SARDAR VALLABHBHAI PATEL INSTITUTE OF TECHNOLOGY, VASAD.

INFORMATION TECHNOLOGY ENGINEERING

2024-25

CERTIFICATE

Date: ___/___/2024

This is to certify that the Summer Internship Work entitled "STENO: ULTIMATE STEGANOGRAPHY SOFTWARE" has been carried out by JAINIL PRAJAPATI (210410116126) under my guidance in fulfilment of the degree of Bachelor of Engineering in Information Technology (7th Semester) of Gujarat Technological University, Ahmedabad during the academic year 2024-25.

Internal Guide	**Head of the Department**
Prof. Priyanka Shah	Prof. Mala Patel
Asst. Prof	IT Dept.
SVIT,VASAD	SVIT,VASAD

Table of Contents

1. Introduction

1.1 Introduction

STENO is an advanced steganography application designed to provide users with a powerful yet user-friendly tool for hiding secret messages within various digital media formats. In an era where digital privacy and secure communication are of paramount importance, STENO offers a comprehensive solution for concealing sensitive information "in plain sight."

Steganography, derived from the Greek words "steganos" (covered) and "graphein" (writing), is the practice of hiding information within other non-secret data to avoid detection. Unlike cryptography, which makes data unreadable, steganography aims to hide the very existence of the secret information.

STENO supports multiple file formats including text, image, and audio, making it a versatile tool for various steganography needs. It employs advanced algorithms to embed data within these files while minimizing detectable changes, ensuring that the presence of hidden information remains inconspicuous.

1.2 Existing System

Current steganography solutions often suffer from several limitations:

1. Limited format support: Many tools specialize in only one type of media (e.g., only images or only audio).
2. Complex user interfaces: Existing tools often require technical expertise, making them inaccessible to average users.
3. Lack of integration: Users often need multiple tools to handle different types of steganography.
4. Weak security measures: Some tools lack robust encryption or password protection for hidden data.
5. Platform dependence: Many solutions are limited to specific operating systems.

1.3 Need for the New System

The development of STENO addresses several critical needs in the field of steganography:

1. Comprehensive solution: A single tool capable of handling multiple file formats.
2. User-friendly interface: Making steganography accessible to users with varying levels of technical expertise.
3. Enhanced security: Implementing strong encryption and password protection for hidden data.
4. Cross-platform compatibility: Ensuring the tool works across different operating systems.
5. Educational value: Providing a platform for understanding and experimenting with steganography techniques.

1.4 Objective of the New System

The primary objectives of STENO are:

1. To provide a user-friendly, all-in-one steganography solution supporting text, image, and audio formats.
2. To implement robust steganography techniques that minimize the detectability of hidden data.
3. To enhance the security of hidden data through password protection and potential encryption.
4. To ensure cross-platform compatibility for wider accessibility.
5. To create an extensible architecture that allows for easy addition of new steganography techniques and file format support in the future.

1.5 Problem Definition

STENO aims to solve the following problem: "How can we create a comprehensive, user-friendly, and secure steganography tool that supports multiple file formats and is accessible to users across different platforms?"

1.6 Software Process Model

The development of STENO follows an Agile software development process, specifically using the Scrum framework. This approach allows for iterative development, frequent stakeholder feedback, and the flexibility to adapt to changing requirements.

1.7 Core Components

The core components of STENO include:

1. Text Steganography Module: Utilizes the SNOW algorithm for hiding data in text files.
2. Image Steganography Module: Implements LSB (Least Significant Bit) technique for image files.
3. Audio Steganography Module: Uses LSB technique adapted for audio samples in WAV files.
4. Graphical User Interface: Built using Python's Tkinter library for cross-platform compatibility.
5. Database Management: Handles user accounts and file metadata using SQLite.

1.8 Project Profile

- Project Name: STENO
- Version: 1.0
- Development Language: Python
- Primary Libraries: Tkinter, OpenCV, Wave
- Database: SQLite

- Version Control: Git
- Repository: https://github.com/enough-jainil/Steno-project.git

1.9 Advantages and Limitations of the Proposed System

Advantages:

1. Multi-format support in a single application
2. User-friendly interface accessible to non-technical users
3. Cross-platform compatibility
4. Enhanced security through password protection
5. Extensible architecture for future improvements

Limitations:

1. Currently lacks support for video steganography
2. Dependent on external tools like SNOW for certain operations
3. Large file processing may be time-consuming on less powerful systems
4. Advanced users may require more customization options

2. Requirement Determination & Analysis

2.1 Requirement Determination

2.1.1 Functional Requirements

1. Text Steganography
   - The system shall allow users to hide messages in text files using whitespace steganography (SNOW algorithm).
   - Users shall be able to encode messages or entire files within text documents.
   - The system shall provide a function to check available space for hiding data in text files.
   - Users shall be able to extract hidden messages from steganographic text files.
2. Image Steganography
   - The system shall support hiding data in image files (PNG, JPG, JPEG, BMP, GIF) using the LSB technique.
   - Users shall be able to embed both text messages and files within images.
   - The system shall provide a function to estimate the maximum data size that can be hidden in a given image.
   - Users shall be able to extract hidden data from steganographic image files.
3. Audio Steganography
   - The system shall support hiding data in WAV audio files using LSB steganography.
   - Users shall be able to embed both text messages and files within audio files.

- The system shall provide a function to estimate the maximum data size that can be hidden in a given audio file.
- Users shall be able to extract hidden data from steganographic audio files.

4. User Interface
   - The system shall provide a graphical user interface for all steganography operations.
   - The interface shall include separate sections for text, image, and audio steganography.
   - The system shall provide file browsing capabilities for selecting input and output files.
   - The interface shall display progress indicators for lengthy operations.
5. Security
   - The system shall implement password protection for all steganography operations.
   - The system shall provide a secure method for storing user credentials and file metadata.
   - The system shall include a password recovery mechanism for registered users.
6. File Management
   - The system shall allow users to specify input and output file locations.
   - The system shall prevent accidental overwriting of existing files without user confirmation.

2.1.2 Non-Functional Requirements

1. Usability
   - The user interface shall be intuitive and require minimal training for basic operations.
   - The system shall provide tooltips and help documentation for all features.
   - The system shall support keyboard shortcuts for common operations.
2. Performance
   - The system shall be capable of processing files up to 100MB in size without significant delay.
   - Steganography operations shall complete within 60 seconds for files up to 10MB on standard hardware.
3. Reliability
   - The system shall handle errors gracefully, providing clear error messages to users.
   - The system shall not corrupt or modify input files during normal operation.
4. Scalability
   - The system architecture shall allow for easy addition of new steganography techniques in future versions.
5. Compatibility

- The system shall run on Windows, macOS, and Linux operating systems.
- The system shall be compatible with Python 3.6 and above.

6. Security
   - The system shall use secure methods for storing passwords and sensitive data.
   - The steganography techniques used shall resist common steganalysis attacks.
7. Maintainability
   - The code shall be well-documented and follow PEP 8 style guidelines for Python code.
   - The system shall use a modular architecture to facilitate future updates and maintenance.

2.1.3 Hardware Requirements

Minimum system requirements for running STENO:

- Processor: 1.5 GHz dual-core processor or equivalent
- RAM: 4 GB
- Storage: 500 MB of free disk space
- Display: 1280x720 resolution or higher

Recommended system requirements:

- Processor: 2.5 GHz quad-core processor or equivalent
- RAM: 8 GB
- Storage: 1 GB of free disk space
- Display: 1920x1080 resolution or higher

2.1.4 Software Requirements

- Operating System: Windows 7 or later, macOS 10.12 or later, Ubuntu 18.04 or equivalent Linux distribution
- Python: Version 3.6 or later
- Additional software:
  - Pip (Python package installer)
  - Tkinter (usually comes with Python installation)
  - OpenCV (for image processing)
  - SNOW (for text steganography)

2.2 Targeted User

STENO is designed for a diverse range of users, including:

1. Privacy-conscious individuals seeking to secure their personal communications.
2. Professionals who need to transmit sensitive information securely.

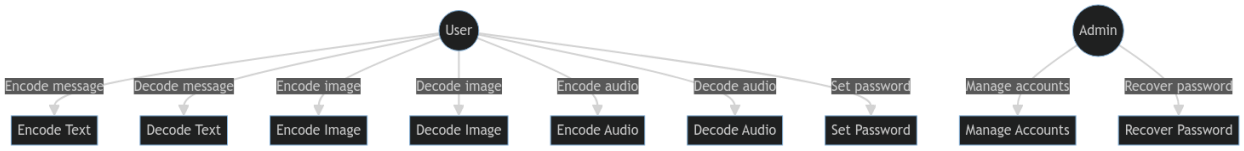
3. Students and educators interested in learning about or teaching steganography concepts.
4. Digital rights activists working to protect freedom of speech in restrictive environments.
5. Security researchers studying steganography techniques and their applications.
6. Small to medium-sized organizations requiring a simple, in-house steganography solution.

The user-friendly interface makes STENO accessible to both technical and non-technical users, while its advanced features cater to more experienced users and professionals in the field of information security.

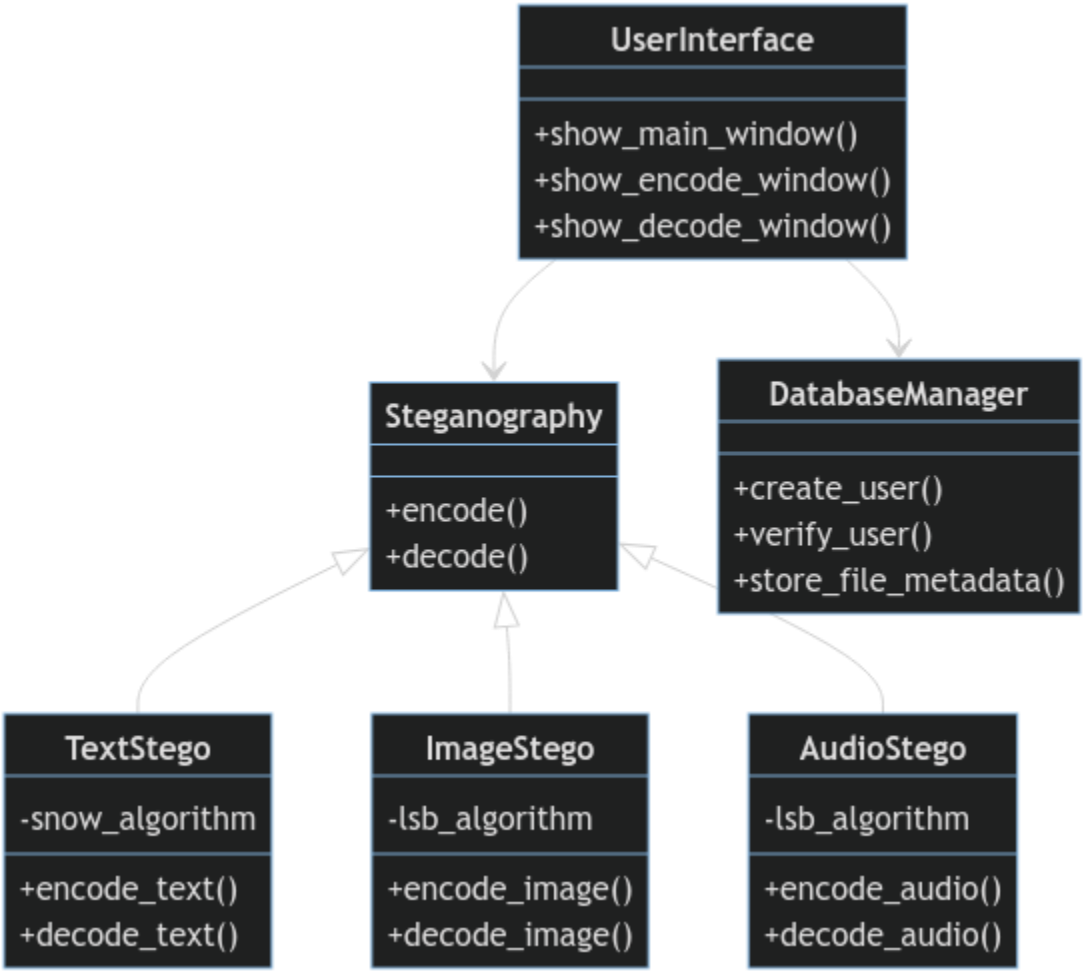
Certainly! I'll create the System Design pages for the STENO project report, including the various UML diagrams and other design elements you've requested.

3. System Design

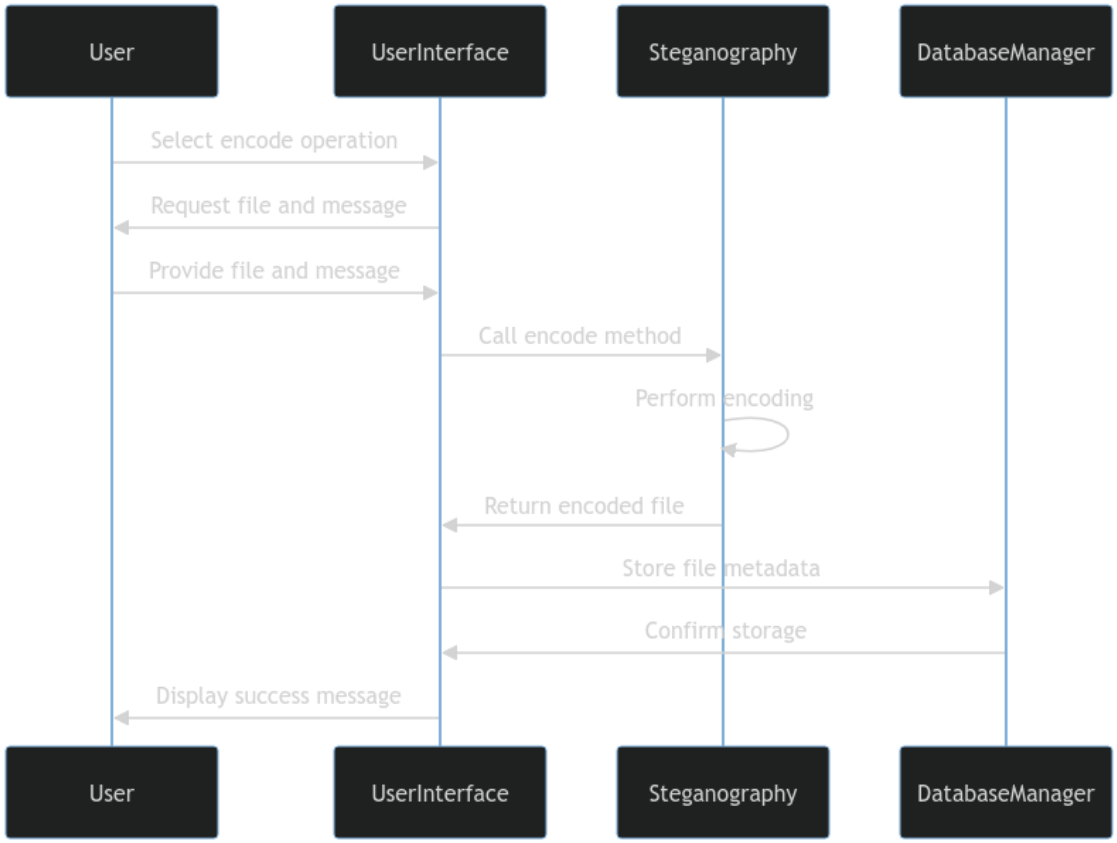
3.1 Use Case Diagram



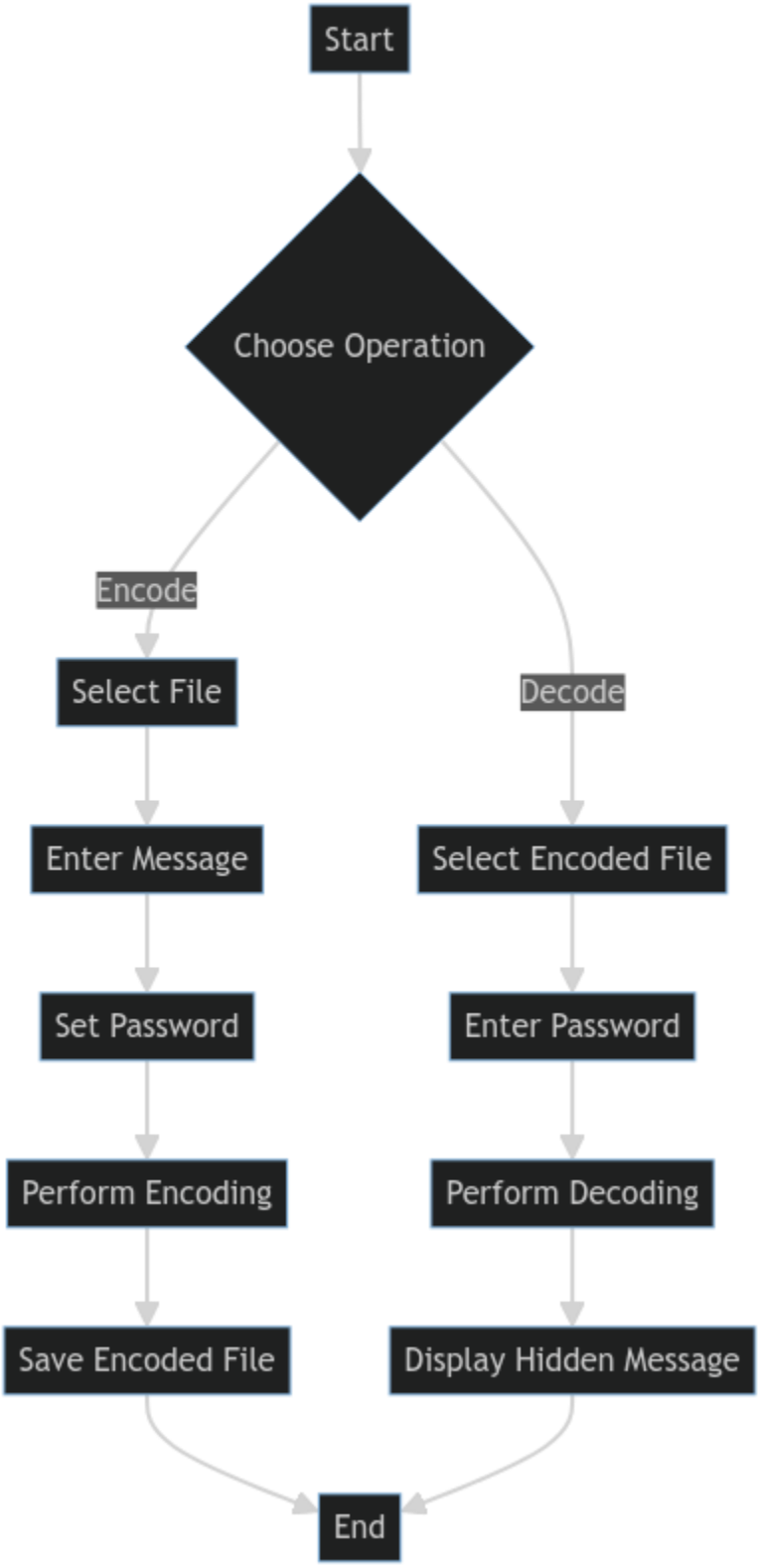
3.2 Class Diagram



3.3 Sequence Diagram



3.4 Activity Diagram



3.5 Data Dictionary

Term	Description	Data Type	Constraints
Username	Unique identifier for user account	String	3-20 characters, alphanumeric
Password	User account password	String	8-30 characters, must include uppercase, lowercase, number
File Path	Location of input/output files	String	Valid file path
Message	Text to be hidden	String	Up to 1000 characters
File Type	Type of file (text, image, audio)	Enum	Only 'text', 'image', 'audio'
Encoding Method	Method used for steganography	Enum	'SNOW', 'LSB'
Timestamp	Time of encoding/decoding operation	DateTime	Valid date and time

3.6 Algorithm Flow

1. Text Steganography (SNOW Algorithm):
a. Encode:
   - Convert message to binary
   - Insert spaces and tabs at end of each line based on binary data
   - Ensure original text content remains unchanged

   b. Decode:
   - Read spaces and tabs at end of each line
   - Convert pattern to binary
   - Convert binary to text message

2. Image Steganography (LSB Algorithm):
a. Encode:
   - Convert message to binary
   - For each bit of the message:
     - Select a pixel
     - Modify the least significant bit of a color channel to match the message bit

     b. Decode:
   - For each modified pixel:
     - Read the least significant bit
   - Combine bits to form binary data
   - Convert binary to text message

3. Audio Steganography (LSB Algorithm):

a. Encode:
   - Convert message to binary
   - For each bit of the message:
     - Select an audio sample
     - Modify the least significant bit of the sample to match the message bit

   b. Decode:
   - For each modified audio sample:
     - Read the least significant bit
   - Combine bits to form binary data
   - Convert binary to text message

These algorithms are implemented within their respective classes (TextStego, ImageStego, AudioStego) and are called by the main Steganography class based on the file type selected by the user.

This System Design section provides a comprehensive overview of the STENO project's architecture and structure. It includes:

1. Use Case Diagram: Illustrating the main functions of the system and user interactions.
2. Class Diagram: Showing the main classes in the system and their relationships.
3. Sequence Diagram: Demonstrating the flow of operations for an encoding process.
4. Activity Diagram: Depicting the general flow of activities in the application.
5. Data Dictionary: Defining key data elements used in the system.
6. Algorithm Flow: Describing the step-by-step process for each type of steganography.

These diagrams and descriptions provide a clear picture of how the STENO system is structured and how its components interact. This information is crucial for developers to understand the system architecture and for maintainers to navigate the codebase.

4. Development

4.1 Coding Standards

Adhering to consistent coding standards is crucial for maintaining readability, efficiency, and ease of collaboration in the STENO project. The following coding standards have been adopted:

4.1.1 Python Style Guide (PEP 8)

The project follows the PEP 8 style guide for Python code. Key points include:

1. Indentation: Use 4 spaces per indentation level.
2. Maximum Line Length: Limit all lines to a maximum of 79 characters.
3. Imports:

- Place imports at the top of the file.
- Group imports in the following order: standard library, third-party, local application.

4. Naming Conventions:
   - Functions, variables, and attributes: lowercase_with_underscores
   - Classes: CapitalizedWords
   - Constants: ALL_CAPS
5. Comments: Use inline comments sparingly. Write docstrings for all public modules, functions, classes, and methods.

4.1.2 Project-Specific Standards

1. File Organization:
   - Place each class in its own file.
   - Use `__init__.py` files to organize modules.
2. Error Handling:
   - Use try-except blocks for error handling.
   - Raise custom exceptions where appropriate.
3. Logging:
   - Use Python's built-in logging module for debugging and error tracking.
   - Avoid print statements for debugging in production code.
4. Testing:
   - Write unit tests for all functions and methods.
   - Aim for at least 80% code coverage.
5. Documentation:
   - Use docstrings to document all modules, classes, and functions.
   - Follow the Google Python Style Guide for docstring formatting.

4.1.3 Version Control Practices

1. Commit Messages:
   - Use present tense ("Add feature" not "Added feature").
   - First line should be a short description (50 chars or less).
   - Followed by a blank line and a more detailed explanation if necessary.
2. Branching:
   - Use feature branches for all new development.
   - Use pull requests for code review before merging into the main branch.
3. Code Review:
   - All code must be reviewed by at least one other developer before merging.
   - Use GitHub's pull request features for code reviews.

4.2 User Interface

The user interface for STENO is designed to be intuitive, user-friendly, and accessible across different platforms. It is implemented using Python's Tkinter library.

4.2.1 Design Principles

1. Simplicity: Keep the interface clean and uncluttered.
2. Consistency: Maintain a consistent look and feel across all windows and dialogs.
3. Feedback: Provide clear feedback for all user actions.
4. Accessibility: Ensure the interface is usable by people with different abilities.

4.2.2 Main Window

The main window serves as the central hub for all operations:

4.2.3 Steganography Operation Windows

Each steganography type (text, image, audio) has its own operation window:

```
() => (
  <div className="bg-gray-100 p-4 rounded-lg shadow-md w-96">
    <h2 className="text-xl font-bold mb-4">Text Steganography</h2>
```

```
    <div className="space-y-4">
      <div>
        <label className="block text-sm font-medium text-gray-700">Input
File</label>
        <div className="mt-1 flex rounded-md shadow-sm">
          <input type="text" className="flex-1 rounded-l-md border-gray-300
p-2" placeholder="Select file..."/>
          <button className="bg-blue-500 text-white rounded-r-md p-
2">Browse</button>
        </div>
      </div>
      <div>
        <label className="block text-sm font-medium text-gray-
700">Message</label>
        <textarea className="mt-1 block w-full rounded-md border-gray-300 p-
2" rows="3"></textarea>
      </div>
      <div>
        <label className="block text-sm font-medium text-gray-
700">Password</label>
        <input type="password" className="mt-1 block w-full rounded-md
border-gray-300 p-2"/>
      </div>
      <div className="flex justify-between">
        <button className="bg-green-500 text-white p-2
rounded">Encode</button>
        <button className="bg-yellow-500 text-white p-2
rounded">Decode</button>
      </div>
    </div>
  </div>
)
```

4.2.4 User Feedback

1. Progress Bars: Used for long-running operations like encoding large files.
2. Message Boxes: Display success messages, errors, and warnings.
3. Tooltips: Provide additional information for UI elements.

4.2.5 Accessibility Features

1. Keyboard Navigation: All functions can be accessed via keyboard shortcuts.
2. High Contrast Mode: An option to switch to a high-contrast color scheme.
3. Scalable Text: Ability to adjust text size for better readability.

4.2.6 Responsive Design

The UI is designed to be responsive, adapting to different screen sizes and resolutions:

1. Flexible layouts using grid and flex designs.
2. Adjustable font sizes based on screen resolution.

3. Collapsible menus for smaller screens.

By following these coding standards and UI design principles, the STENO project aims to create a robust, maintainable, and user-friendly application. The consistent coding practices ensure that the codebase remains clean and easy to understand, while the thoughtful UI design provides an intuitive and accessible user experience.

This Development section provides a comprehensive overview of the coding standards and user interface design for the STENO project. It includes:

1. Coding Standards: Detailing the Python style guide (PEP 8), project-specific standards, and version control practices.
2. User Interface: Covering design principles, main window layout, operation windows, user feedback mechanisms, accessibility features, and responsive design considerations.

The section also includes mockups of the main window and a typical operation window to illustrate the UI design concepts.

This information serves as a crucial reference for developers to maintain consistency in code style and UI design throughout the development process.

5. Future Enhancements

While STENO currently provides a robust set of steganography features, there are several areas where the application could be enhanced to provide more functionality, improved security, and better user experience. This section outlines potential future enhancements for the STENO project.

5.1 Video Steganography Support

Implementing video steganography would significantly expand STENO's capabilities:

- Develop algorithms for hiding data within video frames or audio streams of video files.
- Support popular video formats such as MP4, AVI, and MOV.
- Implement efficient processing methods to handle large video files.
- Ensure minimal quality degradation in the resulting steganographic videos.

5.2 Enhanced Encryption

Improving the encryption methods used in STENO would significantly enhance the security of hidden data:

- Implement strong, standardized encryption algorithms (e.g., AES-256) for all steganography types.
- Introduce a key derivation function (e.g., PBKDF2) to strengthen password-based encryption.

- Implement end-to-end encryption for any future network-based features.

5.3 Advanced Steganography Techniques

Enhancing the steganography algorithms used in STENO could improve data hiding capacity and resistance to steganalysis:

- For images: Implement DCT (Discrete Cosine Transform) based steganography or Wavelet Transform techniques.
- For audio: Explore phase coding or spread spectrum techniques.
- For text: Implement linguistic steganography methods that modify text structure or word choice.

5.4 Web and Mobile Versions

Expanding STENO's availability to web and mobile platforms could significantly increase its accessibility and user base:

- Develop a web application with client-side processing for security.
- Create mobile apps for iOS and Android platforms.
- Implement cloud storage integration for secure file handling across devices.

5.5 Machine Learning Integration

Incorporating machine learning could enhance various aspects of STENO:

- Develop ML models to optimize data embedding for different types of carrier files.
- Implement intelligent steganalysis to assess the security of steganographic outputs.
- Use ML for anomaly detection to identify potential attacks or unauthorized access attempts.

5.6 Blockchain Integration

Integrating blockchain technology could provide additional security and verification features:

- Implement a blockchain-based system for verifying the integrity of steganographic files.
- Develop smart contracts for managing access to hidden data in enterprise settings.
- Explore the use of decentralized storage solutions for enhanced data privacy.

5.7 Plugin System

Developing a plugin architecture could allow for easy extension of STENO's capabilities:

- Create an API for developing custom steganography algorithms.
- Allow integration with other security tools (e.g., encryption software, digital signature tools).

- Enable custom output formats or integrations with other applications.

5.8 Collaboration Features

Adding collaboration features could make STENO more useful in team environments:

- Implement secure file sharing mechanisms for steganographic files.
- Develop a system for managing team access to hidden data.
- Create an audit trail system for tracking steganography operations in a team setting.

5.9 Advanced User Interface

Enhancing the user interface could improve user experience and accessibility:

- Implement a dark mode option for the UI.
- Develop a drag-and-drop interface for file selection and processing.
- Create interactive tutorials to guide new users through the steganography process.
- Implement voice commands for hands-free operation.

5.10 Performance Optimization

Improving STENO's performance, especially for large files, could enhance user experience:

- Implement multi-threading for CPU-intensive operations.
- Optimize memory usage for large file processing.
- Develop a distributed processing system for handling very large files or batch operations.

5.11 Comprehensive Analytics

Adding analytics features could provide valuable insights for users:

- Implement statistics tracking for steganography operations (e.g., capacity utilization, frequency of use).
- Develop reports on potential vulnerabilities or strength of hidden data.
- Create visualization tools for understanding steganography patterns and usage.

5.12 Internationalization and Localization

Expanding language support could make STENO accessible to a global audience:

- Implement a robust internationalization framework.
- Provide translations for major world languages.
- Allow community contributions for translations and locale-specific features.

These future enhancements would significantly expand STENO's capabilities, improve its security and performance, and cater to a wider range of use cases. Prioritization of these enhancements should be based on user feedback, security considerations, and alignment

with the project's overall goals. Implementation of these features would be planned and executed in future development cycles, ensuring that STENO remains at the forefront of steganography technology.

6. Conclusion

The STENO project represents a significant advancement in the field of steganography, providing a comprehensive, user-friendly, and secure solution for hiding sensitive information within various digital media formats. As we conclude this report, it's important to reflect on the project's achievements, its impact, and the road ahead.

6.1 Project Achievements

STENO has successfully met its primary objectives:

1. **Multi-format Support**: The application effectively implements steganography techniques for text, image, and audio files, providing users with flexibility in choosing their preferred medium for data hiding.
2. **User-friendly Interface**: Through careful design and implementation, STENO offers an intuitive graphical user interface that makes complex steganography operations accessible to users with varying levels of technical expertise.
3. **Enhanced Security**: The implementation of password protection for hidden data and an admin account system for password recovery has significantly enhanced the overall security of the steganographic process.
4. **Cross-platform Compatibility**: By leveraging Python and platform-independent libraries, STENO ensures functionality across different operating systems, broadening its accessibility.
5. **Extensible Architecture**: The modular design of STENO lays a strong foundation for future enhancements and additions of new steganography techniques.

6.2 Impact and Significance

STENO contributes to the field of information security and privacy in several ways:

1. **Democratizing Steganography**: By providing a user-friendly tool, STENO makes advanced steganography techniques more accessible to individuals and organizations concerned with data privacy.
2. **Educational Value**: The application serves as an excellent platform for understanding and experimenting with steganography concepts, benefiting students, educators, and researchers in the field.
3. **Practical Application**: STENO offers a practical solution for secure communication, potentially finding applications in journalism, human rights activism, and corporate data protection.
4. **Advancing the Field**: The project's open-source nature and extensible architecture contribute to the broader development and improvement of steganography techniques.

6.3 Lessons Learned

The development of STENO has provided valuable insights:

1. **Balancing Security and Usability**: The project highlighted the challenge of implementing robust security measures while maintaining user-friendliness, emphasizing the importance of finding the right balance.
2. **Importance of Modular Design**: The modular approach adopted in STENO's architecture proved crucial for managing complexity and facilitating future enhancements.
3. **Cross-platform Challenges**: Developing for multiple platforms reinforced the importance of choosing cross-platform libraries and implementing platform-specific checks where necessary.
4. **Performance Considerations**: Handling large files, especially in image and audio steganography, underscored the need for efficient algorithms and consideration of performance optimization techniques.

6.4 Future Outlook

While STENO has achieved its initial goals, there is significant potential for future growth and improvement:

1. **Expanding Capabilities**: The planned addition of video steganography and implementation of more advanced algorithms will further enhance STENO's capabilities.
2. **Enhanced Security**: Future versions aim to incorporate stronger encryption methods and more robust security features to protect against advanced steganalysis techniques.
3. **Broader Accessibility**: Development of web-based and mobile versions could significantly expand STENO's user base and use cases.
4. **Integration of Emerging Technologies**: The potential integration of machine learning and blockchain technologies opens up exciting possibilities for advancing the field of steganography.
5. **Community Engagement**: As an open-source project, STENO has the potential to benefit from community contributions, potentially accelerating its development and feature expansion.

6.5 Final Thoughts

The STENO project demonstrates the potential of combining advanced steganography techniques with user-friendly software design. It addresses the growing need for tools that can secure sensitive information in an era of increasing digital surveillance and data breaches.

While steganography should not be seen as a replacement for strong encryption, it provides an additional layer of security by obscuring the very existence of hidden data.

STENO offers this capability in an accessible format, potentially broadening the use of steganography beyond specialized technical circles.

As digital privacy concerns continue to grow, tools like STENO will play an increasingly important role in protecting sensitive information. The project's future development, guided by user feedback and advancements in steganography techniques, will aim to stay at the forefront of this crucial aspect of information security.

In conclusion, the STENO project not only achieves its goals of creating a comprehensive steganography tool but also lays a foundation for future innovations in the field. It stands as a testament to the power of open-source development in creating tools that enhance digital privacy and security for all.

7. References

1. Provos, N., & Honeyman, P. (2003). Hide and seek: An introduction to steganography. IEEE security & privacy, 1(3), 32-44.
2. Johnson, N. F., & Jajodia, S. (1998). Exploring steganography: Seeing the unseen. Computer, 31(2), 26-34.
3. Kessler, G. C. (2004). An overview of steganography for the computer forensics examiner. Forensic science communications, 6(3), 1-27.
4. Python Software Foundation. (2021). PEP 8 – Style Guide for Python Code. https://www.python.org/dev/peps/pep-0008/
5. Fridrich, J. (2009). Steganography in digital media: principles, algorithms, and applications. Cambridge University Press.
6. Cox, I. J., Miller, M. L., Bloom, J. A., Fridrich, J., & Kalker, T. (2007). Digital watermarking and steganography. Morgan Kaufmann.
7. Wayner, P. (2009). Disappearing cryptography: information hiding: steganography & watermarking. Morgan Kaufmann.
8. Tkinter. (2021). Python interface to Tcl/Tk. https://docs.python.org/3/library/tkinter.html
9. OpenCV team. (2021). OpenCV (Open Source Computer Vision Library). https://opencv.org/
10. SQLite. (2021). SQLite Home Page. https://www.sqlite.org/index.html
11. SNOW. (n.d.). SNOW Home Page. http://www.darkside.com.au/snow/
12. Westfeld, A., & Pfitzmann, A. (1999). Attacks on steganographic systems. In Information Hiding (pp. 61-76). Springer, Berlin, Heidelberg.
13. Simmons, G. J. (1984). The prisoners' problem and the subliminal channel. In Advances in Cryptology (pp. 51-67). Springer, Boston, MA.
14. Katzenbeisser, S., & Petitcolas, F. A. (2000). Information hiding techniques for steganography and digital watermarking. Artech house.
15. Zöllner, J., Federrath, H., Klimant, H., Pfitzmann, A., Piotraschke, R., Westfeld, A., ... & Wolf, G. (1998). Modeling the security of steganographic systems. In Information Hiding (pp. 344-354). Springer, Berlin, Heidelberg.

16. Cheddad, A., Condell, J., Curran, K., & Mc Kevitt, P. (2010). Digital image steganography: Survey and analysis of current methods. Signal processing, 90(3), 727-752.
17. Kipper, G. (2003). Investigator's guide to steganography. Auerbach Publications.
18. Anderson, R. J., & Petitcolas, F. A. (1998). On the limits of steganography. IEEE Journal on selected areas in communications, 16(4), 474-481.
19. Murdoch, S. J., & Lewis, S. (2005). Embedding covert channels into TCP/IP. In Information hiding (pp. 247-261). Springer, Berlin, Heidelberg.
20. Cachin, C. (2004). An information-theoretic model for steganography. Information and Computation, 192(1), 41-56.
21. Agaian, S. S., Caglayan, O. U., & Akopian, D. (2006). A new binary image steganalysis using bit-plane complexity. In 2006 IEEE International Conference on Image Processing (pp. 2013-2016). IEEE.
22. Li, B., He, J., Huang, J., & Shi, Y. Q. (2011). A survey on image steganography and steganalysis. Journal of Information Hiding and Multimedia Signal Processing, 2(2), 142-172.
23. Sallee, P. (2003). Model-based steganography. In International Workshop on Digital Watermarking (pp. 154-167). Springer, Berlin, Heidelberg.
24. Provos, N. (2001). Defending against statistical steganalysis. In 10th USENIX Security Symposium (Vol. 10, pp. 323-336).
25. Fridrich, J., Goljan, M., & Du, R. (2001). Detecting LSB steganography in color, and gray-scale images. IEEE multimedia, 8(4), 22-28.