**CHAT APPLICATION**

This document have all informations and notes concernant the chat app (UML diagrams ,etc ..) .Any thing that you think it may be helpful don’t hesitate to share it here .

**Sprint  1 (5j):**

\_ Graphic design

\_ UML diagrams

\_ Initialization of the project

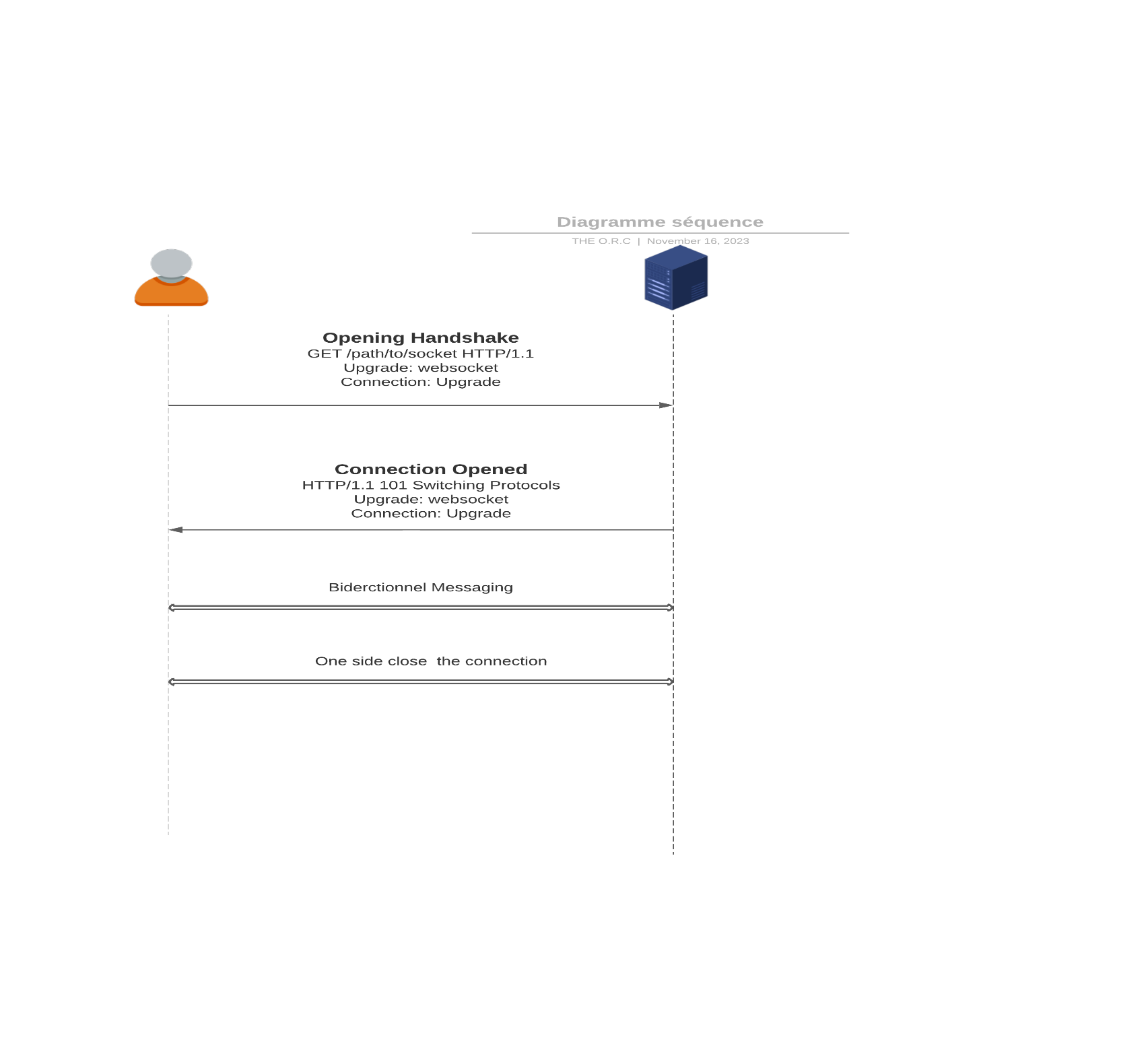
Deadline of project : **30 / 11 / 2023**

WebSocket Protocol est un protocole de communication en temps réel entre des clients (généralement des navigateurs web) et des serveurs sur une seule connexion durée.

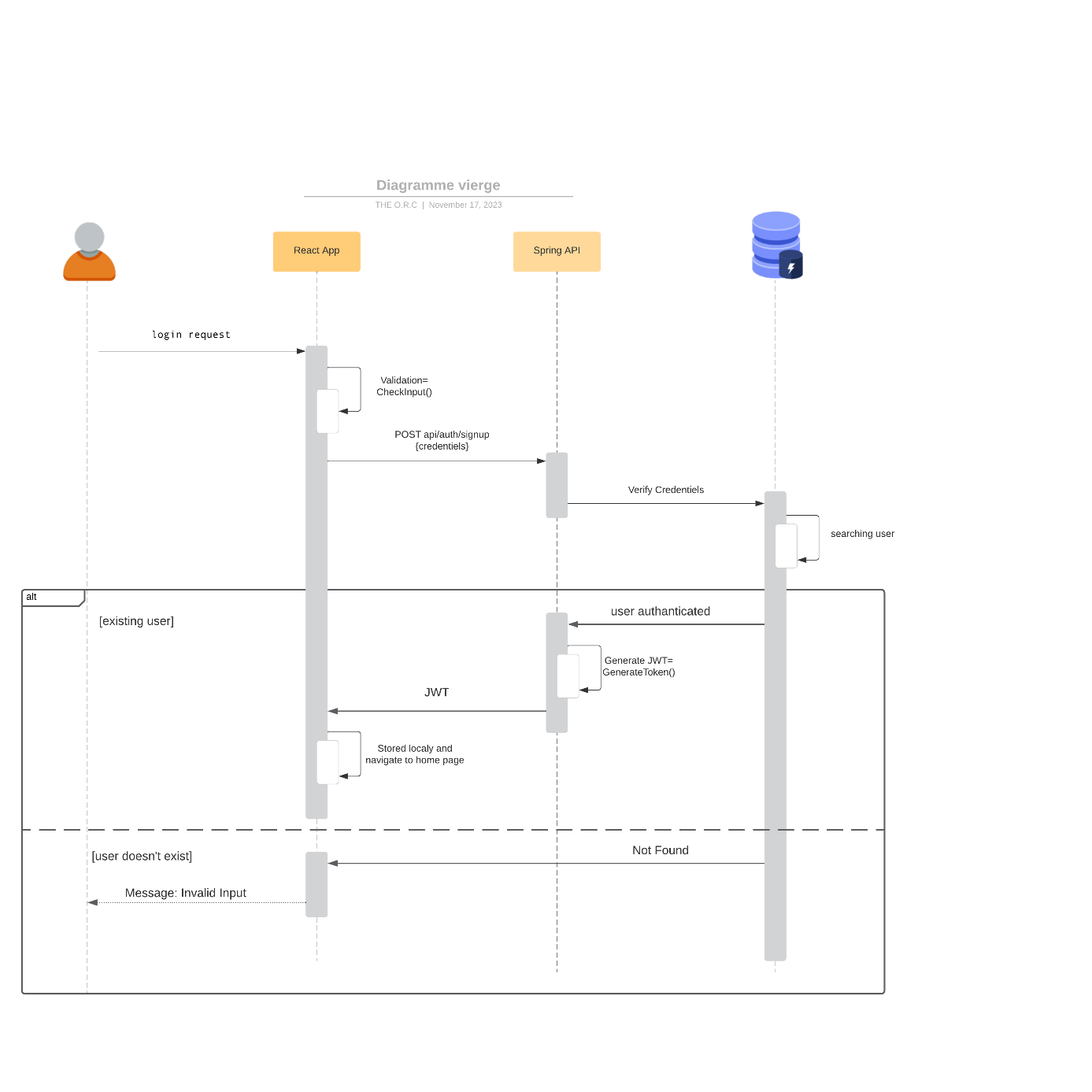
WebSocket permet une communication full-duplex, ce qui signifie que le client et le serveur peuvent s'envoyer simultanément des messages sans attendre de réponse . Cela diffère de l'HTTP traditionnel, qui est généralement basé sur des demandes et des réponses.

**What is the Websocket protocol :**

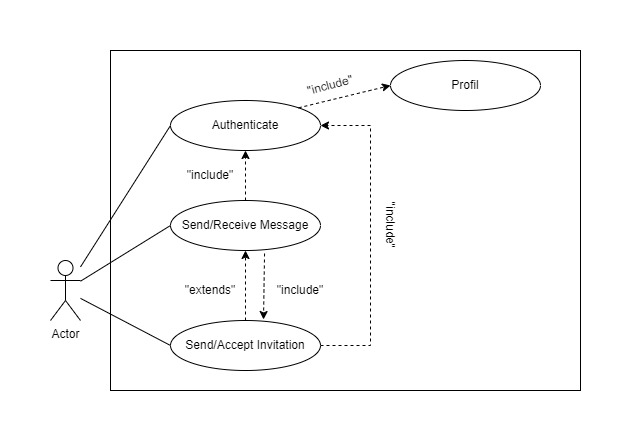




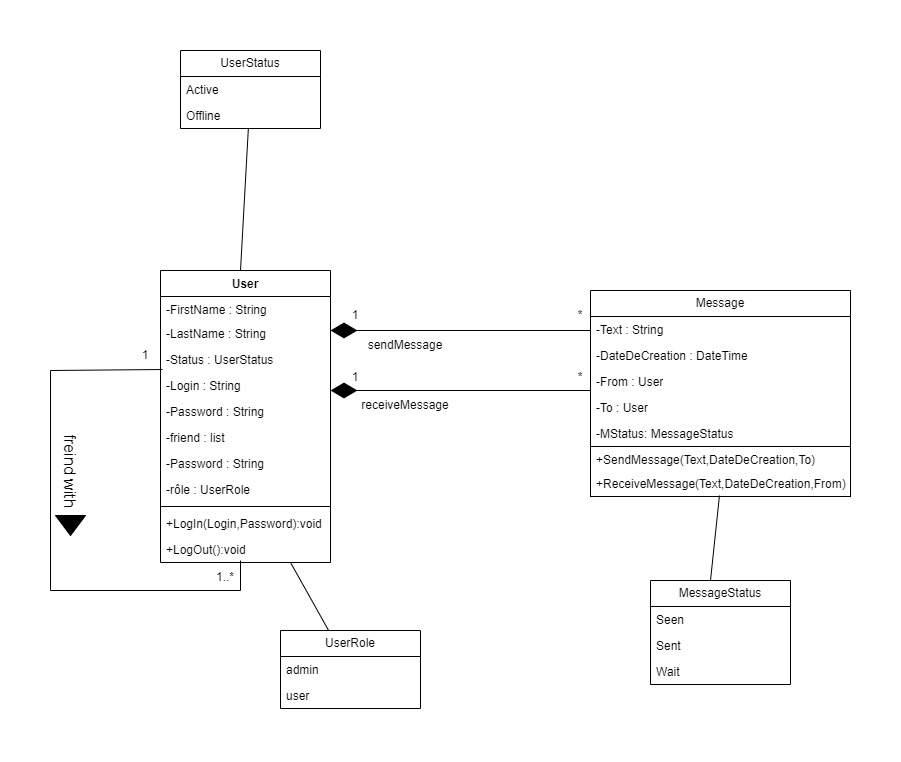
* Diagramme de Séquence pour l’établissement du connexion web socket entre client et serveur :
* Diagramme de séquence pour Sign in :



* Diagramme de cas d’utilisation de l’application de chat



* Diagramme de classe de l’application de chat



Function to strore user data in JSON file

UserData :

{

"users": [

{

"id": 1,

"firstName": "John",

"lastName": "Doe",

"email": "john.doe@example.com",

"password": "password",

"createdAt": "2023-01-01T12:00:00Z"

},

{

"id": 2,

"firstName": "Jane",

"lastName": "Smith",

"email": "jane.smith@example.com",

"password": "hashed\_password",

"createdAt": "2023-01-02T14:30:00Z"

},

]

}

Handle Function :

const [firstName, setFirstName] = useState('');

const [lastName, setLastName] = useState('');

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const handleFirstNameChange = (e) => {

setFirstName(e.target.value);

};

const handleLastNameChange = (e) => {

setLastName(e.target.value);

};

const handleEmailChange = (e) => {

setEmail(e.target.value);

};

const handlePasswordChange = (e) => {

setPassword(e.target.value);

};

const handleSignUp = async (e) => {

e.preventDefault();

try {

const response = await fetch(http://localhost:3001/api/register, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ firstName, lastName, email, password }),

});

if (response.ok) {

console.log('User registered successfully');

} else {

console.error('Registration failed');

}

} catch (error) {

console.error('Error during registration:', error);

}

setFirstName('');

setLastName('');

setEmail('');

setPassword('');

};

Chat json structure :

{

"messages": [

{ "sender": "User1", "message": "Hello, how are you?" },

{ "sender": "User2", "message": "I'm good, thank you! How about you?" },

{ "sender": "User1", "message": "I'm doing well too. What are you up to?" },

{ "sender": "User2", "message": "Just working on some projects. How about you?" },

]

}

Handle Code

import React, { useState, useEffect } from 'react';

const ChatApp = () => {

const [messages, setMessages] = useState([]);

const [newMessage, setNewMessage] = useState('');

useEffect(() => {

// Load chat data from the JSON file when the component mounts

loadChatData();

}, []);

const loadChatData = async () => {

try {

// Fetch chat data from the server or local file (replace with your endpoint or file path)

const response = await fetch('/api/chatData');

if (response.ok) {

const chatData = await response.json();

setMessages(chatData.messages || []);

}

} catch (error) {

console.error('Error loading chat data:', error);

}

};

const handleInputChange = (e) => {

setNewMessage(e.target.value);

};

const handleSendMessage = async () => {

// Add the new message to the messages array

const updatedMessages = [...messages, { sender: 'User1', message: newMessage }];

setMessages(updatedMessages);

// Save the updated messages to the JSON file or server (replace with your endpoint)

try {

const response = await fetch('/api/saveChatData', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ messages: updatedMessages }),

});

if (!response.ok) {

console.error('Failed to save chat data');

}

} catch (error) {

console.error('Error saving chat data:', error);

}

// Clear the input field

setNewMessage('');

};

return (

<div>

<div>

{messages.map((msg, index) => (

<div key={index}>

<strong>{msg.sender}:</strong> {msg.message}

</div>

))}

</div>

<div>

<input type="text" value={newMessage} onChange={handleInputChange} />

<button onClick={handleSendMessage}>Send</button>

</div>

</div>

);

};

export default ChatApp;