test_server.py

```
from fastapi import FastAPI, WebSocket, WebSocketDisconnect
from fastapi.responses import HTMLResponse
from fastapi.middleware.cors import CORSMiddleware
import uvicorn
import os
import asyncio

app = FastAPI()

# Add CORS middleware
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

@app.get("/")
async def get():
    print("Homepage requested")
    with open("test.html", encoding='utf-8') as f:
        return HTMLResponse(f.read())

@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket):
    print("WebSocket connection attempt received")
    await websocket.accept()
```

```
print("WebSocket connection accepted")

try:
    # Keep connection alive with ping/pong
    while True:
        try:
            # Wait for messages with a timeout
            data = await asyncio.wait_for(
                websocket.receive_text(),
                timeout=20.0
            )
            print(f"Received: {data}")
            await websocket.send_text(f"Server received: {data}")

        except asyncio.TimeoutError:
            try:
                # Send ping to keep connection alive
                print("Sending ping...")
                await websocket.send_text("ping")
                continue
            except Exception as e:
                print(f"Error sending ping: {e}")
                break

except WebSocketDisconnect:
    print("Client disconnected normally")
except Exception as e:
    print(f"WebSocket error: {e}")
finally:
```

```
        print("WebSocket connection closed")


if __name__ == "__main__":
    port = int(os.getenv("SERVER_PORT", "8000"))
    print(f"Starting server on port {port}")
    uvicorn.run(
        "test_server:app",
        host="0.0.0.0",
        port=port,
        reload=True,
        ws_ping_interval=20,  # Send ping every 20 seconds
        ws_ping_timeout=30    # Wait 30 seconds for pong response
    )
```

Test.html

```
<!DOCTYPE html>
<html>
<head>
    <title>WebSocket Test</title>
    <style>
        .error { color: red; }
        .success { color: green; }
    </style>
</head>
<body>
    <h2>WebSocket Test</h2>
    <div id="status">Not started yet...</div>
    <button onclick="sendMessage()">Send Test Message</button>
    <div id="messages"></div>
```

```
<div id="debug"></div>

<script>
    let ws;
    let debugDiv = document.getElementById('debug');
    let statusDiv = document.getElementById('status');

    function addDebug(message) {
        console.log(message);
        debugDiv.innerHTML += `<div>${new Date().toISOString()}: ${message}</div>`;
    }

    function updateStatus(message, isError = false) {
        statusDiv.innerHTML = message;
        statusDiv.className = isError ? 'error' : '';
        addDebug(message);
    }

    function connect() {
        updateStatus('Starting connection process...');

        // Log the URL we're trying to connect to
        const wsProtocol = window.location.protocol === 'https:' ? 'wss:' : 'ws:';
        const wsHost = window.location.hostname;
        const wsPort = window.location.port ? `:${window.location.port}` : '';
        const wsUrl = `${wsProtocol}//${wsHost}${wsPort}/ws`;

        updateStatus(`Attempting to connect to: ${wsUrl}`);
```

```
        try {
            ws = new WebSocket(wsUrl);
            addDebug('WebSocket object created');
        } catch (e) {
            updateStatus(`Failed to create WebSocket: ${e.message}`, true);
            return;
        }

        ws.onopen = function() {
            updateStatus('Connected!', false);
            addDebug('WebSocket opened');
        };

        ws.onmessage = function(event) {
            addDebug(`Received message: ${event.data}`);
            document.getElementById('messages').innerHTML += `<p>Received: ${event.data}</p>`;
        };

        ws.onerror = function(error) {
            updateStatus(`WebSocket error: ${error.message || 'Unknown error'}`, true);
            addDebug(`WebSocket error object: ${JSON.stringify(error)}`);
        };

        ws.onclose = function(event) {
            updateStatus(`Connection closed. Code: ${event.code}, Reason: ${event.reason || 'No reason provided'}`, true);
            addDebug(`Close event - wasClean: ${event.wasClean}`);
            // Try to reconnect after 5 seconds
            setTimeout(connect, 5000);
```

```
        };
    }

    function sendMessage() {
        if (!ws) {
            updateStatus('No WebSocket connection exists', true);
            return;
        }

        addDebug(`Current WebSocket state: ${ws.readyState}`);

        if (ws.readyState === WebSocket.OPEN) {
            ws.send("Hello Server!");
            addDebug('Message sent: Hello Server!');
            document.getElementById('messages').innerHTML += '<p>Sent: Hello Server!</p>';
        } else {
            updateStatus(`WebSocket not open (state: ${ws.readyState})`, true);
        }
    }

    // Start connection when page loads
    window.onload = function() {
        addDebug('Page loaded');
        connect();
    };
  </script>
</body>
</html>
```

Web.config

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <!-- Handlers Configuration -->
    <handlers>

      <add name="httpPlatformHandler" path="*" verb="*" modules="httpPlatformHandler" resourceType="Unspecified" />
    </handlers>

    <!-- Enable WebSocket -->
    <webSocket enabled="true" />

    <!-- Python Application Configuration -->
    <httpPlatform processPath="C:\Python311\python.exe"
                  arguments="test_server.py"
                  stdoutLogEnabled="true"
                  stdoutLogFile=".\logs\stdout.log"
                  startupTimeLimit="60">
      <environmentVariables>
        <environmentVariable name="SERVER_PORT" value="%HTTP_PLATFORM_PORT%" />
        <environmentVariable name="PYTHONUNBUFFERED" value="1" />
      </environmentVariables>
    </httpPlatform>

    <!-- Request Filtering -->
```

```
    <security>
      <requestFiltering allowHighBitCharacters="true">
        <requestLimits maxAllowedContentLength="30000000" />
      </requestFiltering>
    </security>

    <!-- Custom HTTP Protocol Headers -->
    <httpProtocol>
      <customHeaders>
        <add name="Access-Control-Allow-Origin" value="*" />
        <add name="Access-Control-Allow-Methods" value="GET,POST,OPTIONS" />
        <add name="Access-Control-Allow-Headers" value="Content-Type" />
      </customHeaders>
    </httpProtocol>
  </system.webServer>
</configuration>
```