UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

# COS 216 Homework Assignment

- Date Issued: **7 March 2022**
- Date Due: **23 May 2022** before **08:00**
- Submission Procedure: **Upload to ClickUP. An assignment upload will be made available.**
- Submission Format: **zip or tar + gzip archive**
- This assignment consists of **3 tasks** for a total of **115 marks**.

## 1 Introduction

During this homework assignment you will be implementing Web Sockets. This assignment is **based on all the practicals for this module**.

You will be creating a webpage that allows multiple users to discuss a news article in an interactive comment section.

On completion, your assignment must provide the following functionality:

- A local NodeJS server polling your PHP API from Wheatley.
- An HTML web page / web site where you would connect to your NodeJS server via a socket to get comment data for a news article on the client side.
- The ability to have multiple connections to the local NodeJS server using sockets, which is how chat messages will be posted.
- Chat functionality for a specific article that has up-to-date data.
- The ability to quote messages in other messages (Reply Functionality).

## 2 Constraints

1. You must complete this assignment individually.
2. You may ask the Teaching Assistants for help but they will not be able to give you the solutions.
3. You must produce all of the source files yourself; you may not use any tool to generate source files or fragments thereof automatically.
4. All written code should contain comments including your name, surname and student number at the top of each file.
5. You assignment must be programmed in NodeJS, HTML, JS, CSS and PHP.

## 3 Submission Instructions

You are required to upload all your source files in an archive, either zipped or tar gzipped, to **clickUP**. No late submissions will be accepted, so make sure you upload in good time.

## 4 Resources

**NodeJS** - `https://www.w3schools.com/nodejs/`
`https://www.tutorialspoint.com/nodejs/index.htm`

**ExpressJS** - `https://www.tutorialspoint.com/expressjs/index.htm`
`https://expressjs.com/`
`https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs`

**Web Sockets** - `https://developer.mozilla.org/en-US/docs/Web/API/WebSocket`
`https://en.wikipedia.org/wiki/WebSocket`
`https://github.com/websockets/ws`
`https://socket.io/`
`https://www.npmjs.com/package/websocket`

**Unix Timestamps** - `https://websiteseochecker.com/blog/what-is-timestamp/`

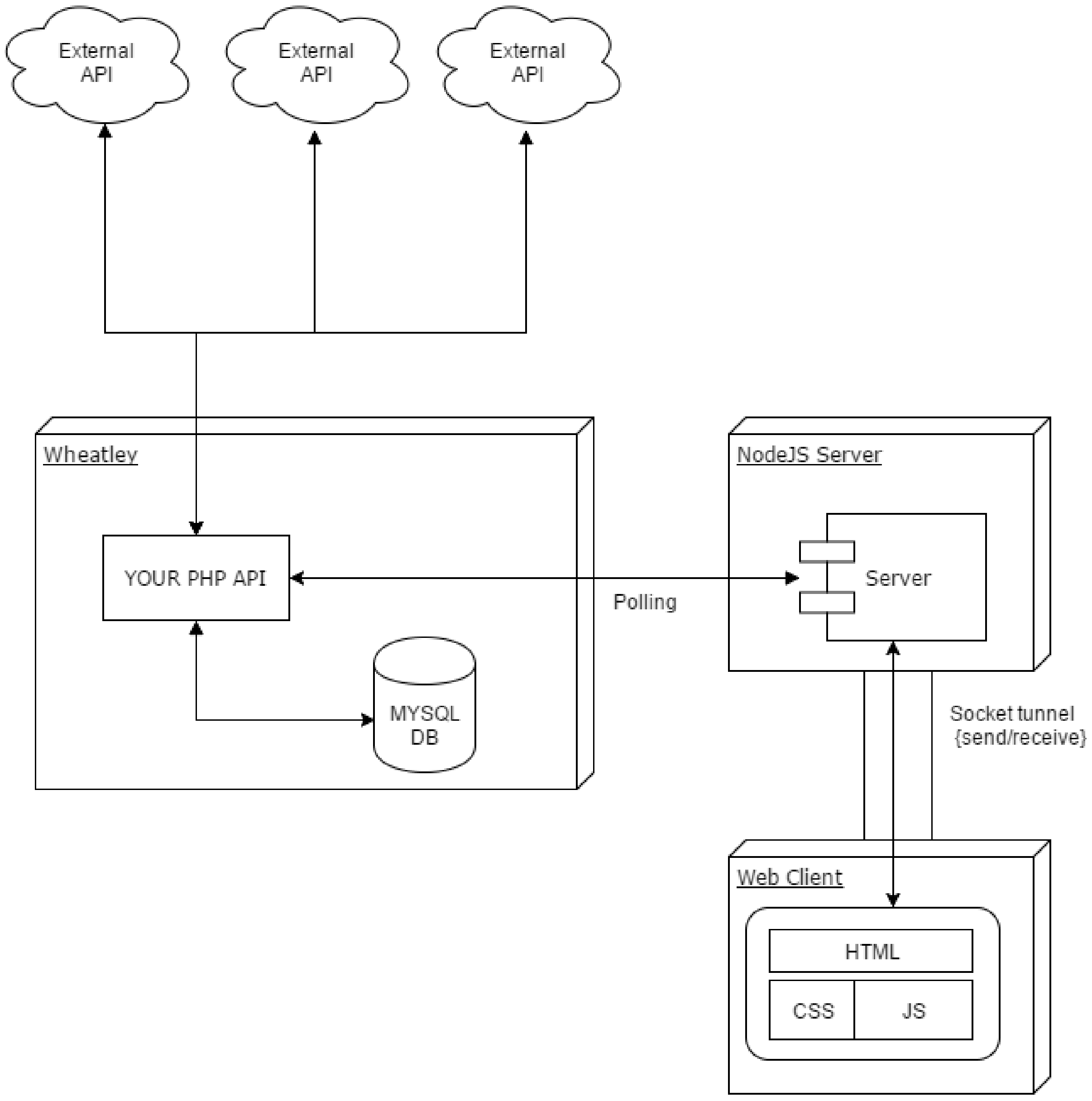
**Toastr** - `https://github.com/CodeSeven/toastr`

## 5 Rubric for marking

| **Task 1 - 'Chat' API type** | **20** |
|---|---|
| Ability to save a message/comment | 10 |
| Ability to retrieve comments in correct order | 10 |
| **Task 2 - Multi-threaded server** | **55** |
| Port can be chosen at run-time | 3 |
| Server uses multi-threading to accept multiple clients simultaneously | 10 |
| 'LIST' command is implemented and fully functioning | 5 |
| 'KILL' command is implemented and fully functioning | 10 |
| 'QUIT' command is implemented and fully functioning | 10 |
| If connection is closed on the client the server closes the socket | 5 |
| Server syncs after X seconds | 2 |
| Server only pushes changes for consequent interactions on the socket | 10 |
| **Task 3 - Web Client** | **35** |
| The client displays notifications if socket disconnects | 2 |
| The client tries to re-establish a broken/closed connection | 3 |
| The client clearly shows change in feed/data | 5 |
| The client shows all the data from the PHP API | 15 |
| The client provides quoting functionality | 10 |
| **Security** | **5** |
| **Upload** | |
| **Not uploaded to clickUP** | **-115** |
| **Bonus** | **5** |
| **Total** | **115** |

## 6 Overview

To understand the flow of how this all works please consult the diagram below.



## 7 Assignment Instructions

### Task 1: 'Chat' API type ............................................................. (20 marks)

In Practical 3, you will have to build your own PHP API which will provide certain functionality types, one such type is 'chat', which you will be implementing in this assignment.

This feature allows access to all messages of the interactive comments section of an article.

You should be able to **store and retrieve messages**, so make sure you structure your requests and modify your database accordingly. You only need to use one news article to demonstrate this, although having a generic solution may earn you bonus marks. The API should allow users to get and send messages.
You should familiarise yourself with the assignment well, and then determine exactly what you need to implement in you API.

**Note: Only registered users can leave messages.**

### Task 2: Multi-threaded NodeJS server ................................................. (55 marks)

For this task you are required to create a **multi-threaded** socket server **(on localhost)**. The server must be coded in **NodeJS locally and not off wheatley**. You are allowed to use libraries for this.
**Note:** Wheatley is a web server that follows the LAMP stack and installing other applications on it brings security issues. You must therefore use localhost for this since the NodeJS server needs to run on an open port and it would be difficult to allow the server to be run on Wheatley as the chances of students using the same port would be high. Also, this is done to avoid some students from intentionally and unintentionally performing port blocking. This is also not possible to do due to UP's firewall.

The server must have the following functionality:

- Be able to specify a port that the server will listen on at run-time (create the socket), using input arguments or prompts.
- Accept multiple client connections and interact with the clients concurrently through sockets.
- The server must utilize functionality of the PHP API you developed/will develop from the practicals.
- Since Wheatley is password protected, you will need to include your login details in the URL as follows:

  username:password@wheatley.cs.up.ac.za/uXXXXXXXX/path_to_api

  It is your responsibility to keep your login details as safe as possible. It is recommended that you store your username and password in a global variable and use that variable throughout. Alternatively, you can secure your details through other means. **Bonus marks may be given depending on how secure your solution is.**
  **NB:** Ensure that you do not submit your code with your passwords included in your clickUP submission.
- The PHP API needs to make use of an **API key** which the NodeJS server would need to authenticate itself to the API (you will implement this in Practical 3). This means that your PHP API would need to generate a random key (e.g 8 alpha-numeric random characters) which you would add to each request you make to the PHP API authenticating the request. More details will be given in Practical 3.
- In order to get the latest data, the server should **sync** to the PHP API you created after every X seconds. Ensure that you choose an appropriate value.
- The server should be smart enough to **only push data that has changed** to the clients instead of pushing all the data every time the PHP API is synced. **Hint:** You can make use of timestamps i.e. your server should keep track of the latest timestamp of the data sent to your PHP API and the clients and, if any data is new, a response will be sent. (Have a look at *Unix Timestamps* linked under Resources)
- The server needs to account for lost sockets (when a socket was created and the client [HTML web page] has closed/disconnected). These lost sockets should be closed in order to allow for new connections.
- The server should have a `LIST` command to list all the connections it currently has.
- The server should have a `KILL` command to close a connection based on some ID.

- The server should have a `QUIT` command that sends a broadcast to all connections notifying that the server will now go off-line, thereafter closing all connections.

To test the functionality of your server you may use `https://www.websocket.org/echo.html` as the client before proceeding to Task 3.

## Task 3: Web Client ........................................ (40 marks)

For this task you are required to develop a web page / web site that will interact with your server (that runs on your local machine [localhost]) you wrote for Task 2. The web client must be implemented in HTML, CSS and JS using Web Sockets. **The client should also be on localhost**.
**Note:** You may use any client side library of your choice (e.g. WebSockets, Socket.io, etc.)

The client must have the following functionality:

- The client should have login functionality (no registration is needed since only registered users can post messages).
- The client should have some way of displaying a live chat in relation to a given news article.
- The client should notify the user when the server has disconnected. This can be done via text, icons, pop-ups, etc. (Have a look at the *toastr* library linked under Resources)
- The client should notify the user when a socket closes and should attempt to re-establish the connection. This can be done via text, icons, pop-ups, etc.
- The client should clearly indicate/show when new data has been received. This can be done by simply adding a small notification and then highlighting a new message in some way, by changing the text colour or showing an animation etc.
- If two tabs are opened on the live chat of the same article, if one tab posts a message it should be visible on the other tab as well. Hence, you need to sync the data using sockets and occasionally send the data to the API for more persistent tracking.
- In addition to basic chat, a chat user should also have the ability to quote another message. The resulting message should clearly indicate which message it is quoting. The quoting should, of course, be visible on all tabs that have the chat open.
- To keep cluttering low, only the 20 most recent messages should be displayed for an article.

**Note: Bonus marks may be given to students who complete all tasks of the assignment and add extra functionality to them.**