A red text on a black background

AI-generated content may be incorrect.

**PROJECT AND TEAM INFORMATION**

## Project Title

|  |
| --- |
| Image-based and Audio-based Steganography for Secure Data Transmission |

## Student/Team Information

|  |  |
| --- | --- |
| Team Name: | Team SilentCipher |
| **Team member 1 (Team Lead)**  Sharma Aayushi  230121926  [sharmaaayushi953@gmail.com](mailto:sharmaaayushi953@gmail.com) |  |
| **Team member 2**  Akansha Rawat  230122459  akansharawat@gmail.com |  |

|  |  |
| --- | --- |
| **Team member 3**  Tanuja  230124746  [tanujachuphal7@gmail.com](mailto:tanujachuphal7@gmail.com) |  |
| **Team member 4**  Rawat Singh Gaurav  230121457  [singhrawatg121@gmail.com](mailto:singhrawatg121@gmail.com) |  |

**PROJECT PROGRESS DESCRIPTION (35 pts)**

## Project Abstract (2 pts)

(Brief restatement of your project’s main goal. Max 300 words).

|  |
| --- |
| In today’s digital world, privacy and secure communication are more important than ever. This project introduces a simple yet powerful web application that lets users hide secret messages inside image and audio files—a technique known as steganography.  The application is built using Python and Flask, making it lightweight and easy to use. Users can upload an image or audio file, type their secret message, and with just a click, the message gets hidden inside the file. The modified file looks and sounds almost exactly the same, but it quietly carries the hidden message. When needed, the user (or the receiver) can upload the stego file back into the app to reveal the original hidden text.  For images, the project uses a method called LSB (Least Significant Bit), which slightly changes the color values of pixels to store information. For audio, it carefully changes parts of the sound wave data without affecting how it sounds to our ears. It also includes features like checking if the file has enough space to hold the message and supports common file formats like PNG and WAV.  This project is not just a cool tech demo—it shows how simple tools can help protect privacy and secure communication. Whether for educational purposes or real-world use, it combines programming, cybersecurity, and multimedia handling in a creative and practical way. |

## Updated Project Approach and Architecture (2 pts)

|  |
| --- |
| Our project is a user-friendly web application that helps people hide and reveal secret messages inside image and audio files using steganography. To build this, we’ve used Python with the Flask web framework, which makes it easy to manage different parts of the app and handle user requests smoothly.  **How It Works:**  The app has three main parts:   1. **User Interface:** This is what users see when they open the app. They can upload an image or audio file, type a secret message, and download the new file with the hidden message inside. Everything runs in a web browser—no special software needed. 2. **Processing Logic:** This is where the actual magic happens. For images, we use LSB (Least Significant Bit) encoding to hide message bits in the colors of pixels without making the image look different. For audio files (WAV format), we hide the message inside sound samples in a way that doesn’t affect how it sounds to human ears. 3. **Server and Routing:** Flask handles all the communication between the user and the backend. When a user uploads a file or submits a message, the server processes it and sends back a response, either showing the hidden message or offering the modified file for download.   **Tools and Libraries:**   * **Flask** – to manage the web interface and server-side logic. * **Pillow (PIL)** – for editing and analyzing images. * **NumPy and Wave** – for working with audio files and encoding data. * **Secrets** – to generate secure file names and manage sessions.   This structure makes the project easy to maintain, upgrade, and even add new features like encryption or support for more file formats in the future. |

## Tasks Completed (7 pts)

|  |  |
| --- | --- |
| Task Completed | Team Member |
| * Image-based Steganography   + Message encoding and decoding within a .png format image * Audio-based Steganography   + Message encoding and decoding within a .wav audio file * Web interface   + User friendly interface with the help of flask | * Aayushi Sharma * Akansha Rawat * Gaurav Singh Rawat * Tanuja |

## Challenges/Roadblocks (7 pts) (Describe the challenges that you have faced or are facing so far and how you plan to solve them. Max 300 words).

|  |
| --- |
| Like most tech projects, building our steganography web app came with its fair share of challenges. Some of them were expected, while others surprised us along the way.   1. **Understanding Audio Files:** Working with audio data was way more complicated than we thought. WAV files store sound in binary, and figuring out how to hide a message without ruining the sound quality took a lot of trial and error. 2. **Limited Space in Images:** Not every image can hold a long message. We had to calculate how much text could be hidden without affecting the image’s quality. If the message was too big, it either caused errors or made the image look strange. 3. **Keeping Media Quality Intact:** Our goal was to hide messages without making it obvious. Doing that while making sure the image or audio looked and sounded the same was a big challenge. A small mistake could easily ruin the file. 4. **Making It Work on All Devices:** We wanted the app to work on different browsers and devices, but ran into issues like file upload limits and compatibility problems. Testing and fixing those bugs took extra time. 5. **Helpful Error Messages:** At first, when something went wrong, the app would just crash or stay blank. We had to improve this by adding clear error messages so users know what went wrong and how to fix it. 6. **Security Features:** We thought about adding password protection or encryption but didn’t have enough time to do it properly. It’s something we’d like to add in the future. 7. **Time Pressure:** Balancing this project with classes, assignments, and exams was tough. Some features had to be simplified or postponed because of time constraints. |

## Tasks Pending (7 pts)

|  |  |
| --- | --- |
| Task Pending | Team Member (to complete the task) |
| **1. Password**  We can implement basic encryption for message protection if a password is provided.  **2. Message Length Limitation**  We can consider more sophisticated techniques like: Error correction (Hamming codes).  **3. Audio Sample Handling**  We can add optional support for stereo audio  **4**. **UI Enhancement**  Drag-and-drop UI for uploads | * Gaurav Singh Rawat * Akansha Rawat * Aayushi Sharma * Tanuja |

## 

## Project Outcome/Deliverables (2 pts)

|  |
| --- |
| The final outcome of our project is a fully functional web application that allows users to hide (encode) and retrieve (decode) secret messages inside image and audio files. It demonstrates how steganography can be used in a practical and accessible way.  **Key Deliverables:**   1. **Web-Based Steganography Tool:** A user-friendly Flask web app where users can upload image or WAV audio files, enter secret messages, and download the encoded (stego) files. 2. **Image Steganography Module:** Uses Least Significant Bit (LSB) technique to hide text in image pixels without visibly changing the image quality. 3. **Audio Steganography Module:** Allows hiding messages in WAV audio files without affecting how they sound to the human ear. 4. **Decode Feature:** Users can upload a stego image or audio file and easily retrieve the hidden message. 5. **Message Capacity Checker:** A tool that checks whether the image has enough space to hide the given message, and shows how much of the image’s capacity will be used. 6. **Error Handling and Feedback:** The app includes clear error messages for unsupported file types, oversized messages, or decoding issues.   Overall, the project successfully delivers a working prototype of a steganography system that’s easy to use and demonstrates the real-world potential of secure communication through hidden data. |

# Progress Overview (2 pts)

|  |
| --- |
| So far, we’ve made solid progress on our steganography web application. The core features are fully developed and tested, including both image and audio message encoding and decoding. The Flask-based web interface is live and functional, allowing users to interact with the system through a simple upload-and-download process.  We’ve implemented:   * LSB-based image steganography for common image formats. * Audio steganography using WAV files with stable encoding/decoding. * A message capacity checker to guide users on input limits. * User-friendly error handling for common issues like wrong formats or oversized messages.   While the basic system is complete, we’re still considering optional improvements like encryption for added security and support for more file formats. The app works well across major browsers and performs reliably for the expected use cases.  Overall, the project is nearly complete, and we are confident about demonstrating it successfully. |

# Codebase Information (2 pts)

|  |
| --- |
| Our project’s code is written in Python, and we rely on Flask to run our web app. The code is neatly split into parts that handle image and audio steganography.   * **Main App (app.py):** This file powers the whole application. It manages web routes, handles file uploads/downloads, and connects the user interface with the underlying logic for hiding and revealing messages. * **Image Processing:** We use the Pillow (PIL) library to work with images. It lets us access and modify pixel data so that we can hide our messages using subtle changes that aren’t noticeable to the eye. * **Audio Processing:** For audio files, we work with the built-in wave module and NumPy. This helps us read WAV files and modify parts of the audio data without messing up the sound quality. * **Support Tools:** The **Secrets** module is used to generate secure filenames and manage sessions. We also have an extra script (audio.py) that focuses on audio steganography, which makes it easier to test and tweak that part of our project independently.   Overall, our codebase is organized and modular, which makes it straightforward to update or add features later—like encryption or support for more file formats. The project is designed to run on a local Flask server, meaning it’s simple to set up and doesn’t need any heavy dependencies. |

## Testing and Validation Status (2 pts)

|  |  |  |
| --- | --- | --- |
| Test Type | Status (Pass/Fail) | Notes |
| * Image-based Steganography * Audio-based Steganography * Working of UI | Pass  Pass  Pass | Secret message is encoded in image and decoded also from the image  Secret message is encoded in audio file and decoded also from the audio file  UI is working fine and integration between image steganography, audio steganography and UI is successful |

# Deliverables Progress (2 pts)

|  |
| --- |
| The outcomes and deliverables of this project include the following:   1. **Image-based Steganography System**: A fully functional module that allows users to securely hide and retrieve messages or files within image files. This system will use techniques like Least Significant Bit (LSB) manipulation, with options for encryption and error detection to enhance security. 2. **Audio-based Steganography System**: A module for encoding and decoding data within audio files (WAV format). This will also utilize LSB manipulation or alternative techniques like Phase Coding to embed data in audio while preserving the quality of the sound. 3. **Documentation**: Complete technical documentation outlining the system architecture, design decisions, and usage instructions. |