

National Textile University
Department of Computer Science
Subject: Operating System
Submitted to: Nasir Mahmood
Submitted by: Esha Mubashir Khan
Reg number: 23-NTU-CS-1151
Lab : 14
Semester: 5th

TASK 1:

services:

```
# MongoDB Database
```

mongodb:

```
image: mongo:7-jammy
```

```
container_name: guestbook-db
```

networks:

```
- guestbook-net
```

volumes:

```
- mongo_data:/data/db
```

```
# Backend API
```

api:

```
build: ./backend
```

```
container_name: guestbook-api
```

environment:

```
- MONGO_URL=mongodb://mongodb:27017/guestbook
```

ports:

```
- "3000:3000"
```

depends_on:

```
- mongodb
```

networks:

```
- guestbook-net
```

```
# Frontend
```

web:

```
build: ./frontend  
container_name: guestbook-web  
ports:  
- "8080:80"  
depends_on:  
- api  
networks:  
- guestbook-net
```

```
networks:  
guestbook-net:  
  
volumes:  
mongo_data:
```

EXPLANATION:

A custom Docker bridge network allows frontend, backend, and database containers to communicate securely using service names instead of IP addresses.

TASK 2:

The screenshot shows a Windows desktop environment with the Visual Studio Code (VS Code) application open. The title bar indicates the workspace is named 'guest-book [WSL: Ubuntu]'. The Explorer sidebar on the left lists files and folders, including 'package.json backend', 'server.js backend', 'index.html frontend', 'Dockerfile frontend', 'docker-compose.yml', 'Dockerfile backend', and 'index.html'. The current file being edited is 'docker-compose.yml'. The code in 'docker-compose.yml' defines two services: 'mongodb' (image: mongo:7-jammy, container_name: guestbook-db) and 'guestbook-api' (build: ./backend, container_name: guestbook-api). The 'api' service has an environment variable MONGO_URL set to mongodb://mongodb:27017/guestbook. The terminal tab at the bottom shows a command-line session where the user runs 'docker exec -it guestbook-web sh'. Inside the container, they attempt to ping 'mongodb' but receive an error message: 'sh: 1: ping: not found'. The system tray at the bottom right shows the date as 12/28/2025 and the time as 11:07 PM.

This screenshot is from the same workspace and environment as the previous one. The terminal output has changed. After running 'ping mongodb' in the previous step, the user now runs '/app # ping mongodb'. The terminal shows a series of successful PING/ PONG responses between the host machine and the MongoDB container, indicating a successful connection. The system tray at the bottom right shows the date as 12/28/2025 and the time as 11:08 PM.

TASK 3:

The screenshot shows the Docker Desktop application window. The left sidebar has a 'Images' section selected, which is highlighted with a grey background. The main area displays a table of 10 local images. The table columns are: Name, Tag, Image ID, Created, Size, and Actions. The 'Actions' column contains icons for each image, including a blue arrow, three dots, and a trash can.

Name	Tag	Image ID	Created	Size	Actions
ollama/ollama	latest	6c76395793f4	17 days ago	6.12 GB	
nginx	trixie	fb01117203ff	19 days ago	227.79 MB	
postgres	15.15-trixie	697ff7029514	20 days ago	632.55 MB	
postgres	15-alpine	2e7b888f2211	24 days ago	392.26 MB	
ubuntu	resolute-20251130	901617b8bedb	29 days ago	134.96 MB	
ubuntu	latest	c35e29c94501	2 months ago	119.25 MB	
mongo	7-jammy	8ddd3db4d263	9 days ago	1.14 GB	
guest-book-web	latest	6699b30319b5	31 minutes ago	224.97 MB	
guest-book-api	latest	927d2d2d8420	22 minutes ago	242.57 MB	

Showing 10 items

Engine running

RAM 3.45 GB CPU 22.06% Disk: 12.24 GB used (limit 1006.85 GB)

ChatGPT can make mistakes. Check important info.

Update available

12°C 11:10 PM 12/28/2025

TASK 4:screenshot of whole project

The screenshot shows the Visual Studio Code interface with the title bar "guest-book [WSL Ubuntu]". The left sidebar displays the file tree under "EXPLORER" and "GUEST-BOOK [WSL: UBUNTU]". The main editor area shows the content of the "Dockerfile backend" file, which defines a Docker container for the frontend. The Dockerfile includes commands for setting up the MongoDB database and exposing port 8080.

```
version: '3.8'
services:
  frontend:
    build: ./frontend
    container_name: guestbook-frontend
    ports:
      - "8080:80"
    depends_on:
      - mongodb
    networks:
      - guestbook-net
    volumes:
      - mongo_data:/data/db

  mongodb:
    image: mongo:7-jammy
    container_name: guestbook-db
    networks:
      - guestbook-net
    volumes:
      - mongo_data:/data/db

networks:
  guestbook-net:
```

TASK 5: Output of both image building commands e.g "docker build -t frontend

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows two main projects:
 - GUEST-BOOK [WSL: UBUNTU]:** Contains `backend`, `frontend`, and `docker-compose.yml`.
 - OPEN EDITORS:** Contains `package.json backend`, `server.js backend`, `index.html frontend`, `Dockerfile frontend`, `docker-compose.yml`, and `Dockerfile backend`.
- Terminal:** Displays a terminal session with the following logs:

```
esh@ESHA-DELL:~/guest-book$ docker build -t guestbook-backend .
[+] Building 0.5s (1/1) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 2B
ERROR: failed to build: solved to solve: failed to read dockerfile: open Dockerfile: no such file or directory
esh@ESHA-DELL:~/guest-book$ cd backend
esh@ESHA-DELL:~/guest-book/backend$ docker build -t guestbook-backend .
[+] Building 13.7s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 159B
=> [internal] load metadata for docker.io/library/node:20-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:20-alpine@sha256:658d0f63e501824d6c23e06d4bb95c71e7d704537c9d9272f4
=> => resolve docker.io/library/node:20-alpine@sha256:658d0f63e501824d6c23e06d4bb95c71e7d704537c9d9272f4
=> [internal] load build context
=> => transferring context: 92B
=> CACHED [2/5] WORKDIR /app
=> CACHED [3/5] COPY package.json .
=> CACHED [4/5] RUN npm install
=> CACHED [5/5] COPY .
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:ae890ac7e470d1cd5f2fafb5d6c94842988e390d8311447f401b272423328
=> => exporting config sha256:6ba1b0b53f36a3b67f132c60e688eeeb8e0647da8fdc54c4d626c18351b177
=> => exporting attestation manifest sha256:5f219bf671cf41e3348e047a341fc9e0f452814d9a5112f6f652f5838c1
=> => exporting manifest list sha256:f96ac0028f56339c7903d58e28ef65c59709e08189e183ae0926f565c09c80ec
=> => naming to docker.io/library/guestbook-backend:latest
=> => unpacking to docker.io/library/guestbook-backend:latest
```
- Bottom Status Bar:** Shows icons for search, file operations, and system status (12°C, battery level, signal strength, etc.).

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

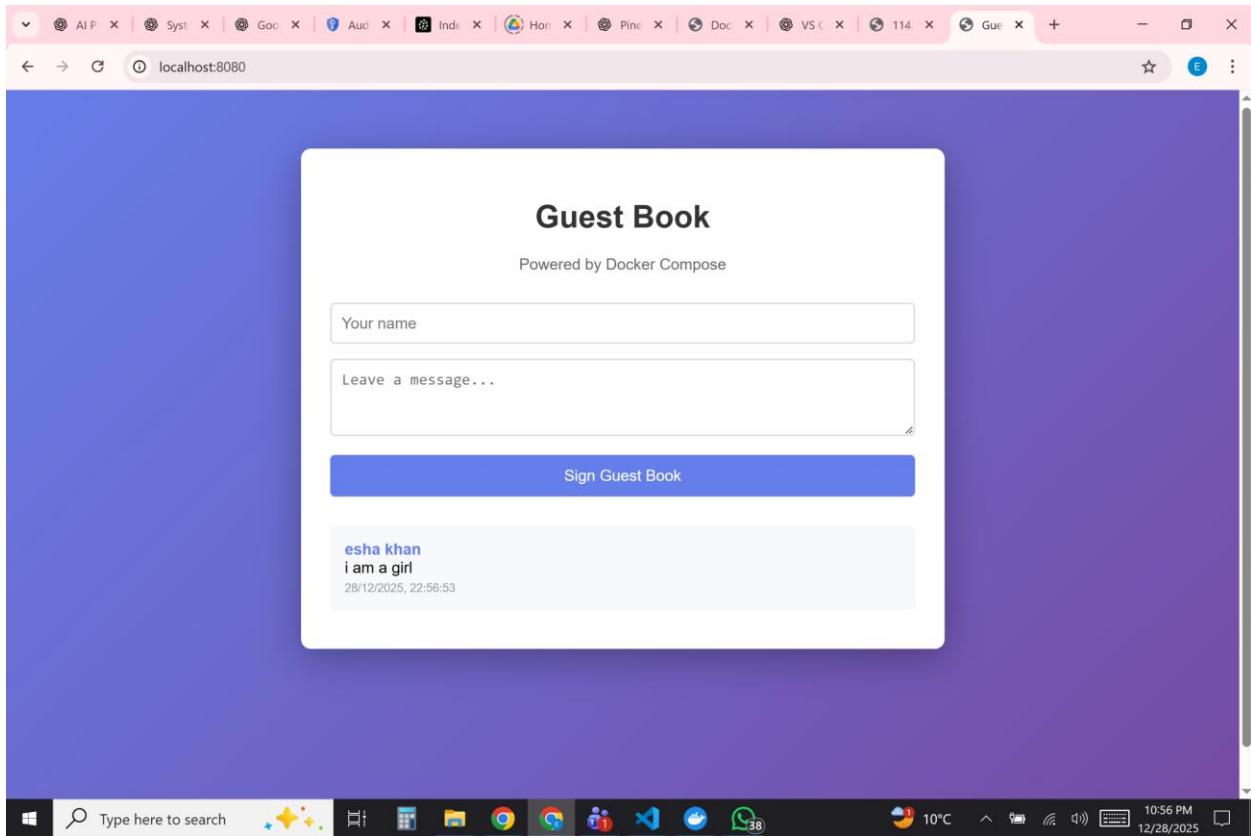
- File Explorer:** On the left, it shows a tree view of files and folders. Under "OPEN EDITORS", there are "package.json backend", "server.js backend", "index.html frontend", "Dockerfile frontend", and "docker-compose.yml". Under "GUEST-BOOK [WSL: UBUNTU]", there are "backend" (with "Dockerfile", "package.json", "server.js"), "frontend" (with "Dockerfile", "index.html"), and "docker-compose.yml".
- Terminal:** The main area displays the output of a terminal session. It shows the user navigating through three Dockerfiles: "backend", "frontend", and "frontends". For each, it runs a "docker build -t" command to create Docker images. The logs include details about the build process, such as layer caching, file copying, and manifest creation.
- Status Bar:** At the bottom, the status bar shows system information including battery level (40%), temperature (12°C), signal strength, and a timestamp (11:16 PM).

```
esh@ESHA-DELL:~/guest-book/backend$ docker build -t guestbook-backend .
=> CACHED [4/5] RUN npm install
=> CACHED [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:aee890ac7e470d1cd5f2fafb5d6c6c04842988e390d8311447f401b272423328
=> => exporting config sha256:6ba1b0b533f36a3b67f132cc0688eeeb8e0647da8fdc54c4d626c18351b177
=> => exporting attestation manifest sha256:5f219bf671cf41e334804a7a341fc9e0f452814d9a5112f6f652f5838c1
=> => exporting manifest list sha256:96eac0028f56339c7903d58628ef65c59709e08189e183ae0926f565c09c80ec
=> => naming to docker.io/library/guestbook-backend:latest
=> => unpacking to docker.io/library/guestbook-backend:latest

esh@ESHA-DELL:~/guest-book/backend$ cd ..
esh@ESHA-DELL:~/guest-book$ cd frontend
esh@ESHA-DELL:~/guest-book$ cd frontends
esh@ESHA-DELL:~/guest-book$ docker build -t guestbook-frontend .
[+] Building 11.4s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 103B
=> [internal] load metadata for docker.io/library/nginx:trixie
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 328
=> [1/2] FROM docker.io/library/nginx:trixie@sha256:fb0117203ff38c2f9af91db1a7409459182a37c87cced5cb42
=> => resolve docker.io/library/nginx:trixie@sha256:fb0117203ff38c2f9af91db1a7409459182a37c87cced5cb42
=> CACHED [2/2] COPY index.html /usr/share/nginx/html/
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:9780bc2faef47e4065401d6369e191e0fad7e14930aae99d8afbae8b90a093dc
=> => exporting config sha256:e9dc37a4e5a9ae97a64d8e29b00f1ac461b2d6143263e70c2c830b0f789a269
=> => exporting attestation manifest sha256:256.008381c16761c3828eff327f2c180c04736f572cc6874c120c5acc2ab1
=> => exporting manifest list sha256:cce80188556a572e2caccd112b4aa8207ccf7dd11a9adb393c7ee3a439af7
=> => naming to docker.io/library/guestbook-frontend:latest
=> => unpacking to docker.io/library/guestbook-frontend:latest

esh@ESHA-DELL:~/guest-book$
```

TASK 6: Webpage at localhost:8080



TASK 7:

Copy contents of docker-compose and both Dockerfiles in the pdf.

Code of docker- compose file :

version: "3.9"

services:

MongoDB Database (only backend network)

mongodb:

image: mongo:7-jammy

container_name: guestbook-db

restart: always

volumes:

```
- mongo_data:/data/db

networks:
  - backend_net

# Backend API

api:
  build: ./backend
  image: backend
  container_name: guestbook-api
  environment:
    - MONGO_URL=mongodb://mongodb:27017/guestbook
  ports:
    - "3000:3000"
  depends_on:
    - mongodb

networks:
  - frontend_net
  - backend_net

# Frontend

web:
  image: nginx:alpine
  container_name: guestbook-web
  restart: always
  ports:
    - "8080:80"
  volumes:
    - ./frontend:/usr/share/nginx/html:ro
```

depends_on:

- api

networks:

- frontend_net

networks:

frontend_net:

backend_net:

volumes:

mongo_data:

Code of dockerfiles :

```
FROM node:lts-alpine3.23
```

```
WORKDIR /app
```

```
COPY package*.json ./
```

```
RUN npm install
```

```
COPY server.js .
```

```
EXPOSE 3000
```

```
CMD ["npm", "start"]
```

To run the file :

```
docker compose up -build
```

use this command in terminal in guestbook.