Flex Qml App.qml

```
import QtQuick 2.15
import QtQuick.Layouts 1.15
import QtQuick.Controls 2.15
import "../lib/utils.js" as Utils
import "../lib"
import FlexUi 1.0

Item {
    id: app
    anchors.fill: parent
    property string appId: "closed-loop-app"
    property string serverUrl: ""
    property bool isLoading: false
    property var currentMeasurement: null

    StackView {
        id: stackView
        anchors.fill: parent
        initialItem: serverConfigPageComponent
    }

    Component {
        id: serverConfigPageComponent
        ColumnLayout {
            id: serverConfigLayout
            anchors.centerIn: parent
            width: parent.width * 0.8
            spacing: 10

            Text {
                text: "Server Configuration"
                font.pointSize: 16
                Layout.alignment: Qt.AlignHCenter
                color: FlexDialog.foregroundColor
            }

            FlexTextInput {
                id: urlInput
                navigationRow: 0
                navigationColumn: 0
                Layout.fillWidth: true
                text: "http://10.10.0.12:8000"
                placeholderText: "Enter server URL"
                //flexDialog: FlexDialog
            }

            FlexButton {
                id: connectButton
                navigationRow: 1
                navigationColumn: 0
                text: "Connect"
                Layout.alignment: Qt.AlignHCenter
                onClicked: {
                    app.serverUrl = urlInput.text;
                    stackView.push(mainPageComponent);
                }
            }

            Connections {
                target: FlexDialog
                function onKeyBackPressed() {
                    FlexDialog.closeDialog()
                }
            }
        }
    }

    Component {
        id: mainPageComponent
        Item {
            id: mainPage
            width: parent.width
            height: parent.height

            property int configurationId: -1
            property string targetThickness: ""
            property string line: ""
```

```
            FlexQmlMeasure {
                id: measureItem
                appId: mainPage.configurationId
                onNewMeasurement: function(measurement) {
                    app.currentMeasurement = measurement;
                    if (measurement && measurement.displayStatus && measurement.displayStatus.showThickness) {
                        currentThicknessText.text = "Current: " + measurement.thicknessString + " " + FlexDialog.unit;
                    } else {
                        currentThicknessText.text = "Current: -.-";
                    }
                }
            }

            ColumnLayout {
                anchors.fill: parent
                anchors.margins: 10
                spacing: 10

                Text {
                    text: "Closed Loop Control"
                    font.pointSize: 16
                    Layout.alignment: Qt.AlignHCenter
                    color: FlexDialog.foregroundColor
                }


                Text{
                    text: "application"
                    font.pointSize: 12
                    Layout.alignment: Qt.AlignLeft
                    color: FlexDialog.foregroundColor
                }

                FlexComboBox {
                    id: configurationComboBox
                    navigationRow: 0
                    navigationColumn: 0
                    Layout.fillWidth: true
                    model: ListModel { id: configurationsModel }
                    textRole: "name"
                    //flexDialog: FlexDialog
                    onSelected: function(index, item) {
                        mainPage.configurationId = item.id;
                    }
                }

                Text{
                    text: "line"
                    font.pointSize: 12
                    Layout.alignment: Qt.AlignLeft
                    color: FlexDialog.foregroundColor
                }

                FlexComboBox {
                    id: lineCombobox
                    navigationRow: 1
                    navigationColumn: 0
                    Layout.fillWidth: true
                    model: ListModel { id: lineModel }
                    textRole: "text"
                    //flexDialog: FlexDialog
                    onSelected: function(index, item) {
                        mainPage.line = item.text;
                    }
                }

                Text{
                    text: "target thickness"
                    font.pointSize: 12
                    Layout.alignment: Qt.AlignLeft
                    color: FlexDialog.foregroundColor
                }

                FlexTextInput {
                    id: targetThicknessInput
                    navigationRow: 2
                    navigationColumn: 0
                    Layout.fillWidth: true
                    placeholderText: "Target Thickness"
                    text: mainPage.targetThickness
                    keyboardType: "numeric"
                    //flexDialog: FlexDialog
                    onTextChanged: {
```

```
                mainPage.targetThickness = Number(text);
            }
        }

        Item{
            Layout.fillHeight: true
        }

        Text {
            id: currentThicknessText
            text: "Current: -.-"
            font.pointSize: 22
            font.bold: true
            Layout.alignment: Qt.AlignHCenter
            color: FlexDialog.foregroundColor
        }

        Item{
            Layout.fillHeight: true
        }

        FlexButton {
            id: sendButton
            navigationRow: 3
            navigationColumn: 0
            text: "Send Data"
            Layout.alignment: Qt.AlignHCenter
            onClicked: {
                sendData()
            }
        }
    }

    BusyIndicator {
        anchors.centerIn: parent
        running: app.isLoading
    }

    FlexPopupDialog {
        id: resultPopup
        title: "Server Response"
    }

    Component.onCompleted: {
        loadConfigurations();
        loadTargetThickness();
        loadLines();
    }

    Connections {
        target: FlexDialog
        function onKeyBackPressed() {
            stackView.pop();
        }

        onTriggerPressed: {
            measureItem.measure();
        }
    }

    function loadTargetThickness() {
        if (!app.serverUrl) return;
        app.isLoading = true;
        var xhr = new XMLHttpRequest();
        var url = "http://localhost:9083/proxy?target=" + app.serverUrl + "/target";
        xhr.onreadystatechange = function() {
            if (xhr.readyState === XMLHttpRequest.DONE) {
                app.isLoading = false;
                if (xhr.status === 200) {
                    var response = JSON.parse(xhr.responseText);
                    mainPage.targetThickness = response.target_thickness.toString();
                    targetThicknessInput.text = mainPage.targetThickness;
                } else {
                    console.log("Error fetching target thickness:", xhr.status, xhr.responseText);
                }
            }
        }
        xhr.open("GET", url);
        xhr.send();
    }

    function loadConfigurations() {
        Utils.httpRequest("GET", "/api/v1/configurations").then(function (data) {
```

```
                    console.log("configurations data:", JSON.stringify(data, null, 2));
                    // Map the server response to the format needed by the ListModel.
                    var modelData = data.map(q => ({
                                                "id": q.id,
                                                "name": q.name,
                                                "isMeasureValid": q.isMeasureValid,
                                                "isReadOnly": q.isReadOnly
                                            }));
                    // Append each item to the ListModel for the ComboBox.
                    modelData.forEach(q => {if(q.isMeasureValid){configurationsModel.append(q)}});
                    // Set loading to false now that data is loaded.
                    app.isLoading = false;
                    // Set the initial selection in the ComboBox.
                    if (configurationsModel.count > 0) {
                        configurationComboBox.currentIndex = 0
                        mainPage.configurationId = configurationsModel.get(0).id;
                    }
                });
            }

            function loadLines(){
                 var url = "http://localhost:9083/proxy?target=" + app.serverUrl + "/lines";
                Utils.httpRequest("GET", url).then(function (data) {
                    lineModel.clear();
                    if (Array.isArray(data)) {
                        data.forEach(function(line) {
                            lineModel.append({ "text": line });
                        });
                    }

                    if(lineModel.count > 0){
                        lineCombobox.currentIndex = 0;
                        mainPage.line = lineModel.get(0).text;
                    }
                });
            }

            function sendData() {
                if (!app.serverUrl || !app.currentMeasurement) return;

                var current_thickness = app.currentMeasurement.thickness;
                var target_thickness = parseFloat(mainPage.targetThickness);
                var current_line = mainPage.line

                var data = {
                    "target_thickness": target_thickness,
                    "current_thickness": current_thickness,
                    "selected_line": current_line
                };

                app.isLoading = true;
                var url = "http://localhost:9083/proxy?target=" + app.serverUrl + "/data";
                Utils.httpRequest("POST", url, data).then(function (response) {
                    app.isLoading = false;
                    console.log("Data sent successfully");
                    resultPopup.text = "Difference: " + response.difference.toFixed(2) + " " + FlexDialog.unit + "\n" + "Message:
" + response.message;
                    resultPopup.open();
                }).catch(function (error) {
                    app.isLoading = false;
                    console.log("Error sending data:", error.status, error.error);
                    resultPopup.text = "Error: " + error.error;
                    resultPopup.open();
                });
            }
        }
    }
}
```

Python server.py example

```
# Import necessary modules
from http.server import BaseHTTPRequestHandler, HTTPServer
import json

# Define the request handler class
class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):
    # Handle GET requests
```

```
    def do_GET(self):
        # If the request path is /target, send the target thickness
        if self.path.startswith('/target'):
            self.send_response(200)
            self.send_header('Content-type', 'application/json')
            self.end_headers()
            # Define the target thickness
            response = {'target_thickness': 75.0}
            # Send the response as JSON
            self.wfile.write(json.dumps(response).encode('utf-8'))
        elif self.path.startswith('/lines'):
            self.send_response(200)
            self.send_header('Content-type', 'application/json')
            self.end_headers()
            # Define the target thickness
            response = ['line 1','line A','line B','line 123','no line']
            # Send the response as JSON
            self.wfile.write(json.dumps(response).encode('utf-8'))
        # Otherwise, send a 404 Not Found response
        else:
            self.send_response(404)
            self.end_headers()

    # Handle POST requests
    def do_POST(self):
        # If the request path is /data, process the received data
        if self.path.startswith('/data'):
            # Get the length of the POST data
            content_length = int(self.headers['Content-Length'])
            # Read the POST data
            post_data = self.rfile.read(content_length)
            # Parse the JSON data
            data = json.loads(post_data)

            # Print the received data
            print("Received data from Flex device:")
            print(f"  Target Thickness: {data.get('target_thickness')}")
            print(f"  Current Thickness: {data.get('current_thickness')}")
            print(f"  Current Line: {data.get('selected_line')}")

            # Calculate the difference between target and current thickness
            target = data.get('target_thickness', 0.0)
            current = data.get('current_thickness', 0.0)
            difference = current - target

            # Send a success response
            self.send_response(200)
            self.send_header('Content-type', 'application/json')
            self.end_headers()
            # Create the response dictionary including the difference
            response = {
                'status': 'success',
                'message': 'Data received adjusting Coating line',
                'difference': difference
            }
            # Send the response as JSON
            self.wfile.write(json.dumps(response).encode('utf-8'))
        # Otherwise, send a 404 Not Found response
        else:
            self.send_response(404)
            self.end_headers()

    # Handle OPTIONS requests for CORS
    def do_OPTIONS(self):
        self.send_response(200)
        self.send_header('Access-Control-Allow-Origin', '*')
        self.send_header('Access-Control-Allow-Methods', 'GET, POST, OPTIONS')
        self.send_header('Access-Control-Allow-Headers', 'Content-Type')
        self.end_headers()

# Define the function to run the server
def run(server_class=HTTPServer, handler_class=SimpleHTTPRequestHandler, port=8000):
    server_address = ('', port)
    httpd = server_class(server_address, handler_class)
    print(f'Starting httpd server on port {port}...')
    print('Run this server and enter http://<your-ip>:8000 in the app.')
    httpd.serve_forever()

# Run the server if the script is executed directly
if __name__ == '__main__':
    run()
```