Music Based Cryptography

A major project report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Chaitanya Dua (201445)

Saurabh Singh (201131)

Under the guidance & supervision of

Dr. Amit Kumar

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology,

Waknaghat, Solan - 173234 (India)

CERTIFICATE

This is to certify that the work which is being presented in the project report titled “Music Based Cryptography” in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by “Chaitanya Dua, 201445” and “Saurabh Singh, 201131” during the period from August 2023 to May 2024 under the supervision of Dr. Amit Kumar, Assistant Professor (SG) Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Chaitanya Dua
(201445)
Saurabh Singh
(201131)

The above statement made is correct to the best of my knowledge.

Dr. Amit Kumar
Assistant Professor (SG)
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat

CANDIDATE'S DECLARATION

We hereby declare that the work presented in this report entitled **'Music Based Cryptography'** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of our own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Amit Kumar** (Assistant Professor SG, Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)	(Student Signature with Date)
Chaitanya Dua	Saurabh Singh
201445	201131

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)

Dr. Amit Kumar

Assistant Professor (SG)

Computer Science & Engineering and Information Technology

Dated:

ACKNOWLEDGEMENT

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully.

We are really grateful and wish our profound indebtedness to Supervisor **Dr. Amit Kumar, Assistant Professor (SG)**, Department of CSE Jaypee University of Information Technology, Waknaghat. Deep Knowledge & keen interest of our supervisor in the field of "**Information Security**" has helped us to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, and valuable advice along with reading many inferior drafts and correcting them at all stage have made it possible for us to complete this project.

We would like to express our heartiest gratitude to **Dr. Amit Kumar,** Department of CSE, for his kind help to finish our project.

We would also generously welcome each one of those individuals who have helped us straight forwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

Chaitanya Dua (201445)

Saurabh Singh (201131)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

MIDI	Music Instrument Digital Interface
LSB	Least Significant Bit
MSB	Most Significant Bit
MP3	MPEG audio layer 3
Music XML	Music Extensible Markup Language
LSTM	Long Short-Term Memory
SSD	Solid State Drive
AWS	Amazon Web Services
MEDS	Music Encryption and Decryption Software
MVT	Model View Template
MAE	Mean Absolute Error
PSNR	Peak Signal-to-Noise Ratio

ABSTRACT

In the realm of information security, music-based cryptography has emerged as a new approach for hiding sensitive, secretive information within the intricacies of music compositions. This technique utilizes the power and complexity of music to embed hidden messages seamlessly into music pieces, rendering them un-noticeable to the human ear. By using various parameters of the music files, such as pitch, velocity, time values of the individual note, it is possible to encode data.

This report dives into the captivating world of art and technology [24], exploring its principles, techniques, and applications. In this, we will examine various methods employed to embed data within musical compositions, analyzing their strengths and limitations. Furthermore, we will investigate the real-life applications of music-based cryptography in diverse domains, such as secure communication, copyright protection, and secret hiding.

Through this comprehensive report, we aim to provide an understanding of cryptography concepts, its potential in artistic fields, and its challenges. We also will try to address limitations that need to be addressed for its wider adoption. By harnessing the power of music to conceal messages, we can open new avenues for security and a unique dimension to the art of music.

As we conclude our exploration, we emphasize the transformative potential of this study, opening gates to a future where music not only serves as a source of artistic expression but also as a means of communication and hiding and encrypting data.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Cryptography is a term referred to a process of hiding or coding information in a way that the person intended to read the message can only read it. Whatever technique the coder is using or creating, it must ensure the following points, as they are the basis of any cryptography, confidentiality, integrity, availability, authenticity, non- repudiation. If all these are points are getting satisfied then your method can be a success.

In today's world, the most important element that exists is data. We started dealing with data when the era of computer and internet started. This data is so important that everything today depends on it. It is a revenue source for many companies. With the abundance of data comes a risk of compromising it. This is where information security field shows its importance. This data can be secured using the concepts and learnings of information security field. So, cryptographic techniques can be used to secure data, or a communication. There are various techniques that exists, as there is a lot of development in this field and will keep on developing in upcoming years. The techniques range from basic to very complex, in accordance with the data or secrecy it is dealing with. There is a lot of scope in exploring new ways to hide data, secure data, and make the internet a safe, fraud-free, and fun place.

There can be many ways to find new techniques for cryptography or steganography, but doing it with what is already present can be creative. There is concept called music-based cryptography and it can widely be explored. We all love and enjoy music and it is a part of our daily lives and it is fascinating that we can hide data using this music. This fascinating technique harness the power of music to conceal data within the tones or notes of music compositions.

Unlike traditional cryptography, which relies on mathematical algorithms and complex codes, music cryptography leverages the inherent structure and expressive qualities of music to embed messages in a way that is un-noticeable to human ear. This technique can be achieved using various music software supported files, like MP3, or MIDI (Music Instrument Digital Interface), etc., this midi file basically has set of instructions for playing music, making it one of the perfect places to hide or conceal data.

In this report, we will embark on a journey to explore the fascinating world of music-based data embedding. We will delve into the principles and techniques that underpin this unique cryptographic approach, analyzing its strengths and limitations.

1.2 PROBLEM STATEMENT

To find a new technique of steganography in music, we must understand how music works, how a tune works, how it is encoded for the software to read, and where can we hide data in it. Deciding on which type of software supported files to target is important. MIDI file can be a good fit for the purpose as the data can be easily encoded and decoded in the information of music stored in it.

The problem at hand pertains to developing a robust solution for encrypting textual messages within existing MIDI (Musical Instrument Digital Interface) files while preserving the original melody without distortion. MIDI files, being a prevalent format for encoding musical compositions, possess inherent structures that define various musical aspects such as pitch, duration, and timing.

The problem statements in developing this technique can be:

- **Encryption and Decryption algorithms**: - Developing an algorithm that successfully encrypts and decrypts the data in MIDI file is challenging. The algorithm should be designed in a way, that it does not alter the important music information that should be clear for the software to play the track. Encryption should be done in a way that its decryption is possible.

- **Music Quality**: - As we are dealing with the tune, and notes of music and making changes in it, there can be a distortion or loss of music quality. There is no use of hiding the data if one can tell by listening music that it is altered. So, the algorithm should be in a way to not distort any melody of music.

- **File Size**: - When we are hiding data in a file, we are increasing the size of it, this should not be the case and the file size should stay constant. This can be a tricky thing to deal with but there exist various compression techniques through which we can achieve this.

By addressing these challenges, the method presented in this report seeks to contribute valuable insights into the potential of music-based cryptography as an innovative and secure method for information protection in the digital age.

1.3 OBJECTIVES

The major objectives of this project are: -

- **To develop suitable algorithm**: - One of the major objectives of this project is to develop a suitable algorithm that encrypts the textual data into the velocity and pitch values of MIDI file. The decryption for the same should be possible and should preserve the original text from the MIDI-encoded data as there are many variables that can alter the workings of it. The algorithm should be useable for all types of MIDI files, allowing a wide range of choices for the user.

- **Preservation of music quality**: - Music is all about its melody, and if it is not there, then there is no point for the hiding to take place. Therefore, the algorithm should preserve the quality of music and reduce the distortion as much as possible. We are mixing art with technology in this project and maintaining the quality of both should be its highest priority.

- **Optimization of file size**: - Moreover, an essential aspect of the algorithms' development is the optimization of the resulting MIDI file size. The data should be hidden in such a way, using compatible compression techniques, that the size increase in the MIDI file remains within acceptable limits, promoting practicality and usability.

- **Building an online platform**: - Building an online interactive platform is the basis of any project. In addition to developing the required algorithms, the project entails creating a user-friendly online platform. This platform should be made offering a seamless integration of the algorithm created, providing a dedicated interface for the user to either encrypt or decrypt or plat the MIDI track. Building it is important to showcase the usage in real-time to users.

These identified objectives serve as the foundational pillars crucial for the success of the project, collectively contributing to achieving its goals. The development of a robust encryption stands necessary, laying the groundwork for secure and effective transformation of textual data into MIDI parameters. Along with this, following other objectives makes a defined structure of this cryptographic technique to deliver at its best.

1.4 SIGNIFICANCE AND MOTIVATION OF THE PROJECT REPORT

1.4.1 SIGNIFICANCE

The exploration of music-based cryptography holds immense significance which are:

- **Innovation: -** Innovation is a process by which a domain, a product, or a service is renewed and brought up to date by applying new processes, introducing new techniques, or establishing successful ideas to create new value. Standing on this definition, this project tries to innovate using the art of music and the advancement of technology to achieve a new unexplored filed.

- **Enhanced Security: -** Music based cryptography offers a unique approach to data security. As we know that any cryptographic technique should follow the following points that is confidentiality, integrity, availability, authenticity, and non-repudiation. The significance of this report stands on this exact same thing that forms its inherent structure. By developing a correct encryption and decryption algorithm this project aims to enhance the security of the confidentiality of the message conveyed by the user unnoticeable.

  With the advancement of technology and its security, there is also a rise in the cyber-attacks which can lead to a lot of financial damage. Thus, enhancing the security and offering a new field to explore, this project thrives towards reducing all these negatives in the society.

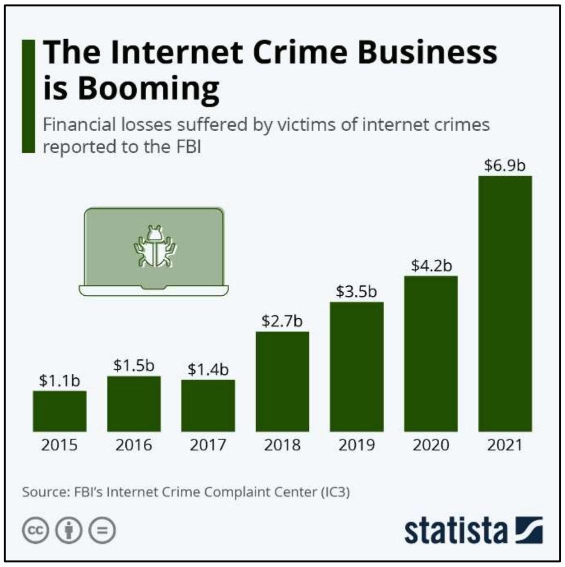


Figure 1.1: Statistics for fraud

- **Copyright Protection: -** This music-based cryptography can serve as a novel tool for copyright protection as well. The owner of the dedicated MIDI file can embed it with the information of its ownership. If this embedded MIDI file is used by any other organization or platform in future then it can be easily protected. This copyright protection works more as a watermark system just like any other application.

  For example, if a creator of an image wants to protect the authenticity of that image, watermark is added. Although the whole purpose of the Watermark is not only for copyright protection but to convey other users that who is the original creator of that image or application. In the same way the, MIDI file created can have its own watermark either for copyright protection or declaring the original ownership of its creator. The advantage of adding this is that it is present in a hidden form and not has a dedicated entry.

- **Enhancing communication: -** This project expands the possibility of general communication. As it uses musical compositions as a vessel for data transmission. This type of creative communication can be particularly useful in scenarios where traditional communication channels are restricted or being monitored for example it can be used by defense forces. This can have many meaningful purposes along with a usage for just amusement.

1.4.2 MOTIVATION

- **Exploring unconventional cryptography: -**The conventional cryptographic methods use complex mathematical expressions and algorithms. This study of musical based steganography or cryptography allows us to explore the unconventional cryptography way to encoding and decoding data. This thus push the boundaries of our imagination and contributes to the development of the advancement of the information security field.

  This creative approach opens the doors of imagination beyond which we can explore and develop new and improve the existing techniques to contribute in security. This motivation of creating something new and unconventional forms a basis of this project.

- **Addressing real-world challenges**: - As discussed earlier, with the advancement of technology, we constantly need to thrive and innovate our security related techniques. This study takes motivation from the existing problems in the real-world scenario, the loopholes, the shortcomings of existing related techniques and thrive to create new and working solution to overcome such issues. This study has potential to address real-world challenges in secure communication, copyright protection, and secure data transmission.

- **Creativity: -** A creative mind can lead to many ideas. Creativity not necessarily lies in creating something new, or inventing something. This can be done doing the existing things in new and in a creative way. As music-based data hiding is dependent on the art and craft of music, therefore in quest of taking a creative approach, this thus forms the motivation of this study.

- **Passion for Art and Technology: -** The combined passion for both art and technology form the most important basis of the motivation behind this study. These two fields have the potential to amplify each other in remarkable ways. This study uses the power of the universal language, that is music, to connect and inspire. This passion turned into practicality can help in developing new ways to address issues in security and solve them. The following image show an old musical cryptogram [21] used for the same in old age.

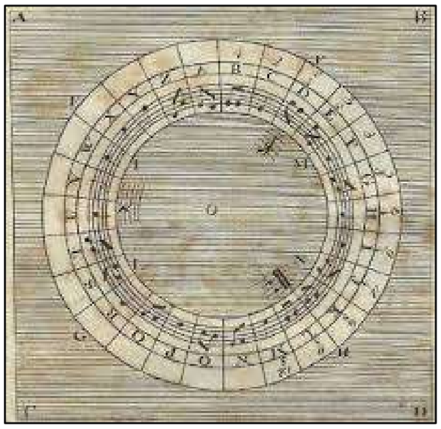


Figure 1.2: Music Cryptogram

It is relatively a new field, but it has already attracted the attention of researchers and security professionals as it promises to offer secure secret transmission of messages through a channel.

1.5 ORGANIZATION OF PROJECT REPORT

CHAPTER 1: INTRODUCTION

This chapter aims to present an overview of the project developed an "Music Based Cryptography." In this project we propose a new method of hiding a textual data, using Least Significant Bit (LSB) method. A music file format, MIDI (Musical Instrument Digital Interface) is being used for its metadata quality. The textual data gets converted to the binary string and gets swapped with the velocity values of the music notes. This chapter describes the problem statement, objectives, significance, and motivation of our project.

CHAPTER 2: LITERATURE REVIEW

This chapter aims to present various literature review of existing research on music-based cryptography. It discusses the various studies conducted on the same topic and if they had any shortcomings in them. It provides a base of further study and helped in finding a research gap. There are six previous year research papers discussed in this section along with some gaps and shortcomings they had in their study.

CHAPTER 4: TESTING

This chapter provides variety of validation points to measure upon. The working is compared to various previous algorithms, showing the efficiency of the study conducted.

CHAPTER 5: RESULTS AND EVALUATION

This chapter provides a detailed evaluation of the system created. The working of the encryption and decryption algorithms are discussed by taking an example and screenshots of the same are added. It was noted that the web application is responsive and providing the desired results.

CHAPTER 6: CONCLUSION

The final chapter summarizes the project's outcomes, suggesting potential improvements and future work. It emphasizes the system's potential to improve some areas of encryption process making even more reliable.

CHAPTER 2

LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

2.1.1 INTRODUCTION

Music and cryptography have been linked to each other since ancient times. The idea behind this study and project is not new. Hidden messages in musical notes were used in various places in ancient times, be it in defense or for personal use. The major reason to attempt the study in this field to find the shortcomings it had, and how it can be tackled.

When a plain text is hidden in the file supporting music, it can distort the melody, or increase the file size. There is a lot of scope in its improvement in the same. The method proposed in the study or project tries to solve the exact same issues. The idea of replacing plaintext characters with musical notes may involve musical notes in the form of visual sequence, vocal or instrumental music. In fact, the fields of music and cryptography have positive correlation. History says that many of the cryptographers have been good musicians. Earlier forms of music cryptography made use of the simple method of replacing characters of the plaintext with musical symbols. That type of encryption was mainly based on pen and paper. As the musicians played the music using the paper in front of it.

With the advancement of technology, there was an advancement in the field of music as well. These days we can make music using computer applications and even edit or change the existing one. This quality of the music makes it easy for hiding textual data. While the current studies in combining music and cryptography is somewhat limited, more than a few proposals in musical cryptography have involved symmetric key encryption, substitution ciphers, and stenography. There are many types of files such as Music XML, MIDI, MP3, etc., in this study we will focus mostly on MIDI as it has all the metadata of the music file, whether it is the name of the composer or the name of the song, or what type of instrument is used. These meta data can be extracted with the help of many tools and languages and can be edited using suitable algorithms.

The focus of this study is to reduce the amount of melody distortion and control the increase in file, making it suitable for real world usage, be it in data transmission or secrecy management.

2.1.2 SURVEY OF THE RELEVANT PAPERS

Deepthi Rao et al [1], proposed a novel method for music encryption. A predefined set of instructions based on Carnatic music is defined. Based on this, a mapping is done using musical notes, to encrypt and decrypt data. This method considers a plaintext alphabet that consists of the 26 letters of English along with space character resulting in a set of 27 characters. The proposed approach makes use of only janaka ragas. Depending on the selected raga, all its swara except for an amsa swara, are assigned with positive weights. Each of these six positions can take a weight of 0 or 1, resulting in 64 possible combinations.

Mattias Wecksten et al [2], suggested method velody 2 consists of encoding encrypted data at high capacity into the velocities of the note-on events, while mimicking humanization available in tools with MIDI support such as Ableton Live. To be able to extract just the embedded message and nothing more, the message length needs to be known. This can be done in many ways, but assuming that most messages will be less than 256 bytes of length one approach is to add an eight-bit message header to indicate the message length. This approach allows for longer messages if that would be required by stacking several blocks after one another. Assuming most messages are less than two blocks in length this approach will have the same or less overhead than an approach where 16 bits would be used to indicate the message length.

Curtis Helsel et al [3], The proposed paper used LSTM networks, cryptographic techniques, steganography, and music editing software. The file format that contains this information is used to create the sequential data that the network will train. To extract the information needed, the music21 library is used. The music21 project allows users to analyse, search, and transform music into a symbolic form. For the ease of use, the Keras library with the TensorFlow backend is used to create the predictive model. The determination for the architecture was done by modifying different components of the model to achieve a convergence of minimal loss within a reasonable number of epochs trained. The network is comprised of the following components: an LSTM layer with a 25 percent dropout rate on the output and a densely connected layer with the number of nodes equal to the number of different note values within the training data.

Reman Krishnan et al [4], The proposed system uses musical passwords to access the files and musical notes to encrypt and decrypt the text file data. When a user accesses a file, a musical keyboard appears and the user is prompted to play the musical password. Users define their own password as a sequence of musical notes of varying length. The musical password further serves as the key during encryption. The output would be a set of musical notes which represents the corresponding text in the file. The encrypted text file is then played as music. The proposed model is a symmetric model, where the same musical password used to encrypt should be used to decrypt data. The system ensures that a given plaintext with a given key generates different encrypted output each time due to usage of random mapping tables.

Da-Chun Wu et al [5], The proposed method proposes Data hiding techniques for steganography, which embed secret data in multimedia imperceptibly, are useful for protecting information security. By taking advantage of the popularity of MIDI files on the Internet, a new data hiding method via MIDI files is proposed, which modifies the velocities of musical note sequences to embed secret data. Initially, musical note sequences with monotonic pitches, each consisting of at least three consecutive notes with pitches either entirely non-decreasing or entirely nonincreasing, are found from an input MIDI file. Next, for each of such musical note sequences, a reference velocity is generated for each non-end note in the sequence by a linear interpolation scheme. The original MIDI file size is also kept unchanged after data embedding, avoiding attracting attentions from hackers. Experimental results show the feasibility of the proposed method.

Yi-Hsin Liu et al [6], In this paper, a new data hiding method is proposed for embedding message data in MIDI files using the values of delta times. The proposed method is performance-preserving because it does not alter the resulting musical expression; it is blind since the embedded message can be extracted without referencing any data of the original MIDI file; and it is reversible as well since the cover MIDI file can be restored completely from the stego-MIDI file during the message extraction process. The method yields reasonable data embedding rates with limited file expansions, and embeds message data with security better than the surveyed performance-preserving blind methods.

2.2 KEY GAPS IN THE LITERATURE

The reviewed research papers collectively contribute to the emerging field of music encryption and data hiding within MIDI files, each presenting unique approaches and methodologies. However, key gaps and areas for improvement can be identified across these works. Firstly, while Deepthi Rao et al [1] propose a method based on Carnatic music, the effectiveness and security of this approach could be further validated through rigorous testing and comparison with existing encryption methods. The utilization of janaka ragas raises questions about the system's adaptability to diverse musical genres.

Mattias Wecksten et al [2] introduce a high-capacity encoding method, but the potential impact on the musical quality and the robustness of the method under various musical contexts should be explored. Curtis Helsel et al [3] leverage LSTM networks, but the paper lacks a comprehensive evaluation of the model's performance under different datasets and musical styles. Reman Krishnan et al [4] propose a unique musical password system, yet the paper would benefit from an in-depth analysis of the system's vulnerability to attacks and the potential for secure key management.

Da-Chun Wu et al [5] focus on data hiding via velocity modification, but the robustness of their approach against different attacks and its scalability to larger MIDI files need further investigation. Finally, Yi-Hsin Liu et al [6] propose a method using delta times, and while the reversibility and performance preservation are highlighted, the security aspects and potential vulnerabilities in real-world scenarios require more exploration. To advance the field, future research could focus on standardized evaluation metrics, comprehensive security analyses, and the development of hybrid approaches that combine the strengths of multiple methods while addressing their respective limitations.

In summary, while the existing research papers contribute valuable insights into the intersection of music and encryption, future work could benefit from standardized evaluation frameworks, enhanced scalability and adaptability, a deeper understanding of user experience considerations, and a more thorough exploration of security aspects to ensure the reliability and effectiveness of these innovative approaches.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

3.1.1 FUNCTIONAL REQUIREMENTS

The following are the functional requirements: -

- **Encryption Functionality: -** The encryption algorithm should be constructed in a way that it correctly takes input text from user. Users should be able to upload the MIDI files. The algorithm should convert the input text into binary data and thus embedded inside the velocity values of the MIDI notes. After all this, encrypted file should be made available to the user without any hassle.
- **Decryption Functionality: -** The decryption algorithm should be done in a way that user should be able to upload the MIDI encrypted file. The system should extract the binary data from the velocity values of the MIDI notes. After a successful extraction, this data should be converted back to the original text and presented in front of the user on the screen.
- **User Interface: -** The web application designed should have a user-friendly interface. The user should not find it difficult to navigate through the website or face any issues. The buttons for encryption and decryption, and forms should be clear and intuitive.
- **Security: -** There should be an input validation on the encryption page, to prevent malicious input. The files that the user is uploading should be securely handled to prevent any unauthorized access. The encryption and decryption processes should be secure and resistant to attacks.
- **Error Handling: -** The designed system should provide meaningful error messages for incorrect inputs. Both the algorithms should be designed in a way to handle errors and avoid any un-necessary system crashes.
- **Metadata Handling: -** As we are dealing with the metadata of the MIDI file format or we can say metadata of the music file, it is important to preserve and handle it well without getting it distorted, as it can affect the important abilities of the file, such as pitch or melody or even size.

3.1.2 NON-FUNCTIONAL REQUIREMENTS

Some of the non-functional requirements of this project are: -

- **Performance: -** The working of the encryption and decryption algorithms should be efficient and not cause any delays. The web application should be responsive, providing quick feedback to user actions.
- **Usability: -** The user interface should be intuitive, requiring minimal training for users. The navigation throughout the website should be easy and direct and should not cause any confusion.
- **Security: -** Whole system should follow best practices for web application security. The encrypted files and the mapping schema should be kept hidden and access to it should be restricted.
- **Reliability: -** The system prepared should be reliable and trustworthy. The algorithms should consistently provide accurate results. It should handle the errors during the process, and provide meaningful error messages to users.
- **Scalability: -** The architecture should allow for easy scalability to accommodate future enhancements for increased user loads. The web system should use tools and software that best in providing the band width required to host a large number of users.

3.1.3 SOFTWARE REQUIREMENTS

3.1.3.1 LANGUAGES USED

- **Python: -** Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented, and functional programming. Python is the optimal choice for developing a music encryption and decryption web application due to its versatility, readability, and a rich ecosystem of libraries and frameworks. It can handle complex tasks while maintain code clarity. Rather than building all its functionality into its core, Python was designed to be highly extensible via modules.
- **HTML: -** HTML, or Hypertext Markup Language, serves as the backbone of the World Wide Web, providing the fundamental structure for creating and presenting

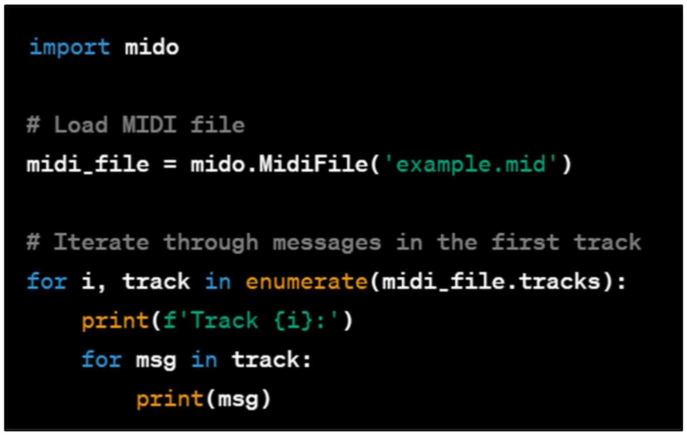
content on web pages. It is a markup language that utilizes a tag-based system to define the elements within a document, such as headings, paragraphs, images, links, and more. HTML is integral to web development, as it forms the core structure that web browsers interpret and render visually.

- **CSS**: - CSS, or Cascading Style Sheets, is a powerful styling language used in web development to control the presentation and layout of HTML documents. Working alongside HTML and JavaScript, CSS plays a pivotal role in enhancing the visual appeal and user experience of web pages. It provides a means to define the look and feel of content, ensuring consistency and enabling responsive design across various devices.

3.1.3.2 LIBRARIES USED

The following libraries will get used in the project: -

- **Mido: -** The mido library in Python is a powerful tool for working with MIDI (Musical Instrument Digital Interface) files. MIDI is a protocol that allows electronic musical instruments, computers, and other devices to communicate and synchronize with each other. The mido library simplifies MIDI file parsing, creation, and manipulation, making it a valuable resource for developers working on music-related projects. MIDI messages are fundamental to MIDI communication, conveying information about musical notes, control changes, and other events. mido facilitates the creation and manipulation of MIDI messages, allowing developers to craft intricate musical compositions.

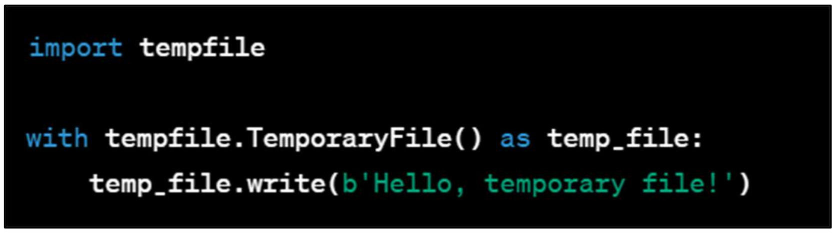


```python
import mido

# Load MIDI file
midi_file = mido.MidiFile('example.mid')

# Iterate through messages in the first track
for i, track in enumerate(midi_file.tracks):
    print(f'Track {i}:')
    for msg in track:
        print(msg)
```

Figure 3.1: Demo code snippet of the mido library module

Mido benefits from an active community of developers and thorough documentation. This makes it easier for newcomers to get started and for experienced developers to troubleshoot issues or implement advanced features. In this mido library, we will use MidiFile, MidiTrack, Message modules particularly. These modules will help us extracting the necessary information of the MIDI file, required for executing of the further process.

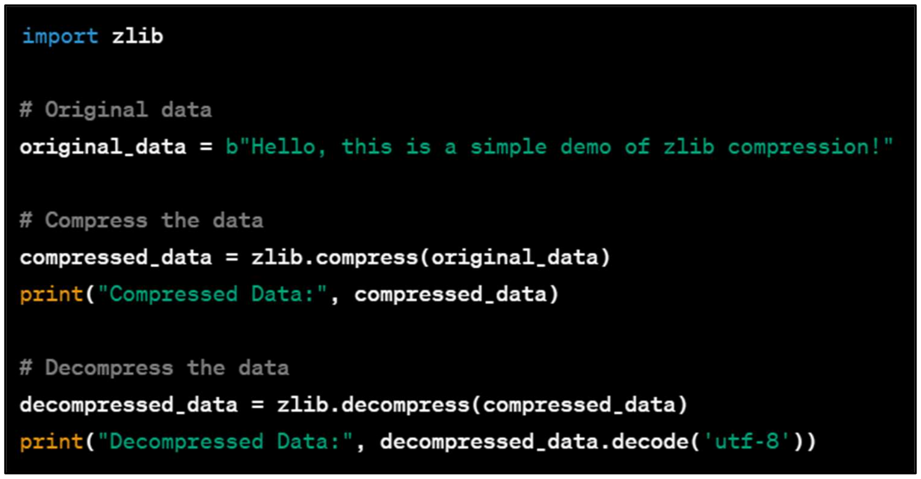
- **Tempfile**: - The tempfile module in Python provides functions and classes for working with temporary files and directories. It is particularly useful when you need to create temporary files or directories during the execution of a program, and you want these files to be automatically cleaned up when they are no longer needed.



```
import tempfile

with tempfile.TemporaryFile() as temp_file:
    temp_file.write(b'Hello, temporary file!')
```

Figure 3.2: Demo code snippet of tempfile library module

- **Zlib: -** Zlib is a software library for data compression that provides high-performance, in-memory compression, and decompression functions. It is a widely used and versatile compression library in the Python programming language. Developed by Jean-loup Gailly and Mark Adler, zlib is designed to be fast, efficient, and portable across various platforms. Zlib is primarily known for its data compression capabilities. It uses the DEFLATE compression algorithm, which combines the LZ77 algorithm and Huffman coding. This results in excellent compression ratios for a wide range of data types.

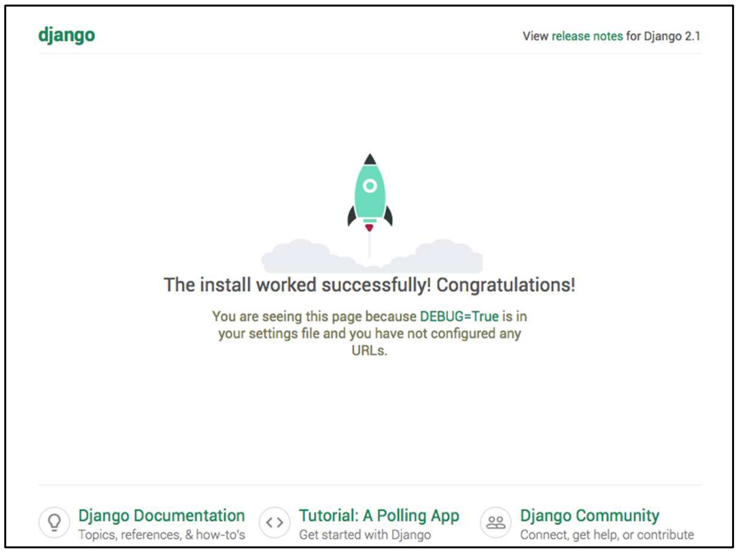


```
import zlib

# Original data
original_data = b"Hello, this is a simple demo of zlib compression!"

# Compress the data
compressed_data = zlib.compress(original_data)
print("Compressed Data:", compressed_data)

# Decompress the data
decompressed_data = zlib.decompress(compressed_data)
print("Decompressed Data:", decompressed_data.decode('utf-8'))
```

Figure 3.3: Demo code snippet of zlib library

3.1.3.3 WEB FRAMEWORK

Django [13]: - Django is a high-level, open-source web framework for building web applications using the Python programming language. It follows the Model-View-Controller (MVC) architectural pattern but refers to it as Model-View-Template (MVT), where the template handles the presentation layer. Originally developed to ease the creation of news websites, Django has evolved into a versatile framework that empowers developers to build robust and scalable web applications. For this project we will use this framework to handle both frontend and backend.


django
View release notes for Django 2.1
Django Documentation
Topics, references, & how-to's
Tutorial: A Polling App
Get started with Django
Django Community
Connect, get help, or contribute

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Figure 3.4: Installation page of django web framework

3.1.4 HARDWARE REQUIREMENTS

- **Network:** A reliable network connection is required for this project, to upload, download or even to access the website. An ethernet or wifi adapter is recommended for the easy use of the system.
- **Storage:** Adequate storage for the operation system, development tools, and project files is important. An SSD is beneficial for the faster code compilation and application testing.
- **Cloud Services:** If this project is to be uploaded, the django application to a production environment, cloud hosting services such as AWS, Google Cloud, or Heroku can provide scalable infrastructure. The hardware requirements will depend on the chosen cloud service provider and the expected traffic.

Its important to note that these hardware recommendations are general guidelines. Along with these, other hardware requirements can arise with some improvisation It is based on the complexity of the project, the number of concurrent users, and specific features that may demand additional resources.

3.2 PROJECT DESIGN AND ARCHITECTURE

3.2.1 METHODOLOGY

In order to use the features of the web application, the users can get themselves enrolled on the website. After a successful registration, the users can login to the software. The website will have the following features:

- **Encryption:** The introductory page will have the encryption button through which the user can reach out to the encryption page. After proceeding successfully, the user can take benefit of the encryption algorithm, by just typing the text and uploading the file they want. On pressing the “Encrypt My File” option the encrypted MIDI file will get directly downloaded on the users’ system.
- **Decryption:** The introductory page also led to the decryption page through the decryption button. After loading the page, the user can decrypt the encrypted file by just uploading the encrypted file. On pressing the “Decrypt My File” button, the decrypted plain text will get displayed on the users’ screen.

- **MIDI player [11]:** On pressing the MIDI player button, the user can take benefit of the MIDI player option. By just uploading the file, the user can play the track and enjoy on the melodies.

This is the basic workflow of the web application. All the features are made in a way, to provide utmost ease to user to navigate through.

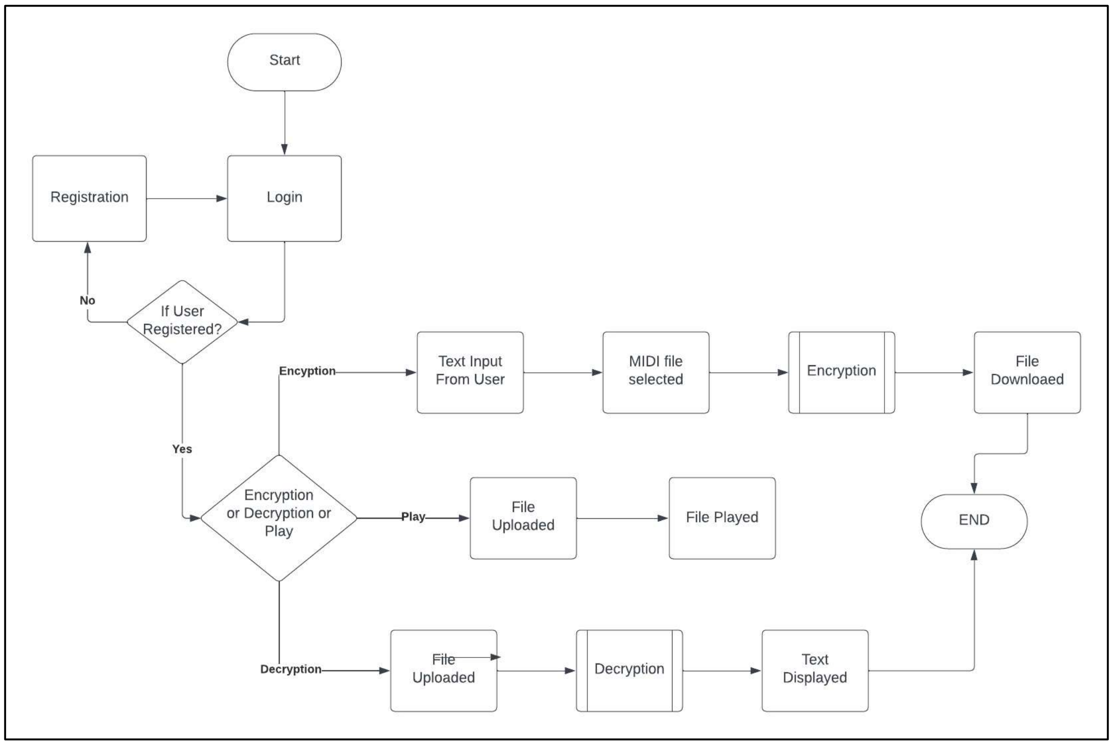


Figure 3.5: Flow Graph

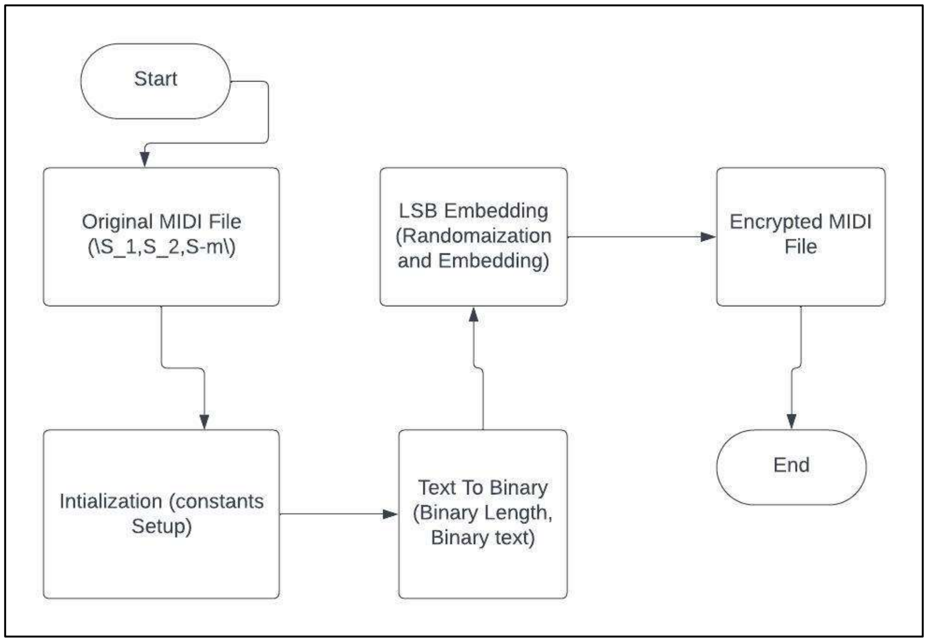

Start
Original MIDI File (\S_1,S_2,S-m\)
Intialization (constants Setup)
Text To Binary (Binary Length, Binary text)
LSB Embedding (Randomaization and Embedding)
Encrypted MIDI File
End

Figure 3.6: Encryption Flow Graph

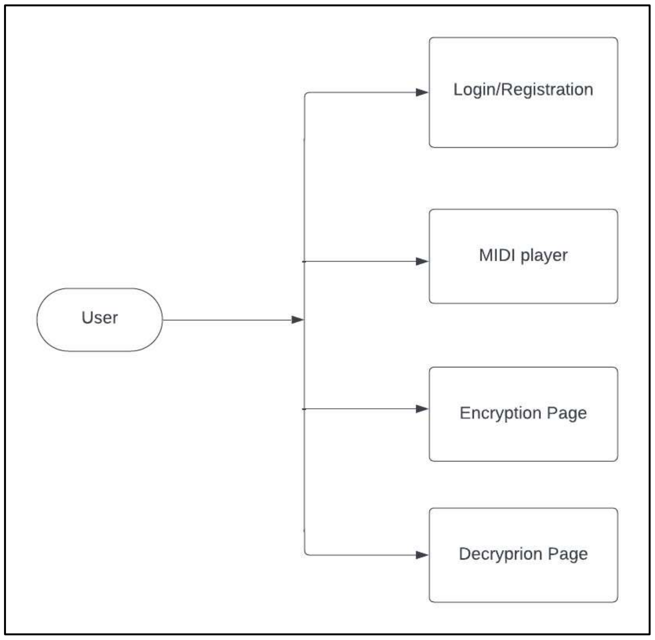


Figure 3.7: Data flow diagram

3.3 IMPLEMENTATION

3.3.1 PREQUISITES

To understand how the algorithms are working, we first need to understand some basic things. The working and composition of the software using is important. Here the music software readable file, the MIDI file is being used. This MIDI stands for Music Instrumental Digital Interface. This type of file is readable for computer and thus a good fit for us to make changes. This MIDI file contains a lot of information of the music stored in it, for example, the name of the song, the name of composer, the instrument used to produce that music, the lyrics, and most importantly, the notes and tunes, etc., This all data stored inside of it is known as the metadata of that music file.

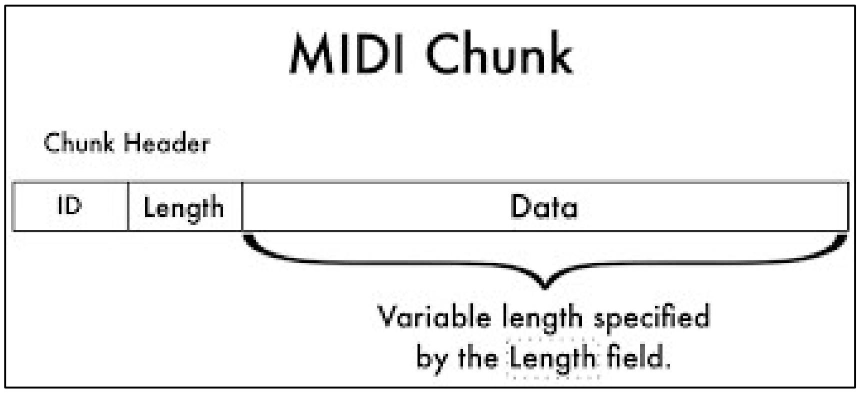


Figure 3.8: The format of MIDI file

There are other files like MIDI, like Music XML, MP3, but none of them can be used for implementing the algorithm because of their shortcomings. **MIDI** (**Musical Instrument Digital Interface**) is a technical standard that describes a communication protocol, digital interface, and electrical connectors that connect a wide variety of electronic musical instruments, computers, and related audio devices for playing, editing, and recording music. A file format that stores and exchanges the data is also defined. Advantages of MIDI include small file size, ease of modification and manipulation and a wide choice of electronic instruments and synthesizer or digitally sampled sounds. A MIDI file is not an audio recording. Rather, it is a set of instructions – for example, for pitch or tempo – and can use a thousand times less disk space than the equivalent recorded audio.

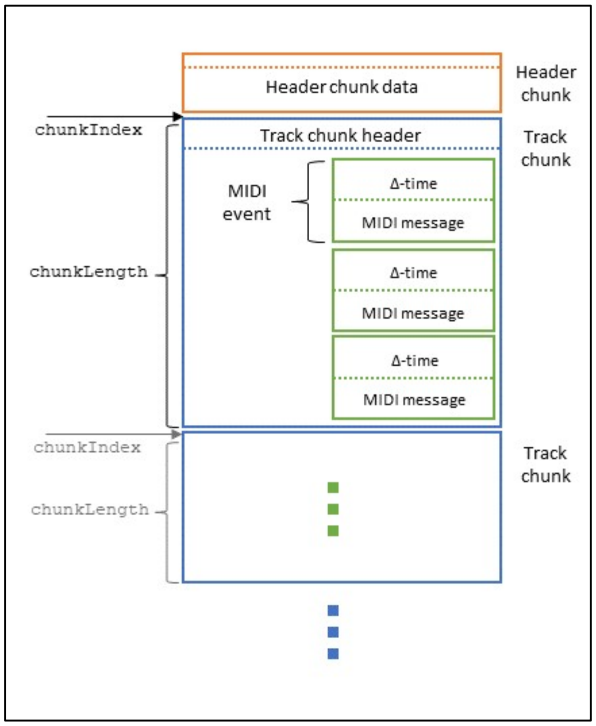


Figure 3.9: MIDI file structure

This designed system will have an encryption and decryption algorithm, integrate using the django python web framework.

3.3.2 ENCRYPTION

1. For implementing the text encryption, we must follow the following steps: -

- We must have all the software needed to execute the process. We need python installed, and visual studio for better arrangement and execution of the codes.
- After installing and getting our systems ready, we need to extract and access all the meta data of the MIDI file. For this, we need to use python libraries like "mido," from this mido, we may need the "midifile," "miditrack," "message" to execute our code.
- We will extract especially the pitch and velocity values of the music track. After extracting, we will convert the values into 8-bit binary values.
- After that, we will take text input from the user and convert each letter into binary values according to a predefined mapping.

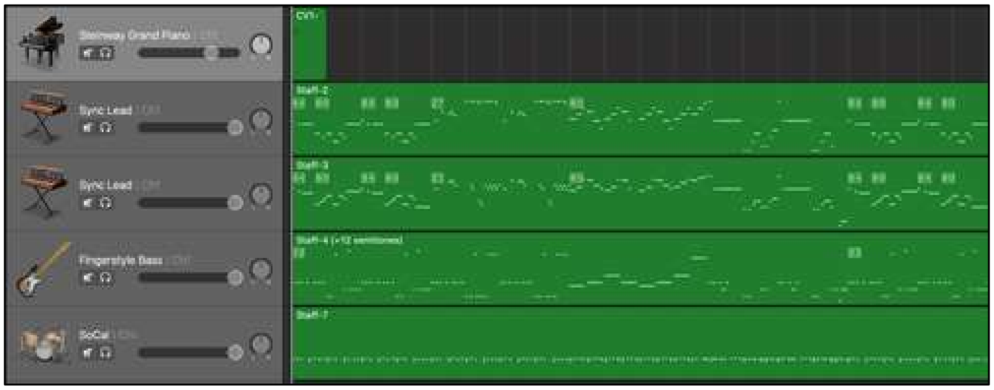


Figure 3.10: Working with MIDI data using mido library

- To implement the encryption, we will use the **Least Significant Bit (LSB)** method. This means that we will change and alter the last bit of every velocity binary value.
- The benefit of using least significant bit and modifying the velocity is that it will alter it will not change the melody of the music. Instead, it will just alter the volume to some extent, but that too will not be significantly increased. It is un-noticeable for human ear.

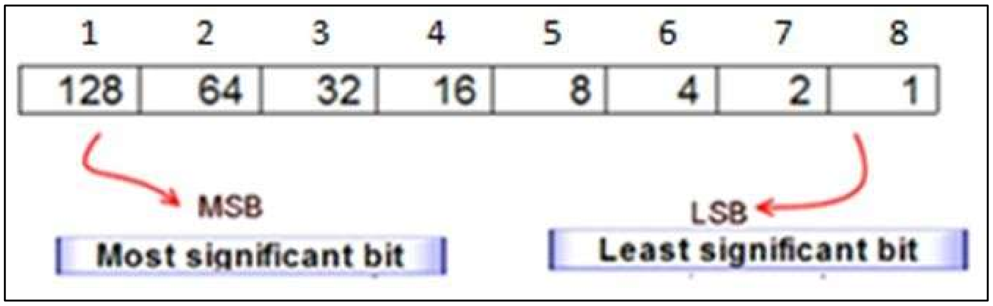


Figure 3.11: Representation of MSB and LSB

- We will use the Least Significant Bit and not the Most Significant Bit as changing the Most Significant Bit will change the volume significantly.
- Now, we will swap every least bit of consecutive velocity values with the text binary values.
- To achieve a complexity, we make changes to every 2nd velocity values or every 4rth or so on. This way we can achieve a strong encryption scheme which is difficult to crack.

- Finally, we can save the encrypted file to our system. This track can be played on MIDI player and has the same melody as that of the original.
- The file size is also not changed significantly in this encryption process keeping the text length to a certain limit.
- For encryption we need to create a separate URL for encryption result. This is important as the result will be downloaded on the user's system.

2. For implementing the image encryption, we must follow the following steps: -

- For image encryption, we need all the same software and hardware requirements as we used in text encryption.
- Additionally, we need to use some more libraries like PIL, shutil, etc.
- Initially, an image will be taken as an input via an upload option on the website.
- This image will be broken down into a binary array data which will be hidden in the track following the same encryption algorithm as text that is LSB.
- This encryption will result in a ".mid file," which will be downloaded on the user's system automatically.
- This encryption of image into a MIDI file significantly increases the size of file, which seems inefficient.
- It is noted and suggested that a very small image, just for a watermarking purpose is uploaded, to keep it as compact as possible.
- The decompression technique if applied will hamper the decryption method and the processing speed.
- This encryption is therefore, stands as a demo algorithm and has a future scope.

3.3.3 DECRYPTION

After successfully encrypting the MIDI file format, an encrypted file will get downloaded on the user's system. To decrypt, we must reverse the steps we had in the encryption process. The text will be in an integrated form in the velocity values of the tune of the music. To extract it, we must know the schema we used in the encryption. Without the mapping and encryption schema this decryption is not possible. Once we know the mapping key and other requirements, just by reversing the function, we can extract it. During decryption, each "note_on" event in the last track of the MIDI file is analyzed to retrieve the least significant bit (LSB) from the velocity value. These LSBs are then concatenated to reconstruct a binary string representing the hidden text.

This decryption should be done in a way that it takes a file as an input and gives only the decrypted text hidden inside of it. If this mapping schema is lost or any kind of deviation is there then it may result in distorted or incomplete extraction of the concealed information. This algorithm is done in a user-friendly way, in which we only take an encrypted file as an input, after this it will smoothy reveal the text.

In case of image decryption, the encrypted file will be taken as an input through an upload option on the website. The velocity values will thus be fetched and combined as an array, and then each pixel value is generated to form an image. This decryption is time taking and needs to be more efficient by integrating some compression and decompression techniques.

3.3.4 WEBSITE

In parallel with the intricate encryption and decryption processes, the development of a user-friendly web interface is also important. For this project, django web framework is being used. This web application serves as a bridge between users and the underlying encryption and decryption functionalities. Through this interface, users can seamlessly encrypt their text into MIDI files or decrypt concealed information from encrypted MIDI files with ease. Users, regardless of their technical expertise, can interact with features of the application. This way it is not limited to only the people who have knowledge of cryptography or music.

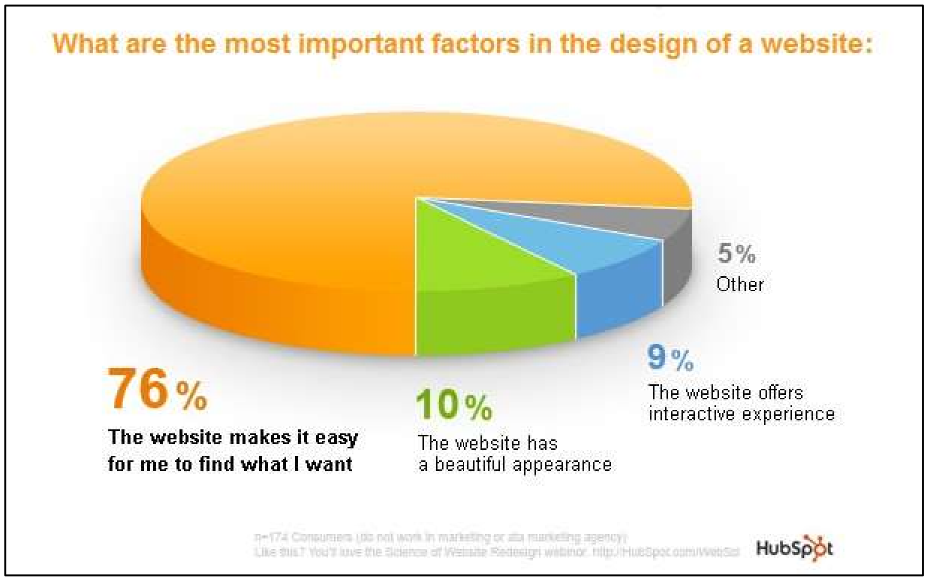
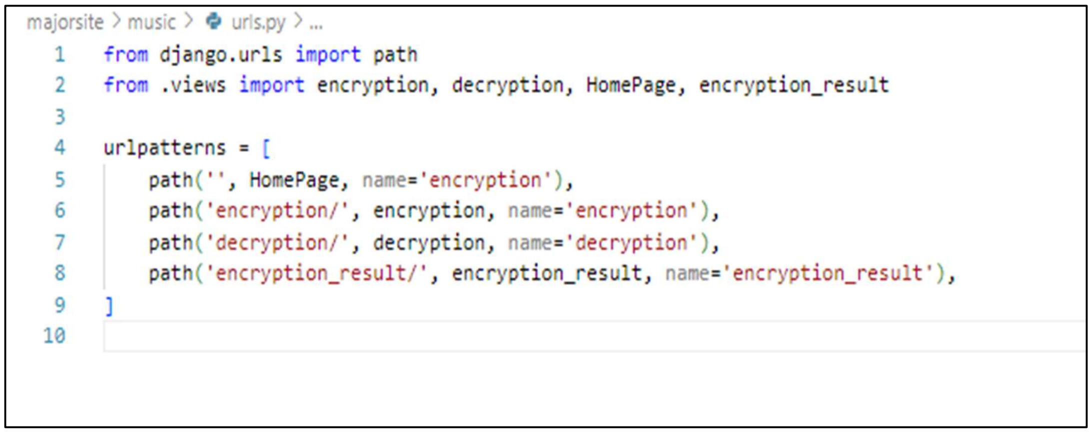


Figure 3.12: Statistics for user-friendly website

Django, a high-level python web framework, was chosen for its efficiency and versatility. It streamlines the development by providing built-in tools and conventions, enabling rapid creation of web applications. It is based on MVT (Model-View-Template) architecture, that means it can support rendering of various webpages and an easy interaction between all these pages. This framework was a good fit here has it has immense security features ensuring a good defense against common web vulnerabilities. Also, there is a lot of documentation available on this, making it easy for a creator to navigate through.

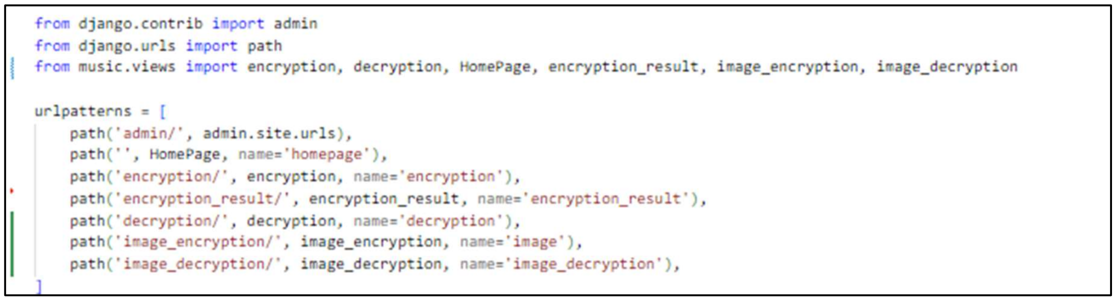
In this web application, there are four web pages, first page is an introductory page, where the user can choose what is to be done. Second page is of encryption, where an input, in the form of text, is taken from the user, and a MIDI file format is uploaded. After this encryption, an encrypted file is pushed on the user's system. Third web page is of the decryption algorithm, where an encrypted file is uploaded and the decrypted text is shown on the screen. The last page is a MIDI player, where the user can play the MIDI track. All these pages together become the Music Encryption and Decryption Software (MEDS).

The Music Encryption and Decryption Software (MEDS) developed using Django serves as an accessible and secure platform for users, transcending technical barriers and making the encryption and decryption processes user-friendly and efficient. The integration of Django not only expedites development but also contributes to the overall reliability and security of the web application. Various pages of the application are joint using the concepts of views and URLs.



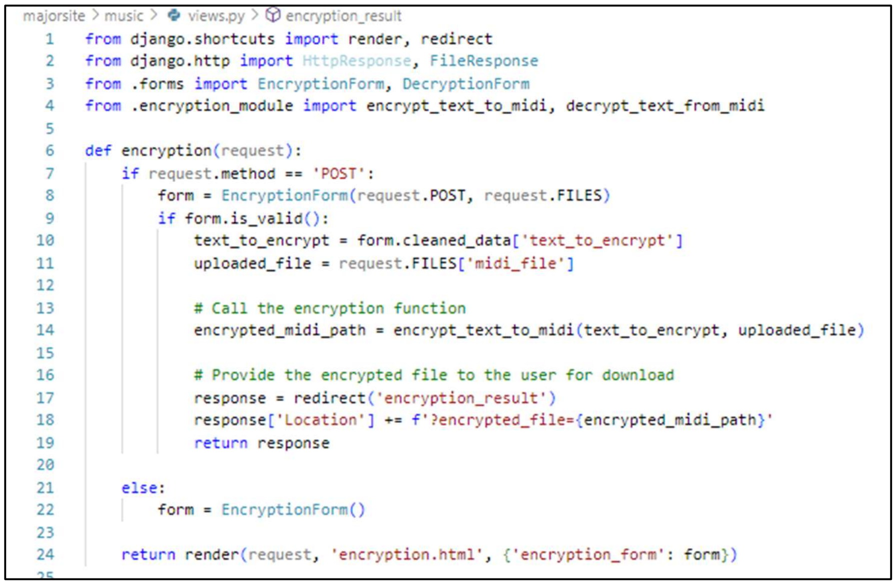
```python
from django.urls import path
from .views import encryption, decryption, HomePage, encryption_result

urlpatterns = [
    path('', HomePage, name='encryption'),
    path('encryption/', encryption, name='encryption'),
    path('decryption/', decryption, name='decryption'),
    path('encryption_result/', encryption_result, name='encryption_result'),
]

```

Figure 3.13: URL for Text



```
from django.contrib import admin
from django.urls import path
from music.views import encryption, decryption, HomePage, encryption_result, image_encryption, image_decryption

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', HomePage, name='homepage'),
    path('encryption/', encryption, name='encryption'),
    path('encryption_result/', encryption_result, name='encryption_result'),
    path('decryption/', decryption, name='decryption'),
    path('image_encryption/', image_encryption, name='image'),
    path('image_decryption/', image_decryption, name='image_decryption'),
]
```

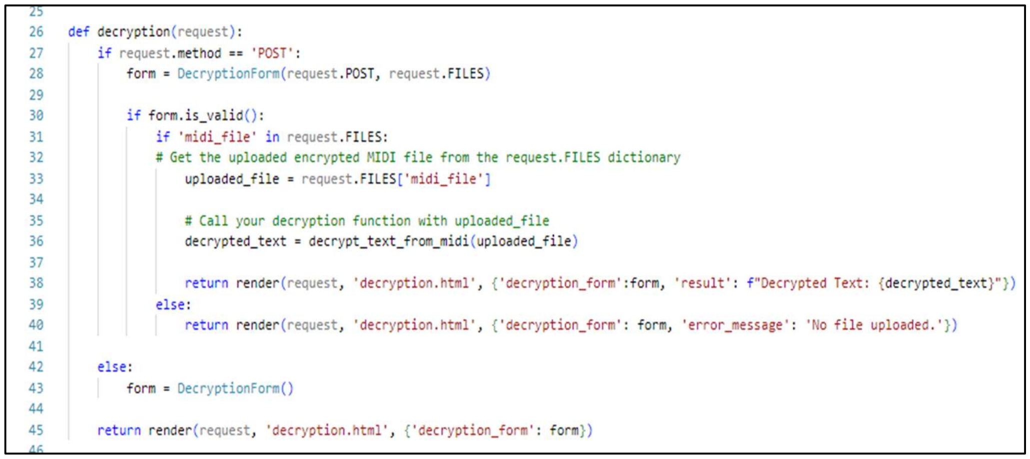
Figure 3.14: URL for Images



majorsite > music > views.py > encryption_result

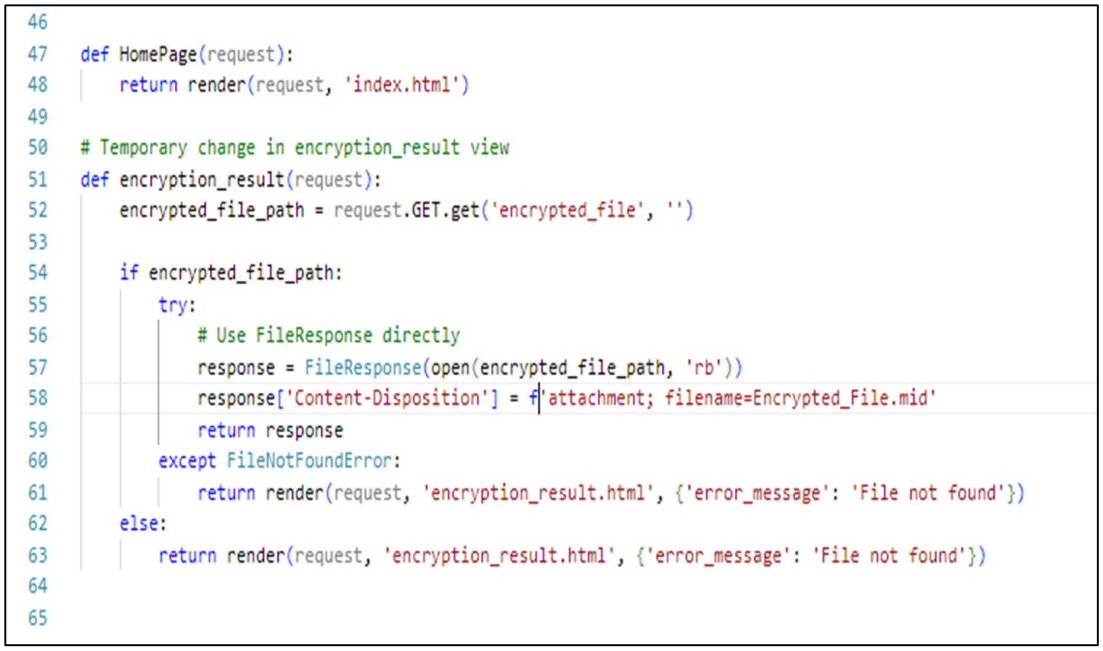
```
from django.shortcuts import render, redirect
from django.http import HttpResponse, FileResponse
from .forms import EncryptionForm, DecryptionForm
from .encryption_module import encrypt_text_to_midi, decrypt_text_from_midi

def encryption(request):
    if request.method == 'POST':
        form = EncryptionForm(request.POST, request.FILES)
        if form.is_valid():
            text_to_encrypt = form.cleaned_data['text_to_encrypt']
            uploaded_file = request.FILES['midi_file']

            # Call the encryption function
            encrypted_midi_path = encrypt_text_to_midi(text_to_encrypt, uploaded_file)

            # Provide the encrypted file to the user for download
            response = redirect('encryption_result')
            response['Location'] += f'?encrypted_file={encrypted_midi_path}'
            return response

    else:
        form = EncryptionForm()

    return render(request, 'encryption.html', {'encryption_form': form})
```

Figure 3.15: Views of text encryption



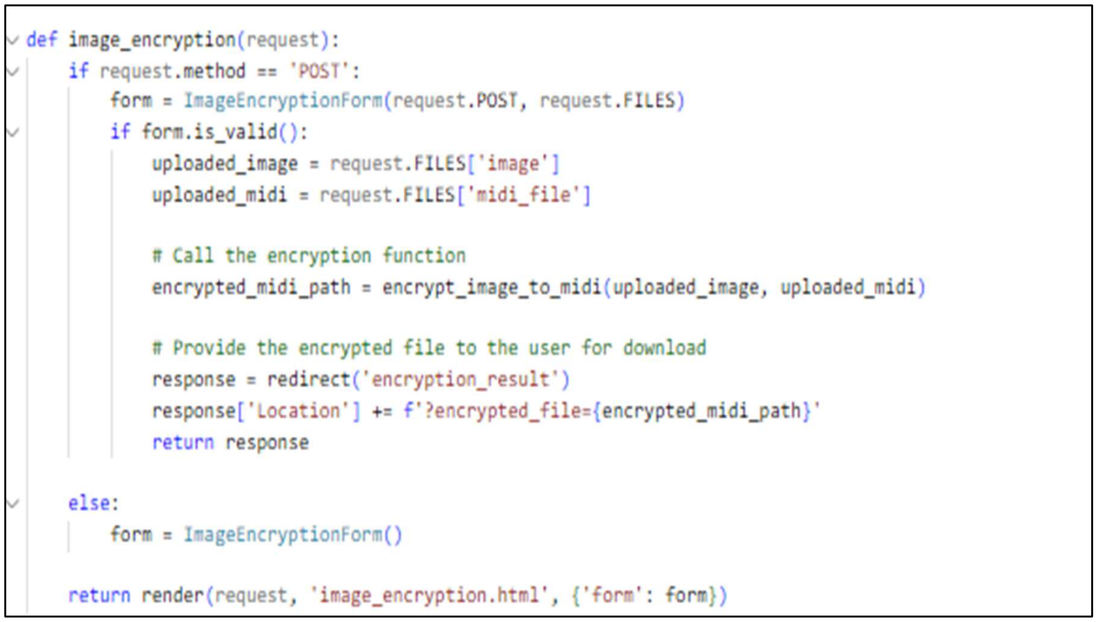
```
def decryption(request):
    if request.method == 'POST':
        form = DecryptionForm(request.POST, request.FILES)

        if form.is_valid():
            if 'midi_file' in request.FILES:
            # Get the uploaded encrypted MIDI file from the request.FILES dictionary
                uploaded_file = request.FILES['midi_file']

                # Call your decryption function with uploaded_file
                decrypted_text = decrypt_text_from_midi(uploaded_file)

                return render(request, 'decryption.html', {'decryption_form':form, 'result': f"Decrypted Text: {decrypted_text}"})
            else:
                return render(request, 'decryption.html', {'decryption_form': form, 'error_message': 'No file uploaded.'})

    else:
        form = DecryptionForm()

    return render(request, 'decryption.html', {'decryption_form': form})
```

Figure 3.16: Views of text decryption



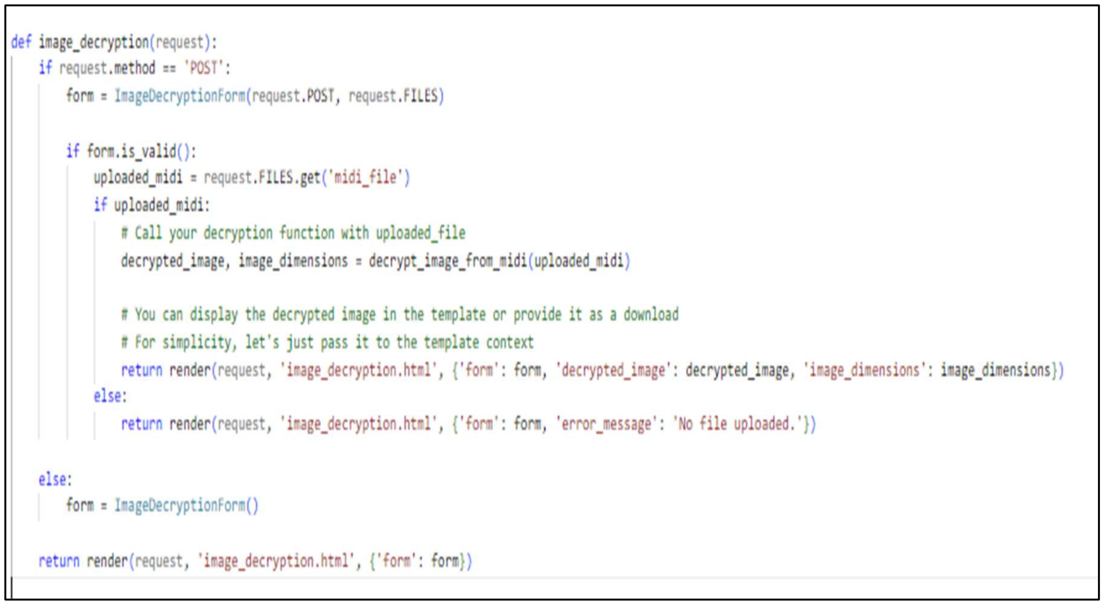
```python

def HomePage(request):
    return render(request, 'index.html')

# Temporary change in encryption_result view
def encryption_result(request):
    encrypted_file_path = request.GET.get('encrypted_file', '')

    if encrypted_file_path:
        try:
            # Use FileResponse directly
            response = FileResponse(open(encrypted_file_path, 'rb'))
            response['Content-Disposition'] = f'attachment; filename=Encrypted_File.mid'
            return response
        except FileNotFoundError:
            return render(request, 'encryption_result.html', {'error_message': 'File not found'})
    else:
        return render(request, 'encryption_result.html', {'error_message': 'File not found'})


```

Figure 3.17: Views of encryption result



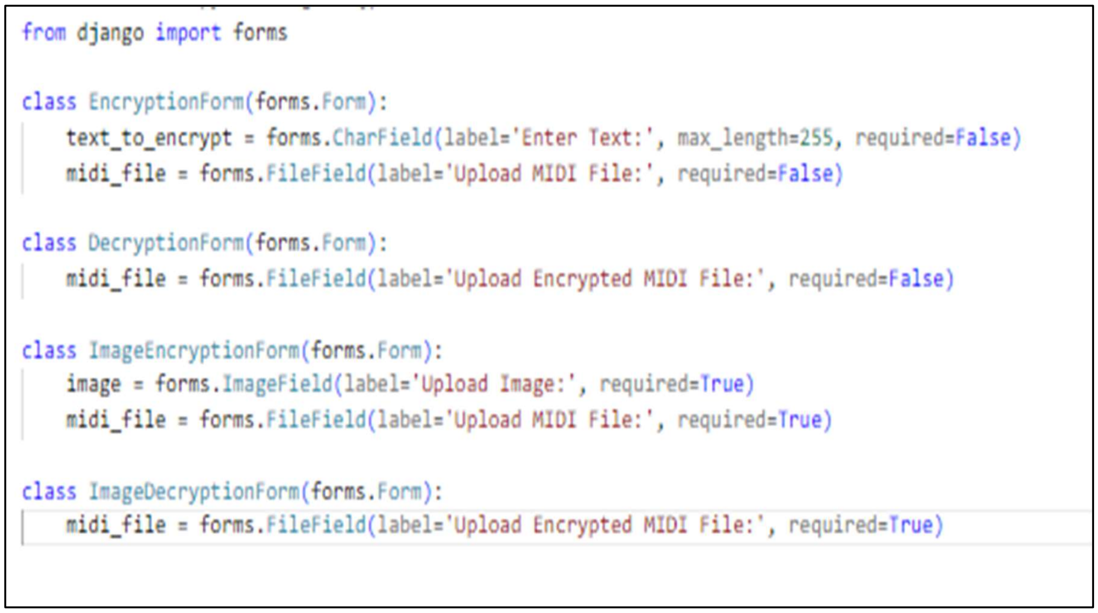
```python
def image_encryption(request):
    if request.method == 'POST':
        form = ImageEncryptionForm(request.POST, request.FILES)
        if form.is_valid():
            uploaded_image = request.FILES['image']
            uploaded_midi = request.FILES['midi_file']

            # Call the encryption function
            encrypted_midi_path = encrypt_image_to_midi(uploaded_image, uploaded_midi)

            # Provide the encrypted file to the user for download
            response = redirect('encryption_result')
            response['Location'] += f'?encrypted_file={encrypted_midi_path}'
            return response

    else:
        form = ImageEncryptionForm()

    return render(request, 'image_encryption.html', {'form': form})
```

Figure 3.18: Views of image encryption



```
def image_decryption(request):
    if request.method == 'POST':
        form = ImageDecryptionForm(request.POST, request.FILES)

        if form.is_valid():
            uploaded_midi = request.FILES.get('midi_file')
            if uploaded_midi:
                # Call your decryption function with uploaded_file
                decrypted_image, image_dimensions = decrypt_image_from_midi(uploaded_midi)

                # You can display the decrypted image in the template or provide it as a download
                # For simplicity, let's just pass it to the template context
                return render(request, 'image_decryption.html', {'form': form, 'decrypted_image': decrypted_image, 'image_dimensions': image_dimensions})
            else:
                return render(request, 'image_decryption.html', {'form': form, 'error_message': 'No file uploaded.'})

    else:
        form = ImageDecryptionForm()

    return render(request, 'image_decryption.html', {'form': form})
```

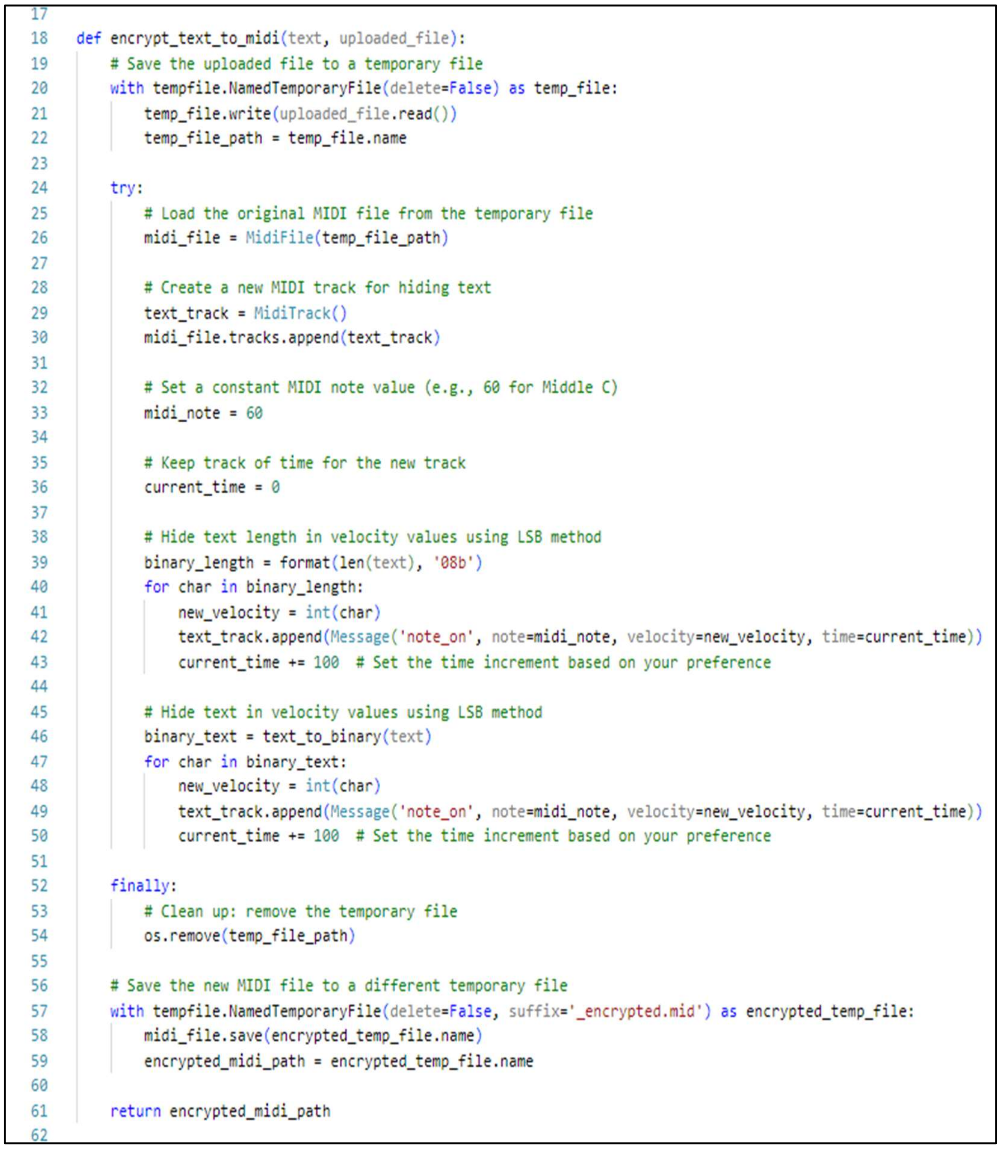
Figure 3.19:Views of image decryption



```
from django import forms

class EncryptionForm(forms.Form):
    text_to_encrypt = forms.CharField(label='Enter Text:', max_length=255, required=False)
    midi_file = forms.FileField(label='Upload MIDI File:', required=False)

class DecryptionForm(forms.Form):
    midi_file = forms.FileField(label='Upload Encrypted MIDI File:', required=False)

class ImageEncryptionForm(forms.Form):
    image = forms.ImageField(label='Upload Image:', required=True)
    midi_file = forms.FileField(label='Upload MIDI File:', required=True)

class ImageDecryptionForm(forms.Form):
    midi_file = forms.FileField(label='Upload Encrypted MIDI File:', required=True)
```

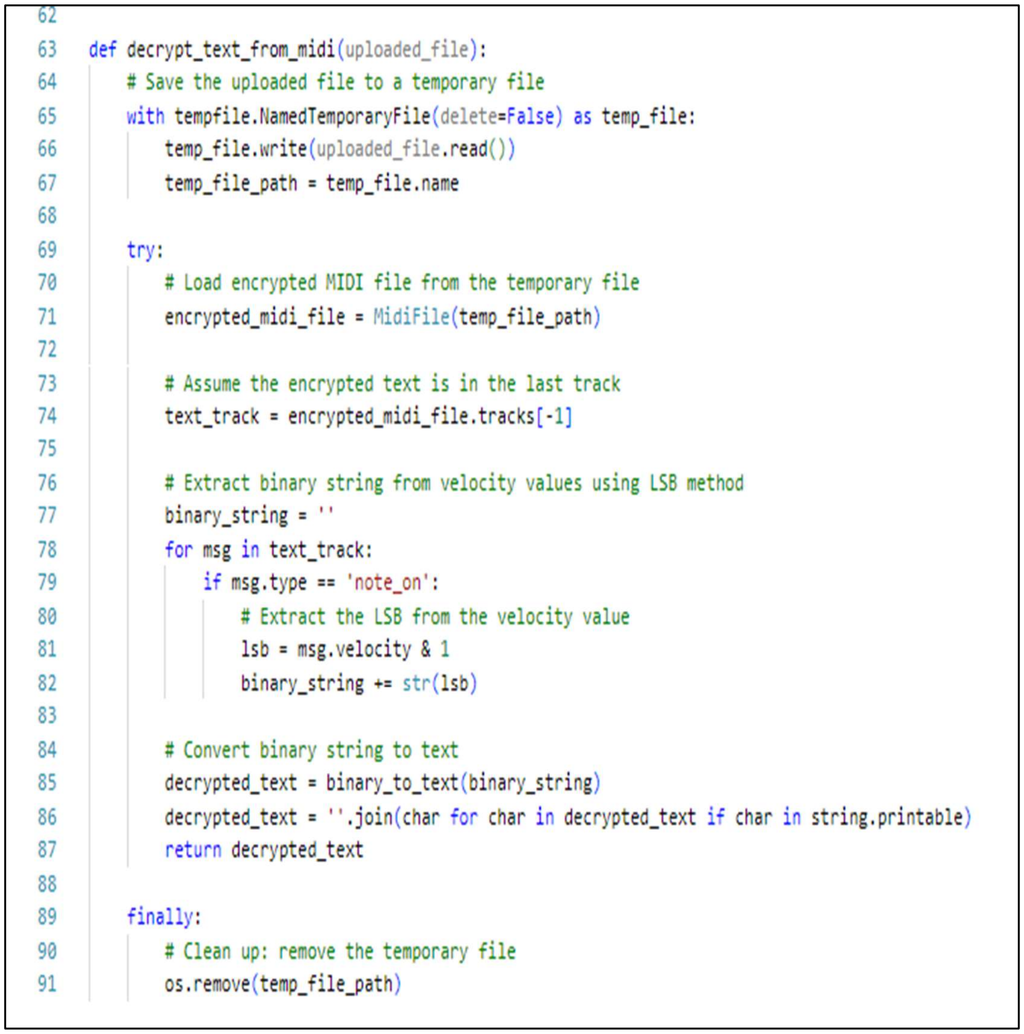
Figure 3.20: Forms

3.3.5 SCREENSHOTS OF THE WEBPAGES AND ALGORITHMS



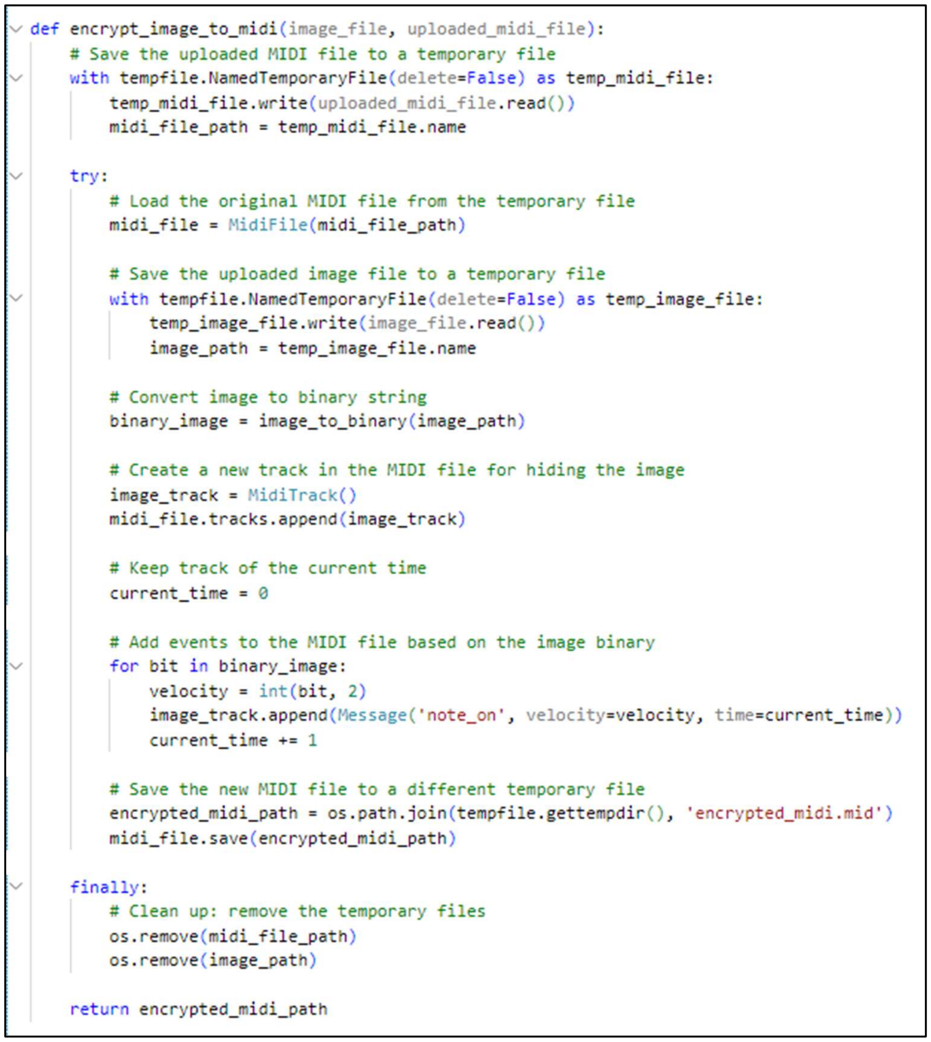
```python

def encrypt_text_to_midi(text, uploaded_file):
    # Save the uploaded file to a temporary file
    with tempfile.NamedTemporaryFile(delete=False) as temp_file:
        temp_file.write(uploaded_file.read())
        temp_file_path = temp_file.name

    try:
        # Load the original MIDI file from the temporary file
        midi_file = MidiFile(temp_file_path)

        # Create a new MIDI track for hiding text
        text_track = MidiTrack()
        midi_file.tracks.append(text_track)

        # Set a constant MIDI note value (e.g., 60 for Middle C)
        midi_note = 60

        # Keep track of time for the new track
        current_time = 0

        # Hide text length in velocity values using LSB method
        binary_length = format(len(text), '08b')
        for char in binary_length:
            new_velocity = int(char)
            text_track.append(Message('note_on', note=midi_note, velocity=new_velocity, time=current_time))
            current_time += 100  # Set the time increment based on your preference

        # Hide text in velocity values using LSB method
        binary_text = text_to_binary(text)
        for char in binary_text:
            new_velocity = int(char)
            text_track.append(Message('note_on', note=midi_note, velocity=new_velocity, time=current_time))
            current_time += 100  # Set the time increment based on your preference

    finally:
        # Clean up: remove the temporary file
        os.remove(temp_file_path)

    # Save the new MIDI file to a different temporary file
    with tempfile.NamedTemporaryFile(delete=False, suffix='_encrypted.mid') as encrypted_temp_file:
        midi_file.save(encrypted_temp_file.name)
        encrypted_midi_path = encrypted_temp_file.name

    return encrypted_midi_path

```

Figure 3.21: Encryption Algorithm of Text



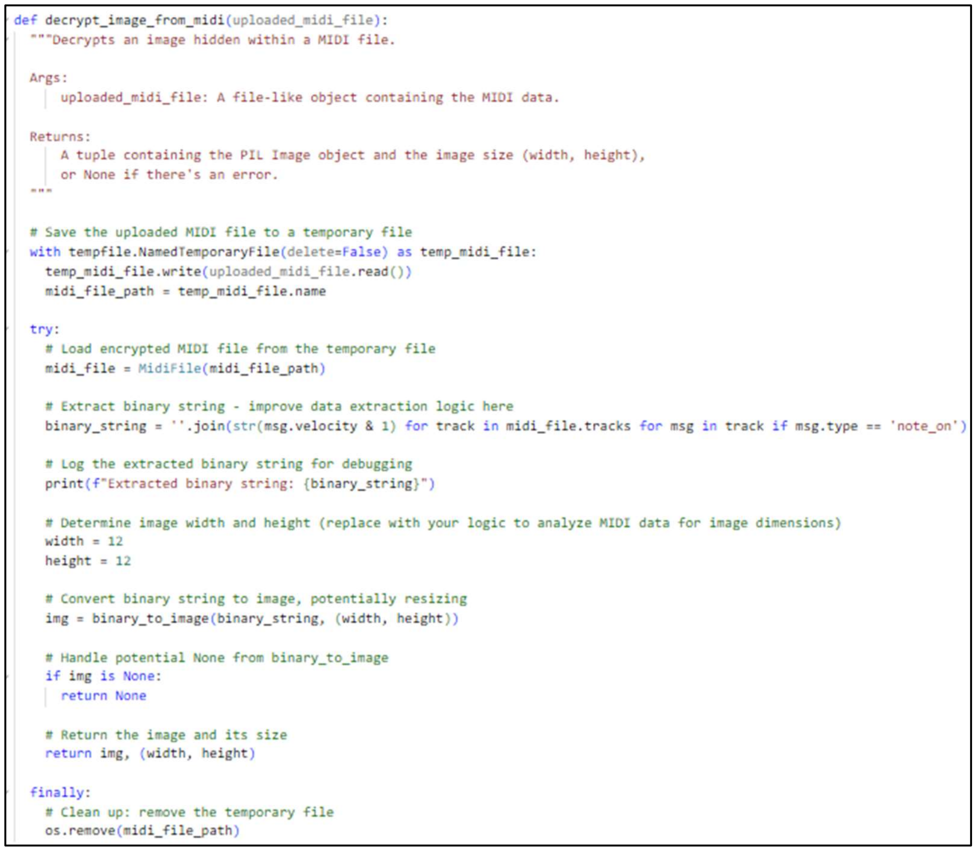
```python

def decrypt_text_from_midi(uploaded_file):
    # Save the uploaded file to a temporary file
    with tempfile.NamedTemporaryFile(delete=False) as temp_file:
        temp_file.write(uploaded_file.read())
        temp_file_path = temp_file.name

    try:
        # Load encrypted MIDI file from the temporary file
        encrypted_midi_file = MidiFile(temp_file_path)

        # Assume the encrypted text is in the last track
        text_track = encrypted_midi_file.tracks[-1]

        # Extract binary string from velocity values using LSB method
        binary_string = ''
        for msg in text_track:
            if msg.type == 'note_on':
                # Extract the LSB from the velocity value
                lsb = msg.velocity & 1
                binary_string += str(lsb)

        # Convert binary string to text
        decrypted_text = binary_to_text(binary_string)
        decrypted_text = ''.join(char for char in decrypted_text if char in string.printable)
        return decrypted_text

    finally:
        # Clean up: remove the temporary file
        os.remove(temp_file_path)
```

Figure 3.22: Decryption Algorithm of Text



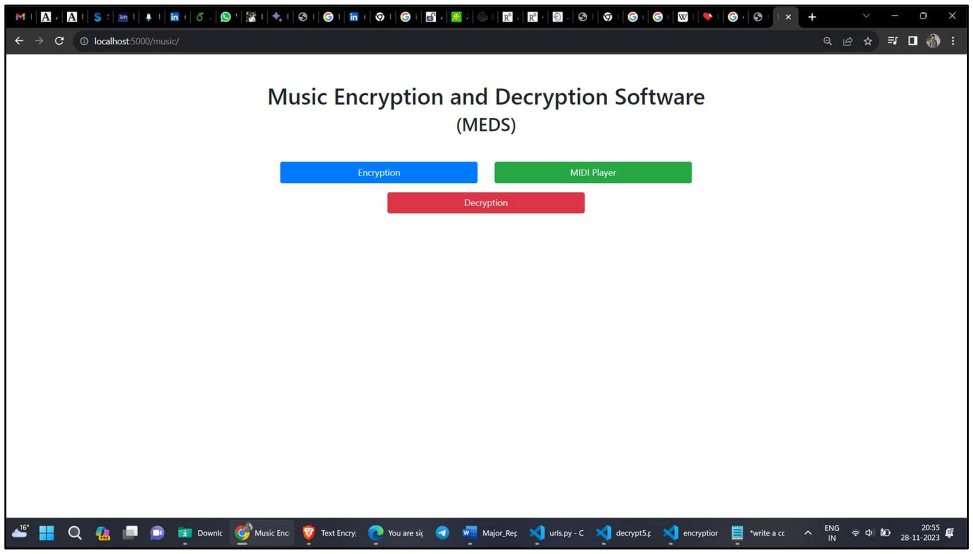
```python
def encrypt_image_to_midi(image_file, uploaded_midi_file):
    # Save the uploaded MIDI file to a temporary file
    with tempfile.NamedTemporaryFile(delete=False) as temp_midi_file:
        temp_midi_file.write(uploaded_midi_file.read())
        midi_file_path = temp_midi_file.name

    try:
        # Load the original MIDI file from the temporary file
        midi_file = MidiFile(midi_file_path)

        # Save the uploaded image file to a temporary file
        with tempfile.NamedTemporaryFile(delete=False) as temp_image_file:
            temp_image_file.write(image_file.read())
            image_path = temp_image_file.name

        # Convert image to binary string
        binary_image = image_to_binary(image_path)

        # Create a new track in the MIDI file for hiding the image
        image_track = MidiTrack()
        midi_file.tracks.append(image_track)

        # Keep track of the current time
        current_time = 0

        # Add events to the MIDI file based on the image binary
        for bit in binary_image:
            velocity = int(bit, 2)
            image_track.append(Message('note_on', velocity=velocity, time=current_time))
            current_time += 1

        # Save the new MIDI file to a different temporary file
        encrypted_midi_path = os.path.join(tempfile.gettempdir(), 'encrypted_midi.mid')
        midi_file.save(encrypted_midi_path)

    finally:
        # Clean up: remove the temporary files
        os.remove(midi_file_path)
        os.remove(image_path)

    return encrypted_midi_path
```

Figure 3.23: Encryption of Image



```
def decrypt_image_from_midi(uploaded_midi_file):
  """Decrypts an image hidden within a MIDI file.

  Args:
      uploaded_midi_file: A file-like object containing the MIDI data.

  Returns:
      A tuple containing the PIL Image object and the image size (width, height),
      or None if there's an error.
  """

  # Save the uploaded MIDI file to a temporary file
  with tempfile.NamedTemporaryFile(delete=False) as temp_midi_file:
    temp_midi_file.write(uploaded_midi_file.read())
    midi_file_path = temp_midi_file.name

  try:
    # Load encrypted MIDI file from the temporary file
    midi_file = MidiFile(midi_file_path)

    # Extract binary string - improve data extraction logic here
    binary_string = ''.join(str(msg.velocity & 1) for track in midi_file.tracks for msg in track if msg.type == 'note_on')

    # Log the extracted binary string for debugging
    print(f"Extracted binary string: {binary_string}")

    # Determine image width and height (replace with your logic to analyze MIDI data for image dimensions)
    width = 12
    height = 12

    # Convert binary string to image, potentially resizing
    img = binary_to_image(binary_string, (width, height))

    # Handle potential None from binary_to_image
    if img is None:
      return None

    # Return the image and its size
    return img, (width, height)

  finally:
    # Clean up: remove the temporary file
    os.remove(midi_file_path)
```

Figure 3.24: Decryption of Image



localhost:5000/music/

Music Encryption and Decryption Software

(MEDS)

Figure 3.25: Introductory Page

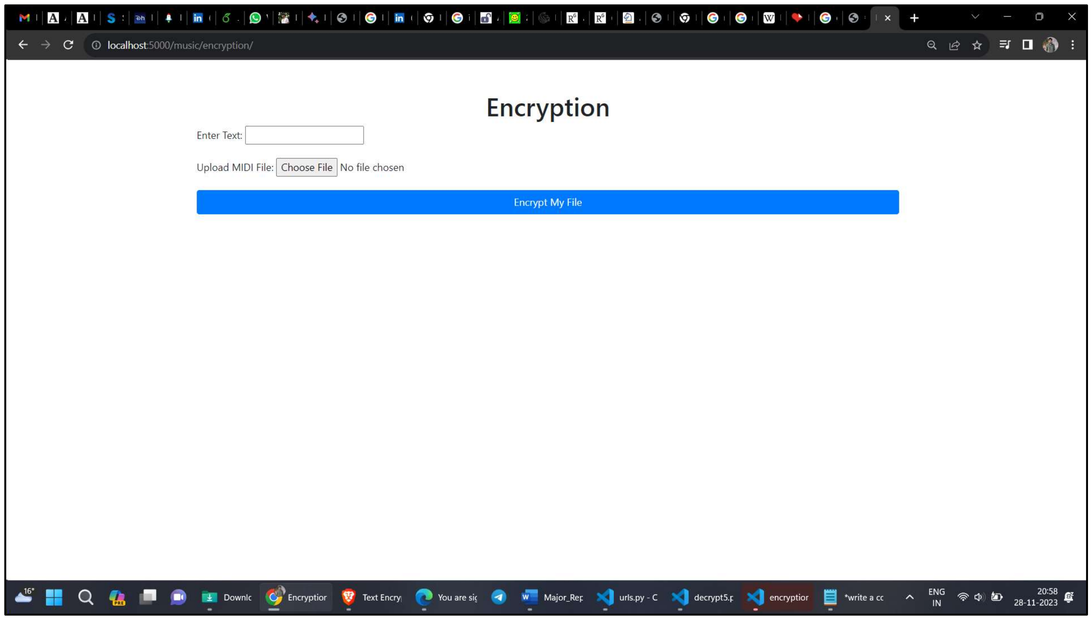

localhost:5000/music/encryption/
Encryption
Enter Text:
Upload MIDI File: Choose File No file chosen
Encrypt My File

Figure 3.26: Encryption Page

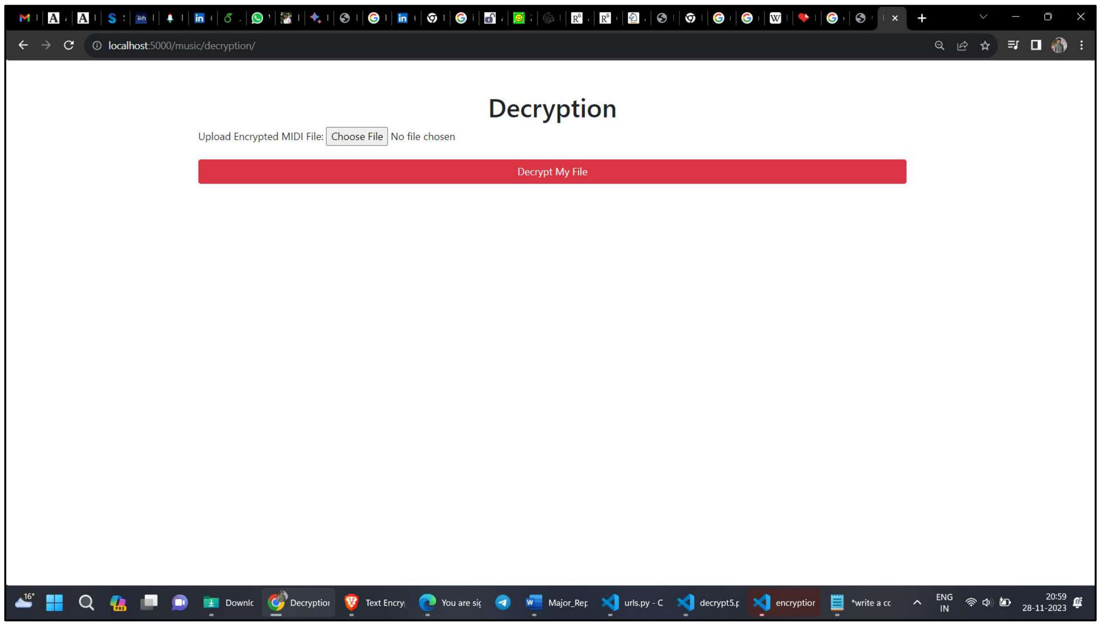

localhost:5000/music/decryption/
Decryption
Upload Encrypted MIDI File: Choose File No file chosen
Decrypt My File

Figure 3.27: Decryption Page

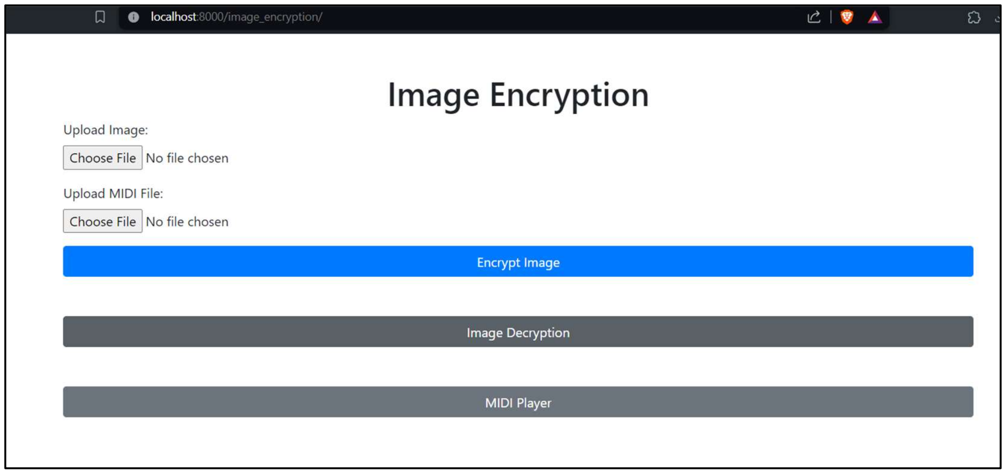


Figure 3.28: Image Encryption Page

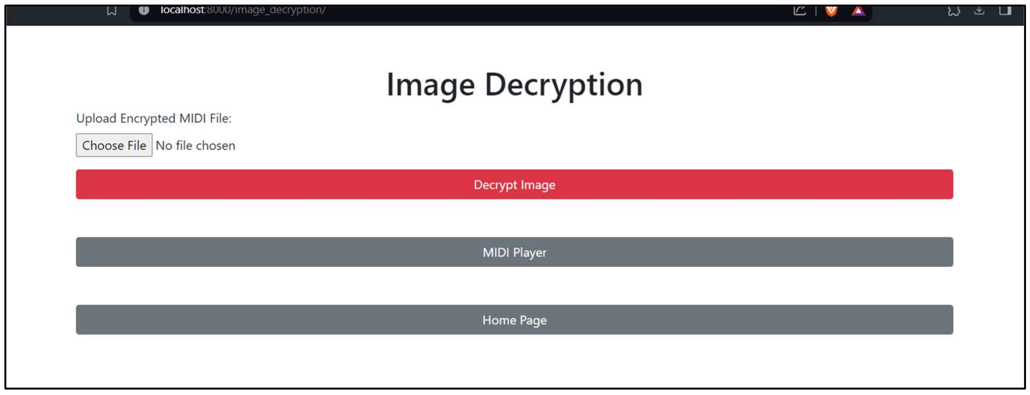

localhost:8000/image_decryption/
Image Decryption
Upload Encrypted MIDI File:
Choose File No file chosen
Decrypt Image
MIDI Player
Home Page

Figure 3.29:Image Decryption Page

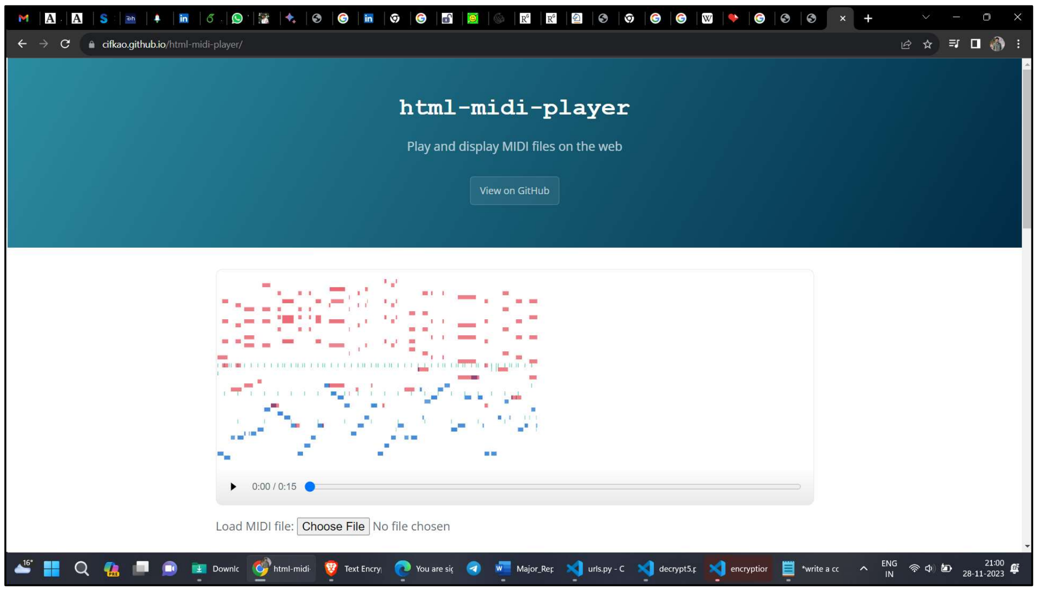

cifkao.github.io/html-midi-player/
html-midi-player
Play and display MIDI files on the web
View on GitHub
0:00 / 0:15
Load MIDI file: Choose File No file chosen

Figure 3.30: MIDI player

3.4 KEY CHALLENGES

The key challenges faced during the development of the project are:

- **Algorithm Design:** Designing the algorithm for both the encryption and decryption was a challenge. There were some decent possibilities to execute the algorithms. But deciding, based on the usability, and practicality, choosing one was tricky. After getting onto one, executing it was another challenge. In order to execute the encryption, there were a lot of error faced due to the time values of the MIDI file. After a lot of efforts and tries, error handling was introduced to the code, and the algorithm led to a success point.
  During the development of the decryption algorithm, the displayed text was having an issue, it was getting displayed with some extra characters on the front side of it. After a lot of efforts, we got successful, but using the string library of the python language to slice the extra characters.
  After a lot of testing, it was established that both the processes were giving out the desired results.
- **Web Application: -** The get the system useful for everyone, we needed to design a website. Choosing what framework is necessary is important. Initially we tried to convert the encryption and encryption to JavaScript as our algorithms were written using python. On failing, we decided to implement it using the django web framework. As this framework is based on python, it was convenient for us to integrate all the work together.
- **Deciding Future Scope:** There was a lot of confusion on deciding the direction of the project. After a lot of research, it was clear on the application of this project. We need to make this algorithm in such a way that we should be able to hide the image data as well. Also, a heavy work on the compression part was also required.

CHAPTER 4

TESTING

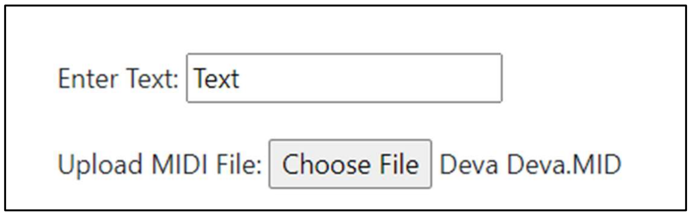
4.1 TESTING STRATEGY

The validation of the Music Encryption and Decryption Software (MEDS) can be approached through various metrics and comparisons. Here are some key aspects to consider:

- **Accuracy of Decryption: -** The designed algorithm is accurately decrypting the crypted music file. We can validate it by matching the decrypted text with the original input. In the following example, we can see the same. Through the following figures, 4.1 and 4.2 we can validate that the user's input "Text," is matching with the displayed text in decryption, that is "Text."
- **File Size Comparison:** - Another thing that validates the algorithms are the file size inflation. In the following figures 4.3 and 4.4, we can see the difference of file inflation, originally the file was of 12.3 KB, and after encryption it became 12.4KB with a margin of inflation of 0.1 KB, which is significantly better than previous algorithms. It should be considered that the size on the disk is same as before, that is 16 KB, which ensures the validation.
- **Performance Metrics:** - The performance of both the algorithms are efficiently faster than previous algorithms. The algorithms will have the features like it incorporates randomization and LSB encoding, which can enhance the embedding process less predictable. It tries to adapt encoding based on the velocity magnitude, providing flexibility in handling different scenarios. Also, the payload capacity depends on the size of the encrypted message stream. The performance of the algorithm depends on the complexity of the MIDI file, the size of the message stream, Mean Absolute Error, Peak Signal-To-Noise Ratio.

4.2 TEST CASES AND OUTCOMES

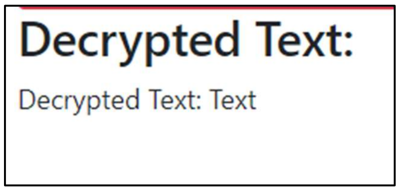
- The test outcomes of the strategies discussed are presented in this section.



Enter Text: Text

Upload MIDI File: Choose File Deva Deva.MID

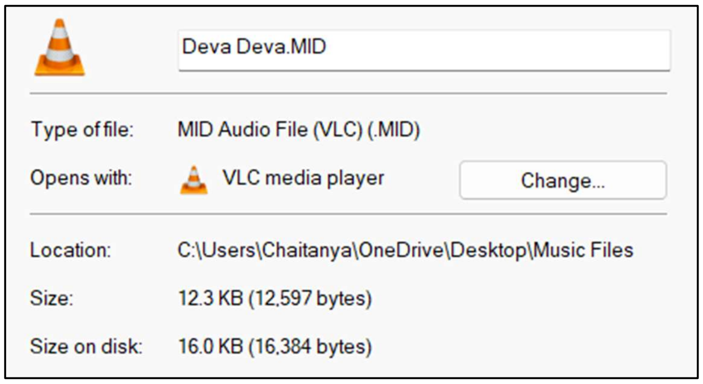
Figure 4.1: User's input



Decrypted Text:

Decrypted Text: Text

Figure 4.2: Decrypted Text



Deva Deva.MID

Type of file: MID Audio File (VLC) (.MID)

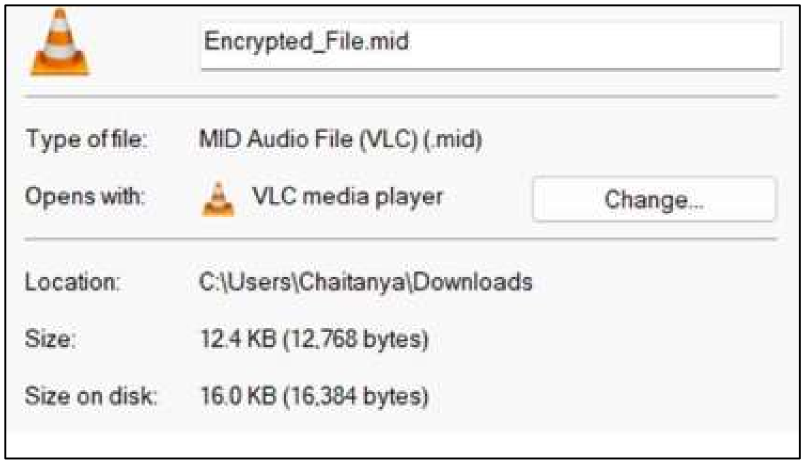
Opens with: VLC media player Change...

Location: C:\Users\Chaitanya\OneDrive\Desktop\Music Files

Size: 12.3 KB (12,597 bytes)

Size on disk: 16.0 KB (16,384 bytes)

Figure 4.3: File size of original file


Encrypted_File.mid

Type of file: MID Audio File (VLC) (.mid)

Opens with: VLC media player

Location: C:\Users\Chaitanya\Downloads

Size: 12.4 KB (12,768 bytes)

Size on disk: 16.0 KB (16,384 bytes)

Figure 4.4: File size of encrypted file

- **Mean Absolute Error:** The Mean Absolute Error is calculated by taking the average of the absolute differences between corresponding elements of the original and encrypted datasets.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i| \tag{4.1}$$

Where: n is the total number of elements

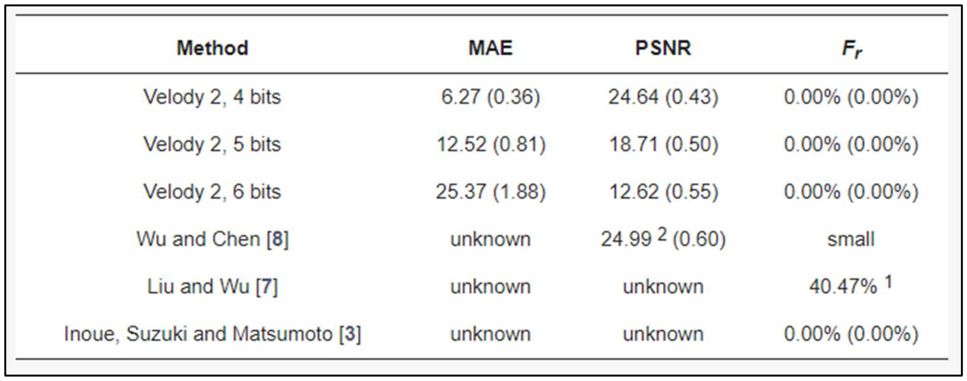
$\boldsymbol{x_i}$ is the ith element of the original dataset

$\boldsymbol{y_i}$ is the ith element of the encrypted dataset

- **Peak Signal-to-Nosie ratio (PSNR):** The Peak Signal-to-Noise Ratio is a logarithmic measurement of the ratio between the maximum possible power of the original signal and the power of the noise that affects the quality of its representation.

$$PSNR = 10.\log_{10}\left(\frac{\max possible\ value^2}{MSE}\right) \tag{4.2}$$

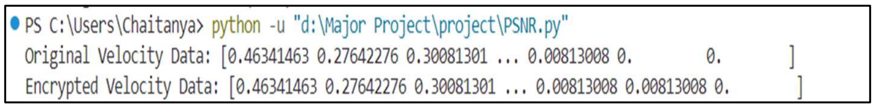
Where: max possible value is the maximum possible value of the data (in this case, the maximum possible velocity value).
MSE is the Mean Squared Error, which is the average of the squared differences between corresponding elements of the original and encrypted datasets.



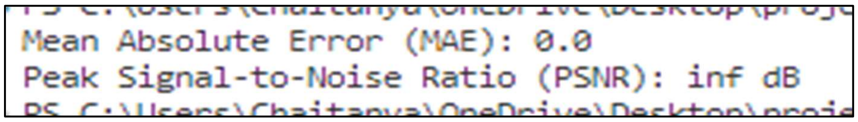
Method	MAE	PSNR	F_r
Velody 2, 4 bits	6.27 (0.36)	24.64 (0.43)	0.00% (0.00%)
Velody 2, 5 bits	12.52 (0.81)	18.71 (0.50)	0.00% (0.00%)
Velody 2, 6 bits	25.37 (1.88)	12.62 (0.55)	0.00% (0.00%)
Wu and Chen [8]	unknown	24.99 [2] (0.60)	small
Liu and Wu [7]	unknown	unknown	40.47% [1]
Inoue, Suzuki and Matsumoto [3]	unknown	unknown	0.00% (0.00%)

Figure 4.5: Previous Algorithm's Data [2]

- The figure 4.5 shows the mean absolute errors, peak signal to noise ratio of some previous algorithms and the file size. The same for our algorithm is:



```
PS C:\Users\Chaitanya> python -u "d:\Major Project\project\PSNR.py"
Original Velocity Data: [0.46341463 0.27642276 0.30081301 ... 0.00813008 0.         0.        ]
Encrypted Velocity Data: [0.46341463 0.27642276 0.30081301 ... 0.00813008 0.00813008 0.        ]
```



```
PS C:\Users\Chaitanya\OneDrive\Desktop\proje
Mean Absolute Error (MAE): 0.0
Peak Signal-to-Noise Ratio (PSNR): inf dB
PS C:\Users\Chaitanya\OneDrive\Desktop\proje
```

Figure 4.6: Results for MAE and PSNR

- **Results:**
  MAE: A value of 0.0 implies that, on average, there is no difference between the original and encrypted velocity values. This suggests a perfect match between the two datasets.
  PSNR: An infinite PSNR also suggests that the quality of the encrypted data is theoretically perfect compared to the original data. PSNR is a measure of how well the original signal can be reconstructed from the noise introduced during encryption. In this case, the absence of noise (MAE = 0) leads to an infinite PSNR.
  File Size: The file size in comparison to the above data, is categorized in "small," as the size increased from 12.3 to 12.4 KB.

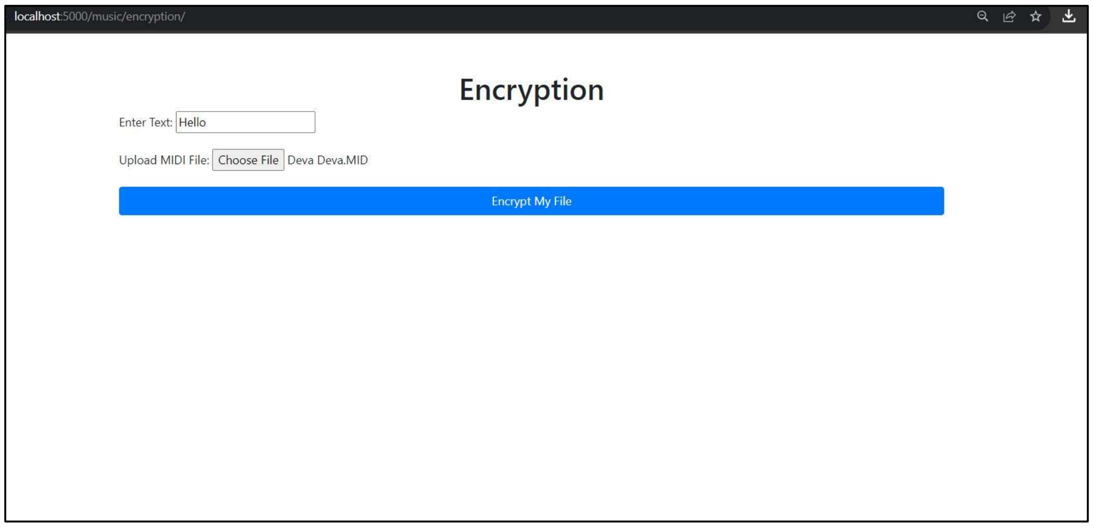
CHAPTER 5

RESULTS AND EVALUATION

5.1 RESULTS

The ecosystem provided a set of successful results. The system is efficiently converting the textual data into binary and then hiding it into the MIDI file. The decryption process is successfully converting the encrypted file to plain text.

Example of the result: - On the encryption page, a text “Hello” was given as an input in the text area, and a file named “Deva Deva.MID” was uploaded. The algorithm successfully encrypted the text into the following file. After this, a file, named “encrypted_file.mid” is downloaded on the user’s computer.



localhost:5000/music/encryption/

Encryption

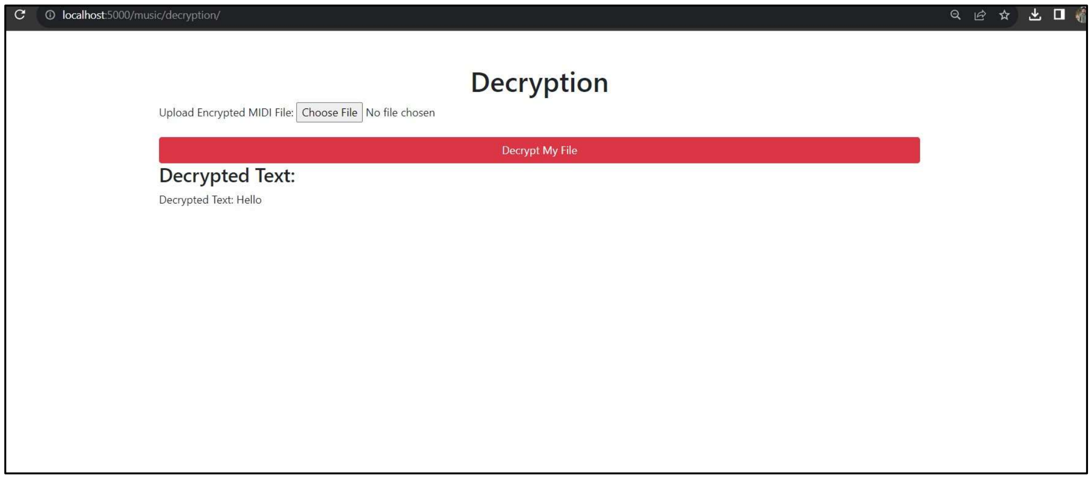
Enter Text: Hello

Upload MIDI File: Choose File Deva Deva.MID

Figure 5.1: Encryption Result

After this process, on uploading the encrypted_file.mid on the decryption page, the algorithm successfully extracted the hidden text. Once the file is loaded, the actual extraction of the hidden information commences. In the encryption process, text is embedded in the velocity values of the MIDI events. During decryption, each "note_on" event in the last track of the MIDI file is analyzed to retrieve the least significant bit (LSB) from the velocity value. These

LSBs are then concatenated to reconstruct a binary string representing the hidden text. Finally, as a precautionary measure and to maintain system cleanliness, temporary files generated during the decryption process are systematically removed.



Decryption

Upload Encrypted MIDI File: Choose File No file chosen

Decrypt My File

Decrypted Text:

Decrypted Text: Hello

Figure 5.2: Decryption Result

In case, the user wants to play the music file, the MIDI player provided plays it without a lag.

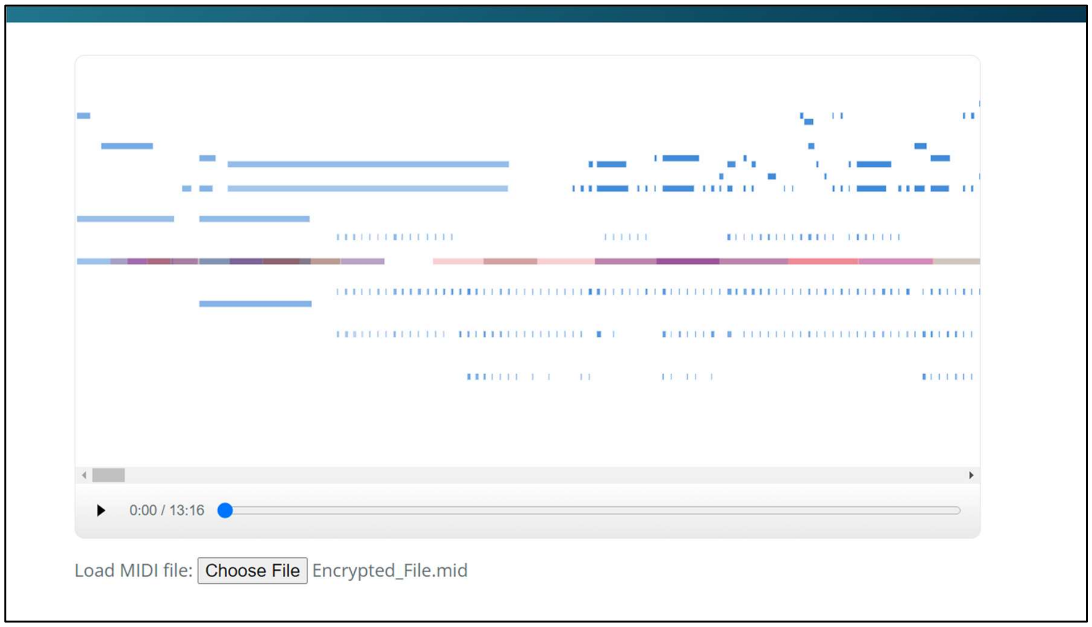


Figure 5.3: Player playing encrypted file

CHAPTER 6

CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSION

In conclusion, the development of the Music Encryption and Decryption Software (MEDS) represents a successful fusion of cryptographic techniques and art. Through the integration of Django, a python web framework, with the encryption and decryption algorithm, we have created a user-friendly platform that empowers users to seamlessly secure and unveil hidden information within MIDI files.

The encryption algorithm, in the MEDS application, offers a secure and efficient method of embedding text within music files. The decryption, on the reverse steps of encryption, effortlessly extract hidden text. The choice of Django as the underlying web framework proves to be instrumental, offering a flexibility as the main algorithms are present in the same language. Its Model-View-template (MVT) architecture facilitates the creation of the distinct web pages, each serving a specific purpose, ranging from text input and file upload to decryption and playback functionalities.

6.2 FUTURE SCOPE

The following are some important future scopes of this project: -

- **Better Compression: -** There is possibility of increasing the compression efficiency of the encryption and decryption algorithms. With the current development in the algorithm, there is a little inflation in the file size of the encrypted file. To maintain the original size, there is a need to improve the algorithm using various compression techniques, like using the zlib library of the python language. The textual data can be reduced to compressed file, then encrypted inside a music file. In decryption, the same can be decompressed. This is must-work-upon future scope and is high on the priority list.
- **Graphical User Interface (GUI) Improvements: -** Enhancing the visual aesthetics of the current web application is another future scope. This will improve the user experience of the MEDS interface, making it more intuitive and engaging. We aim to use the REACT for the frontend in future of improve the display of the software.

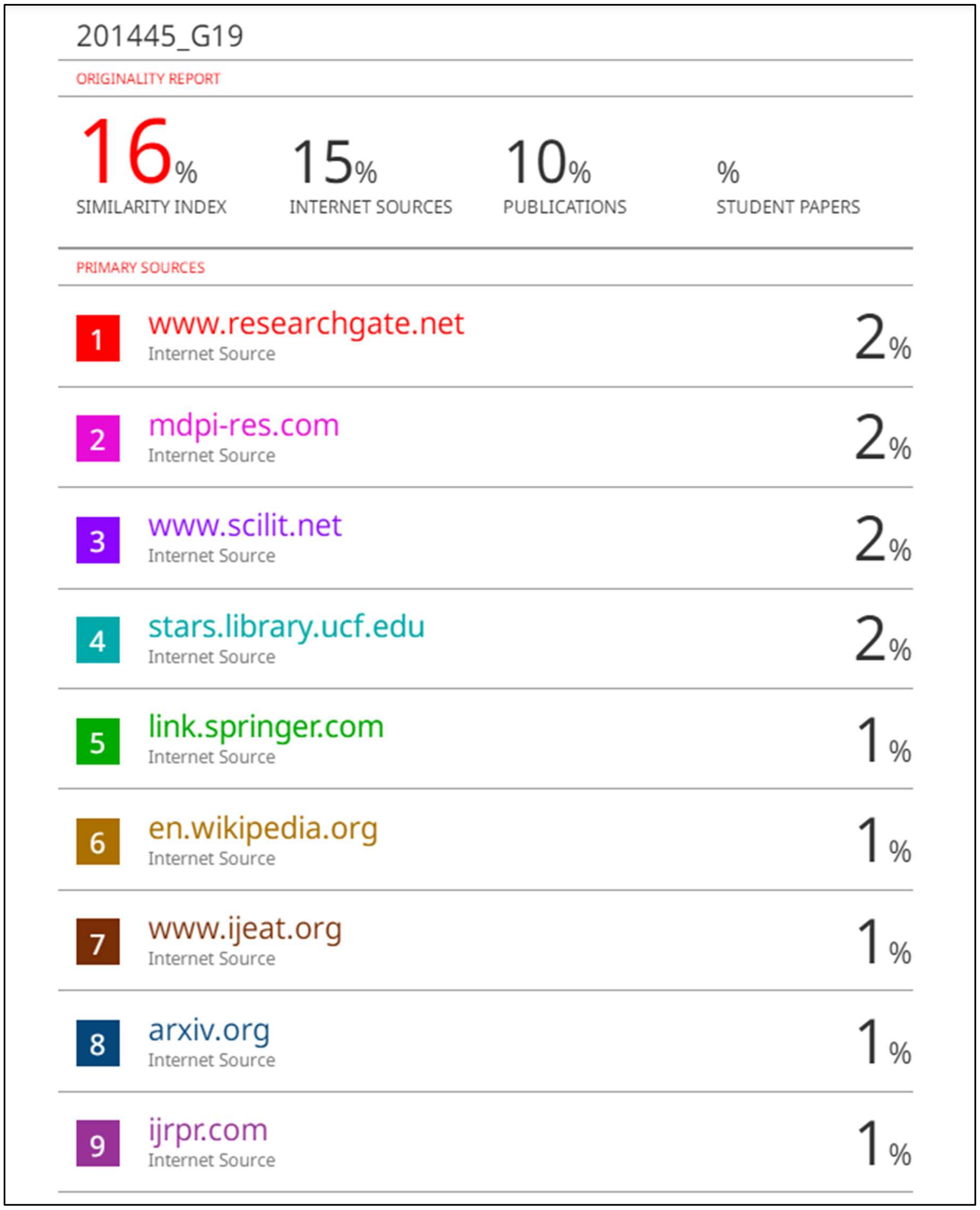
- **Deployment:** - The current system has not yet been deployed for the users to use. Choosing a capable server is also one important future scope to handle the user’s traffic on the software.
- **Improvement in Encryption: -** There is a scope of improvement in the encryption algorithm. Currently, we are hiding textual and image data inside the music files. Textual data seems pretty much efficient but image encryption and decryption is time taking and can be implemented using some better compression integrated processes.

REFERENCES

1. D. Rao and S.G. Koolagudi, "Music Steganography based on Carnatic Music," International Journal of Engineering and Advanced Technology (IJEAT), vol. 9, 2019.
2. E. Järpe and M. Weckstén, "Velody 2—Resilient High-Capacity MIDI Steganography for Organ and Harpsichord Music," Appl. Sci., vol. 11, 2021.
3. C. Helsel, "Musical Cryptography Using Long Short-Term Memory Networks," Honors Theses, University of Central Florida, Orlando, FL, 2019.
4. R. Krishnan, M. S. R. Murthy, and K. R. Venugopal, "An Intelligent Text Encryption System Using Musical Notes," 2011 International Conference on Computer Science and Information Technology (ICCSIT), 2011
5. D. -C. Wu, C. -Y. Hsiang and M. -Y. Chen, "Steganography via MIDI Files by Adjusting Velocities of Musical Note Sequences With Monotonically Non-Increasing or Non-Decreasing Pitches," in *IEEE Access*, vol. 7, 2019
6. Liu, YH., Wu, DC. A high-capacity performance-preserving blind technique for reversible information hiding via MIDI files using delta times. *Multimed Tools Appl* **79**, 17281–17302 (2020). https://doi.org/10.1007/s11042-019-08526-9
7. Y. Huang, H. Zhang, X. Zhang, and J. Cao, "NuText: A Novel Method for Music Encoding and Its Practical Application," IEEE Access, vol. 11, 2023
8. E. Järpe and M. Weckstén, "A Survey of Music Cryptography and Steganography," in IEEE Access, vol. 9, 2021
9. Y. Huang, H. Zhang, X. Zhang, and J. Cao, "Deep Learning-Based Music Steganography for Copyright Protection," IEEE Transactions on Information Forensics and Security, vol. 17, no. 4, 2022
10. S. E. El-Khamy, N. O. Korany, and M. H. El-Sherif, "Audio Steganography Using Discrete Cosine Transform (DCT) & Discrete Wavelet Transformation (DWT)," Multimedia Tools and Applications, vol. 76, 2017
11. D. Singh and G. Kaur, "A Wavelet Transform Based Audio Steganography Scheme Using Least Significant Bit Technique," International Journal of Advanced Research in Computer Science and Engineering (IJARCSE), vol. 5, 2016.
12. CIFKAO, "HTML MIDI Player," Available: https://cifkao.github.io/html-midi-player/. [Accessed: 14 November,2023].

13. Django Project, "Django Documentation," Available: https://docs.djangoproject .com/en/4.2/. [Accessed: 25 October, 2023]
14. D. Dutta and T. Kumar, "A Symmetric Key Algorithm for Cryptography using Music," Int. J. Comput. Appl., vol. 26, 2011.
15. H. J. S. A. A. Sellahewa and H. Jassim, "A high-capacity performance-preserving blind technique for reversible information hiding via MIDI files using delta times," Multimedia Tools and Applications, vol. 78, 2019
16. "MIDI (Musical Instrument Digital Interface)," TechTarget, Available: https://www. techtarget.com/whatis/definition/MIDI-Musical-Instrument-Digital-Interface. [Accessed: 16 September,2023].
17. D. K. Bhattacharyya and S. Nandi, "Efficient Design of ENCA Based Cipher System," IETE Journal of Research, vol. 46, no. 3, pp. 163-173, 2000.
18. E. Sams, "Musical Cryptography," Available: https://ericsams.org/index.php/on-cryptography/333-musical-cryptography?start=2 [Accessed: 26 October, 2023]
19. PuzzleNation, "Musical Cryptography: Hiding Messages in the Music," Available: https://blog.puzzlenation.com/2021/01/21/musical-cryptography-hiding-messages-in-the-music/ [Accessed: 23 October, 2023]
20. D. Bruff, "Episode 36: Musical Cryptography," Available: https://derekbruff.org /blogs/fywscrypto/2019/12/19/episode-36-musical-cryptography/. [Accessed: 28 October, 2023]
21. AudioCipher, "Musical Cryptogram," Available: https://www.audiocipher.com /post/musical-cryptogram. [Accessed: 30 October, 2023]
22. National Institute of Standards and Technology, "Cryptography," Available: https:// www. nist.gov/cryptography. [Accessed: 17 September, 2023]
23. Southern Utah University, "The Tech Impact on the Music Industry," Available: https://online.suu.edu/degrees/business/master-music-technology/tech-impact-music-industry/. [Accessed: 18 September, 2023]
24. Hacker Noon, "Hiding Secrets: Steganography in Digital Arts and NFTs," Available: https://hackernoon.com/hiding-secrets-steganography-in-digital-arts-and-nfts-531z35f0. [Accessed: 20 September, 2023]
25. Lucidchart. (n.d.). Lucidchart Flowchart Maker. Lucidchart. https://www.lucidchart. com/pages/examples/flowchart-maker [Accessed: 21 September, 2023]

APPENDIX



201445_G19

ORIGINALITY REPORT

16%	15%	10%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.researchgate.net Internet Source	2%
2	mdpi-res.com Internet Source	2%
3	www.scilit.net Internet Source	2%
4	stars.library.ucf.edu Internet Source	2%
5	link.springer.com Internet Source	1%
6	en.wikipedia.org Internet Source	1%
7	www.ijeat.org Internet Source	1%
8	arxiv.org Internet Source	1%
9	ijrpr.com Internet Source	1%