**Tạo máy chủ websocket bằng node.js**

Bước 1: Cài đặt Node.js Trước tiên, bạn cần cài đặt Node.js trên máy tính của mình. Truy cập trang web chính thức của Node.js tại https://nodejs.org và tải xuống phiên bản phù hợp với hệ điều hành của bạn. Sau khi tải xuống, hãy cài đặt Node.js bằng cách chạy tệp cài đặt bạn đã tải xuống.

Bước 2: Tạo thư mục dự án Tạo một thư mục trống trên máy tính của bạn để chứa dự án Node.js của bạn.

Bước 3: Khởi tạo dự án Node.js Mở terminal và điều hướng đến thư mục dự án bạn đã tạo. Sau đó, hãy chạy lệnh sau để khởi động dự án Node.js:

npm init -y

Lệnh trên sẽ tạo ra một tệp package.json trong thư mục dự án của bạn.

Bước 4: Cài đặt các gói phụ thuộc Theo gói tiếp theo, bạn cần cài đặt các gói phụ thuộc cần thiết cho dự án của bạn. Trong trường hợp này, chúng ta cần cài đặt expressvà socket.io. Run command after to install the two packages:

npm install express socket.io

Bước 5: Tạo tệp server.js và sao chép mã mẫu Trong thư mục dự án của bạn, tạo một tệp mới có tên là server.js(hoặc bất kỳ tên nào khác mà bạn muốn). Mở tệp này bằng trình chỉnh sửa văn bản đã chỉnh sửa và sao chép đoạn mã mẫu cho máy chủ Node.js

const fs = require('fs');

function writeToLog(message) {

const logMessage = `${new Date().toISOString()}: ${message}\n`;

fs.appendFile('server.log', logMessage, (error) => {

if (error) {

console.error('Error writing to log file:', error);

}

});

}

const express = require('express');

const app = express();

const axios = require('axios');

const server = require('http').createServer(app);

// Lắng nghe các yêu cầu HTTP trên cổng đã chỉ định

server.listen(8080, () => {

console.log('HTTP server is running on port 8080');

});

// Endpoint handler

app.get('/data', async (req, res) => {

try {

const chatSamples = ['Xin Chao!', 'Hello', 'Hi Guy!'];

res.json({ chat\_samples: chatSamples });

} catch (error) {

console.error('Error fetching data:', error);

res.status(500).json({ error: 'Error fetching data' });

console.log('loi gui api');

}

});

const WebSocket = require('ws');

const wss = new WebSocket.Server({ server });

wss.on('connection', (ws) => {

console.log('A client connected.');

writeToLog('A client connected.');

// Handle new message event from client

ws.on('message', (message) => {

console.log('New message:', message);

// Broadcast the new message to all other clients

wss.clients.forEach((client) => {

if (client !== ws && client.readyState === WebSocket.OPEN) {

client.send(message);

console.log('da gui message cho cac client');

writeToLog('A client send message.');

}

});

});

// Handle client disconnect event

ws.on('close', () => {

console.log('A client disconnected.');

writeToLog('A client disconnected.');

});

});

wss.on('listening', () => {

const { address, port } = wss.address();

console.log(`Server is running on ${address}:${port}`);

});

Bước 6: Chạy server Quay lại terminal, đảm bảo bạn đang đứng trong project project và chạy câu lệnh sau để chạy server:

node server.js

Nếu bạn thấy thông báo "Máy chủ đang chạy trên cổng 8080" xuất hiện trong thiết bị đầu cuối, điều đó có nghĩa là máy chủ đã được khởi động thành công.

**Tạo 1 project flutter**

Import websocket

web\_socket\_channel: ^2.0.0

bỏ đoạn code này vào trong file pubspec.yaml như hình dưới

A screenshot of a computer

Description automatically generated

Mở file AndroidMainfest.xml trong android/app/src/main

<uses-permission android:name="android.permission.INTERNET" />

Pass đoạn code này vào trước <application như hình dưới

A screen shot of a computer

Description automatically generated

Pass code này vào main.dart

import 'dart:convert';  
import 'dart:typed\_data';  
import 'package:http/http.dart' as http;  
import 'package:flutter/material.dart';  
import 'package:web\_socket\_channel/io.dart';  
import 'package:web\_socket\_channel/web\_socket\_channel.dart';  
void main() => runApp(ChatApp());  
  
  
class ChatApp extends StatefulWidget {  
 @override  
 \_ChatAppState createState() => \_ChatAppState();  
}  
  
class \_ChatAppState extends State<ChatApp> {  
 final WebSocketChannel channel = IOWebSocketChannel.connect('ws://192.168.0.107:8080');  
 final TextEditingController \_textController = TextEditingController();  
 List<String> messages = []; // Danh sách lưu trữ các tin nhắn  
 String clientName = ''; // Tên client  
 List<String> chatSamples = [];  
 List<String> options = ['Option 1', 'Option 2', 'Option 3'];  
 String selectedChatSample = '';  
 @override  
 void dispose() {  
 channel.sink.close();  
 super.dispose();  
 }  
  
 @override  
 void initState() {  
 super.initState();  
  
 channel.stream.listen((dynamic data) {  
 if (data is String) {  
 setState(() {  
 messages.add(data);  
 });  
 } else if (data is Uint8List) {  
 String message = utf8.decode(data);  
 setState(() {  
 messages.add(message);  
 });  
 } else {  
 print('Received unexpected data type: ${data.runtimeType}');  
 }  
 });  
 fetchData();  
  
 }  
  
 void sendMessage(String message) {  
 final formattedMessage = '$clientName: $message'; // Thêm tên client vào tin nhắn  
 channel.sink.add(formattedMessage);  
 // Thêm tin nhắn mới vào danh sách và cập nhật giao diện  
 setState(() {  
 messages.add(formattedMessage);  
 });  
 }  
  
 void fetchData() async {  
 final url = Uri.*parse*('http://192.168.0.107:8080/data');  
 final response = await http.get(url);  
  
 if (response.statusCode == 200) {  
 final data = json.decode(response.body);  
 List<String> chatSamples = List<String>.from(data['chat\_samples']);  
 setState(() {  
 this.chatSamples = chatSamples;  
 selectedChatSample = this.chatSamples[0];  
 });  
 } else {  
 print('Lỗi khi gọi API: ${response.statusCode}');  
 }  
 }  
   
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 title: 'Chat App',  
 home: Scaffold(  
 appBar: AppBar(  
 title: const Text('Chat App'),  
 ),  
 body: Column(  
 children: [  
 Expanded(  
 child: ListView.builder(  
 itemCount: messages.length,  
 itemBuilder: (context, index) {  
 return ListTile(  
 title: Text(messages[index]),  
 );  
 },  
 reverse: true, // Hiển thị tin nhắn mới nhất lên đầu danh sách  
 ),  
 ),  
  
  
  
 Container(  
 padding: const EdgeInsets.all(8.0),  
 child: Row(  
 children: [  
 Expanded(  
 child: TextField(  
 controller: \_textController,  
 decoration: InputDecoration(  
 hintText: 'Enter a message',  
 ),  
 ),  
 ),  
 IconButton(  
 icon: Icon(Icons.*send*),  
 onPressed: () {  
 final message = \_textController.text;  
 sendMessage(message);  
 \_textController.clear();  
 // Hiển thị thông báo là tin nhắn đã được gửi  
 ScaffoldMessenger.*of*(context).showSnackBar(  
 SnackBar(content: Text('Sent message: $message')),  
 );  
 },  
 ),  
 SizedBox(width: 10),  
 DropdownButton<String>(  
 value: selectedChatSample,  
 onChanged: (String? newValue) {  
 setState(() {  
 selectedChatSample = newValue!;  
 \_textController.text = newValue!;  
 });  
 },  
 items: chatSamples.map<DropdownMenuItem<String>>((String value) {  
 return DropdownMenuItem<String>(  
 value: value,  
 child: Text(value),  
 );  
 }).toList(),  
 ),  
 ],  
 ),  
 ),  
  
 Container(  
 padding: const EdgeInsets.all(8.0),  
 child: Row(  
 children: [  
 Expanded(  
 child: TextField(  
 onChanged: (value) {  
 setState(() {  
 clientName = value;  
 });  
 },  
 decoration: InputDecoration(  
 hintText: 'Enter your name',  
 ),  
 ),  
 ),  
 IconButton(  
 icon: Icon(Icons.*done*),  
 onPressed: () {  
 // Hiển thị thông báo là tên client đã được nhập  
 ScaffoldMessenger.*of*(context).showSnackBar(  
 SnackBar(content: Text('Client name: $clientName')),  
 );  
 },  
 ),  
 ],  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}