CN Project Documentation

Secure Cloud-Based File Storage System

Computer Networks Project

Project Details

- **Title**: Secure Cloud-Based File Storage System (Client-Server Network)
- **Project Members**:
  - Adyansh Aggarwal (PES1UG23AM028)
  - Akash Madisetty (PES1UG23AM035)

Problem Statement

Development of an encrypted file transfer and storage system that ensures secure communication between clients and servers while maintaining data confidentiality.

Project Aim

To build a client-server system where multiple clients can **store encrypted files** on **Google Drive** using **AES-level encryption**, ensuring secure transmission and storage of sensitive data. The system implements **TCP checksum** for data integrity verification and **SSL/TLS encryption** for secure communication channels, providing multiple layers of security for file transfers.

System Architecture

1. Server Components

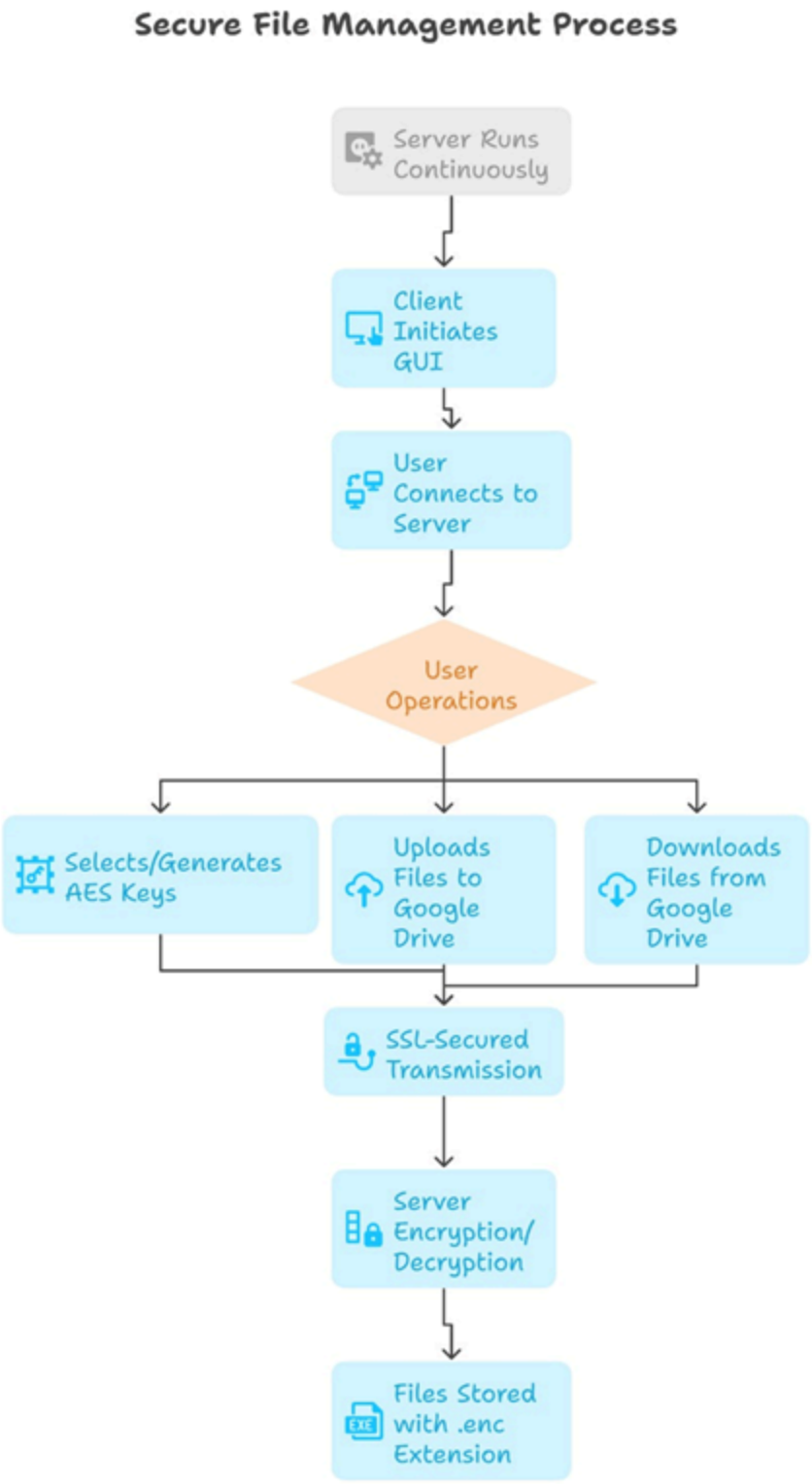
- **server.py**: Main server implementation
- **run_server.py**: Server execution script
- **Server.crt**: SSL certificate
- **Server.key**: SSL private key
- **generate_ssl.py** : Generates the SSL key and certificate for the server
- **encryption.py**: AES encryption/decryption implementation
- **gdrive.py**: Google Drive API integration
- **test_encryption.py**: Encryption testing module
- **Downloads directory**: Temporary storage for incoming files

- **Uploads directory**: Temporary storage for outgoing files

2. Client Components

- **gui.py**: User interface implementation
- **client.py**: Client-side logic
- **Server.crt**: Server's SSL certificate
- **run_client.py**: Client execution script

Process Flow



Security Features

1. **SSL/TLS Encryption**:
   - Secure Socket Layer implementation
   - Server authentication through certificates

- Encrypted communication channel

2. **AES Encryption**:
   - File-level encryption using AES
   - Local key storage on client side
   - Protection against unauthorized access
3. **Data Integrity**:
   - TCP checksum verification
   - Protection against data corruption
   - Secure file transfer protocols

Key Security Considerations

1. **Client-Side Key Management**:
   - AES keys stored locally on client machines
   - **Loss of keys results in permanent data inaccessibility**
   - No server-side key storage
2. **Server Security**:
   - No file storage on server
   - Dedicated encryption/decryption operations
   - SSL certificate validation
3. **Data Protection**:
   - Single encrypted copy in Google Drive
   - Corrupted data handling
   - Secure transmission protocols

Technical Stack

1. **Core Libraries**:
   - Socket library for network communication
   - Crypto library for AES implementation
   - Google Drive API and Google Auth libraries
   - Cryptography and OpenSSL libraries
2. **User Interface**:
   - PyQt5 for GUI implementation
   - Intuitive user experience
   - Secure credential management

File Structure

```
Project Root/
├── Server/
│   ├── server.py
│   ├── run_server.py
│   ├── Server.crt
│   ├── Server.key
│   ├── encryption.py
│   ├── gdrive.py
│   ├── generate_ssl.py
│   ├── test_encryption.py
│   ├── Downloads/
│   └── Uploads/
└── Client/
    ├── gui.py
    ├── client.py
    ├── Server.crt
    └── run_client.py
```

Security Protocols

1. **Authentication**:
   - SSL certificate validation
   - Client-server handshake
   - Secure key exchange
2. **Encryption**:
   - AES-256 encryption standard
   - SSL/TLS for transmission
   - Secure key storage
3. **Data Integrity**:
   - Checksum verification
   - Error detection

Error Handling

1. **Connection Issues**:
   - Graceful disconnection
   - Reconnection protocols - server has appropriate time out for added safety.
   - Error logging
2. **Data Corruption**:

- Checksum validation
- Secure retransmission
- Data integrity verification

3. **Key Management**:
   - Key validation
   - Secure storage
   - Access control

Future Enhancements

1. **Additional Features**:
   - Multi-factor authentication
   - Key recovery mechanisms
   - Enhanced file sharing capabilities
2. **Performance Improvements**:
   - Parallel file processing
   - Optimized encryption algorithms
3. **Security Upgrades**:
   - Advanced encryption standards
   - Improved key management
   - Enhanced access control

Running the Project

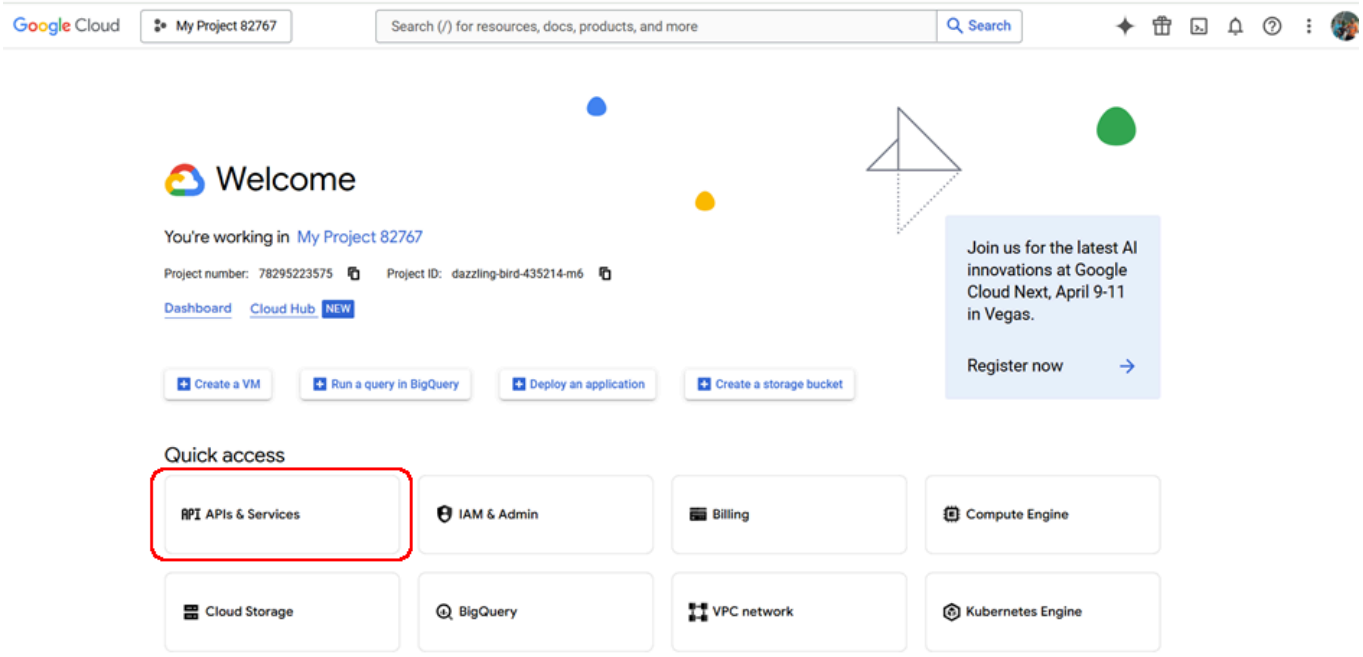
Prerequisites

1. Python 3.x installed
2. Required Python packages:

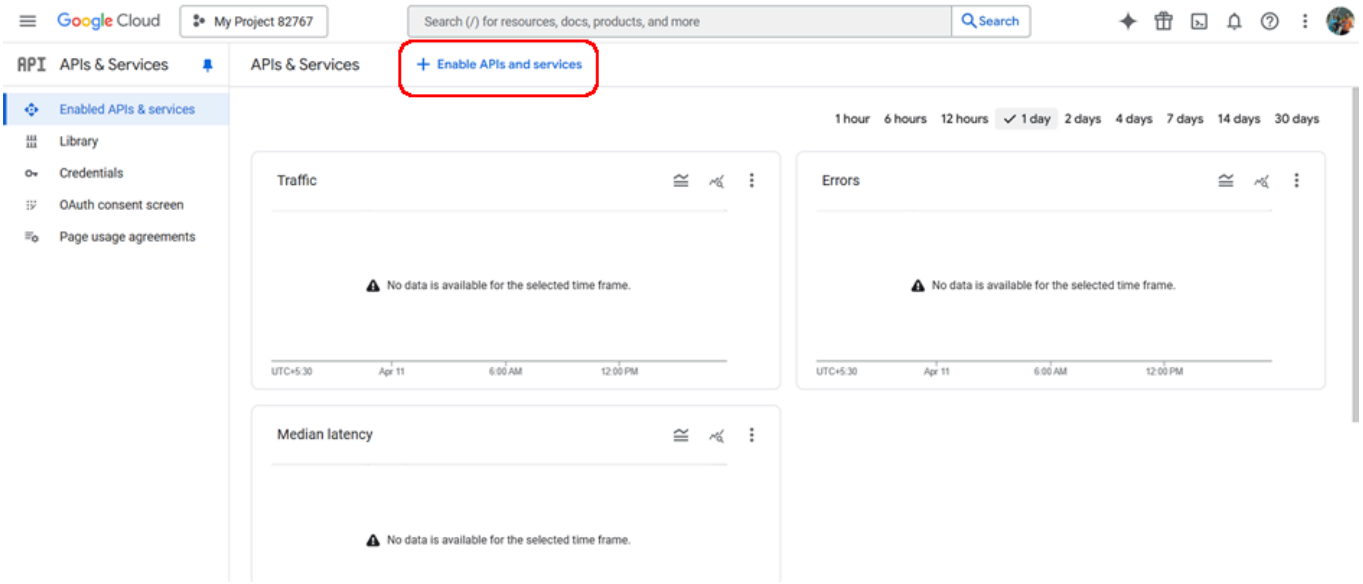
```
pip install pycryptodome
pip install google-auth google-auth-oauthlib google-auth-httplib2 google-api-python-client
pip install PyQt5
pip install cryptography
```

Google Cloud Credentials

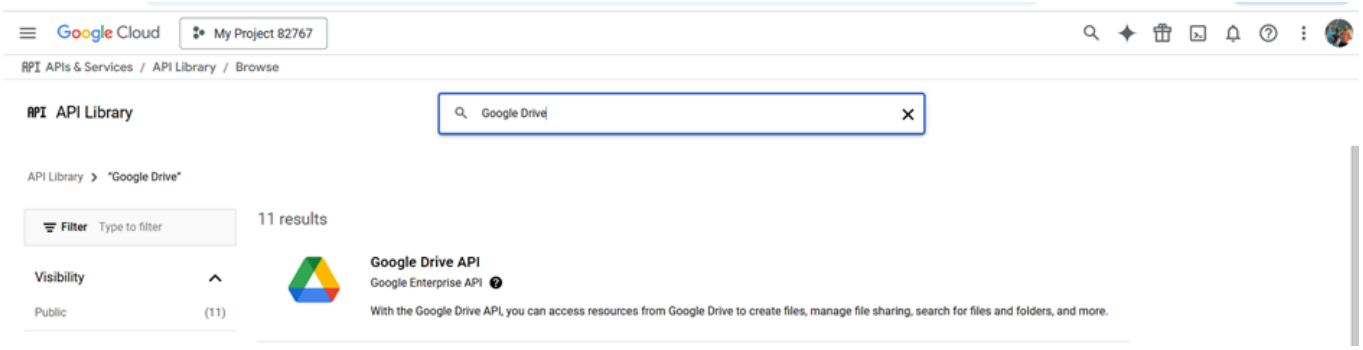
1. Create Google Cloud Console account and create a project
2. Click on APIs & Services



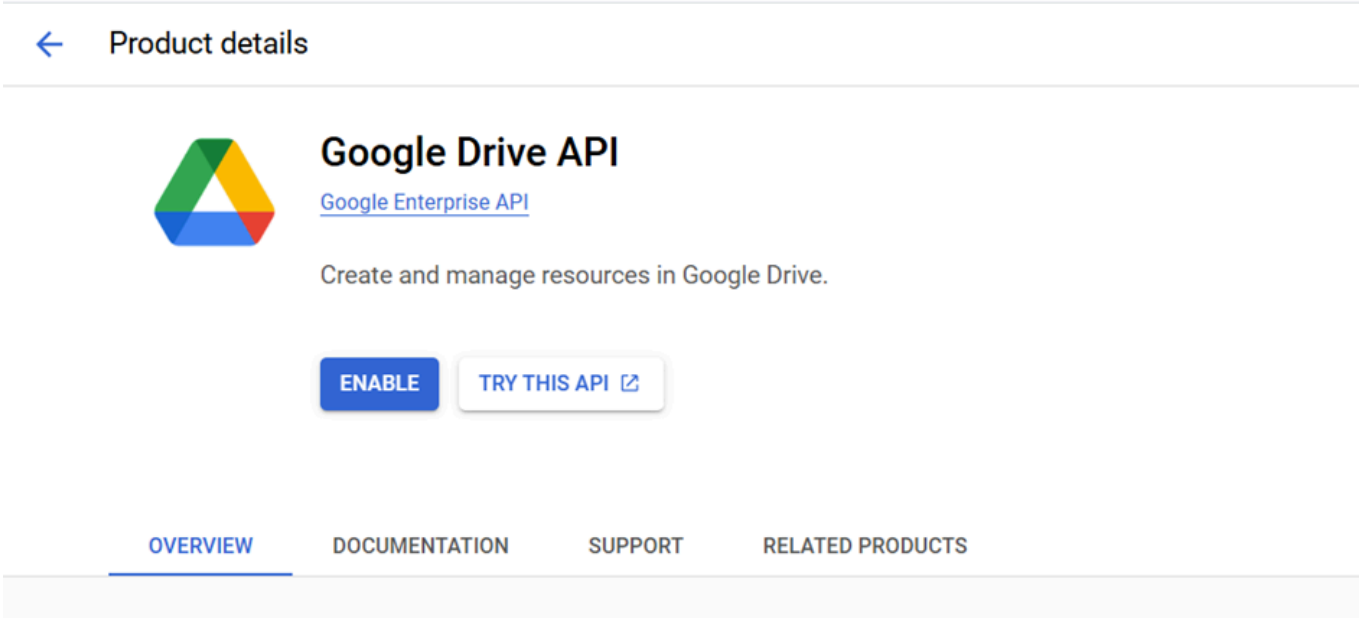
3. Click on Enable APIs and Services



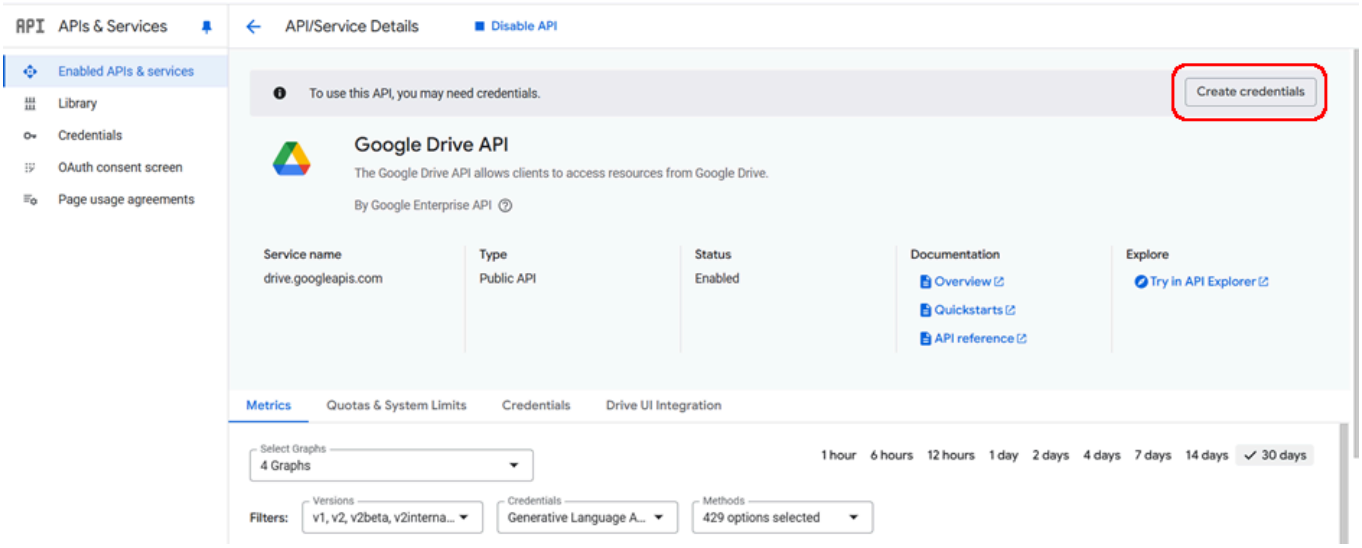
4. Browse and select Google Drive API



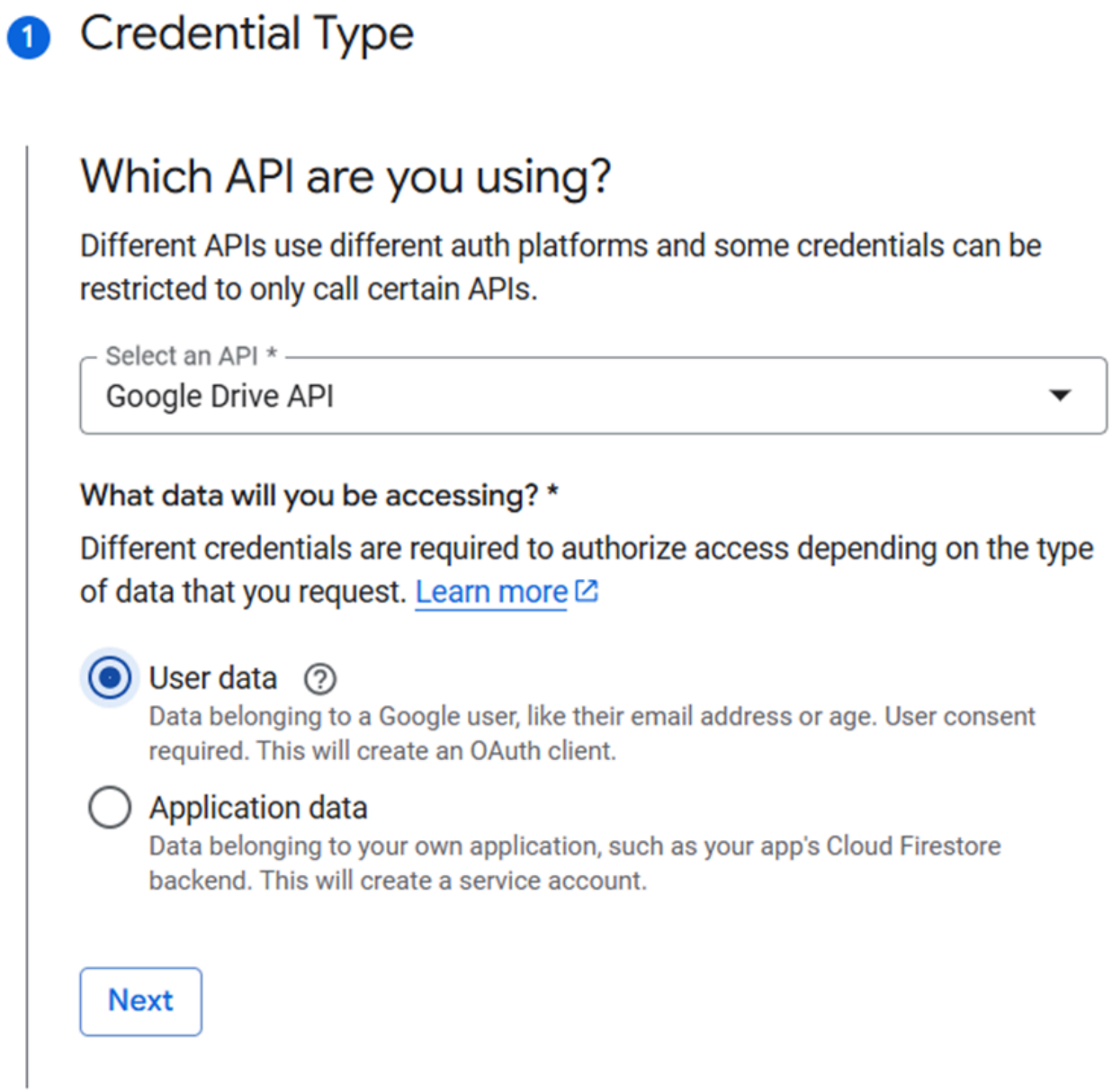
5. Click on Enable



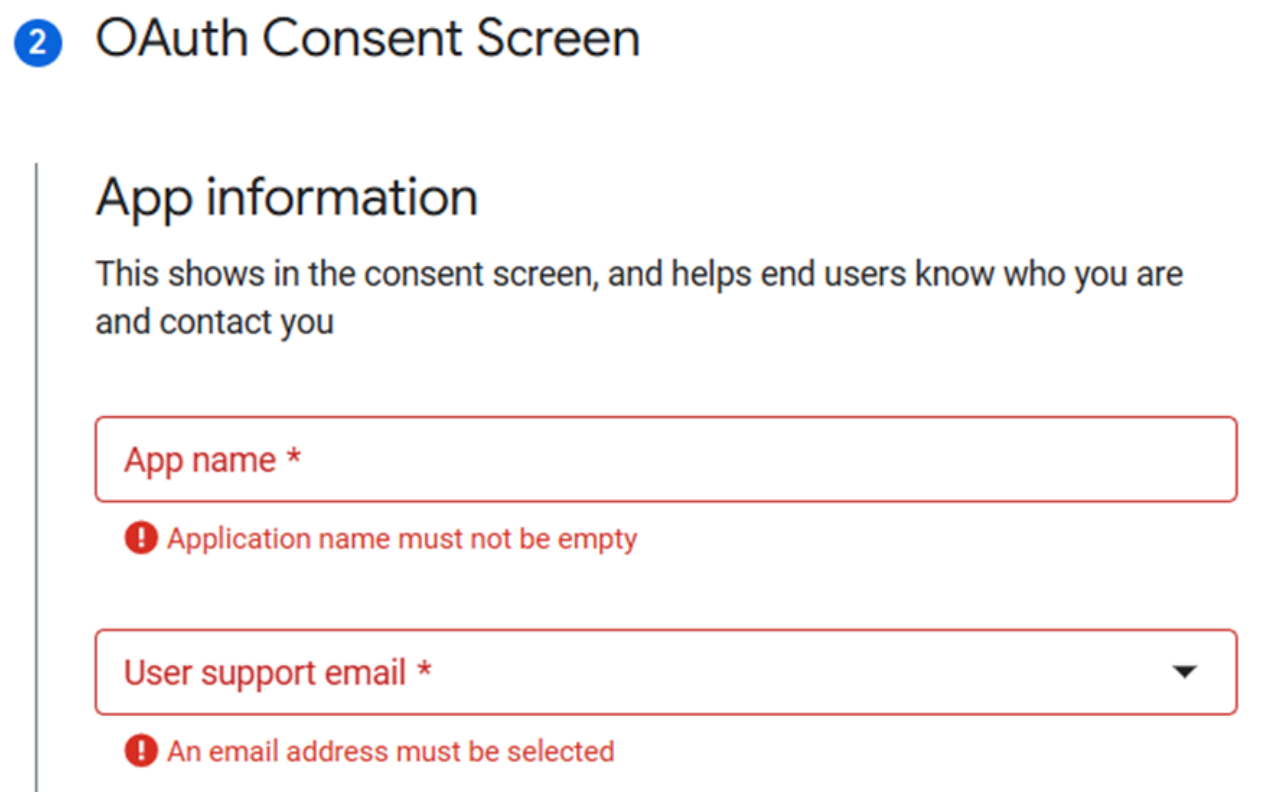
6. This Page will appear and select create credentials



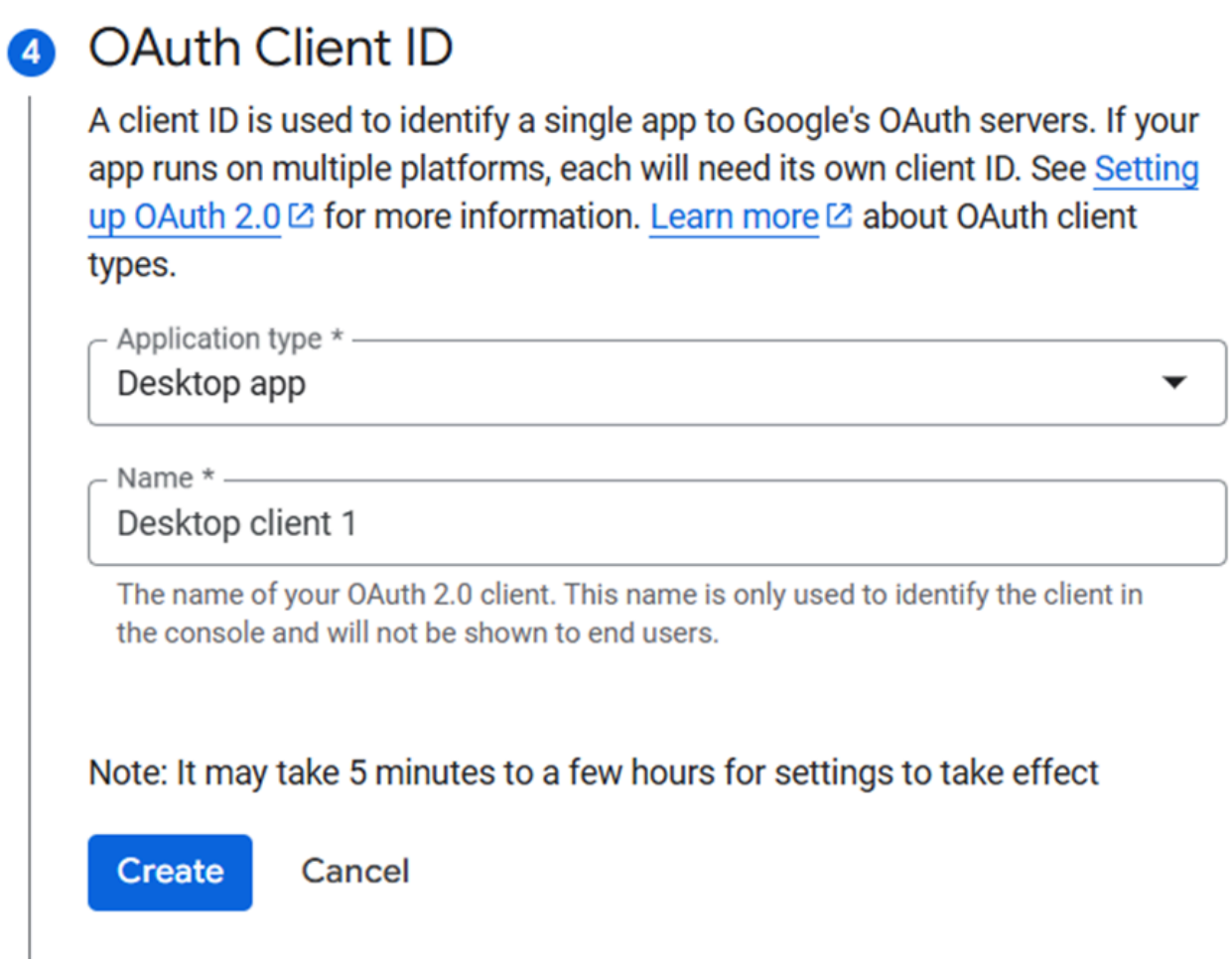
7. Click on User Data and click next



8. Fill App name and e-mail (if prompted for a scope - not required to add)



10. OAuth Client ID – Select Application type and Name



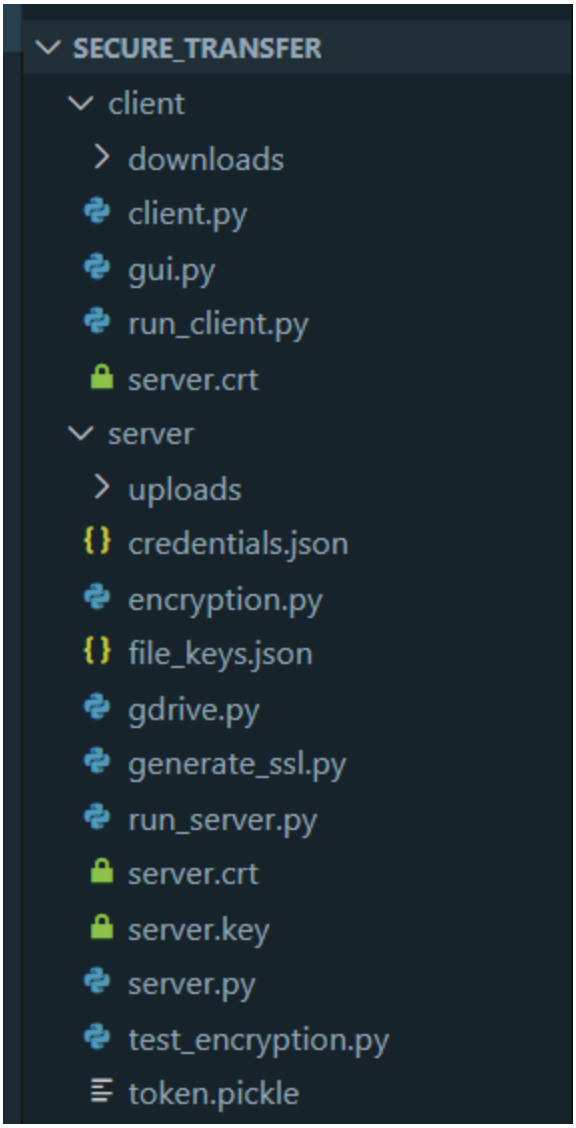
11. Download the credentials and rename it as `credentials.json` and click done
12. Save this `credentials.json` to the Server

Server Setup

1. Generate SSL certificates:
   Use the below command or run the `generate_ssl.py` file in the server

```
openssl req -x509 -newkey rsa:4096 -keyout Server.key -out Server.crt -days 365 -nodes
```

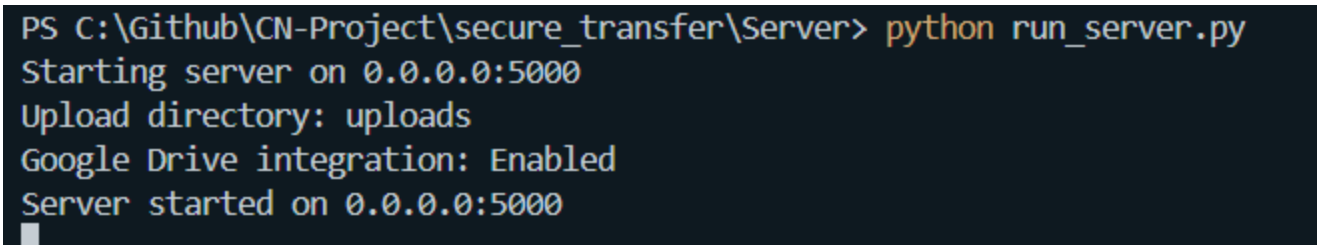
2. Copy the generated `Server.crt` to the client directory



3. Start the server:

```
python run_server.py
```

- The server will start listening on the specified port
- Note down the server's IP address and port number

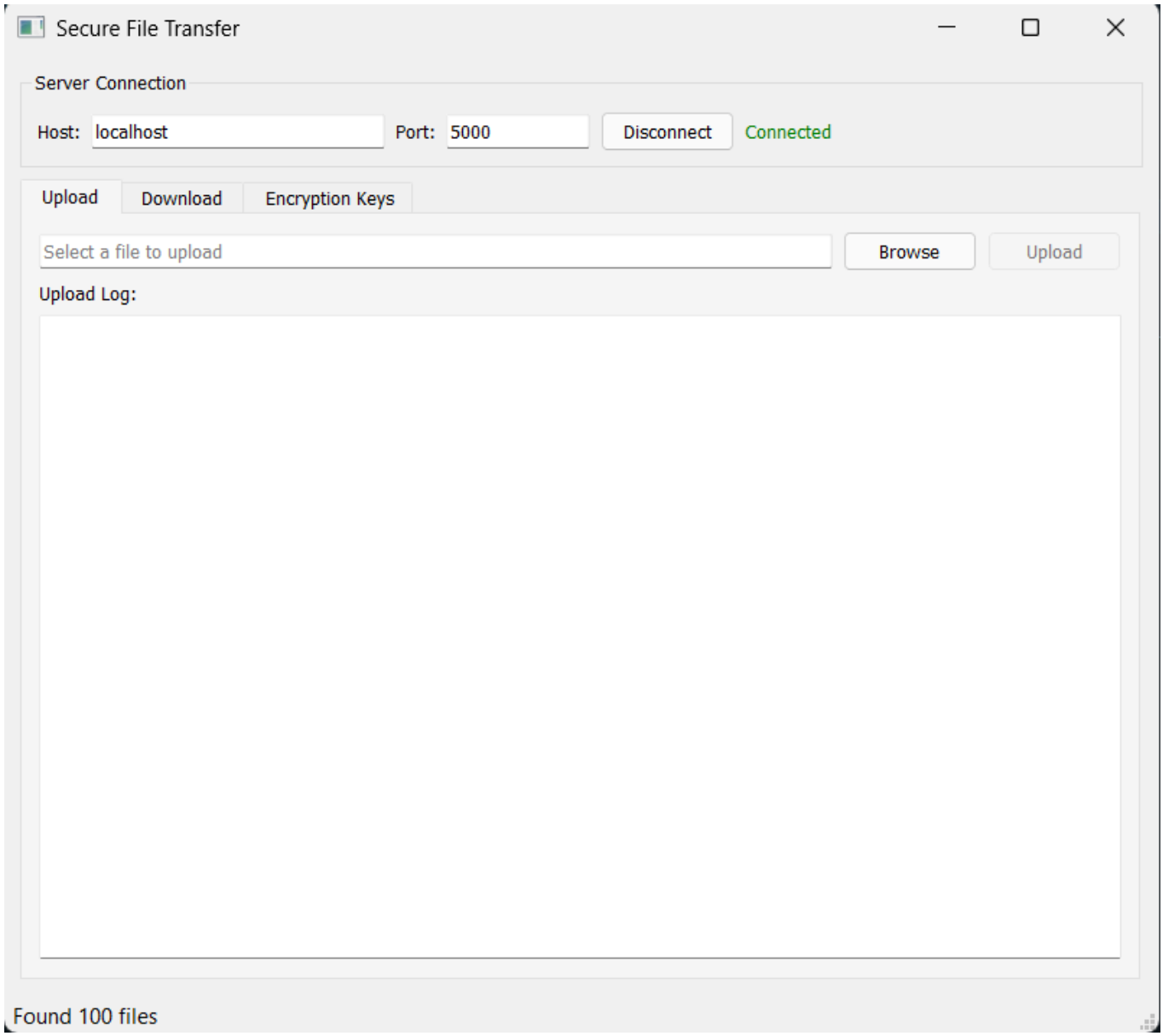


Client Setup

1. Ensure you have the `Server.crt` file in the client directory
2. Start the client application:

```
python run_client.py
```

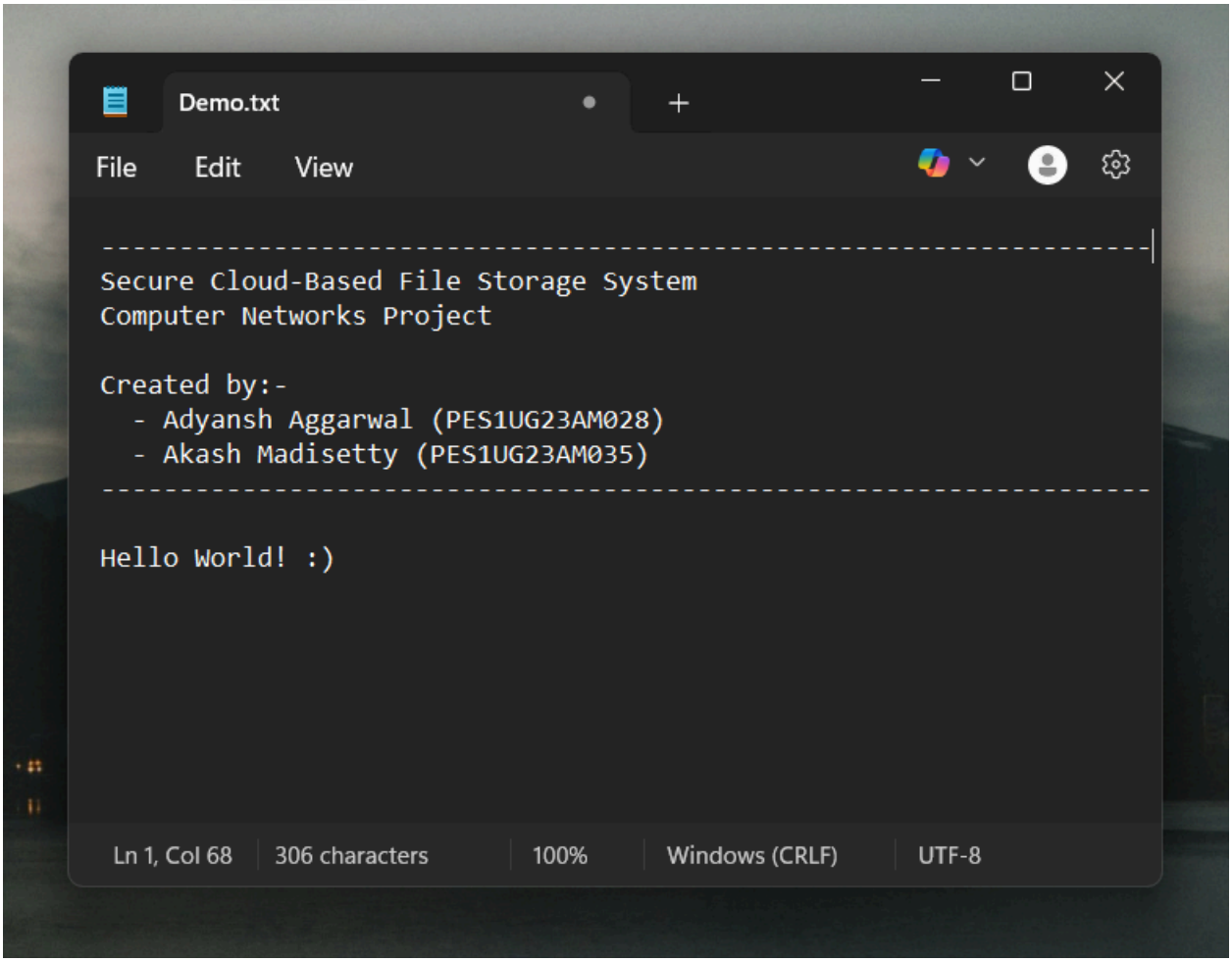
3. In the GUI:
   - Enter the server's IP address
   - Enter the server's port number
   - Click "Connect" to establish a secure connection

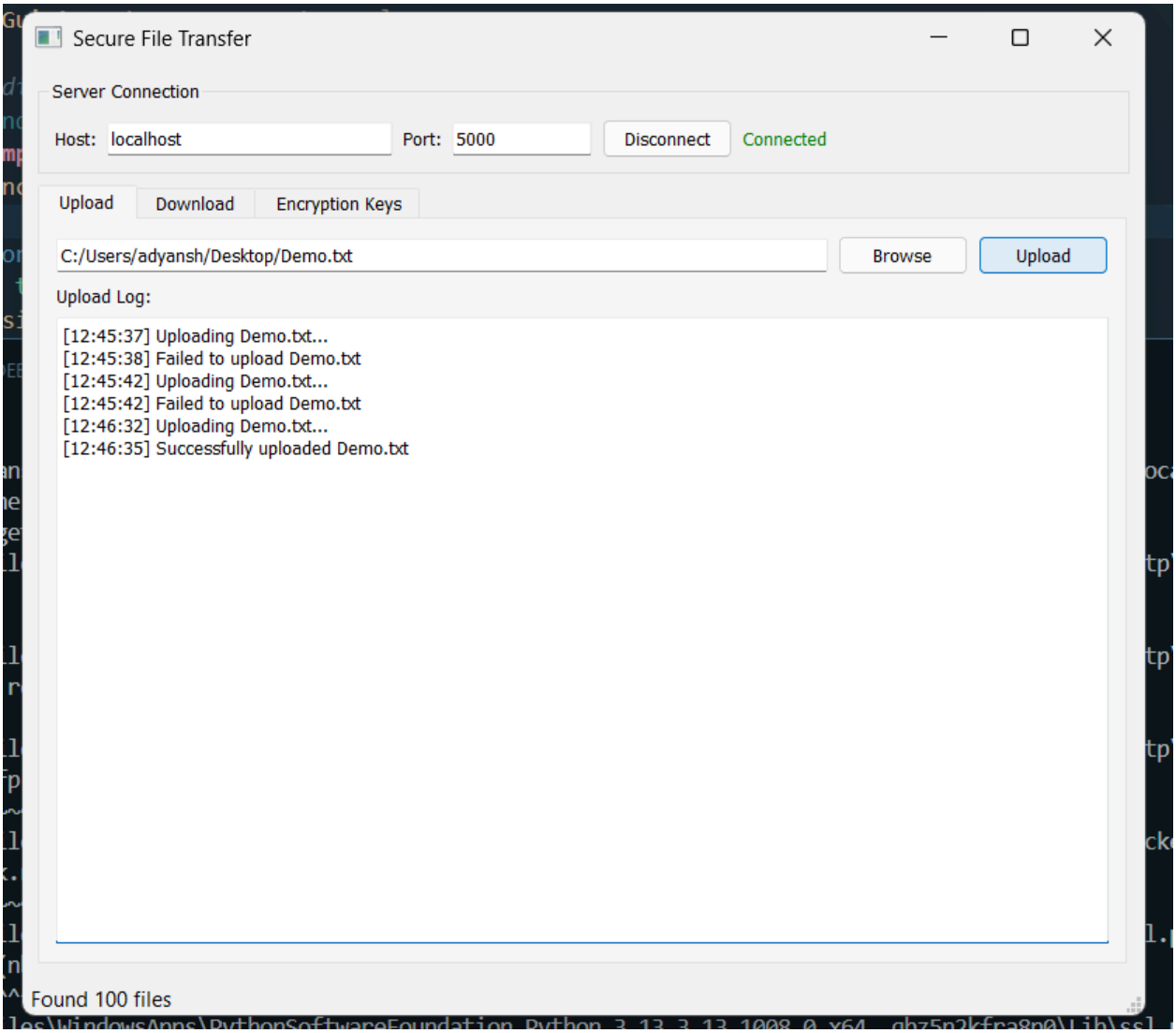


Using the Application

1. **File Upload**:
   - Click "Upload" button
   - Select the file to upload
   - Choose or generate an AES key (clicking upload automatically generates a new key make sure to save this)
   - The file will be encrypted and uploaded to Google Drive

- Let us upload a `demo.txt` for this example

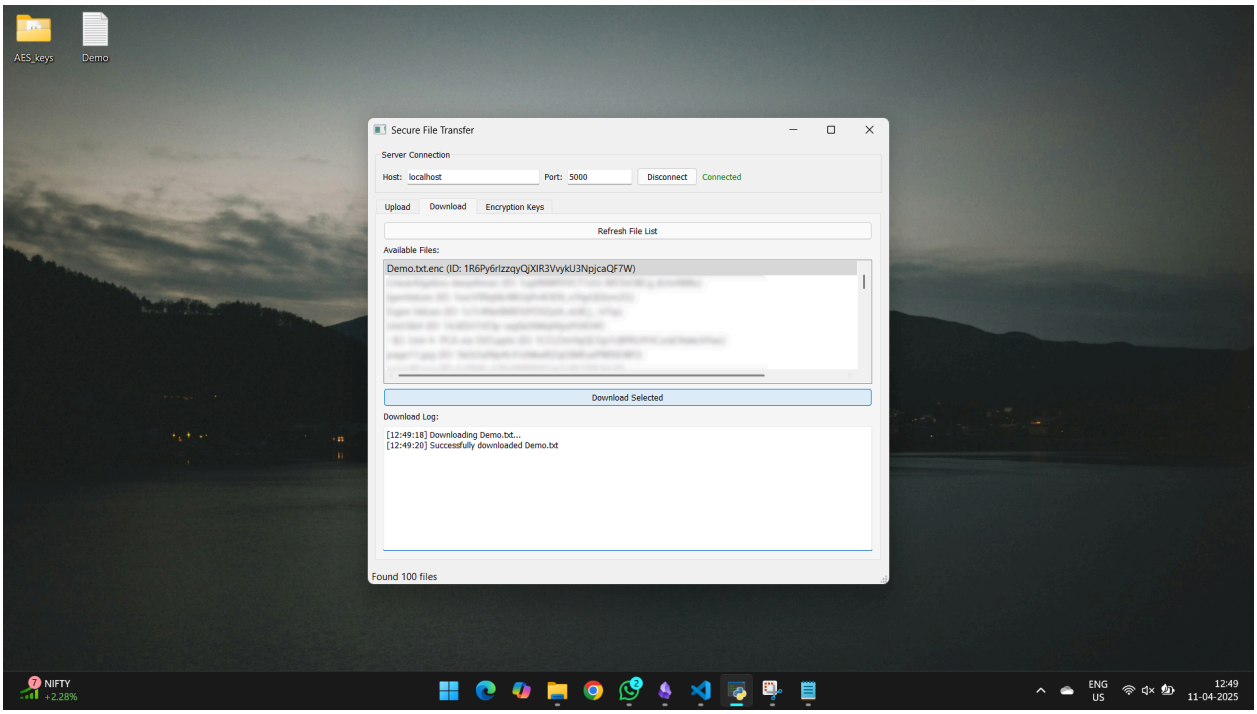




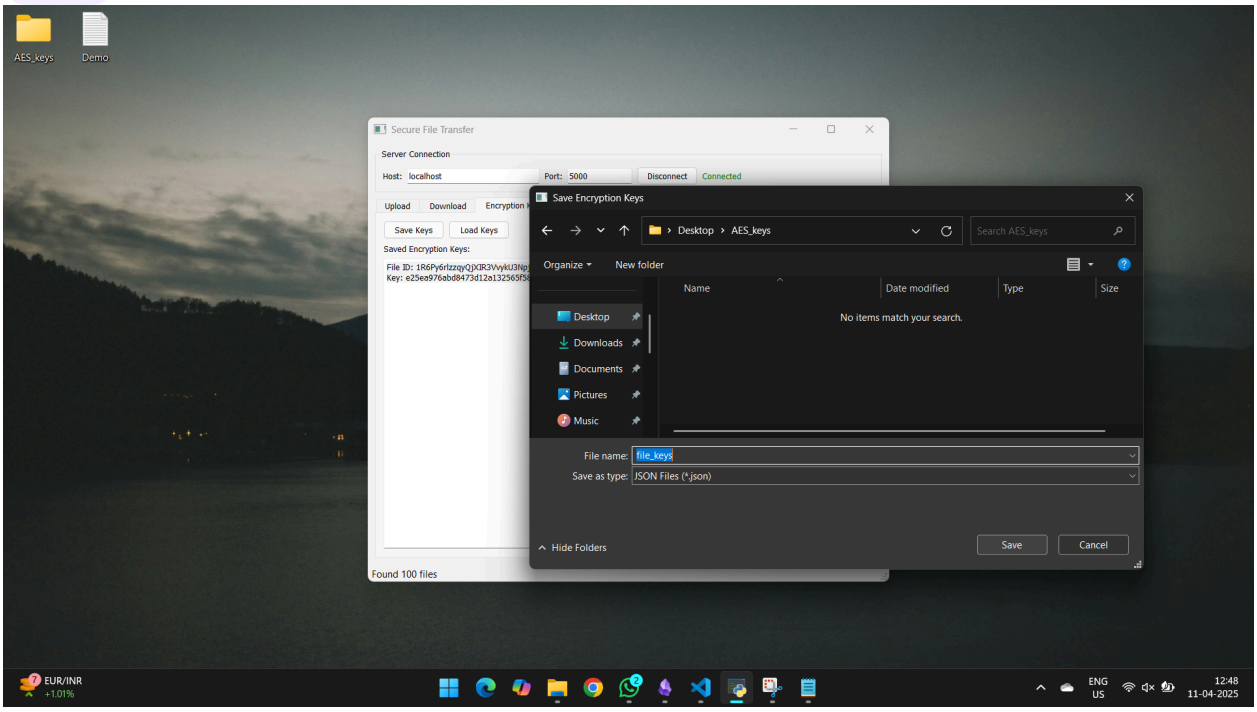
#Note : The Server was disconnected to demonstrate the error logs when file fails to upload

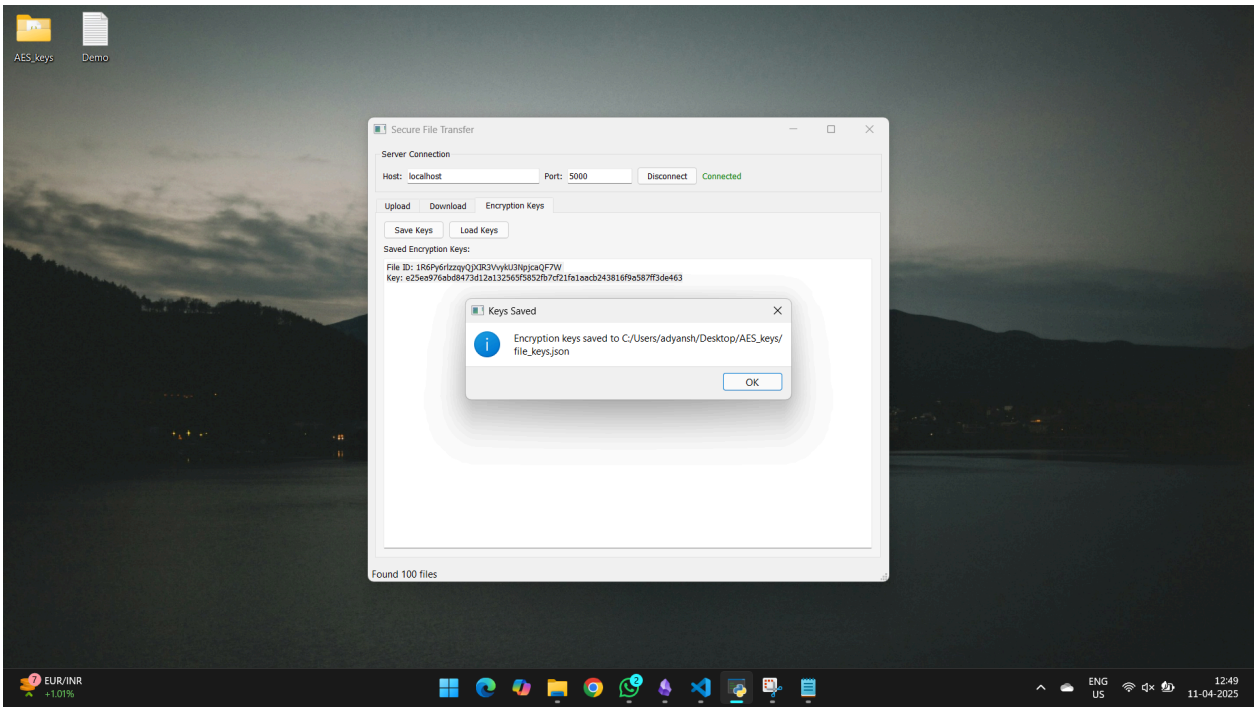
2. **File Download**:
   - Select the file from the list
   - Provide the correct AES key
   - Click "Download"

- The file will be decrypted and is saved locally in `/client/downloads`



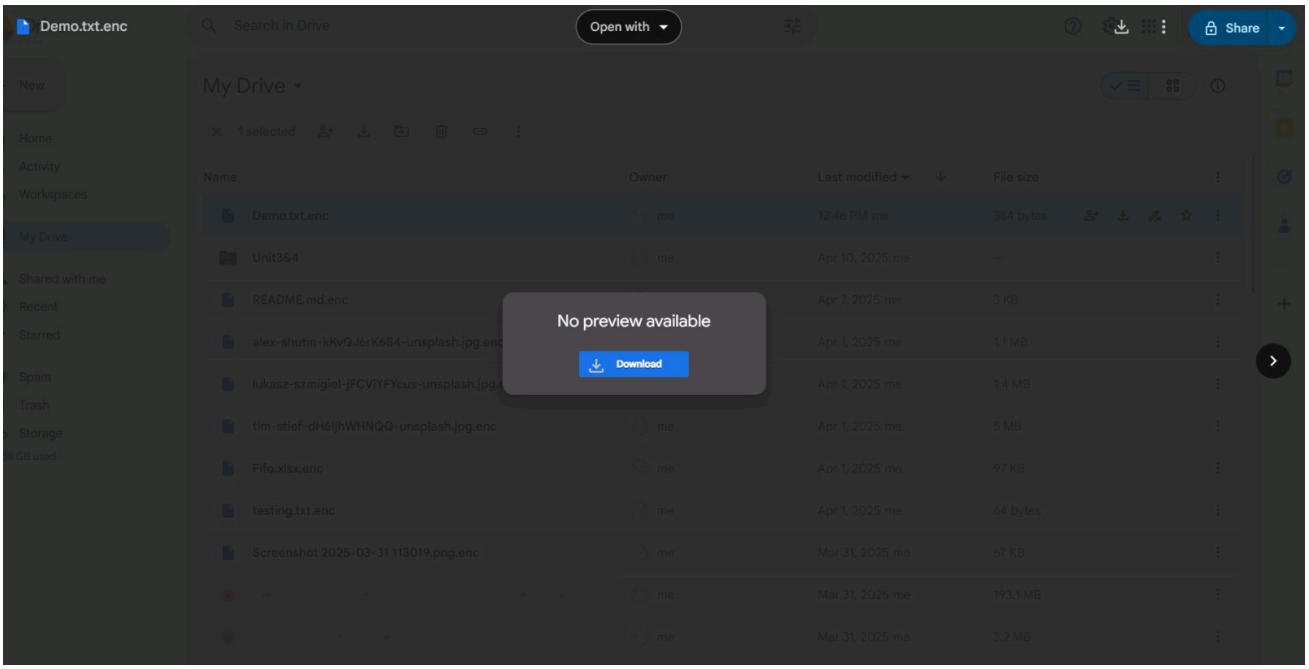
3. **Key Management**:
   - Load a new AES key and when used for a file make sure to save it.
   - Store keys securely as they are required for decryption
   - #Note : Lost keys cannot be recovered



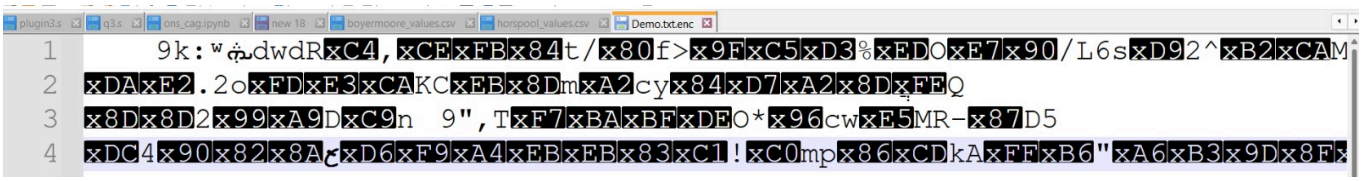


4. **Results**

- The uploaded file will be stored with a `.enc` extension in the Drive. This cannot be viewed even in the drive as the file extension is not supported.



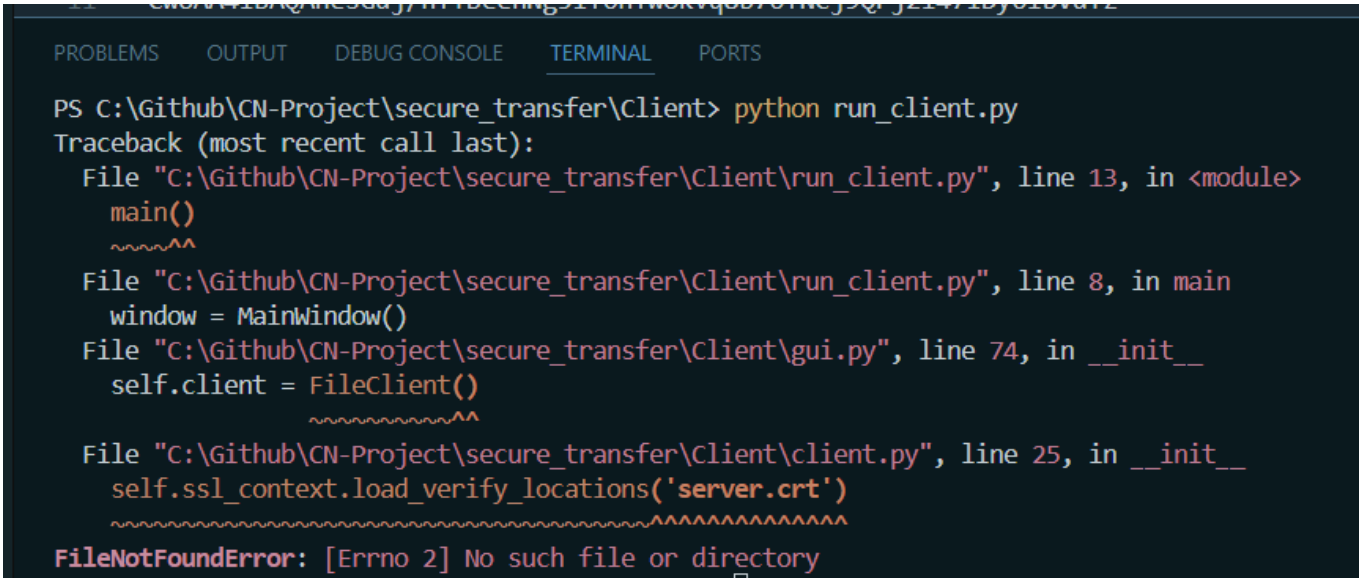
- Incase this file is downloaded from the Drive or a wrong AES key is used the file that is downloaded is corrupted (original `.enc` file is unchanged in the Drive)
  Corrupted file -



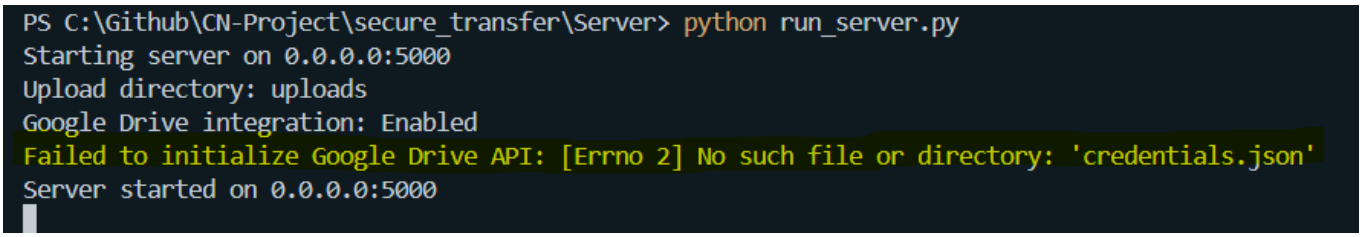
Edge Cases

1. Wrong encryption key provided/ no encryption key provided - in such a case the file still downloads but the data is corrupted.

2. SSL/TLS certificate or key files are missing or incorrect - client fails to launch



3. Missing or invalid `credentials.json`



```
PS C:\Github\CN-Project\secure_transfer\Client> python run_client.py
Connected to server at localhost:5000
Failed to list files: Google Drive integration not enabled
```

Troubleshooting

1. **Connection Issues**:
   - Verify server IP and port
   - Check if server is running
   - Ensure SSL certificate is valid
2. **Authentication Errors**:
   - Verify Google Drive credentials
   - Check SSL certificate validity
   - Ensure proper key management
3. **File Transfer Issues**:
   - Check network connectivity (Client and Server must be on the same network)
   - Verify file permissions
   - Ensure sufficient storage space