

# Audio Integrity Checking Application

Verifying the integrity of audio files.

**Submitted by:**

Name: Soham Banerjee

Roll No: 38

Enrollment No: 12022002016053

Year & Department: 3<sup>rd</sup> Year (6<sup>th</sup> Sem), CSE AIML

Academic Year: 2022-26

**Submitted to:**

Prof. Indrajit Dey & Prof. Pabak Indu

# Audio Integrity Checking Application

## 1. Introduction

In the digital era, the integrity of audio files is crucial for ensuring authenticity and preventing unauthorized modifications. An audio integrity checking application verifies whether an audio file has been altered by generating and comparing hash values. This project aims to develop a GUI-based application to help users verify the integrity of their audio files efficiently.

## 2. Objective

The primary objective of this project is to create a simple and user-friendly GUI application that allows users to:

- Select an audio file for integrity verification.
- Generate a unique hash value (SHA-256) for the selected file.
- Save and compare hash values to detect file alterations.

## 3. Working Process

### 3.1 File Selection

The user selects an audio file using a file dialog box within the application.

### 3.2 Hash Generation

A SHA-256 hash is computed for the selected audio file using the Python `hashlib` library. This hash acts as a unique fingerprint for the file.

### 3.3 Hash Saving

Users can save the generated hash in a text file for future verification.

### 3.4 Integrity Verification

The application allows users to compare a newly generated hash with a previously saved hash to check if the file has been modified.

## 4. Implementation Details

The implementation of the audio integrity checking application is divided into the following parts:

### 4.1 File Selection

```
```python
function select_file():
    open file dialog
    store selected file path
```
```

## 4.2 Hash Generation

```
```python
function compute_hash(filepath):
    open file in binary mode
    read file data in chunks
    generate SHA-256 hash
    return hash value
```
```

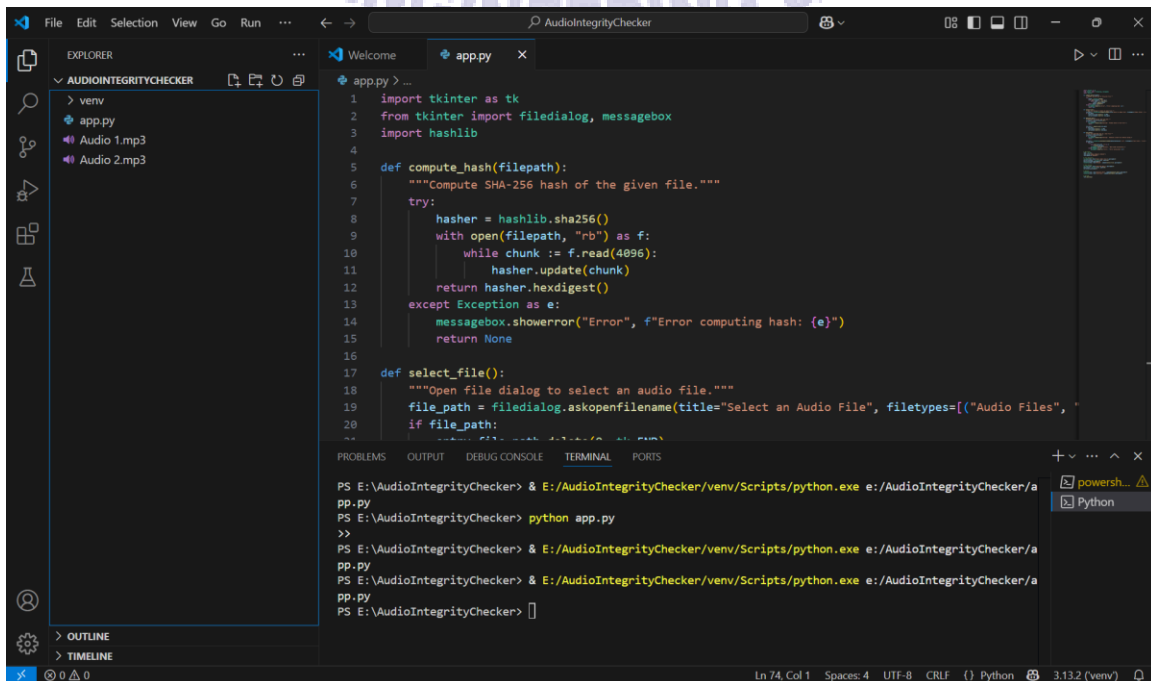
## 4.3 Saving Hash

```
```python
function save_hash(hash_value):
    open save dialog
    write hash value to a text file
```
```

## 4.4 Integrity Verification

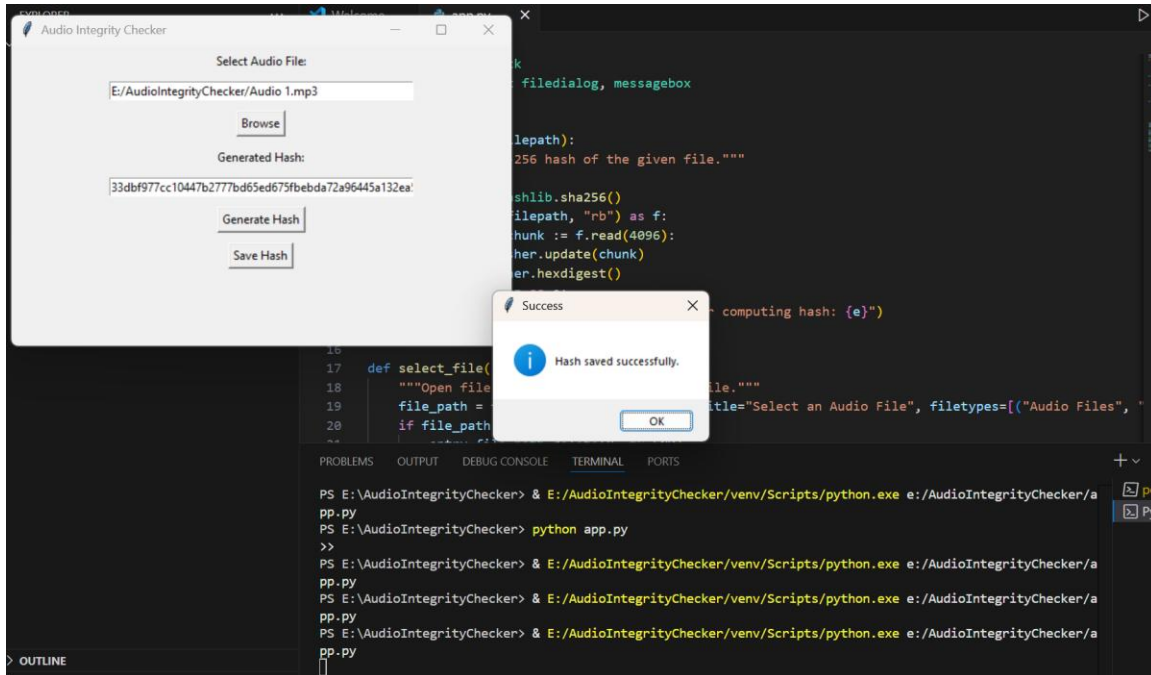
```
```python
function verify_hash(filepath, stored_hash):
    compute new hash from file
    compare new hash with stored hash
    return verification result
```
```

## 5. Results and Observation



```
File Edit Selection View Go Run ... AudioIntegrityChecker
EXPLORER
  v AUDIOINTEGRITYCHECKER
    > venv
    > app.py
    > Audio 1.mp3
    > Audio 2.mp3
  Welcome
  app.py
  1 import tkinter as tk
  2 from tkinter import filedialog, messagebox
  3 import hashlib
  4
  5 def compute_hash(filepath):
  6     """Compute SHA-256 hash of the given file."""
  7     try:
  8         hasher = hashlib.sha256()
  9         with open(filepath, "rb") as f:
 10             while chunk := f.read(4096):
 11                 hasher.update(chunk)
 12             return hasher.hexdigest()
 13     except Exception as e:
 14         messagebox.showerror("Error", f"Error computing hash: {e}")
 15         return None
 16
 17 def select_file():
 18     """Open file dialog to select an audio file."""
 19     file_path = filedialog.askopenfilename(title="Select an Audio File", filetypes=[("Audio Files", ".mp3")])
 20     if file_path:
 21         return file_path
 22     return None
 23
 24 def save_hash(hash_value):
 25     """Save the hash value to a text file."""
 26     save_path = filedialog.asksaveasfilename(title="Save Hash Value", filetypes=[("Text Files", ".txt")])
 27     if save_path:
 28         with open(save_path, "w") as f:
 29             f.write(hash_value)
 30
 31 def verify_hash(filepath, stored_hash):
 32     """Verify the hash of the given file against the stored hash."""
 33     new_hash = compute_hash(filepath)
 34     if new_hash == stored_hash:
 35         return "File is authentic."
 36     else:
 37         return "File is not authentic."
 38
 39 def main():
 40     file_path = select_file()
 41     if file_path:
 42         hash_value = compute_hash(file_path)
 43         if hash_value:
 44             save_hash(hash_value)
 45             print(f"Hash generated and saved: {hash_value}")
 46         else:
 47             print("Error computing hash.")
 48     else:
 49         print("No file selected.")
 50
 51 if __name__ == "__main__":
 52     main()
 53
 54 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
 55 PS E:\AudioIntegrityChecker> E:\AudioIntegrityChecker\venv\Scripts\python.exe E:\AudioIntegrityChecker\app.py
 56 PS E:\AudioIntegrityChecker> python app.py
 57 >>
 58 PS E:\AudioIntegrityChecker> E:\AudioIntegrityChecker\venv\Scripts\python.exe E:\AudioIntegrityChecker\app.py
 59 PS E:\AudioIntegrityChecker> E:\AudioIntegrityChecker\venv\Scripts\python.exe E:\AudioIntegrityChecker\app.py
 60 PS E:\AudioIntegrityChecker> E:\AudioIntegrityChecker\venv\Scripts\python.exe E:\AudioIntegrityChecker\app.py
 61 PS E:\AudioIntegrityChecker>
 62
 63 Ln 74, Col 1 Spaces: 4 UTF-8 CRLF () Python 3.13.2 (venv)
```





**GitHub Link:** [https://github.com/Soham-Project-IEM/Audio\\_Integ\\_Chek\\_CyberS.git](https://github.com/Soham-Project-IEM/Audio_Integ_Chek_CyberS.git)

## 6. Security Considerations

- The use of **SHA-256** hashing ensures a high level of security and uniqueness for file verification.
- No file modifications occur within the application, ensuring the integrity of the original data.
- The application does not store user data, maintaining privacy and security.

## 7. Conclusion

This project successfully implements an audio integrity checking application using Python and Tkinter. The application provides a simple and effective way to verify the authenticity of audio files using cryptographic hashing techniques. Further improvements can include support for additional hashing algorithms and a database for storing hash values securely.

## 8. References

1. Menezes, A., van Oorschot, P., & Vanstone, S. (1996). Handbook of Applied Cryptography. CRC Press.
2. National Institute of Standards and Technology (NIST). (2020). SHA-256 Algorithm Specification.

## 9. Acknowledgment

I would like to express my gratitude to my instructors Prof. Indrajit Dey & Prof. Pabak Indu and peers for their valuable guidance and support throughout the development of this project.