UNIVERSITI MALAYA

WIF3005 Software Maintenance and Evolution

Alternative Assessment

Semester 1 2024/2025

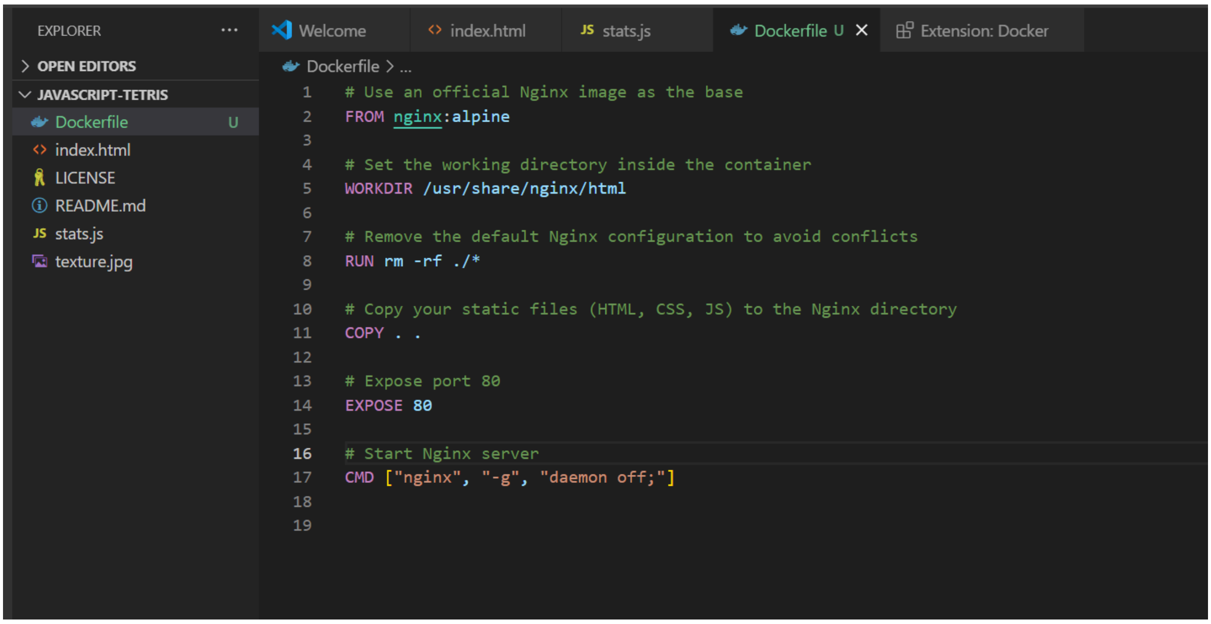
Lecturer: Dr. Nur Nasuha Binti Mohd Daud

Prepared by:

Chong Jia Xuan

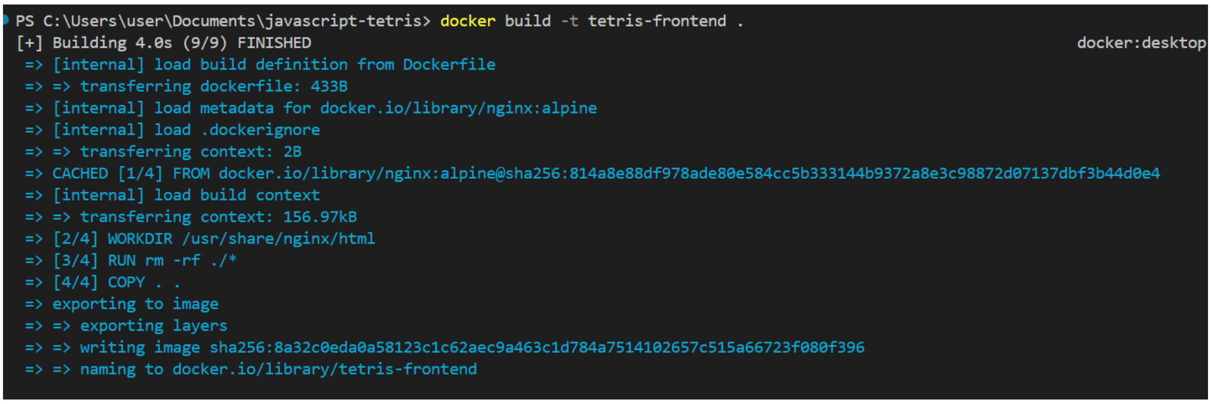
U2102725/1

Q2) Proof of verified build and test in BuildTest folder



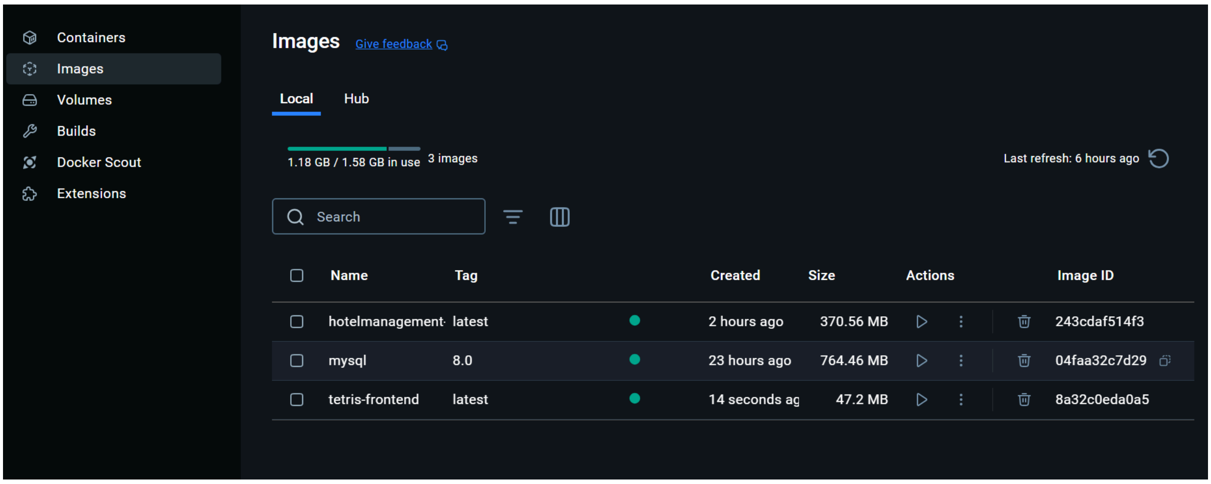
```dockerfile
# Use an official Nginx image as the base
FROM nginx:alpine

# Set the working directory inside the container
WORKDIR /usr/share/nginx/html

# Remove the default Nginx configuration to avoid conflicts
RUN rm -rf ./*

# Copy your static files (HTML, CSS, JS) to the Nginx directory
COPY . .

# Expose port 80
EXPOSE 80

# Start Nginx server
CMD ["nginx", "-g", "daemon off;"]


```

This Dockerfile uses the lightweight Nginx Alpine image to serve static files (HTML, CSS, and JS) for your frontend application. It first sets the working directory to /usr/share/nginx/html, which is where Nginx serves files from by default. The default configuration is cleared using rm -rf ./*, and the static files from your project are copied into this directory. The Dockerfile then exposes port 80 to allow HTTP traffic and starts the Nginx server using the command nginx -g "daemon off;", ensuring the server keeps running in the foreground.



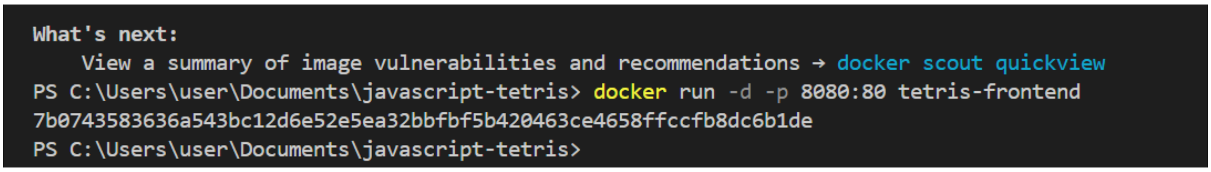
```
PS C:\Users\user\Documents\javascript-tetris> docker build -t tetris-frontend .
[+] Building 4.0s (9/9) FINISHED                                                                    docker:desktop
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 433B
 => [internal] load metadata for docker.io/library/nginx:alpine
 => [internal] load .dockerignore
 => => transferring context: 2B
 => CACHED [1/4] FROM docker.io/library/nginx:alpine@sha256:814a8e88df978ade80e584cc5b333144b9372a8e3c98872d07137dbf3b44d0e4
 => [internal] load build context
 => => transferring context: 156.97kB
 => [2/4] WORKDIR /usr/share/nginx/html
 => [3/4] RUN rm -rf ./*
 => [4/4] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:8a32c0eda0a58123c1c62aec9a463c1d784a7514102657c515a66723f080f396
 => => naming to docker.io/library/tetris-frontend
```

Figure 1: Build the docker image



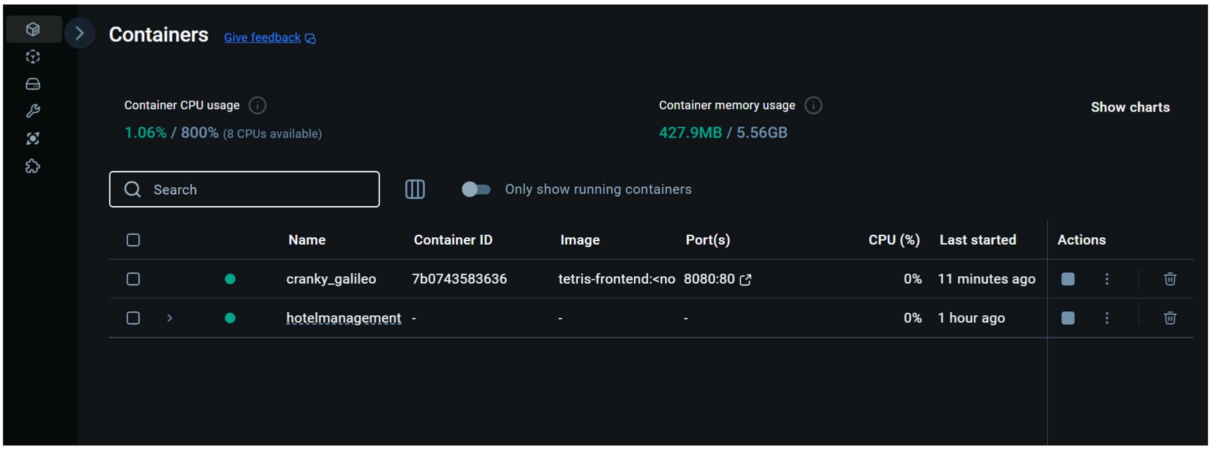
□	Name	Tag		Created	Size	Actions	Image ID
□	hotelmanagement-	latest	●	2 hours ago	370.56 MB	▷ ⋮ 🗑	243cdaf514f3
□	mysql	8.0	●	23 hours ago	764.46 MB	▷ ⋮ 🗑	04faa32c7d29
□	tetris-frontend	latest	●	14 seconds ag	47.2 MB	▷ ⋮ 🗑	8a32c0eda0a5

Figure 2: The docker image has been successfully built.



```
What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\user\Documents\javascript-tetris> docker run -d -p 8080:80 tetris-frontend
7b0743583636a543bc12d6e52e5ea32bbfbf5b420463ce4658ffccfb8dc6b1de
PS C:\Users\user\Documents\javascript-tetris>
```

Figure 3: Run the Docker container

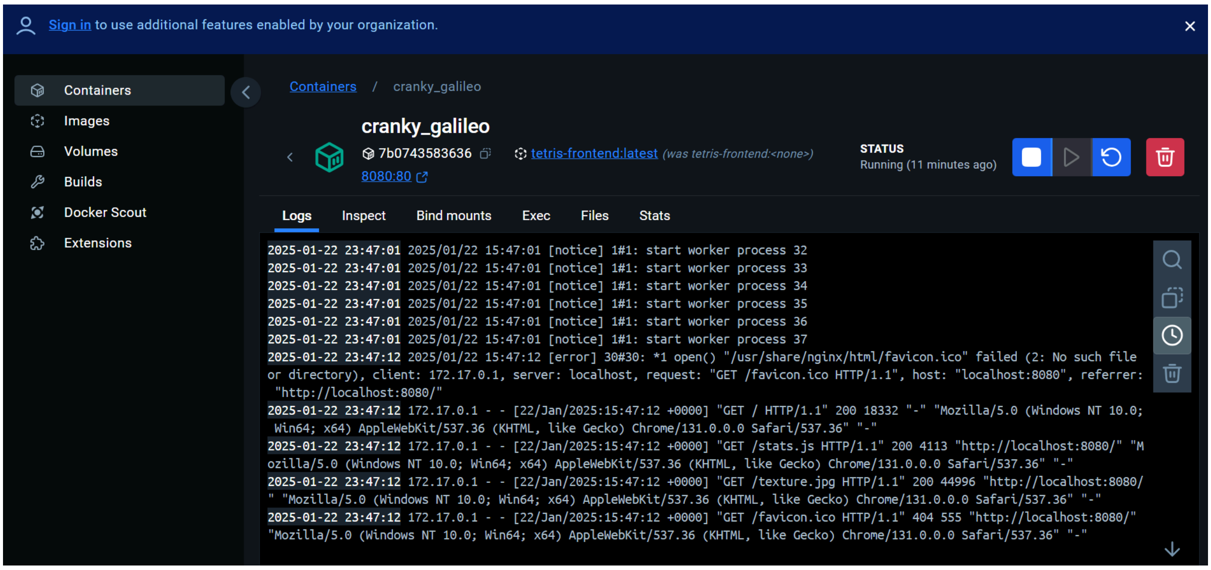


Container CPU usage
1.06% / 800% (8 CPUs available)

Container memory usage
427.9MB / 5.56GB

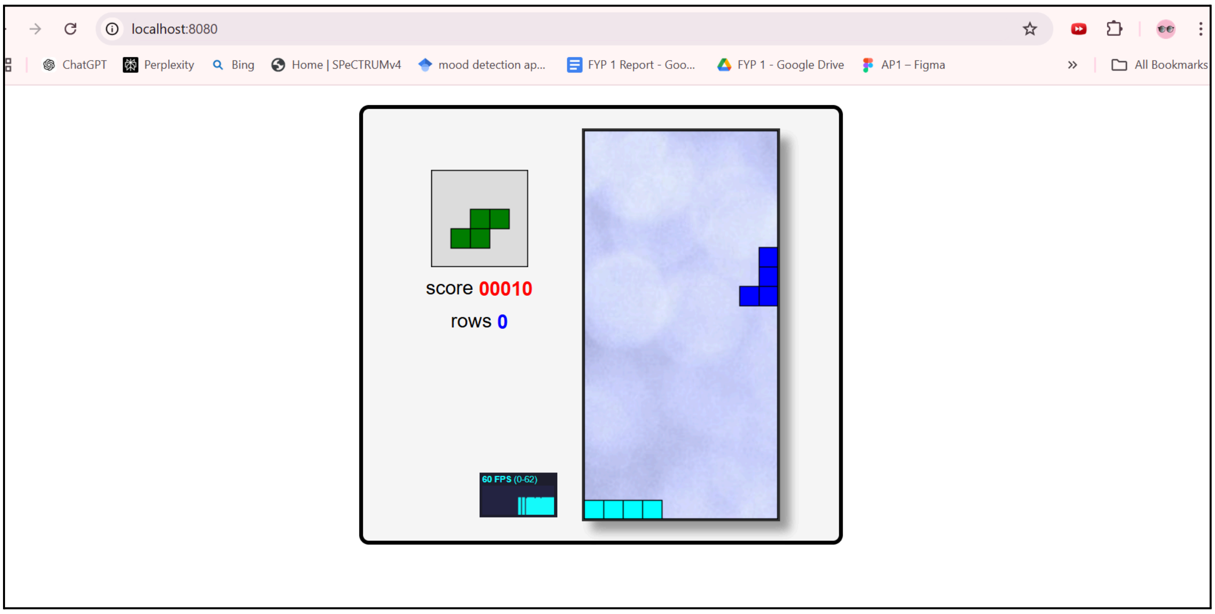
□			Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
□		●	cranky_galileo	7b0743583636	tetris-frontend:<no	8080:80	0%	11 minutes ago	■ ⋮ 🗑
□	›	●	hotelmanagement	-	-	-	0%	1 hour ago	■ ⋮ 🗑

An active Docker container running an Nginx server, serving files for a frontend application. The container is based on the lightweight Nginx Alpine image and is configured to expose port 80 for HTTP traffic.



```
2025-01-22 23:47:01 2025/01/22 15:47:01 [notice] 1#1: start worker process 32
2025-01-22 23:47:01 2025/01/22 15:47:01 [notice] 1#1: start worker process 33
2025-01-22 23:47:01 2025/01/22 15:47:01 [notice] 1#1: start worker process 34
2025-01-22 23:47:01 2025/01/22 15:47:01 [notice] 1#1: start worker process 35
2025-01-22 23:47:01 2025/01/22 15:47:01 [notice] 1#1: start worker process 36
2025-01-22 23:47:01 2025/01/22 15:47:01 [notice] 1#1: start worker process 37
2025-01-22 23:47:12 2025/01/22 15:47:12 [error] 30#30: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file
or directory), client: 172.17.0.1, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "localhost:8080", referrer:
 "http://localhost:8080/"
2025-01-22 23:47:12 172.17.0.1 - - [22/Jan/2025:15:47:12 +0000] "GET / HTTP/1.1" 200 18332 "-" "Mozilla/5.0 (Windows NT 10.0;
 Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
2025-01-22 23:47:12 172.17.0.1 - - [22/Jan/2025:15:47:12 +0000] "GET /stats.js HTTP/1.1" 200 4113 "http://localhost:8080/" "M
ozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
2025-01-22 23:47:12 172.17.0.1 - - [22/Jan/2025:15:47:12 +0000] "GET /texture.jpg HTTP/1.1" 200 44996 "http://localhost:8080/
" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
2025-01-22 23:47:12 172.17.0.1 - - [22/Jan/2025:15:47:12 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://localhost:8080/"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
```

A snapshot of the logs inside a running Docker container hosting an Nginx server. The logs display real-time activity, including HTTP requests and server status, confirming the successful serving of static frontend files



A screenshot of the browser displaying the website served on localhost, showing the frontend files hosted by the Nginx server inside the Docker container.