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

COS 216 Homework Assignment

- Date Issued: **24 February 2025**
- Date Due: **19 May 2025** before **11:00**
- 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 **130 marks**.

NB: Please read through the entire specification before asking questions on Discord/Email, as there is a high likelihood you question may be answered later in the specification.

1 Introduction

During this homework assignment you will be implementing Web Sockets. You will be creating a real-time courier system where a remotely-controlled drone is used to deliver packages to customers. This system will contain a map and use GPS coordinates (longitude and latitude in Decimal Degrees) to show the live location of the drone and the customers. More details will be provided below.

On completion, your assignment must contain the following:

- A PHP API hosted off Wheatley that pulls data from a MYSQL Database.
- A **local** NodeJS socket server polling your PHP API on Wheatley.
- An Angular web client that connects to your NodeJS socket server that will act as the frontend.

2 Constraints

1. You may complete this assignment individually or in groups of 2.
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. **This includes ChatGPT, Gemini etc!**
4. All written code should contain comments including your names, surnames and student numbers at the top of each file.
5. You assignment must be programmed in NodeJS, HTML, JS, CSS and PHP. (Bootstrap is allowed).
6. You are required to use Angular for the frontend client.
7. The API and Database should be hosted off Wheatley, the NodeJS server should be hosted locally.

3 Submission Instructions

You are required to upload all your source files and a ReadME in an archive, either zipped or tar gzipped, to **clickUP**. No late submissions will be accepted (See due date and time at the top of the document), so make sure you upload in good time. Only one member is required to upload the files for the group.

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

Leaflet https://leafletjs.com/index.html

Angular https://v17.angular.io/docs

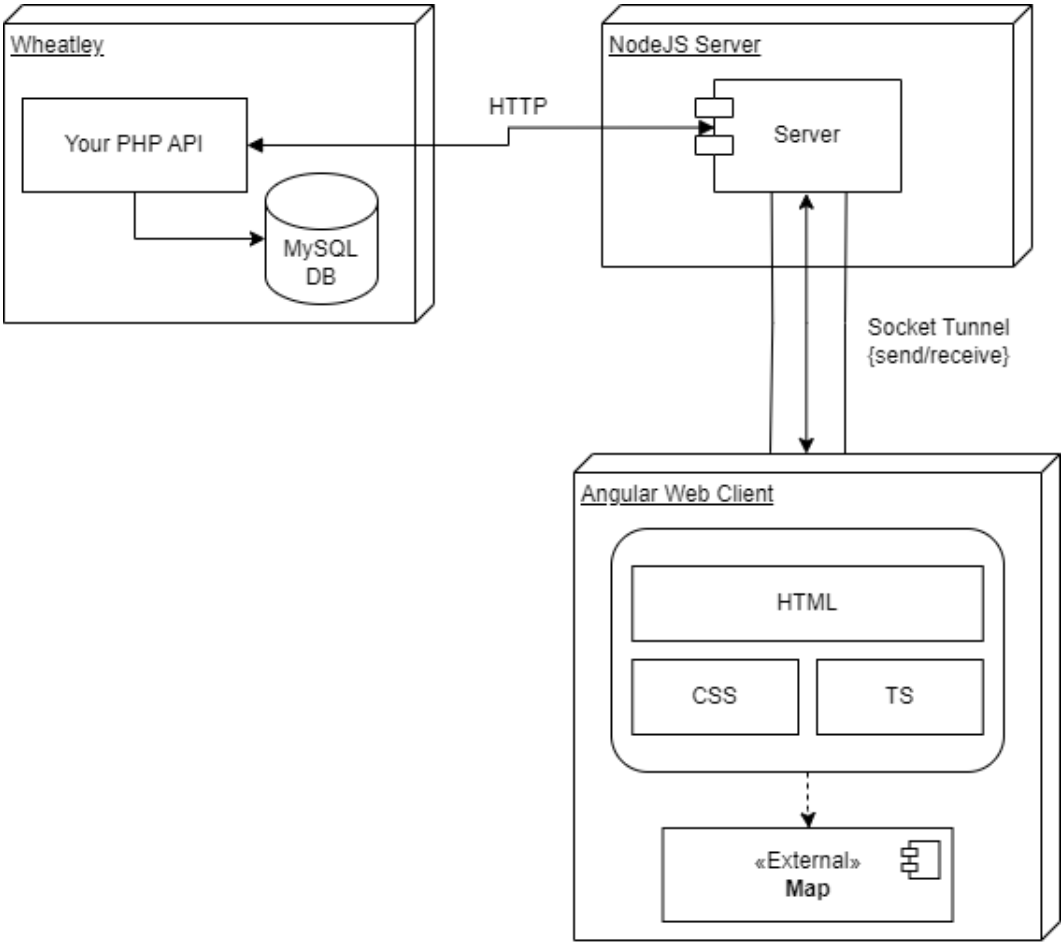
DOM Manipulation https://www.w3schools.com/js/js_htmldom.asp

5 Rubric for marking

Task 1 - PHP API & DB	**45**
MySQL DB	5
Login endpoint	5
CreateOrder endpoint	10
CreateDrone endpoint	5
UpdateOrder endpoint	5
UpdateDrone endpoint	5
GetAllOrders endpoint	5
GetAllDrones endpoint	5
Task 2 - Multi-User server	**58**
Port can be chosen at run-time	3
Reserved Ports cannot be chosen	2
Server can accept multiple clients simultaneously	5
'CURRENTLY_DELIVERING' command is implemented and fully functioning	5
'KILL' command is implemented and fully functioning	5
'QUIT' command is implemented and fully functioning	4
'DRONE_STATUS' command is implemented and fully functioning	5
If connection is closed on the client the server closes the socket	5
Server can send through Order Details to the client using sockets and the API	5
Server can send through Drone Details to the client using sockets and the API	3
Server implements the flow of the Courier System correctly	10
Server updates database	4
ReadMe explanation, Update Everytime vs Update Interval	2
Task 3 - Web Client	**52**
The client can connect to a server through a socket	3
The client allows the user to login	3
The client allows customers to order and view the drone	3
The client is zoomed into Hatfield by default and clearly shows distinct markers for elements on the map	5
The client makes the drone move on the user side on cue from the server	5
Error Messages(Socket disconnected, Location out of range, Delivery may be delayed etc.)	8
Implement the Dust Devils feature	10
The ability to implement the Courier System correctly	15
Security	**5**
Upload **Not uploaded to clickUP**	**-160**
Bonus	**15**
Total	**160**

6 Overview

To understand the architecture of this assignment, please consult the diagram below.



7 Description of Courier System

In this assignment, you are implementing a courier system for a small tech-startup based in Hatfield. This company aims to revolutionize the e-commerce industry by using remote-controlled drones to drop packages to customers. Due to technical constraints, they only operate within a 5km radius of their headquarters (HQ).

Note: The term 'Courier' can refer to a company or employee of a company that transports commercial packages and documents. Thus to prevent ambiguity, the courier **system** will have a courier (individual) who operates the drone.

This system contains two types of users:

- Courier: User who is operating the drone.
- Customer: User who receives packages that they've ordered.

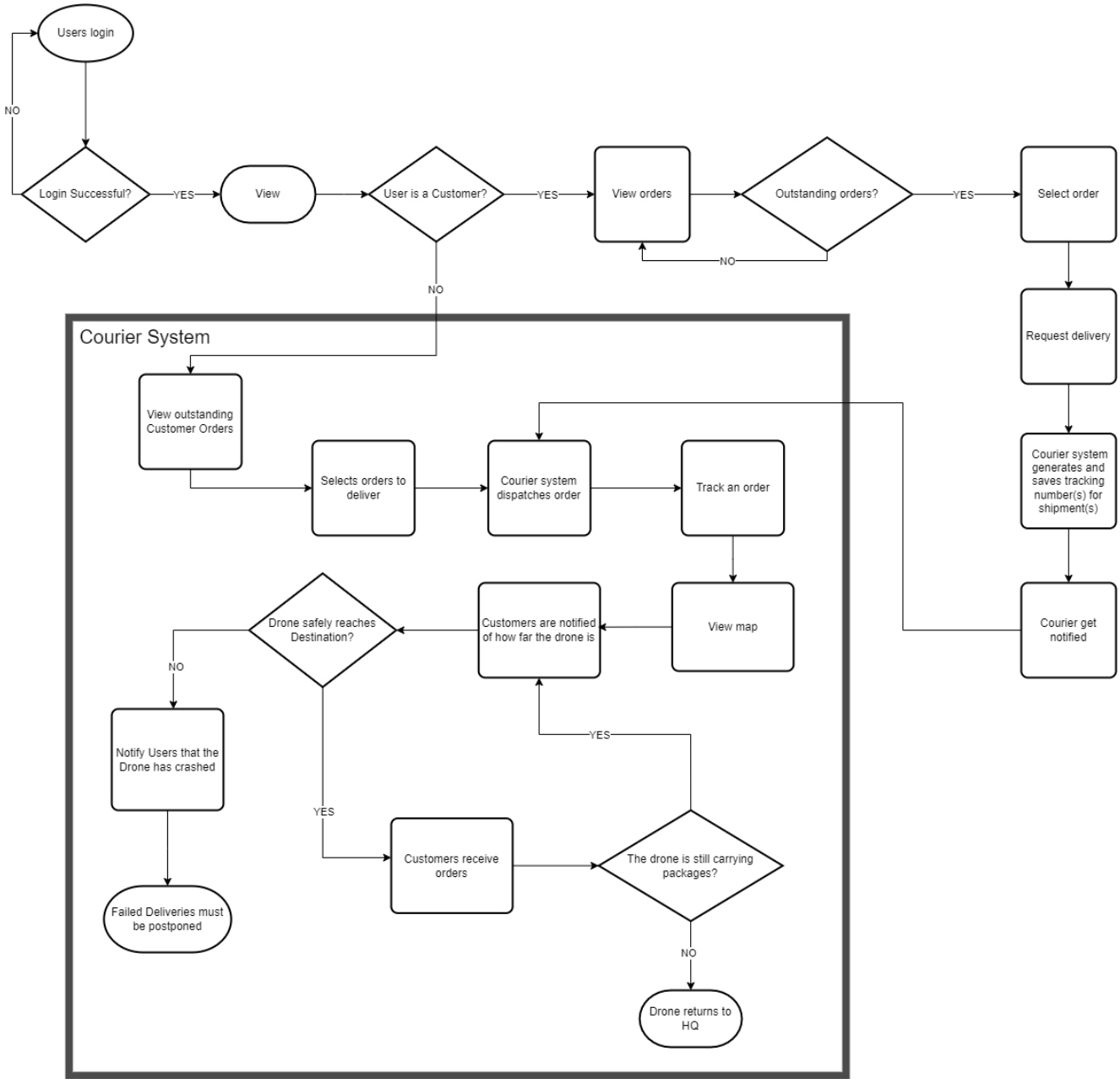
7.1 Courier (aka. drone operator)

The courier will be the user that receives a list of orders to deliver to customers. The courier remotely operates a drone that can carry **up to 7 products at once** to fulfill orders. The courier has the right to cancel or postpone deliveries. The courier is responsible for ensuring that the drone safely returns to the HQ.

7.2 Customer

The customer will be the user that orders one or more products and then receives a tracking number for their order. Their order contains a GPS location for the drone to drop the package. The customer can order up to 7 items at once.

7.3 Flow of the system



8 Assignment Instructions

Task 1: PHP API + Database ..... (25 marks)

You need to create a PHP API that builds on top of your PA3 API by adding Drone-related endpoints for this assignment. This means the Order-related endpoints will be used. It will be hosted off Wheatley, as well as making use of the same MySQL DB for your API. This assignment will require a database with five tables, namely `Users`, `Products`, `Orders`, `Orders_products`, and `Drones`. Almost all of these tables will be carried from your Practical Assignments.

Users are required to login before they can view or interact with any orders. Note that login functionality will be already be implemented in PA3, so it won't carry much weight in the HA.

8.1 API:

Your PHP API will need to handle user login and six other endpoints namely:
`CreateOrder`, `UpdateOrder`, `GetOrder`, `CreateDrone`, `UpdateDrone` and `GetAllDrones`.

- CreateOrder - Adds an order to the database.
- UpdateOrder - Updates the relevant fields in the Orders table.
  - latitude
  - longitude
  - state (Storage, Out_for_delivery or Delivered)
- GetAllOrders - Returns orders that are in the "Storage" state only. When a courier uses this endpoint, they should get all orders. Whereas when a customer uses it, they only get their own orders.

- CreateDrone - Adds a new drone to the database.
- UpdateDrone - Updates the relevant fields in the Drones table.
  - current_operator_id
  - is_available
  - latest_latitude
  - latest_longitude
  - altitude
  - battery_level
- GetAllDrones - Returns all drones in the database.

8.2 Database:

The Orders table will be populated as users order products, therefore you cannot populate it manually. However, you will need to manually create at least 20 Products in your database. The products must be small and light due to the limited carrying capacity of the drone.
The tables must have the following fields (However, you can add more fields if you want):

8.2.1 Users

This table is carried over from PA3. It will store user data:

- id
- username
- password

- email
- type (Distributor, Customer, or Courier)

8.2.2 Products

This table is carried over from PA3. It will store the following data about products:

- id
- title
- brand
- image_url
- categories
- dimensions
- is_available
- distributor

8.2.3 Orders

This table is carried over from PA4. This table will store the data related to orders. It has a **composite primary key** (customer_id, order_id):

- customer_id
- order_id
- tracking_num (Unique 10 character long string that begins with "CS-")
- destination_latitude
- destination_longitude
- state (Storage, Out_for_delivery or Delivered)
- delivery_date

8.2.4 Orders_Products

This table is carried over from PA4. It is a join table to represent the many-to-many relationship between orders and their products.

- id
- order_id
- product_id
- quantity

8.2.5 Drones

This table will contain information about the drones:

- id
- current_operator_id (null or references a userId in the Users table)
- is_available

- latest_latitude
- latest_longitude
- altitude
- battery_level

Task 2: Multi-User NodeJS server ..... (58 marks)

For this task, you are required to create a **Multi-User** 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. It is good practice to block the user from using reserved ports (Ports that have been reserved for other task). However there is still a chance you can choose a port in use, for the sake of this assignment if this happens, just try a different port. The ports you are allowed to use are from 1024 - 49151, make sure the user can only choose a port in this range. *If you are curious about why this is the case, you can read up about it at https://www.techtarget.com/searchnetworking/definition/port-number.*
- Block the user from attempting to open a Reserved Port (Explained above).
- Accept multiple client connections and interact with the clients concurrently through sockets.
- **The server must utilize functionality of the PHP API you developed in Task 1.** Therefore you must call your API endpoints to create, update and get an order.
- The server should have a `KILL` command to close a connection based on their username. This also means that you need to keep track of which socket ID corresponds to which Username since you need the SocketID to close a socket connection.
- 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. Consider the following:
  - If the user was a customer, they will simply have to reconnect to the server.
  - If the user was a courier, there is a possibility that they were operating a drone at the time.
  - If the courier was not operating a drone, they can simply reconnect to the server.
  - But if the courier was operating a drone:
    - The server must notify the customers that the operator is having connection issues and that their delivery has been postponed.
    - All orders that were out for delivery should be reset to the "Storage" state.
    - The drone should be assumed to have crashed. (Remember to reflect this in the database)
- The server should have a `CURRENTLY_DELIVERING` command that shows the order which are currently being delivered. The following information should also be provided:
  - The orderId
  - The name/title of the Products that are in the order
  - The Destination of the order [latitude,longitude]

  - The intended recipient's details

- 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.
- The server should have a `DRONE_STATUS` command which returns the following information:
  - Battery level
  - Altitude
  - Current Operator
  - GPS coordinates
- Take note of the following from the server side:
  - The server should continuously update the position of the delivery drone.
  - The server should notify the user when their package is on its way to them.
  - The server should be able to implement what is shown in the Flowchart under Description of Courier System.

  You have two options of how you can update the database. *There is no incorrect answer here, but you need to motivate your choices.*
  - Every time there is an update - This option mean that you will poll your API `UpdateOrder` every time there is a new order or a change.
  - In an interval - This option will only poll your API `UpdateOrder` in a certain interval, this mean that you will have to keep some local history of the data and compare to the latest data to determined the changes.

  You will need to write a short explanation of your choice and why you believe its the better option. Please write this is a ReadMe.txt that you should include with your ClickUp submission.
- 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.

To test the functionality of your server you may use `https://www.piesocket.com/websocket-tester` as the client before proceeding to Task 3.

Task 3: Angular Web Client ..... (42 marks)

For this task, you are required to develop a webpage/website 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 Angular version 17 (HTML,CSS and Typescript) using Web Sockets. You will also make use of a Leaflet map where all GPS data will be displayed. The client should also be on localhost.
Note: You may use any client side library of your choice (e.g. WebSockets, Socket.io, etc.). To make the changes you can/should make use of DOM Manipulation with JavaScript / Typescript. That way you do not need to redirect to a new page every time which might disconnect the socket.

The client must have the following functionality:

- The user should be able to enter login (Enter username and password). User registration is not needed.
- After successful login, there are 2 options:
  - If the user is a customer, they have two options:

    - They can can view their outstanding orders.
    - They can request delivery of one of their orders.
  - Otherwise, the user is an operator. They have the following options:
    - They can see the status of all drones.
    - They can view the orders that are due for delivery, select a drone and choose which orders to load into the drone.

8.3 The Courier System

The primary objective is for the operator to successfully deliver all packages and return the drone back to HQ. The courier should be able to broadcast notifications to customers if their delivery may be delayed. If the drone fails to deliver all packages, the customers should be informed that something went wrong with their delivery.

8.3.1 The Map

- The map is the central component that users interact with to monitor where the drone is and visualize GPS data.
- All GPS locations will must be rounded off to 4 decimal places.
- To calculate the distance between two points, use the `distance()` function.
- The courier system will operate in a 5km radius around the HQ. Ensure that the drone cannot go beyond this distance and that all orders are within range.
- Both the courier and customer must have a view of a map
  - Your map must be zoomed into Hatfield by default
  - Your map must have distinct markers for the HQ, the drone and customer locations
  - The HQ is located in at the coordinates [25.7472,28.2511]
  - Your map will also have Dust devils randomly located around Hatfield, they will be represented using circles with a radius of 10 meters. The drone operator must avoid entering theses circles on the map to prevent the drone from being destroyed. Here are the details about Dust Devils:
    - There are a random number of Dust Devils on the map (Between 5 and 10)
    - A new group of Dust Devils appear every minute (All at random positions around Hatfield)
    - Dust devils last for 1 minute
    - The location of the Dust Devils must be broadcast to all users and displayed on every user's map

    Further details on how they affect the drone are provided below (See section 8.3.2)
    Note: Make sure that all Dust Devil appear/spawn at least 11 meters away from the drone.
- Customers should be notified about the distance of the drone from their location.

8.3.2 The Drone

- The operator must successfully deliver all packages and the return the drone to HQ to complete a delivery run. There are three attributes to monitor:
  - **Battery life** - The drone has a battery life of **10 minutes**.
  - **Altitude** - The drone operates 20 meters high, and it will loose connection and crash if the altitude **goes above 30 meters**.
  - **Distance** - The drone must stay within a 5km range from the HQ.
- The drone operator must avoid dust devils that appear at random points in the Hatfield and last for 1 minute. If the drone enters a dust devil:

  - It must take one step back, to its previous position. (ie. If you last pressed UP arrow, the drone must go DOWN once to exit the circle)
  - The drones altitude is increased by 5 meters.
- The operator will use the arrow keys to move the drone. Each 'move' changes the drone's latitude or longitude by 0.0001
  - UP arrow increases the longitude
  - DOWN arrow decreases the longitude
  - RIGHT arrow increases the latitude
  - LEFT arrow decreases the latitude

8.3.3 Additional information

- For a package to be delivered: The drone simply needs to hover over the destination and then the operator should mark it as 'Delivered' by clicking on a button.
- Once the drone returns to HQ:
  - The drone's `battery_status` should be set back to 100
  - The `current_operator_id` should be reset to null
  - Its `altitude` must be set to 0
  - `is_available` should be set to true

Note: Bonus marks may be given to students who add extra functionality to them. A maximum of 15 bonus may be awarded (up to 7 for standard extra functionality but to achieve 15 you must add something extra-ordinary!) Bonus marks exceeding full marks will be capped.

Some ideas for bonus marks:

- Implement chat functionality where customers can send messages to the operator.
- Displaying some sort of animation for delivery
- Improving the Courier System to cater for making multiple trips.

Good Luck!