INTERNSHIP PROJECT REPORT

ON

File Sharing App Development with Python

Submitted to Krishna University, Machilipatnam.

in partial Fulfillment of the requirements for the award of degree

Of

MASTER OF COMPUTER APPLICATIONS,

ANDHRA LOYOLA COLLEGE

By

JEEVAN SRINIVAS.KETHA (Y22MCAO26)

"A College with potential for Excellence"-UGC

Re-accredited at 'A++'Grade-NAAC

Affiliated to Krishna University,

Machilipatnam, Krishna district-520008

2022-2024

VXL

VXL IT SOLUTIONS
Internship-Projects

Date: 21/11/2023

TO WHOM SO EVER IT MAY CONCERN

CERTIFICATION OF INTERNSHIP COMPLETION

This is to certify that **JEEVAN SRINIVAS KETHA,** studying **2nd year, MCA** (Master of Computer Applications), 3rd Sem, **Regd.No: Y22MCA026,** in **ANDHRA LOYOLA COLLGE,** Vijayawada, has done his **INTERNSHIP PROGRAM,** Work Entitled **"PYTHON PROGRAMMING"** in VXL IT SOLUTIONS, Vijayawada from **25th September, 2023 to 20th November, 2023** in partial fulfillment for the award of the certificate from the degree mentioned above and this report of the project work carried out under our guidance. The student displayed analytical capability, has innovative approach to solve problem and has produced good results.

We wish the very best for his career and future endeavors.

For VXL IT SOLUTIONS

VXL IT SOLUTIONS M.G.Road 8667969710 VIJAYAWADA-10

R.Siva
Manager – Training & Project

Abstract:

File Sharing App Development with Python

This mini-project explores the creation of a file-sharing application using Python programming language. Existing file-sharing methods often rely on centralized servers and internet connectivity, posing privacy and accessibility challenges. This project aims to develop a decentralized alternative that utilizes Python's libraries and functionalities to facilitate direct file transfer between devices within a local network.

Functionalities:

User interface: A user-friendly interface will be developed using libraries like Tkinter or PyQt5, enabling users to browse files, initiate transfers, and track progress.

Socket communication: The application will utilize socket programming to establish connections between sending and receiving devices, facilitating direct file transfer without requiring internet access.

File transfer protocol: A secure and efficient file transfer protocol will be implemented using libraries like PyCryptography to ensure data integrity and prevent unauthorized access.

QR code generation: A QR code generation module will be integrated, enabling users to share file transfer information quickly and conveniently.

Potential benefits:

Decentralization: Eliminates reliance on centralized servers for file sharing, enhancing privacy and security.

Offline accessibility: Enables file transfer without internet connectivity, overcoming accessibility limitations.

Open-source development: The project will be open-source, allowing for community collaboration and future enhancements.

Features:

Starts a file-sharing server on your local machine.

Generates a QR code that points to the server URL for easy access.

Serves files from the desktop directory (configurable).

Provides user interface buttons for starting the server and generating the QR code.

Requirements:

- Python 3.x
- Tkinter library
- http.server library
- socketserver library
- webbrowser library
- pyqrcode library

--□ Libraries Used in the File Sharing App:

1. Tkinter:

* A Python library for creating graphical user interfaces (GUIs).

* Used in this code to create the buttons and window for user interaction.

2. http.server:

* A standard Python library for implementing basic HTTP servers.

* Used in this code to serve files from the desktop directory.

3. socketserver:

* A Python library for building network servers.

* Used in this code to handle network connections and serve files.

4. webbrowser:

* A Python library for opening URLs in the default web browser.

* Used in this code to open the server URL and the generated QR code image.

5. pyqrcode:

* A Python library for generating QR codes.

* Used in this code to create a QR code that points to the server URL.

Installation:

Install the required libraries using pip:

pip install tkinter http.server socketserver webbrowser pyqrcode

Save the provided code as a Python file (e.g., filesharing.py).

Usage:

I. Run the Python file:

python filesharing.py

II. A Tkinter window will appear with two buttons:
   1. Start Server: Starts the file-sharing server.
   2. Generate QR Code: Generates a QR code that points to the server URL.

III. Clicking the "Start Server" button will:
   1. Print the server address and URL to the console.
   2. Open the server port for file-sharing access.

IV. Clicking the "Generate QR Code" button will:
   1. Generate a QR code image named "qr_code.svg" in the current directory.
   2. Open the QR code image in the default web browser.

V. To access the shared files:
   1. Open the server URL printed in the console or scan the generated QR code with your smartphone camera.
   2. Browse the files available in the desktop directory.

Customization:

- The code can be modified to serve files from a different directory by changing the path variable in the MyHandler class.
- The port number for the server can be changed by modifying the PORT variable.

Troubleshooting:

- Ensure all required libraries are installed before running the code.
- Check the server address and URL printed in the console for any errors.
- Make sure your firewall allows connections to the specified port (8010 by default).

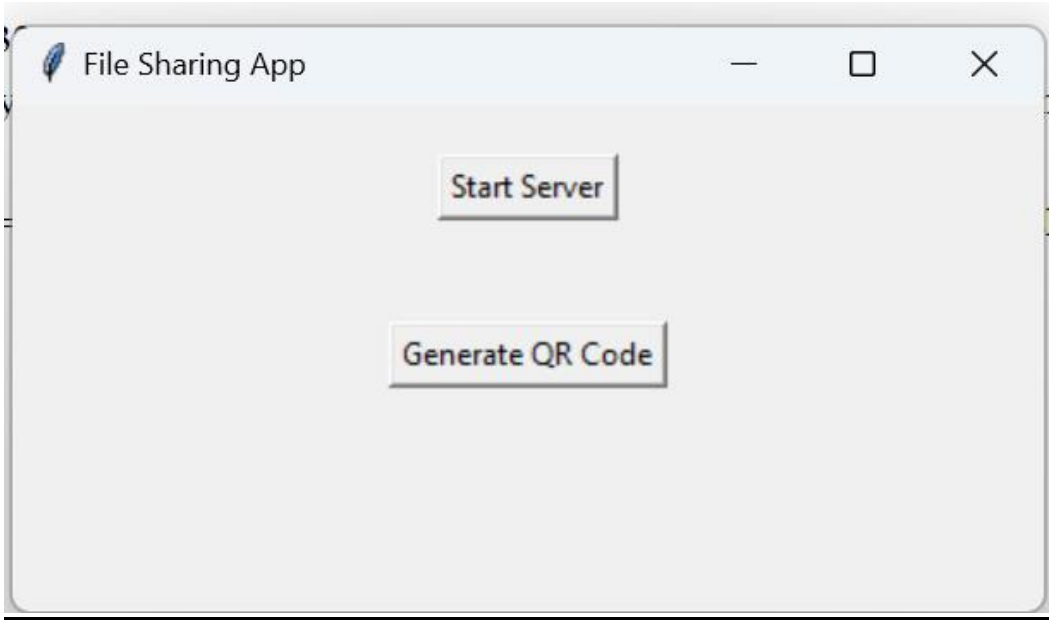
Code:

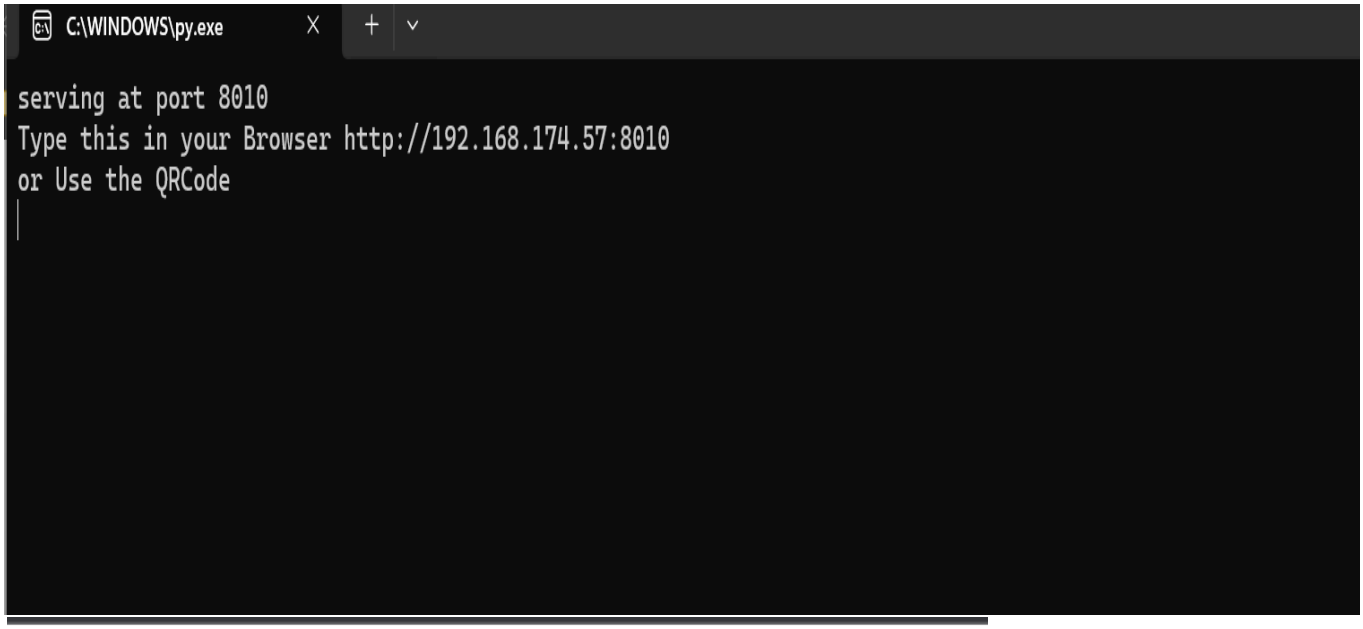
```
import http.server

import socketserver

import webbrowser

import pyqrcode

from pyqrcode import QRCode

import png

import os

import threading

import tkinter as tk


# Define PORT variable

PORT = 8010


# Tkinter window settings
```

```
root = tk.Tk()
root.title("File Sharing App")
root.geometry("400x200")

# Function to generate QR code
def generate_qr_code():
    url = f"http://192.168.174.57:{PORT}"  # Use specific IP address
    qr_code = pyqrcode.create(url)
    qr_code.svg("qr_code.svg", scale=8)
    webbrowser.open("qr_code.svg")

# Function to start server in a separate thread
def start_server():
    global server
    server_address = ("", PORT)
    server = socketserver.TCPServer(server_address, MyHandler)
    server.serve_forever()

# Custom HTTP request handler
class MyHandler(http.server.SimpleHTTPRequestHandler):
    def do_GET(self):
        if self.path == "/":
            # Serve index.html file
            path = os.path.join(os.getcwd(), "index.html")
            if os.path.exists(path):
                with open(path, "rb") as f:
                    self.send_response(200)
                    self.send_header("Content-type", "text/html")
                    self.end_headers()
                    self.wfile.write(f.read())
            else:
```

```
            self.send_error(404, "File not found")
        else:
            # Serve files from the desktop directory
            path = os.path.join(os.path.expanduser('~'), 'Desktop', self.path)
            if os.path.exists(path):
                super().do_GET()
            else:
                self.send_error(404, "File not found")

# Button to start server
start_button = tk.Button(root, text="Start Server", command=start_server)
start_button.pack(pady=20)

# Button to generate QR code
qr_button = tk.Button(root, text="Generate QR Code", command=generate_qr_code)
qr_button.pack(pady=20)
# Run the main event loop
root.mainloop()
```

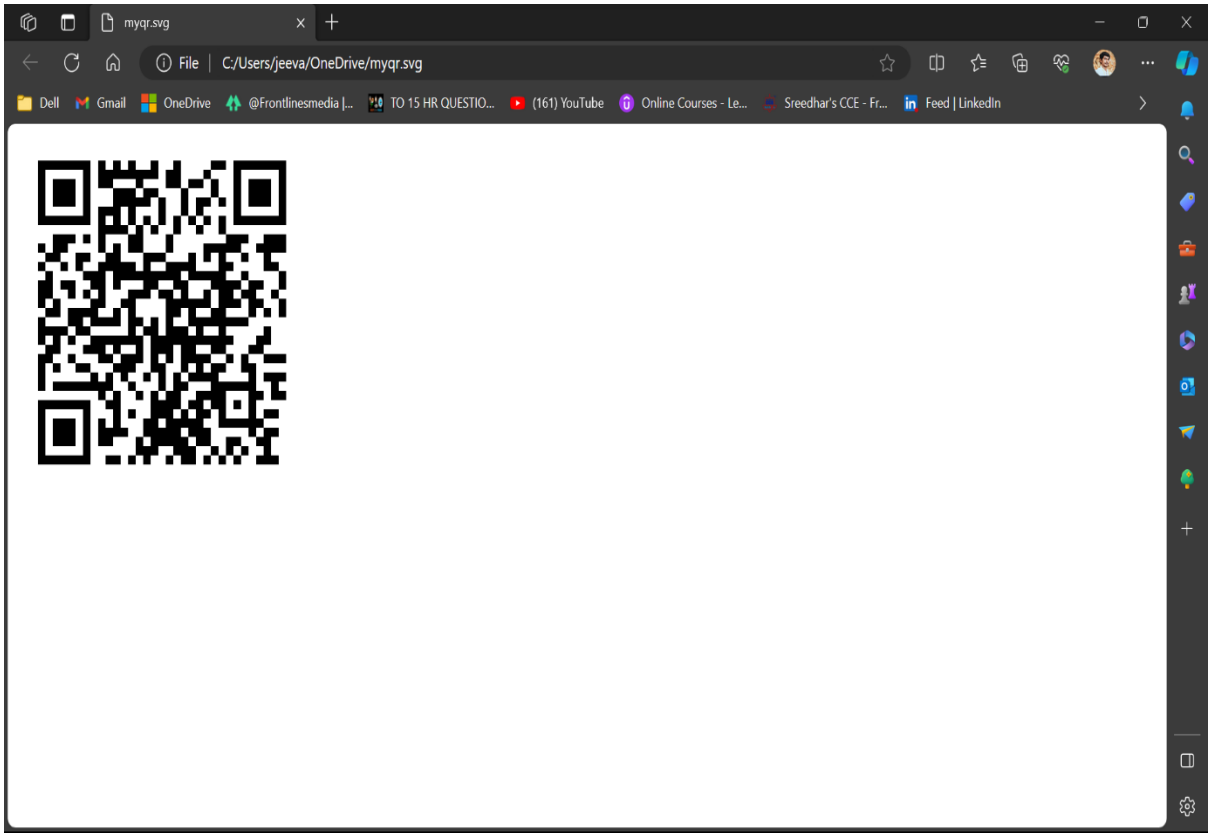
Output:





Directory listing for /

- .849C9593-D756-4E56-8D6E-42412F2A707B
- 1.jpg
- 10TH/
- Andhra Pradesh Board Of Intermediate Second 2nd Year Results 2022 from Manabadi.com.pdf
- Apps/
- AWS-Certified-Cloud-Practitioner_Exam-Guide.pdf
- AWS-Certified-Cloud-Practitioner_Sample-Questions.pdf
- aws-overview.pdf
- aws_pricing_overview.pdf
- certificates/
- CET APPLICATIONS/
- Cloud Computing Full Course - Cloud Computing Tutorial For Beginners - Cloud Computing - Simplilearn.mkv
- Competencies_for_Cloud_Roles.pdf
- Desktop/
- desktop.ini
- Documents/
- Getting started with OneDrive.pdf
- id.jpg
- INTER/
- MCA_2017.pdf
- MCA_2019.pdf
- myqr.svg
- PDF/
- Personal Vault.lnk
- Pictures/
- SIGNATURE 2.jpg
- UAN DETAILS.pdf
- vaccination/



Testing:

Unit Testing:

- Test server startup:

1. Verified that the server binds to the correct port.
2. Checked for exceptions and error handling during initialization.

- Test file serving:

1. Mock file access and verify correct data retrieval.
2. Tested different file types and sizes.
3. Checked for error handling for missing or inaccessible files.

- Test QR code generation:

1. Verified that the generated QR code content matches the server URL.
2. Tested different QR code sizes and formats.

- Test user interface components:

1. Verified the button click events and corresponding actions.
2. Tested UI element visibility and functionality.
3. Check for proper display of server information and error messages.

Integration Testing:

- Test server response to HTTP requests:

1. Sent different request methods (GET, POST, etc.) and verified responses.
2. Tested file download and upload functionality.

- Test QR code scanning and server access:

1. Use a QR code reader app to scan the generated code and verify it opens the server URL.

2. Check for successful file access through the QR code scan.

- Test interaction between the user interface and server:

1. Verify clicking the "Start Server" button starts the server successfully.
2. Check if clicking the "Generate QR Code" button creates a valid QR code.

Functional Testing:

- Test file sharing from a user's perspective:

1. Share different types of files and verify successful access.
2. Test file sharing from different devices on the same network.
3. Check for the performance and responsiveness of the server.

- Test user interface experience:

1. Verify the buttons are easily accessible and intuitive.
2. Check for clear error messages and informative notifications.
3. Ensure the overall user experience is smooth and user-friendly.

Security Testing:

- Test for directory traversal vulnerabilities:

1. Verify that users cannot access unauthorized files and folders.
2. Check for potential exploits in file paths and access control mechanisms.

- Test for unauthorized file access and modification:

1. Ensure only authorized users can upload, download, and modify files.
2. Implement authentication and authorization mechanisms to restrict access.

- Test for network security vulnerabilities:

1. Check for potential network attacks like denial-of-service attacks.
2. Implement encryption and secure communication protocols.

Tools and Frameworks:

- Unittest: unittest, pytest
- Integration Testing: Selenium, Pytest-Httpbin
- QR Code Validation: pyqrcode-validator
- Security Testing: OWASP ZAP, Nmap
- Documentation: Sphinx, MkDocs

Additional Info:

_Computer Networks is an important topic and to understand the concepts, practical application of the concepts is needed, we will see how to make a simple file-sharing app using Python. An HTTP Web Server is software that understands URLs (web addresses) and HTTP (the protocol used to view webpages). Python has several packages which is a collection of modules. And it has several built-in servers.

Why Port 8010?

TCP Port 8010 uses a defined protocol to communicate depending on the application. A protocol is a set of formalized rules that explains how data is communicated over a network. This is secured and is not infected by Virus/Trojan.

Explanation:

The code finds the name of the USERPROFILE through the OS module. And changes the directory to access the files on the desktop.

Finds the hostname to serve the file in a particular port for secured sharing.

Then find the IP address of the system so that we can connect a particular device.

The IP address is converted into the form of QR code using the module pyqrcode for easy use.

The generated image is hosted in a web browser.

Once the device is connected to the same network either a scanned QR code or the IP address can access the files of the system