# Client JWT Authentication Setup

This guide describes how to implement a minimal client-side JWT authentication workflow using TypeScript and SignalR.  
  
All code should be placed inside the 'src' folder, for example in a file like 'auth-client.ts'.

## 1. Install Dependencies (if not installed)

Install the required packages using npm:

npm install @microsoft/signalr

## 2. Create the Client File

Create a file called 'auth-client.ts' in the 'src' folder with the following content:

import { HubConnectionBuilder } from "@microsoft/signalr";  
  
const credentials = {  
 username: "user",  
 password: "pass"  
};  
  
async function authenticate(username: string, password: string): Promise<string> {  
 const response = await fetch("http://localhost:5000/api/auth/authenticate", {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ username, password }),  
 });  
  
 if (!response.ok) {  
 const errText = await response.text();  
 throw new Error(`Authentication failed: ${response.status} ${errText}`);  
 }  
  
 const json = (await response.json()) as { token: string };  
 return json.token;  
}  
  
(async () => {  
 try {  
 const token = await authenticate(credentials.username, credentials.password);  
  
 const connection = new HubConnectionBuilder()  
 .withUrl("http://localhost:5000/hub/cache", {  
 accessTokenFactory: () => token,  
 })  
 .build();  
  
 connection.onclose(err => {  
 console.error("Connection closed:", err);  
 });  
  
 await connection.start();  
 console.log("Connected with token:", token);  
  
 } catch (err: any) {  
 console.error("Error:", err.message);  
 }  
})();

## 3. Compile and Run

Compile the TypeScript code:

npx tsc

Then run the resulting JavaScript with Node.js:

node src/auth-client.js

## Note

Ensure your backend server is running on http://localhost:5000 and is configured to return a valid JWT from the /api/auth/authenticate endpoint.