Computer Networks

CHAT SERVER

TRAVIS - 1113545
FERRAN - 1113542
NOLAN - 1113543

OVERVIEW

- **Summary:**

This project demonstrates practical implementation concepts in computer networks, combining web technologies and QR code utilities.

- **Key Features:**

A server-based web application.
Frontend: HTML, CSS.
Backend: Python-powered server.

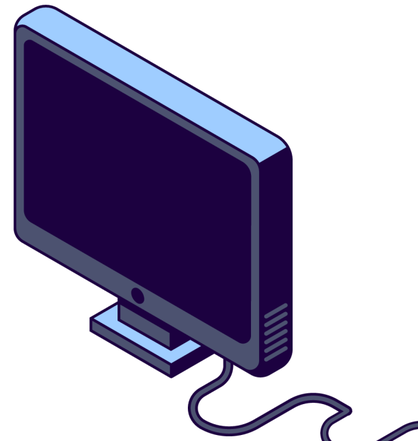
- **Real-World Applications:**

QR code generation for user data.
Web-based interactions between users and servers.

SYSTEM DESIGN

Components

1. **Backend**:
   - server.py: Handles requests and serves HTML pages.
   - Processes QR codes with import_qrcode.py.
   - Communicates with users.json to manage user data.
2. **Frontend**:
   - HTML Pages:
     - index.html: Landing page for users.
     - thank_you.html: Confirmation after data submission.
     - 404.html: Handles errors gracefully.
3. **Data Management:**
   - users.json: Stores user information in JSON format.
   - form_data.txt: Logs form submissions.



Flow Diagram

Include a simple flowchart in your slides showing:

User → Web Browser → Server → Backend Processing → Response (HTML/JSON).

TECHNICAL DETAILS

Key Snippets from server.py:

1. Setting up the Server

```
from http.server import HTTPServer, BaseHTTPRequestHandler
from urllib.parse import parse_qs
import os
import json
from datetime import datetime
```

2. Handling GET Requests

```
def do_GET(self):
    """Handle GET requests."""
    if self.path == '/':
        self.serve_file('index.html', 200)
    elif self.path.startswith('/static/'):
        self.serve_file(self.path.lstrip('/'), 200)
    elif self.path.startswith('/uploads/'):
        self.serve_file(self.path.lstrip('/'), 200)
    else:
        self.serve_404()
```

3. Handling POST Requests

```
def do_POST(self):
    """Handle POST requests."""
    if self.path == '/chat':
        self.handle_chat_submission()
    elif self.path == '/upload':
        self.handle_file_upload()
```

4. Dynamic Content Injection

```python
def serve_file(self, filename, status_code):
    """Serve static files and inject dynamic content."""
    try:
        file_path = os.path.join(os.getcwd(), filename)
        if filename.endswith('.html'):
            with open(file_path, 'r', encoding='utf-8') as file:
                content = file.read()
            if filename == 'index.html':
                content = content.replace("{{chat_messages}}", self.generate_chat_messages())
                content = content.replace("{{uploaded_files}}", self.generate_uploaded_files())
            self.send_response(status_code)
            self.send_header('Content-Type', 'text/html')
            self.send_header('Content-Length', len(content))
            self.end_headers()
            self.wfile.write(content.encode('utf-8'))
        else:
            with open(file_path, 'rb') as file:
                content = file.read()
            self.send_response(status_code)
            self.send_header('Content-Type', self.get_mime_type(filename))
            self.send_header('Content-Length', len(content))
            self.end_headers()
            self.wfile.write(content)
    except FileNotFoundError:
        self.serve_404()
```

5. File Upload Functionality

```python
def handle_file_upload(self):
    """Handle file uploads."""
    form = FieldStorage(fp=self.rfile, headers=self.headers, environ={'REQUEST_METHOD': 'POST'})
    file_field = form['file']

    if file_field.filename:
        os.makedirs('uploads', exist_ok=True)
        file_path = os.path.join('uploads', file_field.filename)
        with open(file_path, 'wb') as output_file:
            output_file.write(file_field.file.read())
```

Client-Server Model:

- **S**erver (server.py) acts as a central hub.
- Clients (browsers) interact via HTTP requests.

Data Transmission Protocols:

- **H**TTP for request/response communication.

QR Code as a Communication Tool:

- **E**ncodes data for secure and efficient sharing.

JSON Data Exchange:

- **U**sed for lightweight data storage and sharing.

APPLICATIONS IN COMPUTER NETWORKS

CODE WALKTHROUGH

CHALLENGES AND LESSONS LEARNED

Challenges

- **U**nderstanding HTTP request-response cycles.
- Handling JSON for dynamic data exchange.

Solutions

- **U**sed Python’s Flask/Django for backend.
- Designed a modular codebase for easy debugging.

FINAL TAKEAWAYS

- Demonstrated a functioning client-server system.
- Learned practical applications of networking principles.
- Explored innovative uses of QR codes and JSON in network communication.

THANK YOU