encoded Image:** The user can then download the image with the hidden message.

4. Image Decoding Module:

- **Upload Encoded Image:** Users upload the image containing the hidden message.
- **Retrieve Hidden Message:** The module extracts and displays the hidden message from the uploaded image.

5. QR Code Generation Module:

- **Input Text:** Users input the text they want to encode into a QR code.
- **Create QR Code:** The module generates a QR code representing the input text.

6. Image Quality Check Module:

- **Compare Two Images:** Users upload two images for comparison.
- **Calculate PSNR:** The module calculates the Peak Signal-to-Noise Ratio (PSNR) between the two uploaded images to assess image quality.

This flowchart provides a clear overview of the functionalities offered by the Flask application, including image steganography, QR code generation, and image quality assessment.

4.2.7 Processing

The processing flow of the system can be broken down as follows:

1. Image Upload (encode & decode functionality):

- **User Input:** The user interacts with the application through the /encode or /decode routes.
- **Image Upload:** In both routes, the user uploads an image file using the image field in a multipart form-data request (POST method).

2. Preprocessing (encode functionality):

- **Image Opening:** The uploaded image is opened using the Image.open function, creating an image object. (This step is specific to the encode functionality.)

3. Data Processing (encode & decode functionality):

- **Secret Data Extraction (encode):** In the /encode route, the user-provided secret data is extracted from the secret_data form field.
- **Data Encoding/Decoding:**
    - **Encoding (encode):** The encode_image function takes the image object and secret data as input. It performs the core logic of hiding the secret data within the image using techniques inspired by Quantum Neural Networks (QNNs).
    - **Decoding (decode):** The decode_image function takes the uploaded image object as input. It extracts the hidden secret data from the image using the same principles employed during encoding.

4. Output Generation (encode & decode functionality):

- **Image Download (encode):** If both image and secret data are provided in the /encode request, the encoded image is saved in memory using a BytesIO object. The encoded image is then sent back to the user as a downloadable file named "encoded_image.png" with the appropriate MIME type.
- **Decoded Data Display (decode):** If an image is uploaded in the /decode request, the decoded secret data is extracted and displayed directly in the response using an f-string.

5. QR Code Generation:

- **Data Acquisition:** The /generate_qr_code route expects JSON data containing the qr_data field in the request body. This data represents the text content to be encoded into a QR code.
- **Error Handling:** If no qr_data is provided, an error message with a 400 Bad Request status code is returned.
- **QR Code Creation:** The qrcode library is used to generate a QR code object with appropriate settings (version, error correction, box size, border). The provided qr_data is added to the QR code object.
- **Image Generation:** The QR code object is converted into an image using the make_image function with desired fill and background colors.
- **Base64 Encoding:** The generated QR code image data is converted to Base64 format for efficient transmission within the JSON response.
- **Response Generation:** A JSON response is created containing the Base64-encoded QR code data under the qr_code key.

6. PSNR Calculation:

- **Image Uploads:** The /calculate_psnr route expects two image files to be uploaded using the original and modified fields in a multipart form-data request.
- **Image Opening:** Both uploaded images are opened using the Image.open function, creating corresponding image objects.
- **PSNR Calculation:** The calculate_psnr function takes the original and modified image objects as input. It calculates the Peak Signal-to-Noise Ratio (PSNR) between the two images, which is a metric for image quality comparison. (The commented-out lines suggest potential calculations for individual PSNRs, which are not used in the current implementation.)
- **Response Generation:** A JSON response is created containing the calculated PSNR value in decibels (dB) with two decimal places of precision under the psnr key.

7. Application Execution:

- The if **__name__** == '**__main__**': block ensures the Flask application code is only executed when the script is run directly (not imported as a module).

- The app.run(debug=True) line starts the Flask development server in debug mode, allowing for automatic code reloading during development.

This breakdown highlights the processing flow within the Flask application for each functionality based on the user interactions and the code's functionalities.

4.3 Summary

Chapter 4 of the detailed design focuses on the quantum-inspired image steganography system, outlining its structural chart, module relationships, and functionalities. The system is broken into six modules: Input, Preprocessing, Encoding, Decoding, Explainability, and Output, which sequentially handle tasks such as image upload, resizing, embedding secret data, and extracting it. Flowcharts illustrate the processes within each module, from data gathering to image encoding, decoding, and quality assessment using metrics like PSNR. A Flask-based application integrates the system's functionalities, including QR code generation and steganography. It employs preprocessing, quantum-inspired randomness, and robust error-handling techniques for secure, user-friendly interaction and data processing.

Chapter 5

IMPLEMENTATION

5.1 Implementation Requirements

The implementation of the quantum-inspired image steganography system requires both hardware and software resources to operate efficiently and effectively. These resources support the system's image processing, encoding and decoding tasks, and the overall user interface.

5.1.1 Hardware Requirements

1. **Processor**:

- **Intel Core i7** or an equivalent multi-core processor is recommended to handle image processing tasks and computations efficiently, especially when dealing with large image files and encoding/decoding operations.
- **GPU**: A NVIDIA GTX 1650 or higher is needed for faster processing, especially if additional machine learning-based processing is required for the quantum-inspired techniques. GPUs are optimized for matrix calculations and can significantly speed up image manipulation tasks.
- **RAM**: At least **16GB RAM** is required to handle the large image datasets, ensure smooth operation, especially when processing multiple high-resolution images simultaneously.
- **Storage**: A minimum of 50GB SSD storage is recommended to store images, model data, and system logs, ensuring fast read and write speeds during encoding and decoding tasks.

5.1.2 Software Requirements

1. **Operating System**:
   - Ubuntu (Linux) is recommended for optimal performance, particularly when using Python and related libraries for image processing. However, Windows 10 or 11 can also be used during development.
   - **Software Tools:**

- **Python:** For backend processing, including image manipulation and quantum-inspired steganography encoding/decoding.
- **Flask:** For developing the web-based user interface.
- **Pillow (PIL):** For image processing tasks like encoding and decoding.
- **HTML, CSS, JavaScript:** For front-end web design and interactivity.

2. **Libraries and Frameworks**:

- **Python 3.x**: Python is the primary programming language used for this project. Python's extensive support for image processing, web development, and libraries for machine learning makes it the ideal choice for building and implementing the quantum-inspired image steganography system.
- **Flask:** Flask is used as the web framework to build the user interface of the application. It provides the necessary tools to handle user inputs, manage file uploads, and serve the system's functionality through simple routes and templates. Flask's lightweight nature and flexibility make it a suitable choice for developing the application's backend.
- **Pillow (PIL):** For image preprocessing, such as resizing, normalization, and augmentation.
- **NumPy:** NumPy is used for efficient array manipulation and processing. It helps in converting images into arrays, facilitating pixel-level operations like applying quantum-inspired transformations and encoding data into image pixels.
- **JavaScript:** JavaScript is used for enhancing interactivity on the frontend of the web application.
- **HTML & CSS:** HTML is used to create the structure of the web application, including forms for uploading images and entering text. CSS is used to style the web page, ensuring it is visually appealing and responsive across different devices.
- **Bootstrap:** Bootstrap is used to design a responsive and user-friendly interface.

5.2 Programming Language Selection

5.2.1 Python

Python is chosen as the primary programming language for the implementation of the quantum-inspired image steganography system for several reasons:

- **Versatility**: Python is a highly versatile language that can handle tasks ranging from image processing (using libraries like Pillow) to web development (via Flask). This makes it ideal for building a comprehensive system that involves both backend logic and user interface functionalities.
- **Image Processing Libraries:** Libraries such as Pillow (PIL) and NumPy provide efficient image manipulation capabilities, which are essential for the encoding and decoding processes in steganography. These libraries make it easy to work with pixel-level operations and perform quantum-inspired transformations on images.
- **Readable Syntax**: Python's clean, easy-to-read syntax simplifies the development and maintenance of the code, which is crucial for an evolving project. Its readability makes debugging and future updates easier, especially for a complex system like steganography, where precision and clarity are key.
- **Community Support**: Python has a large and active community that continuously develops and maintains a wide range of open-source libraries and frameworks. This community support ensures access to comprehensive documentation, tutorials, and troubleshooting solutions, making Python an ideal choice for developing a robust and scalable steganography system.

5.2.2 Key Features of Python

- **Extensive Libraries**: Python offers a rich ecosystem of libraries that facilitate various tasks, including image processing, web development, and data manipulation, essential for the quantum-inspired image steganography system.
    - **Pillow (PIL)** for image manipulation and processing.
    - **NumPy** for efficient array handling and pixel-level image transformations.

- **Flask** for web deployment, enabling the creation of a user-friendly interface.
- **JavaScript** for frontend interactivity and real-time feedback during encoding/decoding.

- **Cross-Platform Compatibility**: Python is inherently cross-platform, allowing the steganography system to run seamlessly across different operating systems (Linux, macOS, Windows). This ensures that the application can be deployed and used in diverse environments without any significant modifications to the code.
- **Ease of Integration**: Python integrates easily with other languages and tools, enabling flexibility in expanding the system's capabilities. For instance, it can be combined with C/C++ for performance-intensive tasks, or JavaScript for adding more dynamic features to the web application, allowing the steganography system to evolve and adapt to various use cases and future requirements.

5.3 Description About Tools and GUI Used

5.3.1 Pillow and NumPy

Pillow (PIL) and NumPy are essential tools for image processing and pixel-level manipulation in the quantum-inspired image steganography system.

- **Pillow:** Pillow is used for handling image formats, converting between them, and applying transformations like resizing and rotation. It is central to the encoding and decoding processes of hidden data within images.
- **NumPy:** NumPy is used for efficient array handling, allowing image pixels to be represented as arrays for easy manipulation. This is crucial for applying quantum-inspired transformations and for encoding data into image pixels.

5.3.2 Flask for Deployment

Flask is a lightweight Python web framework used to deploy the quantum-inspired image steganography system as a web application.

- **API Development**: Flask is used to create RESTful APIs, enabling users to interact with the system by uploading images, encoding text into images, or decoding text from images. It serves as the bridge between the backend (image processing) and the frontend (user interface).

- **User Interaction:** Flask provides an interface where users can easily upload images, encode or decode data, and download results, making the process seamless and accessible.

5.3.3 GUI Framework

React.js was used to create a responsive user interface where users can easily upload images and view classification results. React enables dynamic updates to the UI based on the model's output, making the user experience smoother.

- **Real-Time Feedback**: The frontend sends images to the Flask API and displays results **User** and explanations in real-time.
- **Experience**: The GUI provides a simple, intuitive interface that does not require technical knowledge to use.

5.3.4 HTML, CSS, and JavaScript for Frontend

HTML, CSS, and JavaScript are used to build the user interface for the web application.

- **HTML:** HTML is used to structure the web pages, including forms for uploading images and text, as well as buttons for initiating encoding or decoding processes.
- **CSS:** CSS is used to style the web pages, ensuring a responsive and visually appealing interface. It enables users to interact with the system on various devices without issues.
- **JavaScript:** JavaScript is used to enhance the frontend by providing interactivity, such as image previews and dynamic feedback during encoding and decoding operations. It allows the interface to respond to user actions in real-time without requiring page reloads.

5.4 Coding Guidelines for Programming Language Used

5.4.1 Code Structure

- **Modular Design**: The code is organized into functions and classes, each handling a specific task, such as image manipulation, quantum-inspired transformations, encoding, decoding, or handling API requests. This promotes reusability, scalability, and easier debugging.

- **Separation of Concerns**: Each module is designed to perform a distinct responsibility, such as image processing, web interface handling, or data validation, ensuring that the system remains well-structured and manageable.

5.4.2 Naming Conventions

- **Variables**: Variable names are descriptive and self-explanatory (e.g., encoded_image, pixel_data, quantum_angle), making the code more understandable.
- **Functions**: Functions are named using snake_case (e.g.,encode_image, apply_quantum_rotation) to follow Python conventions and maintain consistency.
- **Classes**: Class names are written in CamelCase (e.g.,ImageProcessor, SteganographyApp) to distinguish them from variables and functions, ensuring readability and clarity.

5.4.3 Commenting and Documentation

- **Docstrings**: Each function and class is documented with clear and concise docstrings that describe their purpose, input parameters, and return values, improving code readability and maintainability.

Inline Comments: Inline comments are added to explain complex sections of code, such as quantum-inspired transformations or pixel manipulation logic, to help future developers understand the rationale behind specific implementations.

5.5 Pseudo Code for Each Module with Description

5.5.1 Input Module

```
def validate_input(image, secret_text):
    # Check for valid image format (e.g., jpg, png)
    if image_format not in ['jpg', 'png']:
        raise Error("Invalid image format")
    # Check if image size is within acceptable range
    if image.size > max_size:
        raise Error("Image too large")
    # Check if secret text is within the allowable character limit
    if len(secret_text) > max_text_length:
        raise Error("Text too long")
    return image, secret_text
```

Description: This module validates user input by checking the image format, size, and the length of the secret text. It ensures the inputs are suitable for encoding or decoding processes.

5.5.2 Preprocessing Module

```
def preprocess_image(image):
    # Resize the image to a fixed size
    image = resize(image, (256, 256))
    # Convert image to an array for processing
    image_array = convert_to_array(image)
    # Normalize pixel values between 0 and 1
    image_array = normalize(image_array)
    return image_array
```

Description: This module prepares the image for encoding or decoding by resizing it, converting it into an array, and normalizing the pixel values. These steps ensure compatibility with the quantum-inspired transformations.

5.5.3 Encoding Module

```
def encode_text_into_image(image_array, secret_text):
    # Convert text into binary representation
    binary_text = ''.join(format(ord(char), '08b') for char in secret_text)
    # Iterate over pixel values to encode the binary text
    index = 0
    for pixel in image_array:
        if index < len(binary_text):
            pixel = modify_pixel_with_bit(pixel, binary_text[index])
            index += 1
    return image_array
```

Description: This module encodes the secret text into the image by modifying pixel values based on the binary representation of the text. It ensures that the embedded data remains hidden without significantly altering the image quality.

5.5.4 Decoding Module

```
def decode_text_from_image(image_array):
    # Extract least significant bits from pixel values
    binary_text = ''
    for pixel in image_array:
        binary_text += extract_least_significant_bit(pixel)
    # Convert binary text back to string
    decoded_text = ''.join(chr(int(binary_text[i:i+8], 2)) for i in range(0, len(binary_text), 8))
    return decoded_text
```

Description: This module retrieves the embedded text by extracting the least significant bits from the image pixels and reconstructing the original secret text.

5.5.5 Explainability Module

```
def explain_encoding(image, secret_text):
    # Highlight pixel regions modified during encoding
    modified_pixels = get_modified_pixels(image, secret_text)
    # Visualize these regions on the image
    explanation_image = highlight_pixels(image, modified_pixels)
    return explanation_image
```

Description:
This module provides an explanation of the encoding process by identifying and highlighting the pixel regions modified to embed the secret text. It enhances transparency and user trust in the encoding process.

5.6 Summary

This chapter details the implementation of the quantum-inspired image steganography system, covering hardware and software requirements for efficient operation. Python is chosen for its versatility, supported by libraries like Pillow and NumPy, with Flask handling the web interface. The chapter includes pseudo code for modules such as input validation, image preprocessing, encoding, decoding, and explainability, ensuring a robust and user-friendly system.

Chapter 6

SYSTEM TESTING

System Testing in a quantum-inspired image steganography system ensures that all components function cohesively and meet the specified requirements. This phase verifies that processes such as image encoding, decoding, and user interactions with the web interface operate seamlessly. The testing also evaluates the system's robustness, accuracy, security, and performance under various scenarios.Key tests include validating input formats, encoding and decoding accuracy, and ensuring quantum-inspired randomness effectively secures hidden data while maintaining image quality. The system is tested for user-friendliness, responsiveness, and compatibility across devices and platforms.

6.1 Key Phases of System Testing:

1. Core Components to Test:

- **Image Processing Module:** Handles image resizing, normalization, and pixel-level manipulation for encoding and decoding.
- **Encoding and Decoding Algorithms:** Ensures that the quantum-inspired randomness secures data effectively while preserving image quality.
- **Web Interface:** Provides user-friendly interaction for uploading images, entering secret text, and downloading encoded results.
- **System Logging and Error Handling:** Logs user activities and identifies errors during encoding or decoding tasks.

6.2 Key Aspects of System Testing in QNN-Based Image Steganography System:

1. Functionality Testing:

- Input Validation:
    - Validate image formats (e.g., PNG, JPEG).
    - Ensure proper handling of image sizes and secret text lengths.
- Encoding Process:
    - Test embedding secret text into images while preserving visual itegrety.
- Decoding Process:
    - Verify that the embedded text can be accurately retrieved from encoded images.

2. Security Testing:

- Data Integrity:
    - Verify that encoded images do not reveal hidden patterns detectable by steganalysis tools.
- Unauthorized Access::
    - Test against unauthorized attempts to decode hidden data without permissions.
- Quantum-Inspired Randomness:
    - Ensure pixel modifications follow unpredictable patterns, enhancing security.
    - Use cryptographic techniques like zero-knowledge proofs if required.

3. Performance Testing:

- Encoding and Decoding Time:
    - Evaluate the time required to process high-resolution images.
- System Load:
    - Test system performance under multiplesimultaneous requests.

4. Integration Testing:

- Verify seamless interaction between the backend (Flask), image processing libraries (Pillow, NumPy), and the frontend (HTML, CSS, JavaScript).

5. Usability Testing:

- User Interface:
    - Test system performance under multiplesimultaneous requests.
- Device Compatibility:
    - Validate the application's accessibility across PCs, smartphones, and tablets.

6. Error Handling and Logging:

- Test system behavior for invalid inputs, unsupported formats, or interruptions.
- Validate that error messages are clear and logs provide actionable information.

7. Validation of Results:

- Confirm that secret text is correctly encoded and decoded without data loss.
- Verify that visual quality of the original image remains intact after encoding.

Tools Used in Testing:

- **Pillow (PIL):** For validating encoding and decoding image processes.
- **NumPy:** For testing array manipulations during encoding and decoding.

- **Postman:** To test API interactions between the frontend and backend.
- **JMeter:** For performance and load testing under simultaneous requests.
- **Selenium:** For automating and validating frontend functionality.

Example Test Scenarios:

- **Input Validation:** Test for unsupported image formats, large file sizes, and excessive text lengths.
- **Encoding and Decoding Accuracy:** Validate correct embedding and retrieval of secret text.
- **Performance Testing:** Simulate multiple concurrent requests and measure response times.
- **Error Handling:** Test for invalid inputs and interruptions during encoding, ensuring proper recovery mechanisms.

System testing ensures the quantum-inspired image steganography system is secure, reliable, and user-friendly, meeting all specified requirements for deployment in real-world applications.

Outcome of System Testing:

A successful system test ensures that the quantum-inspired image steganography system is **secure, reliable, scalable, and user-friendly**, meeting all specified requirements for deployment in real-world scenarios.

Unit Testing:

Unit testing is a mechanism where individual modules of the project are tested in isolation to ensure they function correctly. It is also referred to as differentiation testing, as each module is tested independently to identify faults and errors. Using module-level designs as a reference, significant functionalities are evaluated to detect issues within their boundaries.

This section outlines the successful test case for embedding and extracting hidden data from images using Quantum Neural Network (QNN)-inspired techniques. The primary focus is to verify the correct encoding of secret data into an image and accurate decoding of this data from the encoded image. The testing process also includes generating QR codes and calculating the Peak Signal-to-Noise Ratio (PSNR) to ensure the integrity and quality of the encoded images.

Variability in image formats, resolutions, and dimensions during data embedding can impact the performance of the system. Standardizing image formats and dimensions, as well as prioritizing high-quality images, helps reduce noise and maintain the fidelity of the hidden data. Proper loading, processing, and modification of the images are critical for ensuring the system's high performance and achieving the expected outcomes.

Factors such as color depth, pixel density, and image compression must be optimized to minimize the degradation of the image quality during the encoding process. This step verifies the correct embedding and retrieval of hidden data in the system, a crucial prerequisite for successful steganography and data security. Any errors in encoding or decoding can disrupt the pipeline and compromise the system's ability to conceal or retrieve sensitive information effectively. By ensuring that the encoding and decoding processes function accurately, the system lays the foundation for consistent data hiding, secure communication, and effective image analysis.

Table 6.1: Test Case for Dropdown Options Display

Test Case Sl. No	1
Test Name	Verify dropdown options are displayed correctly.
Test Feature	Ensure the dropdown menu includes all required options.
Output Expected	Dropdown menu displays the following options: • Select an option • Encode Data into Image • Decode Data from Image • Generate QR Code • Calculate PSNR
Output Obtained	Dropdown menu displays all options as expected.
Result	Successful as shown in Snapshot 7.1.1

Table 6.1 outlines the test case for ensuring that the dropdown menu displays all required options related to the image steganography system. These options, including encoding, decoding, QR code generation, and PSNR calculation, form the foundation for accessing system functionalities. Proper display ensures user convenience and seamless navigation across all tasks.

Table 6.2: Test Case for "Generate QR Code" Option

Test Case Sl. No	2
Test Name	Generate QR Code from text.
Test Feature	Verify that a QR code is successfully generated from the provided text.
Output Expected	QR code image returned as a Base64-encoded string.
Output Obtained	QR code generated and returned as Base64 string.
Result	Successful as shown in Snapshot 7.1.4

Table 6.2 outlines the test case for testing the QR code generation functionality. This feature enables the conversion of textual data into a QR code for efficient and secure sharing. Ensuring this process is reliable guarantees that users can generate scannable QR codes with high readability and data integrity. The functionality must handle various data types, lengths, and special characters seamlessly to ensure versatility. It also ensures that the generated QR codes comply with standard formats and can be scanned using any QR code reader without compatibility issues. Proper testing includes evaluating the resolution, size, and contrast of the QR code to ensure optimal performance across different devices and lighting conditions. Moreover, error correction capabilities embedded within the QR code need to be validated, ensuring the code remains scannable even if partially damaged. This functionality plays a crucial role in securely embedding sensitive information, such as URLs or encrypted text, for safe transmission. By ensuring robustness in QR code generation, the system enhances user confidence and establishes a secure medium for data sharing in steganographic applications. Additionally, testing ensures that no data corruption occurs during QR code generation, guaranteeing reliability. This forms a foundation for implementing more complex data-hiding techniques securely.

Table 6.3: Test Case for "Encode Data into Image" Option

Test Case Sl. No	3
Test Name	Encode secret data into an image.
Test Feature	Verify that secret data is successfully encoded into an image.
Output Expected	Encoded image is returned as a downloadable PNG file.
Output Obtained	Encoded image returned successfully for download.
Result	Successful as shown in Snapshot 7.1.5

Table 6.3 outlines the test case for verifying the functionality of encoding data into an image. This step involves integrating secret data into an image using quantum-inspired techniques. Ensuring the functionality works as expected is critical for embedding sensitive information securely into images with minimal distortion to the original data. The encoding process should handle various data formats, including text, binary, or encrypted files, and seamlessly integrate them into the image. Robust testing ensures the embedded data remains intact and recoverable, even when the encoded image is subjected to compression, resizing, or noise. Additionally, the encoding algorithm must ensure that the visual quality of the image is preserved to avoid suspicion, maintaining its usability in real-world scenarios. Quantum-inspired methods enhance the security of the encoding process, making it resistant to conventional steganalysis techniques. The functionality also needs to address compatibility with different image formats such as PNG, JPEG, and BMP, ensuring broad applicability. Proper error handling mechanisms must be in place to alert users if the image lacks the capacity to hold the desired data. The success of this feature guarantees a reliable foundation for the secure exchange of hidden information in multimedia files. By validating this functionality, the system ensures a critical step in achieving robust and advanced steganographic solutions.

Table 6.4: Test Case for "Decode Data from Image" Option

Test Case Sl. No	4
Test Name	Decode secret data from an image.
Test Feature	Verify that secret data is successfully decoded from the given image.
Output Expected	Decoded text matches the original secret data.
Output Obtained	Decoded text retrieved successfully.
Result	Successful as shown in Snapshot 7.1.6

Table 6.4 describes the test case for verifying the decoding process of retrieving embedded data from an image. This functionality is essential for accurately extracting hidden information using the steganographic method. Proper decoding ensures that sensitive data can be reliably retrieved without data loss or corruption. The process involves analyzing the image's encoded regions and reconstructing the original data, maintaining its integrity and usability. Robust testing of this feature ensures that the decoding mechanism is resilient to potential distortions, such as compression, noise, or resizing, that the image may undergo. It is crucial that the method works across diverse image formats and resolutions to ensure compatibility and flexibility. The system should provide clear error messages if the decoding fails due to incomplete or corrupted input data. Moreover, the feature must validate the authenticity of the extracted data to ensure it matches the original payload. The accuracy of the decoding process is vital for preserving the trustworthiness and practicality of the steganographic solution. By guaranteeing a high success rate in retrieving data, this feature forms the backbone of secure and reliable information sharing in sensitive applications. Testing this step ensures the system meets its promise of data confidentiality and precision.

Table 6.5: Test Case for Verifying "PSNR" Ratio

Test Case Sl. No	5
Test Name	Verify "Calculate PSNR" option.
Test Feature	Check that selecting this option loads the corresponding functionality.
Output Expected	"Calculate PSNR" functionality is triggered, and a form to upload two images for comparison is displayed.
Output Obtained	Correct functionality triggered for PSNR calculation.
Result	Successful as shown in Snapshot 7.1.7

Table 6.5 describes the test case for calculating the Peak Signal-to-Noise Ratio (PSNR). This functionality evaluates the quality of images after encoding or decoding. A high PSNR value indicates minimal distortion during the steganographic process, which is vital for maintaining the integrity of the visual data. The calculation involves comparing the original and processed images to determine the mean squared error, which inversely correlates with the PSNR value. This metric is critical for assessing the balance between data hiding capacity and image quality, ensuring the encoded image remains visually acceptable. By validating PSNR, the system ensures that the encoding process does not compromise the usability or perceptibility of the image. Testing also verifies the accuracy of the PSNR algorithm across different image formats, resolutions, and compression levels. A consistent PSNR value across various scenarios confirms the robustness of the steganographic technique. A low PSNR value should trigger an alert or suggestion to optimize encoding parameters for better results. This functionality ensures that the system achieves a delicate balance between security, data capacity, and image fidelity while promoting user confidence and usability.

Table 6.5: Test Case to Verify default Selection

Test Case Sl. No	6
Test Name	Verify default selection.
Test Feature	Ensure the dropdown menu's default option is "Select an option ".
Output Expected	Default dropdown selection is "Select an option."
Output Obtained	Default option correctly displayed.
Result	Successful

Table 6.6 outlines the test case for verifying the default state of the dropdown menu. The default option, "Select an option," ensures user clarity by prompting them to choose a specific functionality. This step ensures a user-friendly interface and avoids accidental triggering of unintended operations.

6.3 Summary

This chapter covers system testing of the quantum-inspired image steganography system to ensure functionality, security, and performance. Key tests validate encoding, decoding, input formats, and image quality preservation. Components like the image processing module, algorithms, and web interface are evaluated using tools like Pillow, NumPy, Flask, and Selenium. Test cases include QR code generation, PSNR calculation, and usability across devices. The chapter ensures the system is robust, secure, and ready for deployment.

Chapter 7

RESULTS AND DISCUSSION

The quantum-inspired image steganography system offers enhanced security, robustness, and usability compared to traditional steganography methods. The use of quantum-inspired randomness ensures the hidden data is highly secure and resistant to modern steganalysis tools, while maintaining the quality of the original image. The system provides a user-friendly interface, enabling seamless encoding and decoding of secret data. Despite its advantages, challenges such as optimizing processing time for high-resolution images, ensuring scalability for large datasets, and enhancing cross-platform compatibility require further refinement for broader adoption.

7.1 Experimental Results

7.1 Snapshot of Home Page

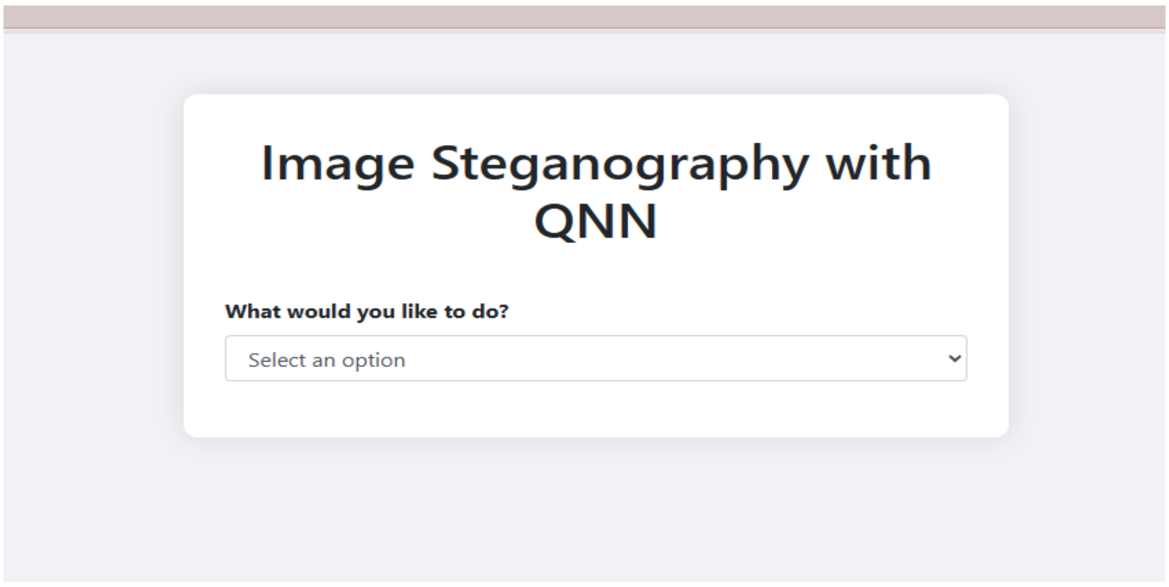

Figure 7.1 : Snapshot of Home Page

This figure 7.1 as snapshot is the homepage for an **Image Steganography Application** using **Quantum Neural Network (QNN)** techniques. Users can choose options to encode, decode, generate QR codes, or compare images seamlessly using a dropdown menu.

7.2 Home Page with Feature Selection

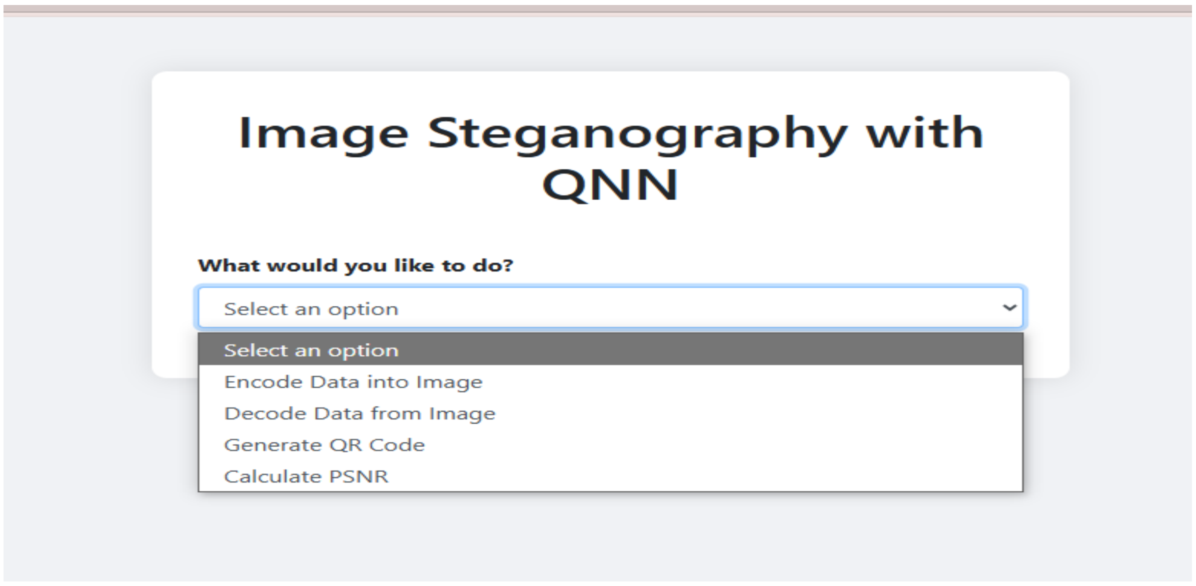

Image Steganography with QNN

What would you like to do?

Select an option

Figure 7.2 : Home Page with Feature Selection

The figure 7.2 as snapshot offers a dropdown menu with options for encoding/decoding data, generating QR codes, and calculating PSNR, enabling secure and efficient image processing.

7.3 QR Code Generation Interface

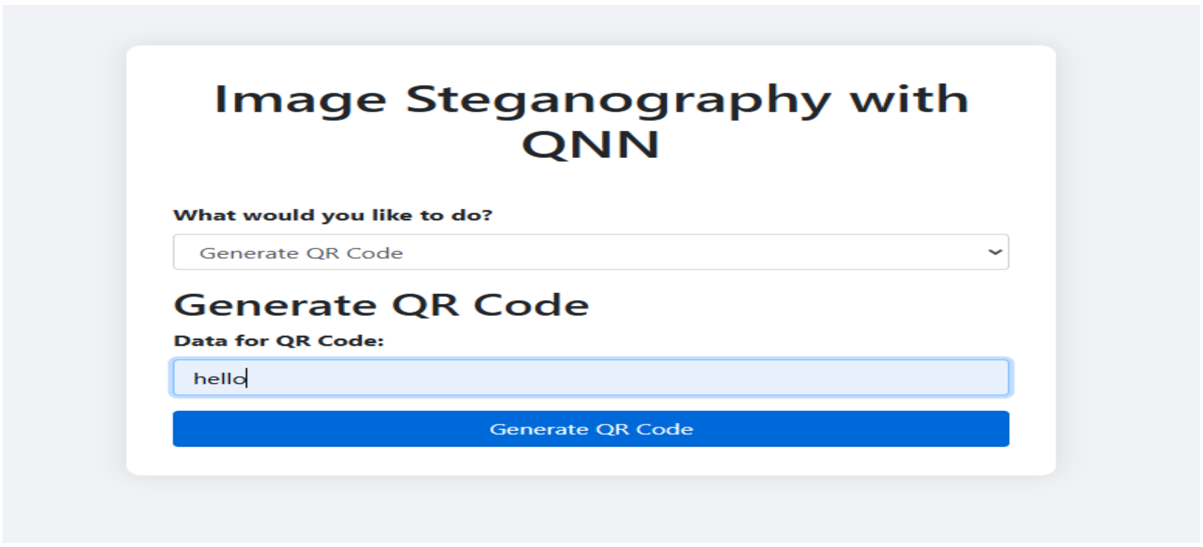

Image Steganography with QNN

What would you like to do?

Generate QR Code

Generate QR Code

Data for QR Code:

hello

Figure 7.3 : QR Code Generation Interface

This figure 7.3 as snapshot allows the user to generate a **QR Code** by inputting custom text or data. The generated QR Code can be downloaded for further use in steganography or other applications.

7.4 QR Code Generated Interface

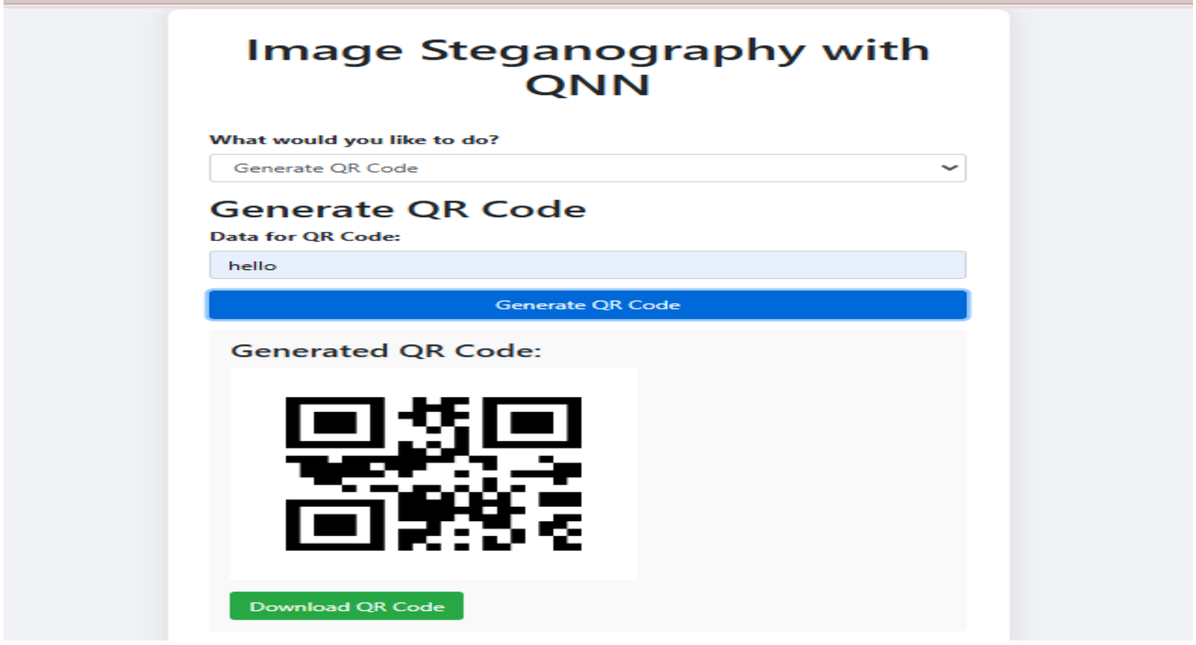
Image Steganography with QNN

What would you like to do?

Generate QR Code

Generate QR Code

Data for QR Code:

hello

Figure 7.4 : QR Code Generated Interface

The figure 7.4 as snapshot showcases an interface for generating QR codes. Users can input data, click "Generate QR Code," and download the resulting QR code.

7.5 Data Encoding into QR Code using QNN

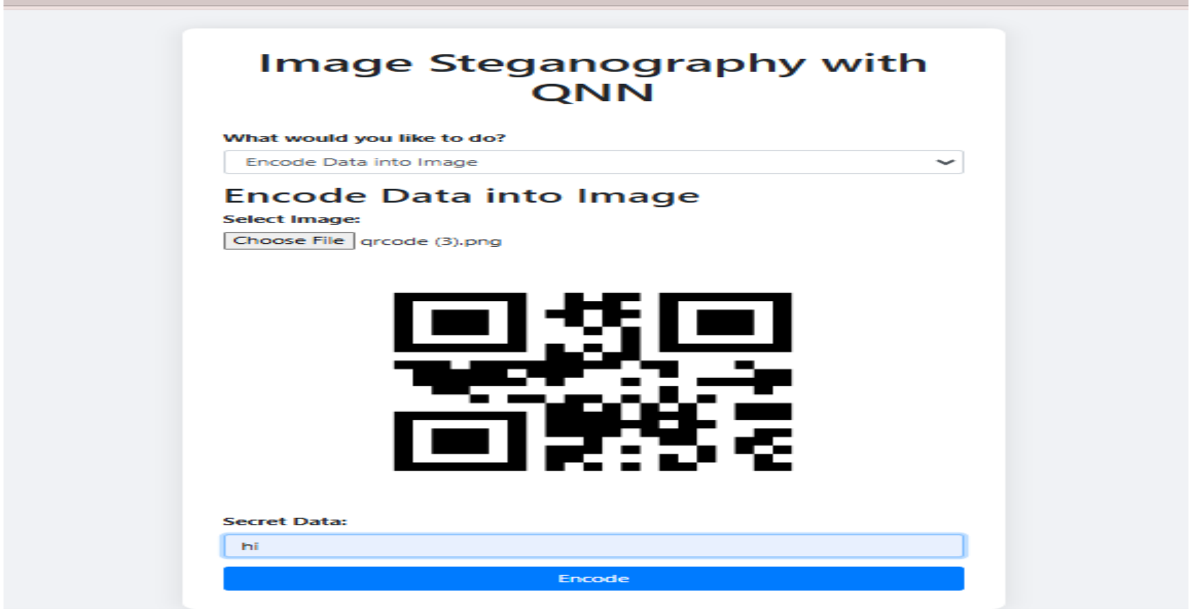
Image Steganography with QNN

What would you like to do?

Encode Data into Image

Encode Data into Image

Select Image:

Choose File qrcode (3).png

Secret Data:

hi

Figure 7.5 : Data Encoding into Image using QNN

The figure 7.5 as snapshot demonstrates an interface for encoding secret data into a QR code image. Users can upload a QR code image, input secret data (e.g., "hi"), and click "Encode" to embed the data.

7.6 Data Decoding from QR Code using QNN

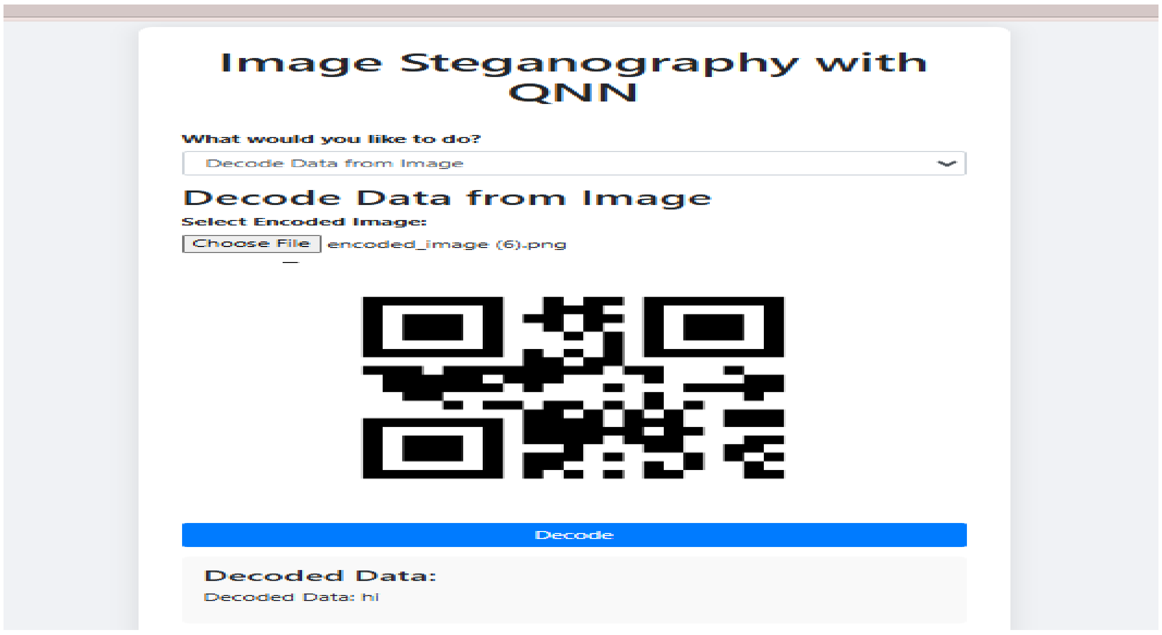

Image Steganography with QNN

What would you like to do?

Decode Data from Image

Decode Data from Image

Select Encoded Image:

Choose File encoded_image (6).png

Decode

Decoded Data:

Decoded Data: hi

Figure 7.6 : Data Decoding from Image using QNN

The figure 7.6 as snapshot displays an interface for decoding embedded data from a QR code image. Users can upload an encoded image, click "Decode," and view the extracted data.

7.7 PSNR Value Calculation Interface

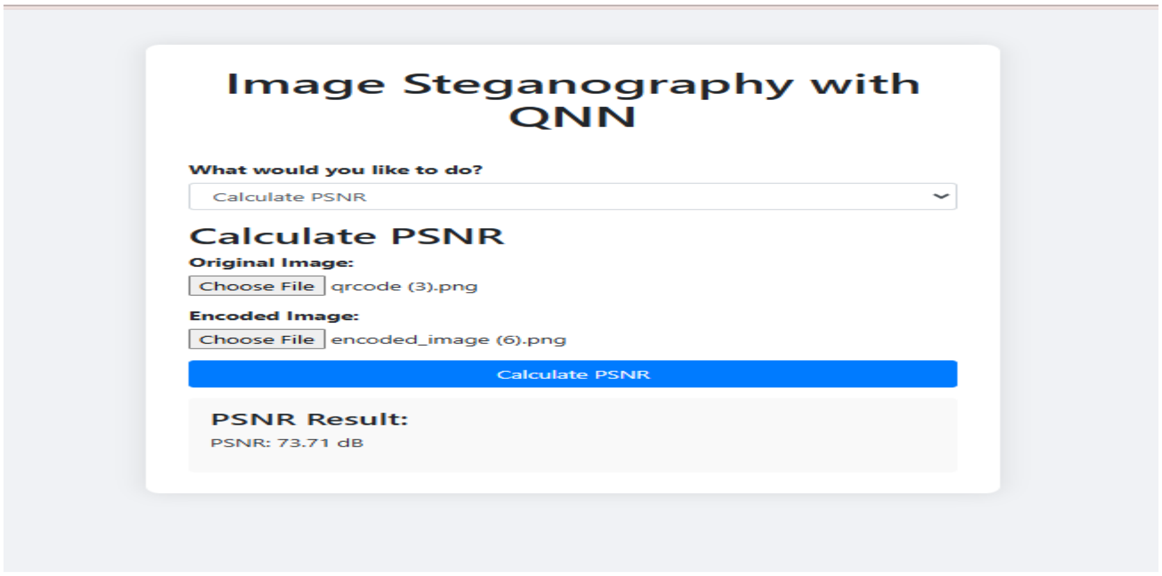

Image Steganography with QNN

What would you like to do?

Calculate PSNR

Calculate PSNR

Original Image:

Choose File qrcode (3).png

Encoded Image:

Choose File encoded_image (6).png

Calculate PSNR

PSNR Result:

PSNR: 73.71 dB

Figure 7.7 : PSNR Value Calculation Interface

This figure 7.7 as snapshot enables users to calculate the **Peak Signal-to-Noise Ratio (PSNR)** between two images.

7.8 Quantum Inspired v/s Traditional Steganography

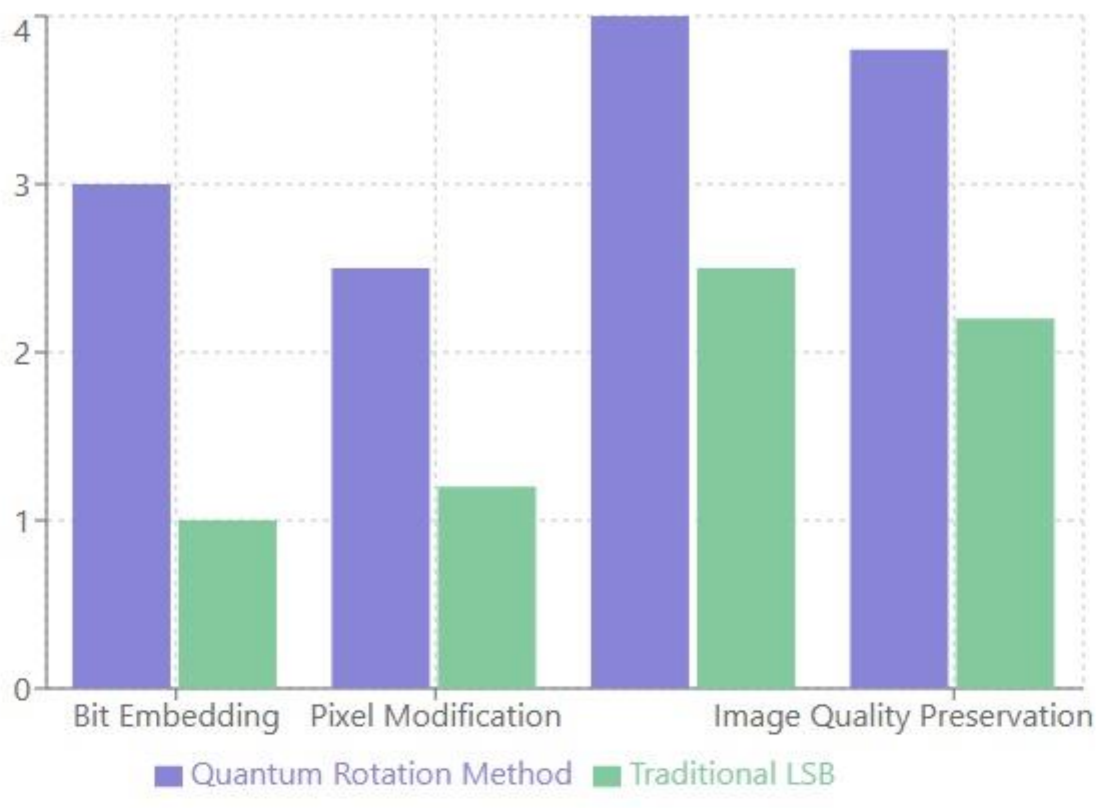

Figure 7.8 : Quantum Inspired v/s Traditional Steganography

Key Observation:

- Quantum rotation method shows improved performance across metrics
- Higher scores indicate better steganographic capabilities
- Metrics include embedding efficiency,pixel modification,data capacity and image quality preservation

7.2 Summary

This chapter presents the experimental results in section 7.1, which consists of snapshots of the Home page, Home Page with Feature Selection, QR Code Generation Interface, Data Encoding into Image using QNN, Data Decoding from Image using QNN and PSNR Value Calculation Interface, The graph 7.8 shows that the Quantum Rotation Method outperforms the traditional LSB method in steganography across various metrics, indicating better performance..

Chapter 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 Conclusion

The quantum-inspired image steganography system represents a groundbreaking approach to secure and covert communication. By leveraging principles of quantum randomness, the system ensures that hidden data is highly secure, resistant to detection by modern steganalysis tools, and maintains the quality of the original image. The integration of a user-friendly web interface simplifies the process of encoding and decoding secret data, making the system accessible to both technical and non-technical users. This innovative approach addresses the limitations of traditional methods, such as predictable patterns in pixel modifications and limited security, providing a robust solution for safeguarding sensitive information.

The successful implementation of this system demonstrates its potential in various fields, including personal data protection, corporate security, and confidential communication. However, challenges such as optimizing processing times for high-resolution images, ensuring scalability for large-scale applications, and enhancing cross-platform compatibility need to be addressed for widespread adoption. Despite these challenges, the quantum-inspired image steganography system lays a solid foundation for secure data transmission in the digital era and highlights the potential of integrating quantum-inspired techniques into practical solutions.

8.2 Future Enhancement

Future enhancements for the quantum-inspired image steganography system could focus on improving processing speed for large datasets and high-resolution images. Integrating advanced cryptographic techniques like homomorphic encryption and expanding support for multiple media formats will enhance security and versatility. Optimizing for real-time encoding, decoding, and mobile-friendly interfaces will make the system more accessible, paving the way for broader applications in secure communication and data protection.

REFERENCES

[1]. Jehn-Ruey Jiang “A Quick Overview of Quantum Machine Learning” IEEE 2023.

[2]. Manjunath T D, Biswajit Bhowmik “Quantum Machine Learning and Recent Advancements” IEEE 2023.

[3]. Shahid Rahman, Jamal Uddin, MuhammHilal Ahmad Bhat, Farooq Ahmad Khanday,brajesh Kumar Kaushik, Faisal Bashir,and Khurshed Ahmad Shah “Quantum Computing: Fundamentals,Implementations and Applications” IEEE 2022.

[4] Zakarya,hameed Hussain, Ayaz Ali Khan, Aftab Ahmed And Muhammad Haleem “A Comprehensive Study of Digital Image Steganographic Techniques”IEEE 2023.

[5].S. Sravani, R. Ranjith ” Image Steganography For Confidential Data Communication” IEEE 2021.

[6]. Dr.Mayank Srivastava, Pratibha Dixit, Shikha Srivastava “Data Hiding using Image Steganography” IEEE 2023.

[7]. Juned Ahmed Mazumder, K. Hemachandra “Image Steganography Using the Fusion of Quantum Computation and Wavelet Transformation” IEEE 2019.

[8]. R S Randhawa, Amey R Hasabnis, Sanghmitra Rai “Quantum-Based Colour Image Steganography” IEEE 2023