Here’s a **well-structured report** you can use for your **Secure File Sharing System** task.

**Secure File Sharing System – Project Report**

**Prepared by:** Ammad Aziz  
**Internship:** Internee.pk – Cybersecurity Internship  
**Task:** Secure File Sharing System  
**Date:** September 2025

**1. Objective**

The main goal of this project was to **develop a secure file-sharing portal** that ensures **confidentiality, integrity, and security** of files being shared between Internee.pk and external parties without relying on external cloud storage.

**2. Key Features**

* **End-to-End Encryption**
  + Uses **AES-256-GCM** for encrypting files.
  + Uses **RSA-2048** for secure key exchange.
* **Local Storage Support**
  + All files are securely encrypted and stored locally.
* **Key Management**
  + Generates **RSA key pairs** for encryption and decryption.
  + Private keys are kept safe with optional password protection.
* **Signed Token System**
  + Tokens use **HMAC-SHA256** for integrity.
  + Tokens have a configurable **expiration time**, simulating signed URLs.
* **Simple Desktop Interface**
  + Built with **Python Tkinter** for a user-friendly GUI.
  + Easy navigation with tabs for Keys, Encrypt & Share, and Decrypt.

**3. Tools & Technologies**

| **Component** | **Technology Used** |
| --- | --- |
| Programming | Python 3.10+ |
| GUI Framework | Tkinter |
| Cryptography | cryptography library |
| File Management | os, zipfile, pathlib |
| Signing & Hashing | hmac, hashlib |

**4. How It Works**

**Step 1 – Key Generation**

* Generate RSA public/private key pair.
* Share the **public key** with recipients.
* Keep the **private key** secure.

**Step 2 – Encrypt & Share**

* Select a file and the recipient’s public key.
* The file is encrypted locally into a secure .sfs bundle.
* Optionally generate a signed token with an expiry time.

**Step 3 – Verify & Decrypt**

* Recipient verifies the token (optional).
* Select the .sfs bundle and private key.
* Decrypts the original file securely.

**5. Security Mechanisms**

| **Layer** | **Algorithm / Method** |
| --- | --- |
| File Encryption | AES-256-GCM |
| Key Wrapping | RSA-OAEP with SHA-256 |
| Token Signing | HMAC-SHA256 |
| Integrity Check | GCM Tag Verification |

**6. Advantages**

* No external dependencies on cloud servers.
* Full control over keys and encrypted files.
* Cross-platform: works on Windows, Linux, and macOS.
* End-to-end encryption ensures **data confidentiality**.

**7. Future Enhancements**

* Integrate **cloud storage** (AWS S3, GCP Storage, or Azure Blob) with signed URLs.
* Add **multi-user support** for organizations.
* Implement **automatic key rotation** and key recovery.
* Add support for **mobile interface** using a web or Flutter-based front-end.

**8. Screenshots *(Optional Section)***

*(Include screenshots of the GUI showing key generation, encryption, and decryption steps if needed.)*

**9. Conclusion**

This project successfully demonstrates a **local secure file-sharing system** that implements **industry-standard cryptographic methods**. It provides a foundation for secure exchanges without depending on external services, while also being extendable for cloud-based integrations in the future.

Would you like me to **export this report as a PDF** for submission?