REPORT

OVERVIEW

This project links 2 containers together, one is based on an image that contains a web server with PHP version 7.4 installed. The other container is based on a MySQL image that will manage our database.

WEB SERVER

- Dockerfile
  - We will be using an apache image which has PHP installed. A `RUN` command will be used to install PDO (PHP Data Objects) which will handle the database connections.
  - `COPY` command to copy the PHP files over to the webserver directory.
  - `EXPOSE 3306` exposes the MySQL port.
- PHP Files
  - The `index.php` file will be the main website.
  - We will initialize the database connection using `dbh.inc.php` which will include a PDO extension that interacts with the database. PDO will be able to manage errors and handle database operations.
  - To be able to add functionality to the website, we have to create another PHP file called `insert.php`. This file allows the website to receive data and inserts it into the database using the PDO statements and then redirects it to the website to be viewed.

DATABASE

- Dockerfile
  - We will be using the latest mysql image and setting an environment variable `MYSQL_ROOT_PASSWORD` to set up a root password.
  - We will be exposing the port 3306 as it is the default MySQL port.
  - The COPY command will allow the .sql script to be initialized when the container runs for the first time.
- studentDB.sql
  - This is the sql script that will be initialized when the container is running.
  - It contains attributes:
    - `student_id INT PRIMARY KEY`
    - `name VARCHAR(255) NOT NULL`
    - `age INT`
    - `cgpa DECIMAL(3, 2)`
    - `additional_data TEXT`

MOUNTED VOLUME FOR DATABASE

We will be mounting a volume to the directory `./db` which allows us to access data from outside the container. This ensures the data remains persistent when the container is stopped or restarted. It will be mounted through the `docker-compose.yml` file which will be explained below.

DOCKER-COMPOSE.YML

- `services`: These are the containers that make up the whole application. In this case, they are web and db.
- `build`: This will build using the Dockerfile in the directory specified.
- `volumes`: This will mount `./site` directory to `/var/www/html` which allows us to access the PHP files and edit them while the containers are running. For the db service it allows us to access the database files on `./db`
- `ports`: For the web server we will be mapping port 8080 on the host machine to 80 on the docker container which is the default web server port. For the MySQL database, we will be mapping port 3307 on the host machine to port 3306 on the container which is the default MySQL port.
- `depends_on`: This is used on the web service which ensures the db service runs first.
- `restart`: Setting this on `always` which ensures the service restarts if it is stopped for any reason.

RUNNING APPLICATION

- To run this container, open a terminal.
- Navigate to the directory of the `docker-compose.yml` file.
- Run command `docker-compose up`
- After docker finishes making containers, enter `127.0.0.1:8080/index.php`

Mustafa Othman (**2206191**)
Youssef Wael (**2206188**)
Malak Wessam (**22010268**)
Ahmed Hamed (**22011938**)
Rawan Mohamed (**20201322594**)