# A PROJECT REPORT

on

"Implementation of LSB based Image Steganography: HideNSeek"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE AND ENGINEERING

BY

AMAN SAHU 1805278

UNDER THE GUIDANCE OF
MANAS RANJAN LENKA

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA - 751024
April 2022

A PROJECT REPORT

on

# “Implementation of LSB based Image Steganography: HideNSeek”

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR’S DEGREE IN
COMPUTER SCIENCE AND ENGINEERING

BY

AMAN SAHU 1805278

UNDER THE GUIDANCE OF
MANAS RANJAN LENKA


SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA -751024

April 2022

# KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

# CERTIFICATE

This is certify that the project entitled

"Implementation of LSB based Image Steganography: HideNSeek"

submitted by

AMAN SAHU 1805278

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Technology (Computer Science & Engineering) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2021-2022, under our guidance.

Date: 16/04/2022

Manas Ranjan Lenka
Assistant Professor,
School of Computer Engineering,
KIIT University, Bhubaneswar.

# Acknowledgements

I am profoundly grateful to **MANAS RANJAN LENKA** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion. I would also like to thank MR. BHABANI SHANKAR PRASAD MISHRA, the Dean of Computer Engineering for providing us with all the resources and opportunities to help me sharpen my skills.

AMAN SAHU

# ABSTRACT

HideNSeek is an end to end application which allows for hiding a message in a given cover image using Least Significant Bit based Image Steganography techniques, and also allows for extracting the said embedded message from the cover image. The website offers the users forms to fill out with the text message they want to hide and the cover image onto which they want to hide the message. The website is built with the help of React.js. The API's (Application Programming Interfaces) on the server side are built using Flask, the logic for embedding the message is also written using Python.

Cryptography is the study of hiding the meaning of messages, Steganography is the study wherein the existence of a message is hidden from the world. An encrypted message, although hides the meaning of the message, it fails to conceal the presence of the message. When looking at an encrypted message, it becomes obvious that there is something being tried to hide. Steganography allows us to embedded messages into various types of other media including text documents, images, audio files and video files. This way, the existence of the message is unknown.

**Keywords:** Steganography, Cryptography, LSB, Embedding, Image, AES, Flask, React

# Contents

# List of Figures

# Chapter 1

# Introduction

HideNSeek is a full stack application which implements the Image Steganography technique in order to embed a given message onto a color image.

Steganography is a very useful technique where we can hide the existence of our message by hiding it in different forms of media. Image Steganography uses Image as a medium to hide the message from the naked eye.

Encryption is the process of hiding the meaning of a message. HideNSeek uses AES-256 bit encryption for hiding the meaning of the message and then proceeds to embed said ciphered message into the provided cover image.

Such an application could prove critical to hide and transmit sensitive data or information over digital mediums as the application implements the age old saying of “hiding in plain sight” in the modern world, where unless one is specifically performing Steganalysis on the images, it would be hard to know the existence of any meaning hidden in the image.

This report structures itself by introducing the application, followed by the basic concepts and literature involved in the implementation of the inner workings of the application, then moving on to specifics about the application itself and then concluding with the results and future works that can be done to improve on the application

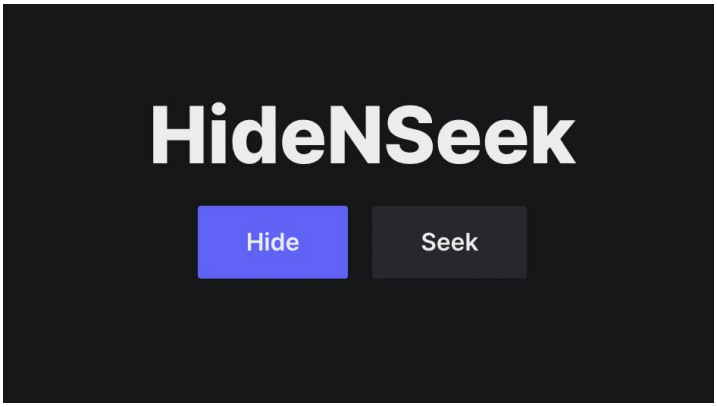


Figure 1.1: HideNSeek webpage snippet

# Chapter 2

# Basic Concepts

This section contains the basic concepts about the related tools and techniques used in this project.

## 2.1 Cryptography

Cryptography provides for secure communication. It consists of two processes, encryption and decryption. Encryption takes in a secret key and then uses that to transform the inputted plain text into an encrypted ciphertext. Decryption uses the same algorithm and key to transform the ciphertext back to the original message/plain text.

Cryptography uses techniques that help to protect information using concepts of mathematics and rule based calculations known as algorithms to transform messages in ways that make them harder to decode.

The main takeaway for this project, is that cryptography revolves around hiding the meaning of the data so that it is not easily understood by third parties involved.

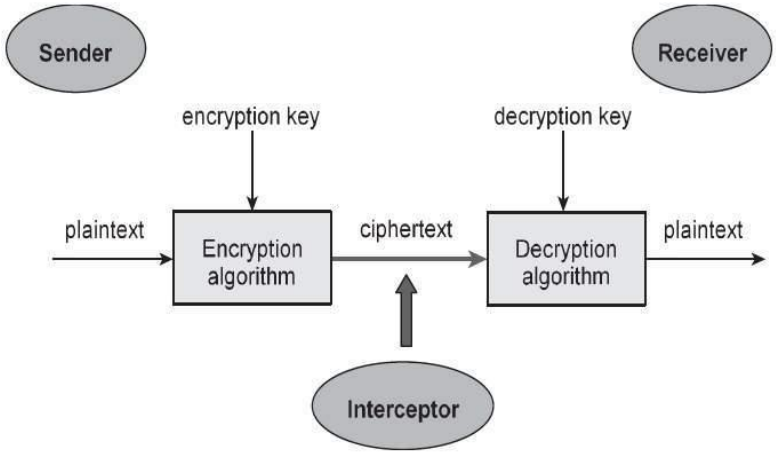


Figure 2.1: Simple Cryptographic System

## 2.2 Steganography

Steganography is the practice of hiding a message in other messages or a physical object. In computing/electronic contexts, a computer file, message, image, or video is hidden in another file, message, image, or video.

When compared to cryptography, cryptography is the practice of protecting the contents of a message alone, but steganography deals with hiding the fact that a secret message is being sent and also hiding its contents.

The advantage that steganography has over cryptography is that the secret message does not attract attention to itself since plainly visible encrypted messages arouse interest despite them being difficult to decode.

Image Steganography is the process of hiding data within an image file. The image onto which the message will be embedded is called the cover image and the image obtained after steganography, after having the message embedded onto it is called the stego image or the embedded cover image.

Different types of Image Steganography techniques are:

- Least Significant bit insertion
- Masking and Filtering
- Redundant Pattern Encoding
- Encrypt and Scatter
- Coding and Cosine Transformation

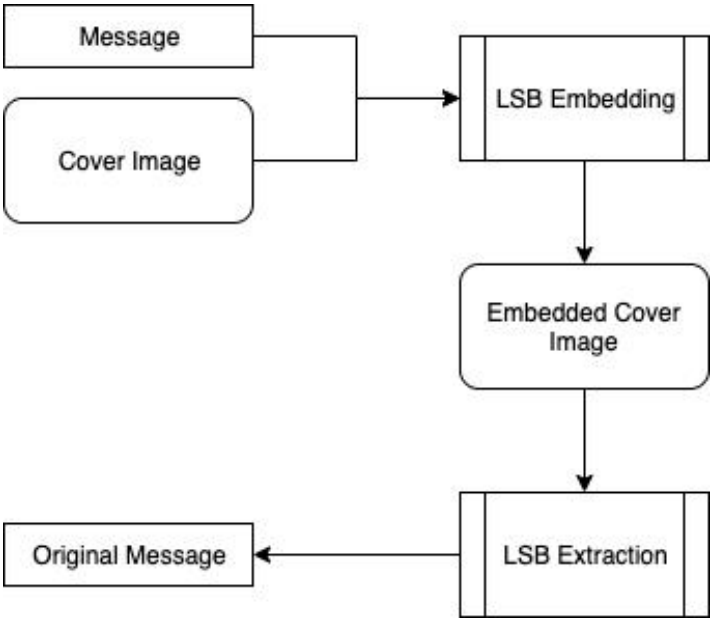


Figure 2.2: Image Steganography Block Diagram

# Chapter 3

# Problem Statement / Requirement Specifications

- To implement Least Significant Bit based Image Steganography.

- To build a website, the frontend of the website must be interactive and must take in an HTML form with appropriate plain text, cover image and key during “Hide”(embedding) portion of the steganography, and key, embedded cover image during the “Seek”(extraction) portion of the steganography.

- The front end of the website/webpage must be able to communicate with a backend server which provides for two API(Application Programming Interface) endpoints one for embedding and one for extraction process of the Steganography procedure.

## 3.1 Project Planning

To meet the project requirements the project can be divided into the following sections each with the specific tasks:

- Front end of the website:
  - Build an interactive HTML Form
  - Make API calls
  - Add Styling
  - Use a better library to make the website more functional and organised (e.g. react.js)
  - Add other web pages to the website to explain the use of the website

- API's
  - An API to receive a POST request for "Hide" portion, returns a downloadable stego image on success
  - An API to receive a POST request for "Seek" portion, returns the extracted plain text on success

- Logic for Image Steganography
  - Take in an image and plain text with key as input
  - Encrypt the plain text with the key
  - Embed cipher text into the image
  - Save the stego image
  - Take in the stego image and key as input
  - Extract the encrypted text
  - Decrypt the extracted text using the key to reveal the original plain text

## 3.2 System Design

### 3.2.1 Block Diagram

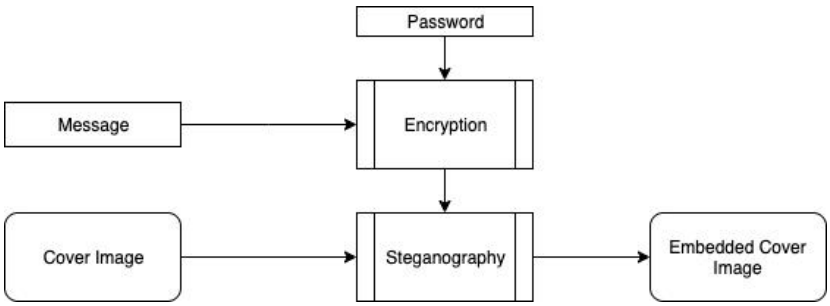


Figure 3.1: Block diagram for "Hide": Embedding portion of implemented Image Steganography

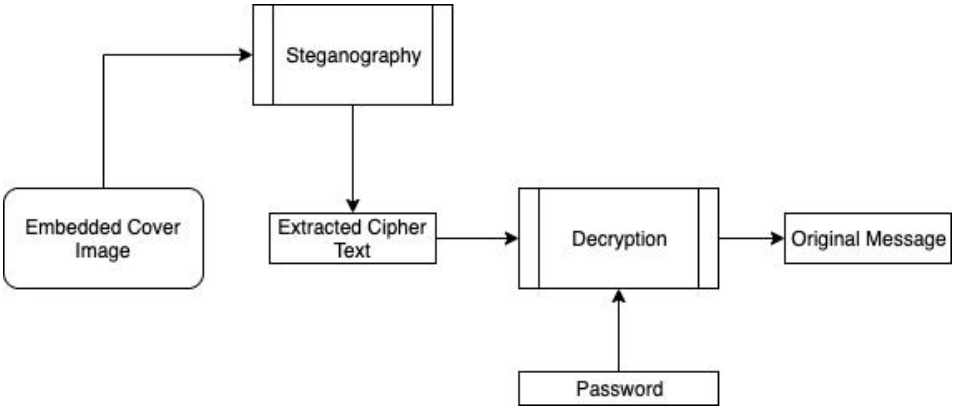


Figure 3.2: Block Diagram for “Seek”: Extraction portion of implemented Image Steganography

# Chapter 4

# Implementation

## 4.1 Implementation Choices

**Python**:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language, created by Guido van Rossum.

Features of Python:

- It is an interpreted and high level language.
- It is open source and free
- It is Interactive − One can sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python supports Object Oriented Programming
- It has a lot of libraries and support for various functionalities built-in.
- Due to a lot of libraries and support, it becomes very easy to use for implementing various functionalities since they donot have to be implemented from scratch

**React.js**:

ReactJS is a declarative, efficient, and versatile JavaScript library for building reusable UI components. It's an open-source, component-based front end library which is responsible just for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.

**Flask**:

Flask is a micro web framework written using Python. It's categorized as a micro-framework as it does not require specific tools or libraries. There are no database abstraction layer, form validation, or any other components in Flask. Flask has extensions that can add features for object-relational mapping, form validations, uploading and other open authentication technologies.

## 4.2 Implementation Details

Before the entire logic behind the Steganography implementation, Fernet library was used since it already had implementations for encryption. A user defined class was created to be able to use Fernet for encrypting and decrypting data, abstracting the other implementational details such as key/token generation using inputted password.

Different types of Image Steganography techniques are:

- Least Significant Bit insertion
- Masking and Filtering
- Redundant Pattern Encoding
- Encrypt and Scatter
- Coding and Cosine Transformation

Image Steganography was implemented by embedding the message in the least significant bits of the image pixels. This procedure was proposed by Arun Kumar Singh et al. (2015).

The paper proposes the following,
For the input message: “Hello”
Each character is represented in its byte form, where
H is 01001000
e is 01100101
etc.
Since we are working with RGB images, each pixel has three values, reach value corresponding to one band of the image.
Let pixel1 = (R, G, B) = (121, 123, 134).
Then each value of pixel1 in 8bit binary would be
121 = 01111001
123 = 01111011
134 = 10000110

Now, for every bit of the input character ‘H’ = 01001000, we replace the last bit of one band value of a pixel consecutively,
So, for
121 = 01111001 for H(**0**1001000) => 0111100**0** = 120
123 = 01111011 for H(0**1**001000) => 0111101**1** = 123
134 = 10000110 for H(01**0**01000) => 1000011**0** = 134

Thus, for the resultant stego-image, the corresponding pixel would become (120, 123, 134).
Such process is continued for all bits of each character of the input message.

So, in essence to embed a message of n characters, we would need at least [(n*8)/3] number of pixels in the cover image.

The program is implemented in Python 3.8. Before embedding the plain text, the plain text is first encrypted using the above mentioned user-defined class for encryption. Then the cipher text is padded with a custom word/end marker to mark the end of the cipher text, to help in extraction of cipher text during the “Seek” part.

The website was built using the react.js library, making use of hooks, components and state values to make the website responsive.

The API end points are built using Flask. Django was not used because the use case was less intensive and did not require the Object-Relation Model (ORM), which is the main feature of Django.

Both the end points are POST HTTP requests, taking the form data as the body of the request, then using the above mentioned classes and code of the Image Steganography implementation, to do the required processing and then return the appropriate response.

## 4.3 Screenshots

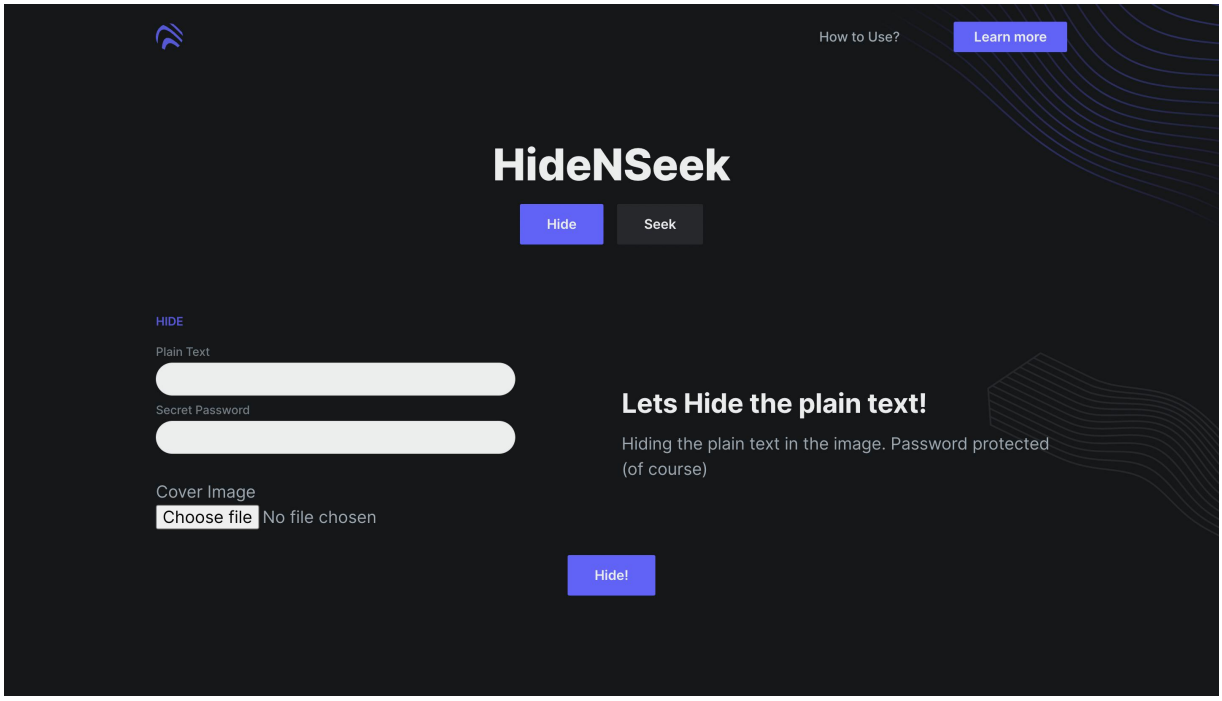


Figure 4.1: “Hide” section on the website

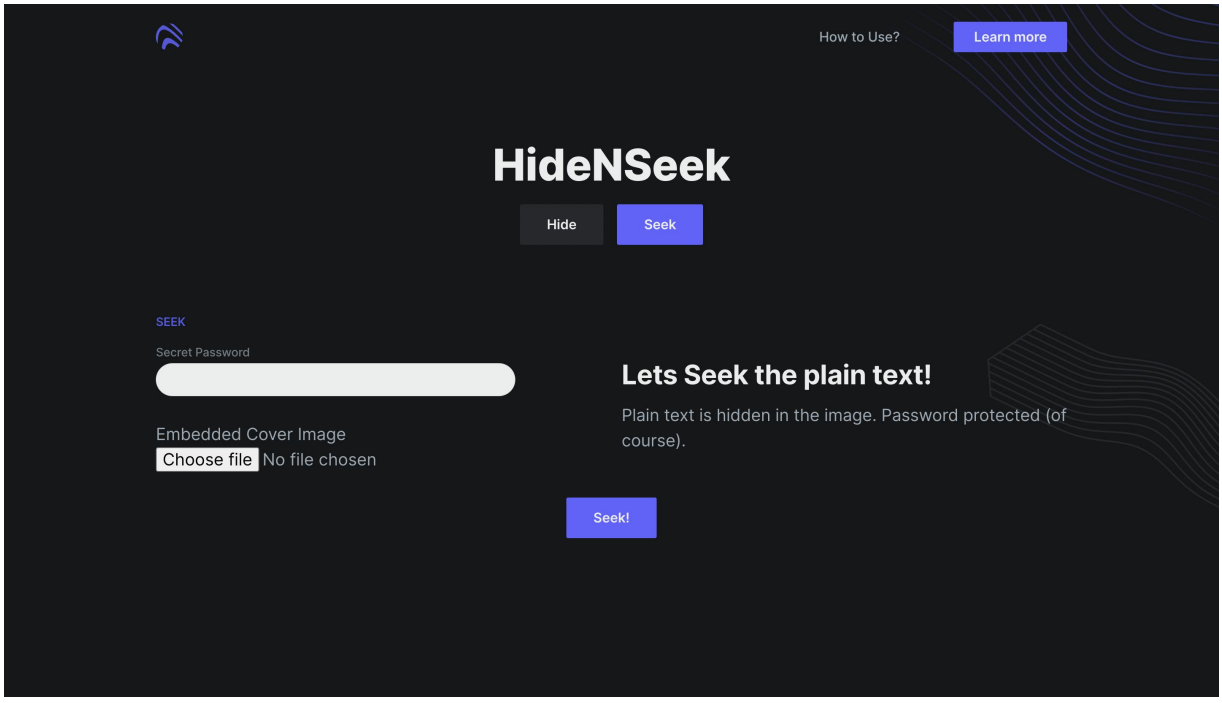


Figure 4.2: “Seek” section on the website

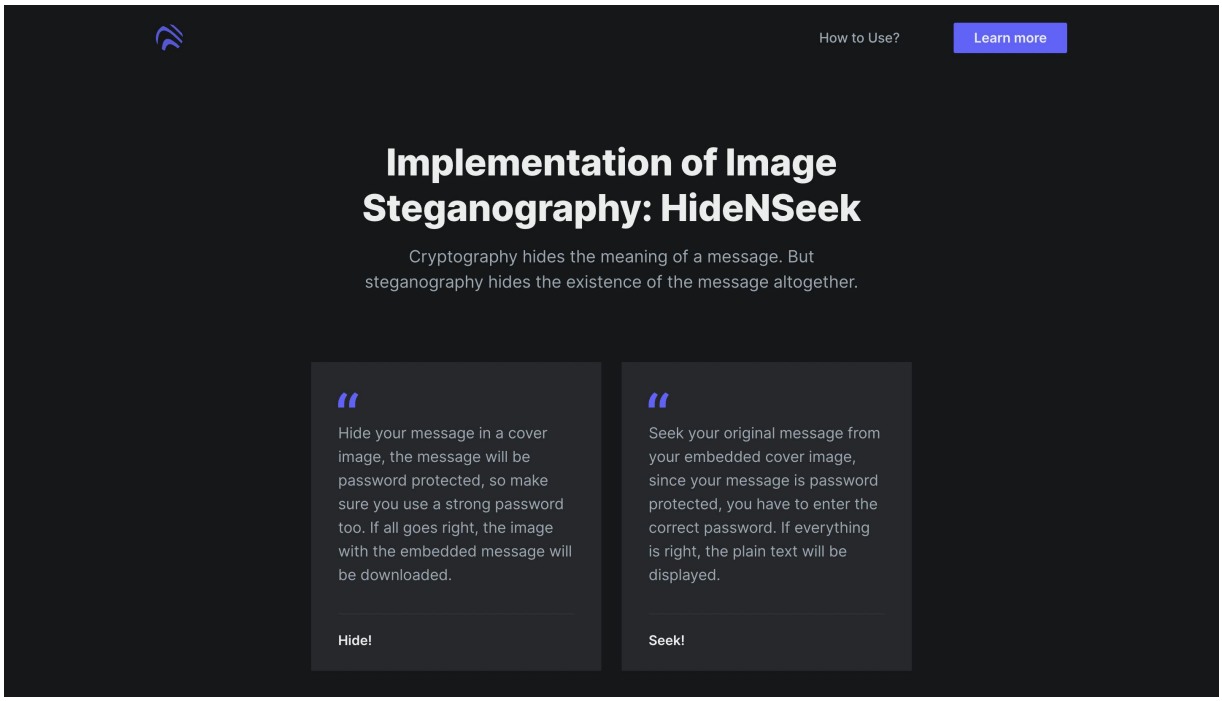


Figure 4.3: “How to” section

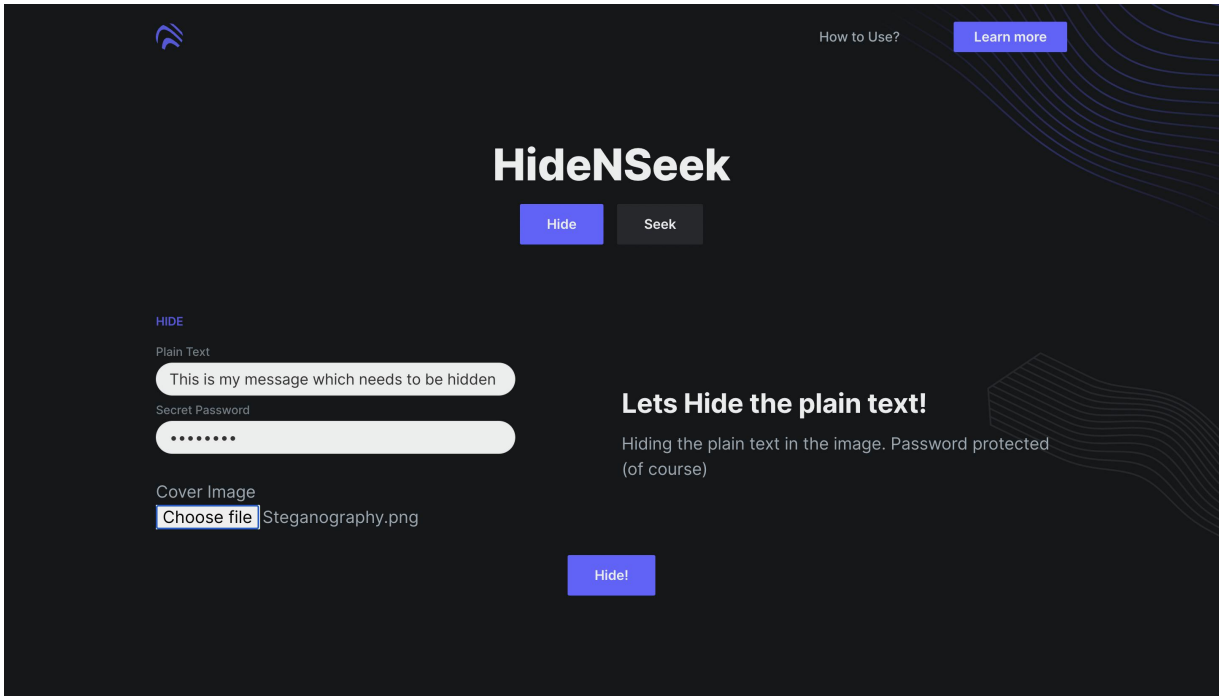


Figure 4.4: Filling the form for “Hide”

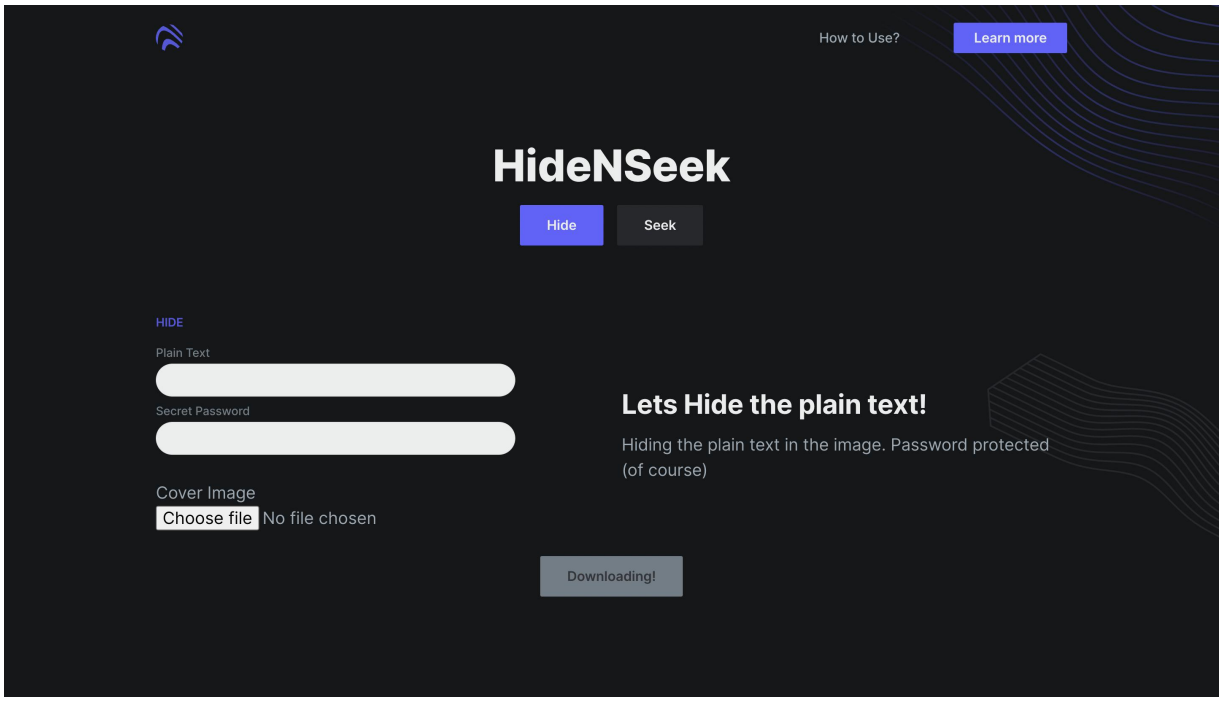


Figure 4.5: Processing, download in progress

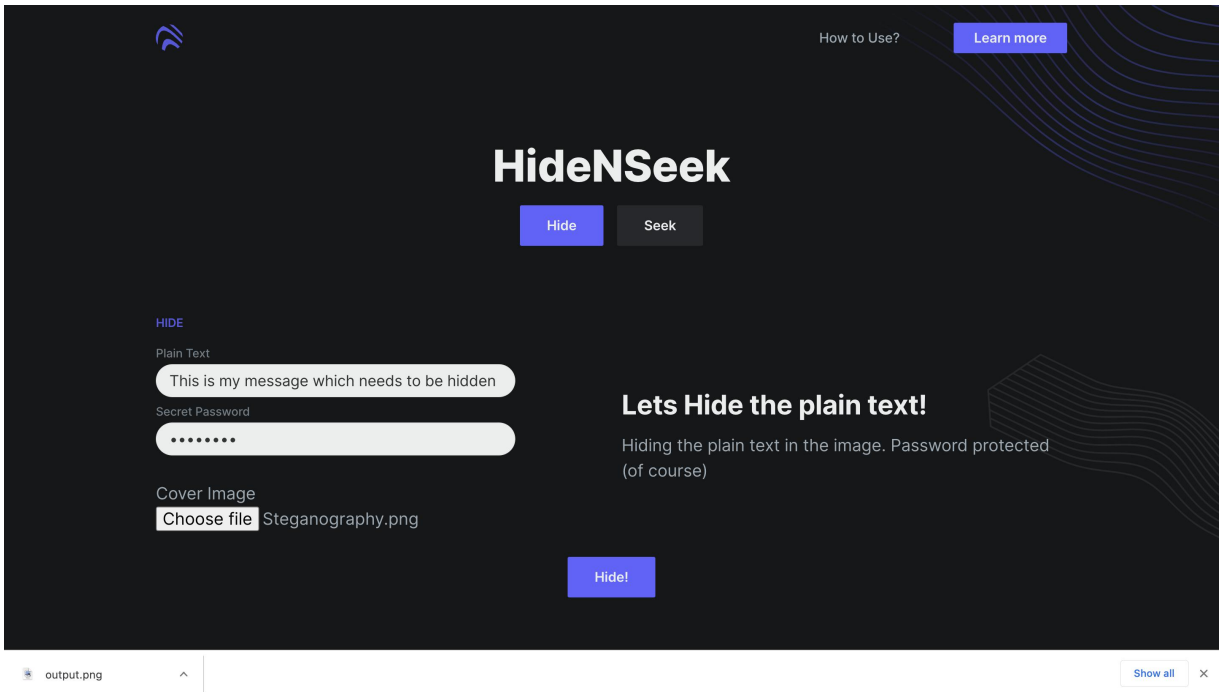


Figure 4.6: Steg-image downloaded as output.png

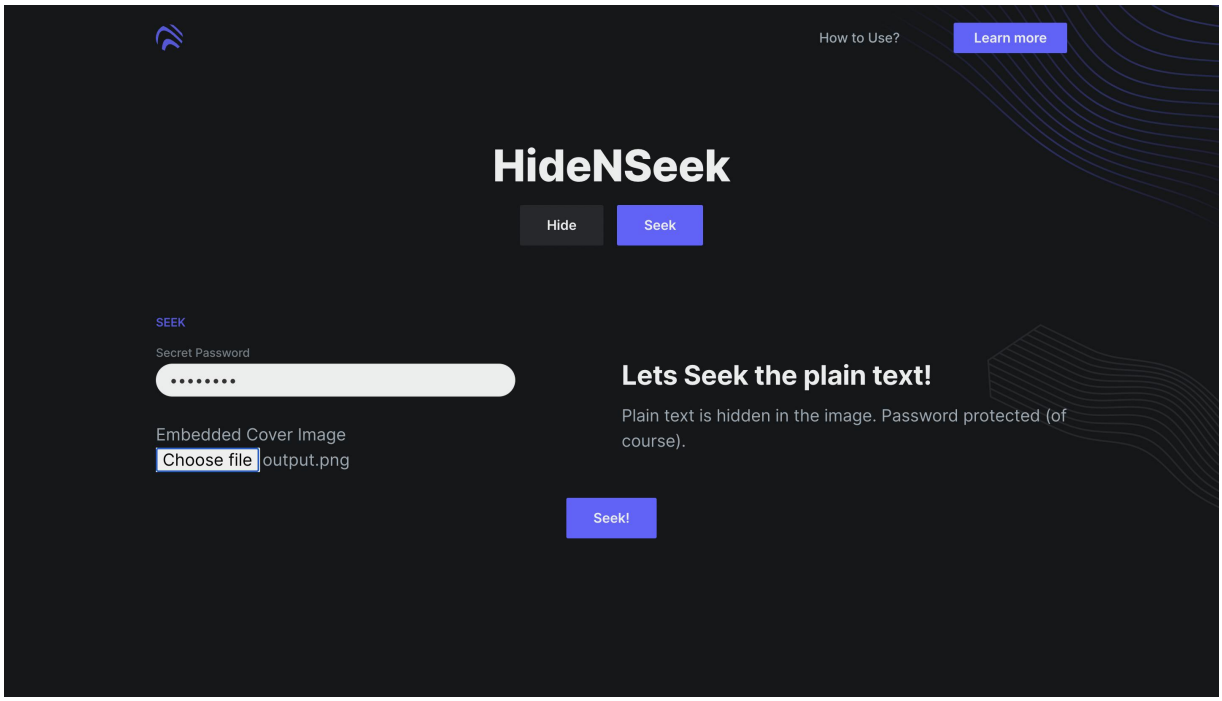


Figure 4.7: Form filled for “Seek”

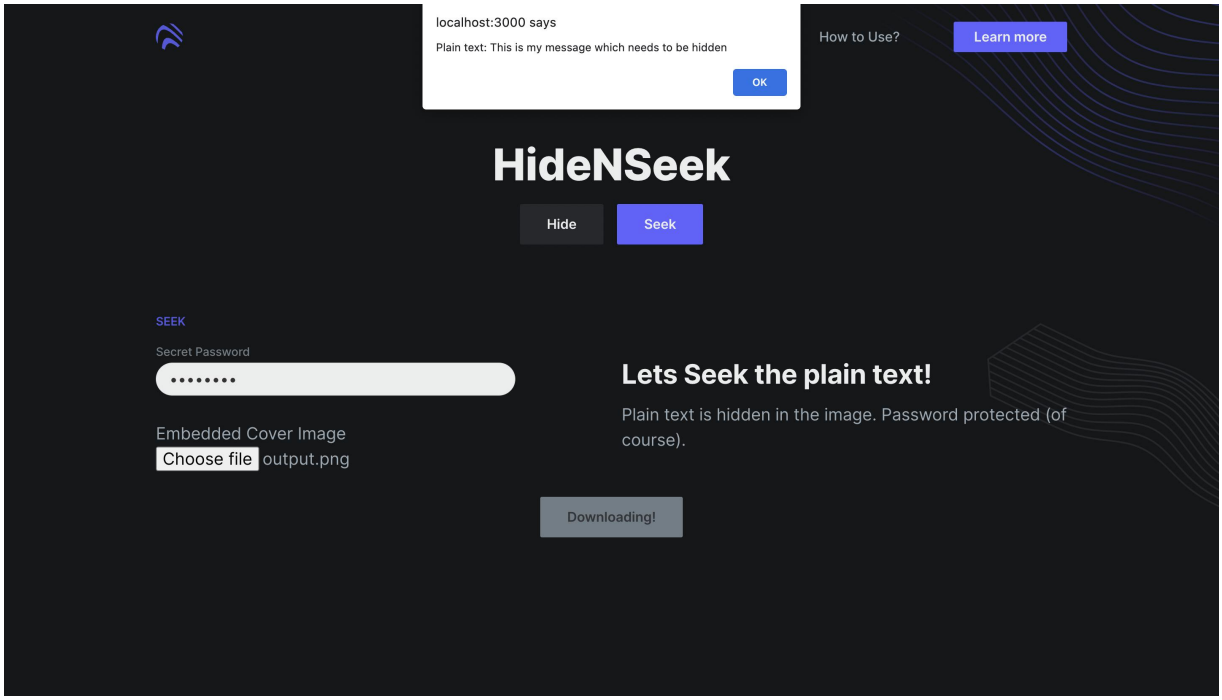


Figure 4.8: Extracted plain text being displayed in the alert

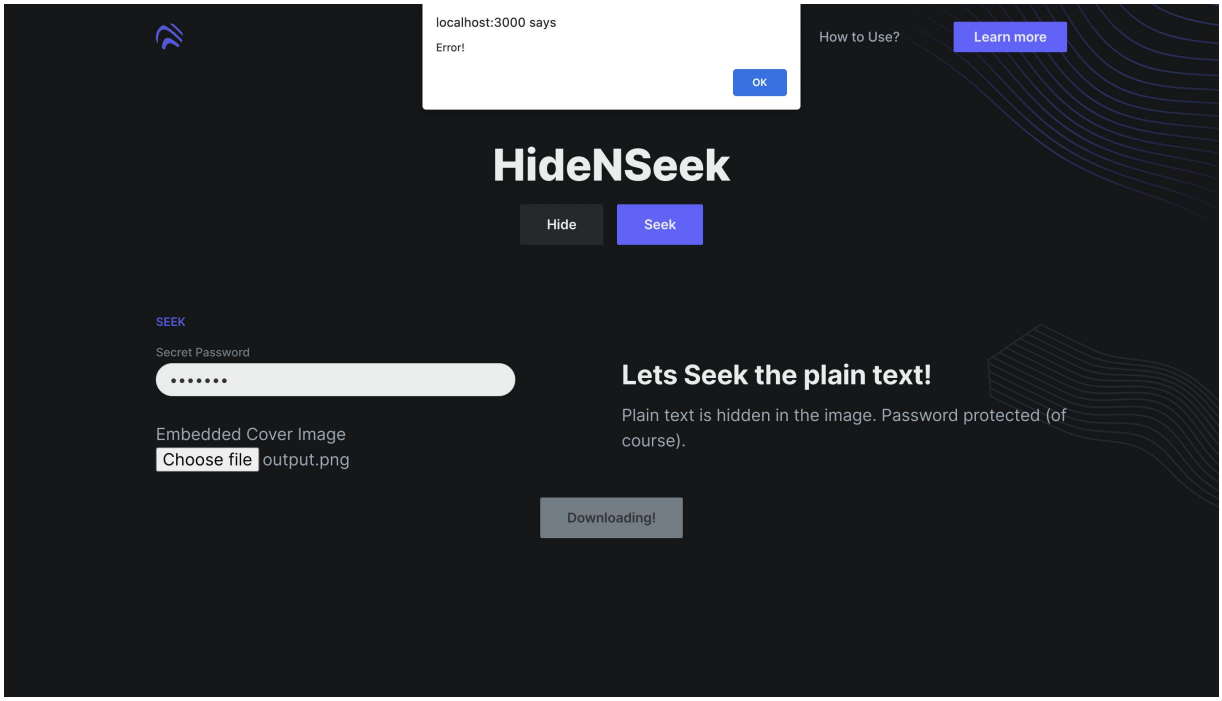


Figure 4.9: Error screen

# Chapter 5

# Conclusion and Future Scope

## 5.1 Conclusion

HideNSeek implements Least Significant Based Image Steganography by hiding the message in the least significant bits of the pixels of the cover image, in the similar manner, the embedded text is extracted from the stego image. As an added security feature, the message is first encrypted using AES-256 symmetric encryption algorithm. The project consists of an interactive website built using react.js with different sections for embedding the message, extracting the message and also a section showing how the application is to be used. The project makes use of Flask to create HTTP application programming interface endpoints for embedding, extracting the message to/from the cover image.

## 5.2 Applications

The particular implementation of Image Steganography provided by this project can be used for confidential information transmission, covert communication and also for confidential data storage

## 5.3 Future Scope

The project can be further extended by adding support to embed image messages onto the cover image, similarly, the message can be further made to undergo different transformations to make extraction of original message more complex than vanilla steganography.

## *References*

[1 ]Arun Kumar Singh, Juhi Singh, Dr. Harsh Vikram Singh, Steganography in Images Using LSB Technique International Journal of Latest Trends in Engineering and Technology (IJLTET) Vol. 5 Issue 1 January 2015 430 ISSN: 2278-621X.

[2] Stuti Goel, Arun Rana, Manpreet Kaur,"A Review of Comparison Techniques of Image Steganography", IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE) e-ISSN: 2278-1676,p-ISSN: 2320-3331, Volume 6, Issue 1 (May. - Jun. 2013),

[3] M. Pavani1, S. Naganjaneyulu, C. Nagaraju, "A Survey on LSB Based Steganography Methods" International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume 2 Issue 8 August, 2013 Page No. 2464-2467.

[4] https://github.com/cruip/open-react-template

[5] https://flask.palletsprojects.com/en/2.1.x/

[6] https://www.python.org/

[7] https://reactjs.org/

[8] http://www.viprefect.com/application-areas.

[9] http://www.asciitable.com/.

[10] The IEEE website. [Online]. Available: http://www.ieee.org/