1. **Docker Setup**

* executed the docker image for intel CPU by writing this command in the terminal  
  **docker run -p 30000:8080 yondermakers/yonder-devops-tech-assessment:latest**

This binds the container’s Java-SpringBoot standard port to local:30000 (-p option is used to map a port on the host machine to a port in the Docker container )

1. **Verification**

* Accesed **‘http://localhost:30000/’** in a web browser to verify the container is running properly

Here I found the theoretical questions:

**Question 1: Exemplify two data structures that you know and describe some situations where you would use them**

* **Arrays: ideal for storing and accessing sequential data due to their ability to access elements by index quickly**

I would use them whenever the order of the elements matters or when I need to frequently access the elements by their position (concrete example: managing a sequence of daily tasks)

* **Hash Tables: efficient for storing or looking-up key-value pairs using unique keys**

I would use them when I need quick data retrieval, like caching data for fast access or counting the frequency of items in a collection (concrete example: counting words in a document)

**Question 2: You open a browser and access** [**http://www.tss-yonder.com**](http://www.tss-yonder.com)**. What is the IP address behind this website and how does the browser know how to get the correct IP?**

* The IP is: **172.67.73.177** (I typed ping <http://www.tss-yonder.com> in the command prompt)
* The browser gets the associated IP address by performing a DNS (Domain Name System) lookup in a distributed database system (the DNS names servers)

**Question 3: Exemplify two transport protocols and think of two applications that would use each of them.**

* **TCP (Transmission Control Protocol):** connection-oriented transport protocol that ensures reliable, ordered delivery of data between applications over a network, it is a reliable delivery system that ensures the package to arrive intact, in order, and confirms receipt, suitable for when the completeness and accuracy of the delivery are crucial.
* **Web Browsers**(HTTP/HTTPS): ensures web pages load correctly
* **Email** (SMTP, IMAP): guarantees reliable sending and receiving of emails
* **UDP (User Datagram Protocol):** connectionless protocol that allows for the transsmision of data without establishing a connection between the sender and the receiver, it is a quick, direct method of sending data that don’t sweat the small stuff, like occasional loses or order, making it suitable for real-time applications where speed is the essence
* **Online Gaming:** reduces latency for real-time gameplay
* **Streaming services (VoIP, Live Video):** offers faster streaming with minimal delay, ideal for live audio and video

**Question 4: You wrote a chat web application in your favourite programming language. You need to host this somewhere and run it so that the entire world can start using it. Describe how you would do that and the tools you use.**

* **Choosing a hosting platform**

I would choose a cloud-based hosting service like AWS (Amazon Web Services). I am thinking about AWS as a giant global computer where I could run my chat app so anyone can access it online, like renting a super-powerful computer that never turns off.

* **Setting Up the Environment**

I would containerize the application using Docker, because it would simplify the process of packaging and running the application in any environment (Docker containers wrap up an application with all its dependencies into a single package). This ensures the application runs the same way, regardless of where it is deployed (like a plant can live anywhere as long as it is in its own pot, that provides it all it needs in order to grow).

For managing and scaling the Docker containers, I would use Kubernetes, because it automates the deployment, scaling and management of containerized applications. It is important for a chat application to handle how many instances are running based on the demand and to adjust those numbers as needed without manual intervention.

* **Database Management**

For the database, I would go with Amazon DynamoDB. It is like a high tech filling system that AWS looks after for you. It is great because it can handle lots of data really quickly, perfect for a chat app where messages are flying in and out all the time.

* **Deployment**

For continuous deployment, I would set up a system that automatically checks and updates my chat app everytime I am making changes to it, like an efficient robot assistant that tests changes automatically and updates the app online without hassle. For this, I could use GitHub Actions – If my code is on GitHub, I can use this. It’s like telling GitHub “Hey, every time I add changes here, please run the tests and update the app if everything looks ok”.

* **Domain Name**

I would register a domain name for the application through a registrar like GoDaddy or Namecheap for easier access by users.

* **Monitoring**

I would start by selecting a monitoring tool that aligns with my requirements, such as Datadog. Then I would integrate this tool with my app and hosting environment, following the setup instructions provided. This ensures that I am alerted to potential issues in real-time.

* **Security Measures**

To keep my chat app safe, I would do two main things: use a Web Application Firewall (WAF), like having a bouncer at the door of my app, checking for troublemakers, stops hackers and bad traffic from getting in, and encrypt data – I would make sure all messages and user info are encrypted when they are stored and when they are sent across the internet.

**Question 5: Now your application is famous but unfortunately it has a lot of bugs. You want only you and a couple of friends to be able to access it until you patch it. Describe two ways to achieve this.**

* I would set up a VPN that only me and my friends have access to. This way, the app is still online, but only those connected to the VPN can reach it.
* I would configure the app’s server or use a firewall to only allow access fro specific IP addresses – mine and my friends’ (like having a guest list for an exclusive party: if the name -in this case, the IP- is on the list, the access is allowed).

**Question 6: Your application is ready for the public once again. You realize that you forgot about security and any network administrator can see the messages that a useer sends or receives. How would you improve your application to prevent this? Is there any way to do this so that not ven the application owner (you) can see the messages between two random users?**

To improve security for my application and ensure that the messages are private, even from network administrators or myself, I would implement end-to-end encryption (E2EE) for the messages. When E2EE is used, a message only appears in decrypted form only for the sender and the receiver of the message (the two ends). With E2EE, the key that can encrypt and decrypt messages remains saved on a user’s device. The app saves a key on the sender’s phone and a key on the receiver’s. The sender encrypts the message with his key, transmits the encrypted message and the receiver’s phone automatically applies the key and decrypts the message.

**Question 7: What are cookies and what are they used for? Find a cookie used by** [**http://www.tss-yonder.com**](http://www.tss-yonder.com) **and copy its name and value. What do you think is its purpose?**

A cookie is a technology that remembers something about the visitor of a website. Without cookies, a website is like a goldfish who loses its memory everytime a user visits it. Cookies are small text files containing unique data to identify the computer to the network. When visiting a website, it gives the browser a cookie to store in a cookie file that is placed in the browser’s folder on the hard drive. The next time the website is visited, the browser will give back the cookie to identify the visitor. Then the website loads with a personalised experience.   
**Name: PHPSESSID**

**Value: 79iokk4m1l1070a2fcnlc9gqdd**

**Question 8: While writing your application you need to cerate more worker processes for processing some data. How can you create child processes in your favourite language? What are the possible states of a process?**

Since my favourite language is Python, I can create child processes using the ‘**multiprocessing’** module which allows the application to run parallel tasks by creating child processes.

A basic way to create a child process:

**from multiprocessing import Process**

**def my\_function(name):**

**print(f‘Hello {name}’)**

**if \_\_name\_\_ == ‘\_\_main\_\_’:**

**p = Process(target=my\_function, args=(‘World’,))**

**p.start()**

**p.join()**

**Question 9: Your application is running but it still has a few problems. Occasionally, it return an error page. How can you find the PID of your application? What would you do to debug it?**

I would type in the command prompt:

**tasklist | findstr my\_application\_name**

this will list applications and services matching ‘my\_application\_name’ along with their PIDs

To debug my application, I would start by reviewing my application’s logs for error messages or clues about what went wrong, and if needed I would require to more advanced tools like using a debugger and monitor resources.

**Question 10: What DBMS would you use to store your application data and why? How would you store the passwords of each user?**

As I said in my previous response, I would use Amazon DynamoDB to store my application data, because as a fully managed NoSQL database service, it eliminates the administrative overhead of operating and scaling a distributed databse, allowing me to focus more on development rather than on maintaining a database.

For storing passwords securely, in DynamoDB, I would hash passwords before storing (utilize a one-way hashing algorithm like Argon2), incorporate salting (add a unique salt to each password before hashing it, this ensures that each hash is unique, even if two users have the same password). For each user, store the hash and the salted password along with the unique salt used. This allows for passwords verification without needing to store or know the actual password.

**Python Application Development**

This script is designed to automate the process of fetching, filtering, and exporting driving license records from the fictional Driving License Authority of Cluj's API. It leverages the Python programming language, utilizing libraries such as **requests** for API communication and **pandas** for data manipulation and exporting to **Excel**.

**Class: ‘DrivingLicenseAuthorityAPI’**

**‘\_\_init\_\_(self, base\_url)’ –** initializes the API client with the provided ‘base\_url’ for the API

**‘fetch\_data(self, total\_records = 150)’ –** fetches ‘total\_records’ number of driving license records from the API, in batches of 30

**‘list\_suspended\_licenses(self, data\_points)’ –** filters and returns a list of suspended licenses from the fetched data

**‘extract\_valid\_licenses(self, data\_points)’ –** filters and returns a list of licenses that are valid as from today’s date

**‘count\_licenses\_by\_category(self, data\_points)’ –** counts and categorizes licenses based on their category type

**Main Function**

1. Initializes the API client
2. Fetches driving license records from the API
3. Prompts the user to select an operation:

* List suspended licenses
* Extract valid licenses
* Count licenses by category

1. Depending on the operation selected, processes and exports data to an Excel file

**The folder includes the 4 Excels:**

* + **‘all\_drivers\_records.xlsx’**
  + **‘suspended\_licenses.xlsx’**
  + **‘valid\_licenses.xlsx’**
  + **‘license\_counts\_by\_category.xlsx’**