# **P2P File Sharing System**

**Submitted By:**  
**Aun Ali** – **22k-4029**  
**Uzair** **Ahmed– 21k-4070**

**Department of Artificial Intelligence**  
**National University of Computer & Emerging Sciences**

## 1. Motivation

The aim of this project is to develop a decentralized P2P File Sharing System that enables users to share files directly without the need for a central server. In the current digital landscape, secure and efficient file sharing is crucial, yet most existing solutions depend on centralized infrastructure, which introduces potential bottlenecks and single points of failure. This project leverages socket programming and modern web technologies to build a resilient, efficient, and secure file-sharing application, emphasizing direct user-to-user communication.

## 2. Overview

### 2.1 Significance of the Project

This project highlights three key areas:

* Demonstrates practical use of socket programming for peer communication.
* Integrates low-level networking with Django, showcasing multi-technology implementation.
* Addresses security by implementing encrypted file transfers.

Academically, it offers hands-on exposure to core concepts in networking, web development, and cybersecurity.

### 2.2 Description of the Project

The system enables users to share files over a network via a web interface built with Django. Users specify the receiver's IP and port and select files to send. File transfers occur directly through socket connections, with Fernet encryption ensuring secure transmission. A dedicated receiver server handles incoming files. The frontend provides a simple interface for file selection, status updates, and previewing text files.

The system focuses on direct file sharing between known peers rather than full peer discovery, keeping implementation manageable while demonstrating P2P communication principles.

### 2.3 Background of the Project

Inspired by early decentralized systems like Napster and BitTorrent, this project combines their P2P principles with modern technologies:

* **Python socket programming** for peer communication [1]
* **Django framework** for backend and interface management [2]
* **Fernet encryption** for securing transfers [3]
* **HTML/CSS/JavaScript** for frontend design [4]

### 2.4 Project Category

This is a **product-based** project that implements a fully functional file-sharing application designed for practical use.

## 3. Features / Scope / Modules

* **Direct File Sharing**  
  Users connect directly via IP and port, with socket-based file transfers and no central server.
* **Secure File Transfer with Encryption**  
  Files are encrypted using Fernet before sending and decrypted on receipt, ensuring confidentiality.
* **Web-Based User Interface**  
  Developed with Django, HTML, CSS, and JS, the interface provides a user-friendly experience with input forms and transfer feedback.
* **File Content Preview**  
  Text files can be viewed directly in the browser before downloading.
* **Efficient File Transfer**  
  Files are read and written in chunks to support large file transfers without memory issues.
* **Separate Receiver Server**  
  Operates alongside the Django app to manage incoming transfers, keeping the interface responsive.

## 4. Project Planning

| **Week** | **Tasks** | **Responsibility** |
| --- | --- | --- |
| Week 9 | Research P2P concepts and socket programming | Aun, Uzair |
|  | Set up environment and tools | Uzair |
|  | Design system architecture | Aun, Uzair |
| Week 10 | Implement socket communication | Aun |
|  | Set up Django structure | Uzair |
|  | Create receiver server | Aun |
| Week 11 | Implement file-sharing functionality | Aun |
|  | Develop Django views and templates | Uzair |
|  | Begin frontend structure | Uzair |
| Week 12 | Implement encryption | Aun |
|  | Optimize chunked transfer | Aun |
|  | Finalize frontend | Uzair |
| Week 13 | Debug and performance test | Aun, Uzair |
|  | UI improvements | Uzair |
|  | Documentation and presentation | Aun, Uzair |

## 5. Project Feasibility

### Technical Feasibility

* Python sockets for robust networking
* Django for rapid backend development
* Cryptography library for encryption
* No specialized hardware needed
* Risks like network issues were handled with error management

### Economic Feasibility

* All tools are open-source
* Runs on standard hardware
* No cost beyond time investment
* Low operational costs

### Schedule Feasibility

* 5-week schedule was sufficient
* Modular design enabled parallel development
* Scope limited to direct sharing made the timeline realistic
* Included buffer for refinement

## 6. Hardware and Software Requirements

**Hardware:**

* Network-enabled computer
* Minimum 4GB RAM
* 1GB free disk space
* Network interface card

**Software:**

* Python 3.8+
* Django 5.2
* Cryptography library
* JavaScript-enabled web browser
* OS: Windows, macOS, or Linux

## 7. Diagrammatic Representation of the Overall System

## **+------------+ HTTP +---------------+ TCP/Socket +------------+**

## **| Browser | <---------------> | Django Server | <----------------------> | Receiver |**

## **| (Frontend) | | (Backend) | | Peer |**

## **+------------+ +---------------+ +------------+**

## **| | |**

## **|--- Enter IP & Port ------------>| |**

## **|--- Upload File ---------------->| |**

## **| |--- Encrypt File ------------------------->|**

## **| |--- Send File ---------------------------->|**

## **|<-- Transfer Status -------------| |**

## **|--- Request File Preview ------->| |**

## **|<-- File Content ----------------| |**

## 8. References

[1] Python Software Foundation, Python 3.11 Socket Programming Docs  
[2] Django Software Foundation, Django Documentation. [https://docs.djangoproject.com](https://docs.djangoproject.com/)  
[3] Cryptography.io Documentation. <https://cryptography.io/en/latest/>  
[4] MDN Web Docs, JavaScript Fetch API. [https://developer.mozilla.org](https://developer.mozilla.org/)

WORKING DEMO:













