

Cloud Project

Supervised by:

- Eng. Sara Eldesouky

Members and Rules

Name	ID	Role
أحمد حسين حسن دويدار	20225926030	Database
حمزة حسين يوسف عمران	22011501	Back-End
معاذ مصطفى عبد الحميد مصطفى	22010263	Front-End
إبراهيم مهاب عمر	22010172	Docker
عبدالرازق جمعة محمد عبدالرحمن	22010133	Docker

The Database Code

The student table database code

We make course table.

We add primary key course_id

We add foreign key university id refers to university table.

```
--  
-- Table structure for table `courses`  
--  
  
CREATE TABLE `courses` (  
  `course_id` int(11) NOT NULL,  
  `course_name` varchar(255) DEFAULT NULL,  
  `course_description` text DEFAULT NULL,  
  `course_duration_months` int(11) DEFAULT NULL,  
  `course_start_date` date DEFAULT NULL,  
  `university_id` int(11) DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

And this code is to insert some data in the created table

```
--  
-- Dumping data for table `courses`  
--  
  
INSERT INTO `courses` (`course_id`, `course_name`, `course_description`, `course_duration_months`,  
  `course_start_date`, `university_id`) VALUES  
(1, 'Data Mining', 'Introduction to data mining techniques and algorithms.', 4, '2024-05-01', 1),  
(2, 'Computing Intensive', 'Advanced topics in computing and intensive computing applications.', 5, '2024-  
06-01', 2),  
(3, 'Data Science Methodology', 'Fundamentals of data science methodologies and best practices.', 6, '2024-  
07-01', 3),  
(4, 'Cloud Computing', 'Principles and practices of cloud computing technologies.', 4, '2024-08-01', 4),  
(5, 'Regression', 'Statistical regression analysis and modeling techniques.', 3, '2024-09-01', 1),  
(6, 'Artificial Intelligence', 'Exploring AI concepts, algorithms, and applications.', 5, '2024-10-01', 2),  
(7, 'Survey Methodology', 'Design and implementation of survey methodologies.', 4, '2024-11-01', 3);  
-----
```

This mysql code is to create the student table

We make student table

We add primary key student id

We add foreign key course id refers to course table.

We add foreign key university id refers to university table.

```
-- Table structure for table `students`--  
  
CREATE TABLE `students` (  
  `student_id` bigint(11) NOT NULL,  
  `name` varchar(255) DEFAULT NULL,  
  `age` int(11) DEFAULT NULL,  
  `cgpa` float DEFAULT NULL,  
  `course_id` int(11) DEFAULT NULL,  
  `university_id` int(11) DEFAULT NULL,  
  `student_image` blob DEFAULT NULL,  
  `year_of_student` int(11) DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

And this code is to add some entries

```
--
-- Dumping data for table `students`
--

INSERT INTO `students` (`student_id`, `name`, `age`, `cgpa`, `course_id`, `university_id`, `student_image`,
`year_of_student`) VALUES
(22010133, 'abd elrahman mohamed gomaa', 20, 3, 3, 1, NULL, 2),
(22010172, 'omar mohabibrahim', 20, 3, 4, 1, NULL, 2),
(22010263, 'moaaz mostafa abd alhamid', 20, 3, 6, 1, NULL, 2),
(22011501, ' Hamza Hussain Yousef', 20, 3, 5, 1, NULL, 2),
(20225926030, 'Ahmed hussein dewdar', 23, 2.7, 1, 1, NULL, 3);
-----
```

This is the university code and some sample data

We make university table.

We add primary key university id

```
--
-- Table structure for table `university`
--

CREATE TABLE `university` (
  `university_id` int(11) NOT NULL,
  `university_name` varchar(255) DEFAULT NULL,
  `university_location` varchar(255) DEFAULT NULL,
  `university_ranking` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
--
-- Dumping data for table `university`
--

INSERT INTO `university` (`university_id`, `university_name`, `university_location`, `university_ranking`)
VALUES
(1, 'Alexandria University', 'Alexandria', 1),
(2, 'Al-Qahira University', 'Cairo', 2),
(3, 'Ain Shams University', 'Cairo', 3),
(4, 'Al Mansoura University', 'Mansoura', 4);
```

And these are some necessary codes to make the tables connected with keys

```
ALTER TABLE `courses`
  ADD PRIMARY KEY (`course_id`),
  ADD KEY `university_id` (`university_id`);

--
-- Indexes for table `students`
--
ALTER TABLE `students`
  ADD PRIMARY KEY (`student_id`),
  ADD KEY `fk_course_id` (`course_id`),
  ADD KEY `fk_university_id` (`university_id`);

--
-- Indexes for table `university`
--
ALTER TABLE `university`
  ADD PRIMARY KEY (`university_id`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `courses`
--
ALTER TABLE `courses`
```

```

MODIFY `course_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;
--
-- AUTO_INCREMENT for table `students`
--
ALTER TABLE `students`
  MODIFY `student_id` bigint(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=20225926031;
--
-- AUTO_INCREMENT for table `university`
--
ALTER TABLE `university`
  MODIFY `university_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
--
-- Constraints for dumped tables
--
-- Constraints for table `courses`
--
ALTER TABLE `courses`
  ADD CONSTRAINT `courses_ibfk_1` FOREIGN KEY (`university_id`) REFERENCES `university` (`university_id`);
--
-- Constraints for table `students`
--
ALTER TABLE `students`
  ADD CONSTRAINT `fk_course_id` FOREIGN KEY (`course_id`) REFERENCES `courses` (`course_id`),
  ADD CONSTRAINT `fk_university_id` FOREIGN KEY (`university_id`) REFERENCES `university` (`university_id`);
COMMIT;

```

And here is some sample data and the course table code

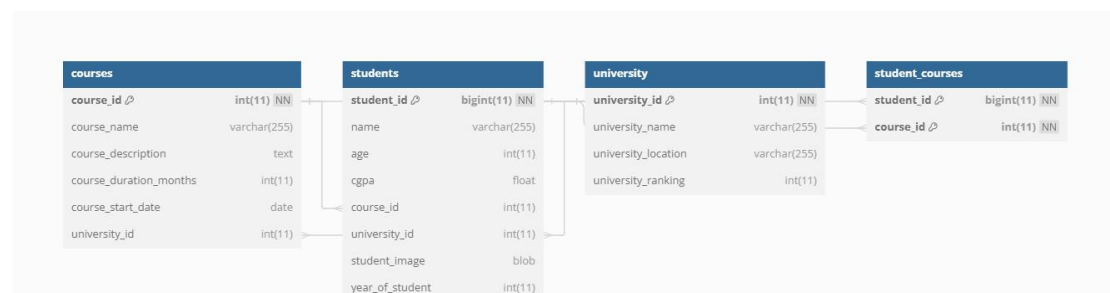
```

CREATE TABLE student_courses (
  student_id bigint(11) NOT NULL,
  course_id int(11) NOT NULL,
  PRIMARY KEY (student_id, course_id),
  FOREIGN KEY (student_id) REFERENCES students (student_id),
  FOREIGN KEY (course_id) REFERENCES courses (course_id)
);

INSERT INTO `student_courses` (`student_id`, `course_id`) VALUES (22011501, 1);
INSERT INTO `student_courses` (`student_id`, `course_id`) VALUES (22011501, 2);
INSERT INTO `student_courses` (`student_id`, `course_id`) VALUES (22011501, 3);
INSERT INTO `student_courses` (`student_id`, `course_id`) VALUES (22011501, 4);

```

Schema:



The Front-End and Back-End

We have implemented a search bar that allows users to find a student's data by entering their ID. When the search tag is clicked, a new page will display the student's image in an image section, and their information in a student information section, including their age, name, ID, and CGPA. We used PHP to display the table of courses the student has taken, and for the rest of the elements, we used HTML to build the table and other necessary sections to present the student's data. To add style, we used CSS.

Here is the first-page HTML code

This code is to show the search bar to enter the student id and send it to the PHP file in order to process it

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Student Search</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body style="background-color: white;">
  <!-- Search bar section -->
  <div class="search-container">
    <form action="start.php" method="post">
      <label for="searchInput" class="inline-element">Student ID:</label>
      <input type="text" id="searchInput" name="searchInput" placeholder="Enter your search term...">
      <button id="searchButton" type="submit" name="submit"><i class="fas fa-search"></i></button>
      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.2/css/all.min.css"
integrity="sha512-SnH5WK+bZxgPHs44uWIX+LLJAJ9/2PkPKZ5QiAj6Ta86w+fsb2TkcmfRyVX3pBnMfcV7oQPJk19QevSCWr3W6A="
crossorigin="anonymous" referrerpolicy="no-referrer" />
    </form>
  </div>
</body>
</html>
```

This is the code output

Student ID: 

And here is the PHP code we used to create the connection with the database

```
<?php
$dbservername="localhost";
$dbusername="root";
$dbpassword="";
$dbname="students database";

$conn=mysqli_connect($dbservername,$dbusername,$dbpassword,$dbname);
?>
```

And here is the new page that will show up code in PHP file which contains both PHP code and HTML also

This php code is to include the connection from the dbh.inc.php file which have the connection code and to query for the information of the input which is the student id

```
<?php
include_once 'include/dbh.inc.php';

if(isset($_POST['searchInput'])) {
    $studentId = $_POST['searchInput'];

    if(!empty($studentId)) {
        // Using prepared statement to prevent SQL injection
        $sql = "SELECT s.student_id, s.name, s.age, s.cgpa, u.university_name, u.university_location,
u.university_ranking,
                c.course_name, c.course_id, c.course_description, c.course_duration_months,
c.course_start_date,
                s.student_image
        FROM students s
        INNER JOIN university u ON s.university_id = u.university_id
        INNER JOIN student_courses sc ON s.student_id = sc.student_id
        INNER JOIN courses c ON sc.course_id = c.course_id
        WHERE s.student_id = ?";

        $stmt = mysqli_stmt_init($conn);
```

This code to make sure the connection is established and to fetch the data according to the query above

```
if(mysqli_stmt_prepare($stmt, $sql)) {
    mysqli_stmt_bind_param($stmt, "i", $studentId);
    mysqli_stmt_execute($stmt);
    $result = mysqli_stmt_get_result($stmt);

    if(mysqli_num_rows($result) > 0) {
        $studentData = mysqli_fetch_assoc($result);
    } else {
        $studentData = false;
    }
} else {
    // Error handling for query preparation failure
    $errorMessage = "Database query failed";
}
} else {
    // Error handling for empty input
    $errorMessage = "Please enter a student ID";
}
}
```

Now this is the HTML code inside the start.php file
This code is as we have mentioned above to make the search bar

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Student Search</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>

<div class="search-container">
  <form method="post">
    <label for="searchInput" class="inline-element">Student ID:</label>
    <input type="text" id="searchInput" name="searchInput" placeholder="Enter student ID...">
    <button type="submit" id="searchButton"><i class="fas fa-search"></i></button>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.2/css/all.min.css"
integrity="sha512-SnH5WK+bZxgPHs44uWIX+LLJAJ9/2PkPKZ5QIAj6Ta86w+fsb2TkcmfRyVX3pBnMfcV7oQPJk19QevSCWr3W6A="
crossorigin="anonymous" referrerpolicy="no-referrer" />
  </form>
</div>
<br><hr><br>
```

This code is to show the student info as the id, name, age, cgpa


```
<div id="searchResults">
  <?php if(isset($studentData)): ?>
    <?php if($studentData): ?>
      <!-- Display student information -->


      <!-- Student Image -->
      <div class="container">
        <div class="image-frame">
          
        </div>

        <!-- Student Info -->
        <div class="about-section">
          <h2 class="about-heading">Student Information</h2>
          <div class="student-info">
            <div class="info-container">
              <div class="info-item">
                <span>Name:</span>
                <span><?php echo $studentData['name']; ?></span>
              </div>
              <div class="info-item">
                <span>ID:</span>
                <span><?php echo $studentData['student_id']; ?></span>
              </div>
              <div class="info-item">
                <span>CGPA:</span>
                <span><?php echo $studentData['cgpa']; ?></span>
              </div>
              <div class="info-item">
                <span>Age:</span>
                <span><?php echo $studentData['age']; ?></span>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
<br><br>
```

The prior code should play this page

Student ID:





Student Information

Name: Hamza Hussain Yousef

ID: 22011501

CGPA: 3

Age: 20

And this code is to show the student enrollment courses

```
<!-- Display enrolled courses -->
<div>
  <table border="3" width="80%">
    <tr id="head">
      <td>COURSE NAME</td>
      <td>COURSE ID</td>
      <td>COURSE DESCRIPTION</td>
      <td>COURSE DURATION (Months)</td>
      <td>COURSE START DATE</td>
    </tr>
    <?php mysqli_data_seek($result, 0); ?>
    <?php while($row = mysqli_fetch_assoc($result)): ?>
      <tr>
        <td><?php echo $row['course_name']; ?></td>
        <td><?php echo $row['course_id']; ?></td>
        <td><?php echo $row['course_description']; ?></td>
        <td><?php echo $row['course_duration_months']; ?></td>
        <td><?php echo $row['course_start_date']; ?></td>
      </tr>
    <?php endwhile; ?>
  </table>
</div>
<?php else: ?>
  <p>No student found with the provided ID.</p>
<?php endif; ?>
<?php elseif(isset($errorMessage)): ?>
  <p><?php echo $errorMessage; ?></p>
<?php endif; ?>
</div>
</body>
</html>
```

This is the table that should show up in the same page

COURSE NAME	COURSE ID	COURSE DESCRIPTION	COURSE DURATION (Months)	COURSE START DATE
Data Mining	1	Introduction to data mining techniques and algorithms.	4	2024-05-01
Computing Intensive	2	Advanced topics in computing and intensive computing applications.	5	2024-06-01
Data Science Methodology	3	Fundamentals of data science methodologies and best practices.	6	2024-07-01
Cloud Computing	4	Principles and practices of cloud computing technologies.	4	2024-08-01

Dockerization

Dockerization, or containerization, is a method of packaging, distributing, and running applications and their dependencies within isolated environments called containers. Docker is one of the most popular containerization platforms.

Here's how it works:

1. **Containerization:** Containers are lightweight, standalone, and executable packages that contain everything needed to run an application, including the code, runtime, system tools, libraries, and settings. Unlike virtual machines, containers share the host operating system kernel and only package the application's specific dependencies.
2. **Docker Engine:** Docker provides a platform and tools for building, shipping, and running containers. The Docker Engine is the core component that manages containers on a host system.
3. **Dockerfile:** Docker uses a special file called a Dockerfile to define the environment and configuration for building a container image. The Dockerfile specifies the base image, dependencies, environment variables, and commands needed to set up the application inside the container.
4. **Container Image:** Once you have a Dockerfile, you use the Docker CLI (Command Line Interface) to build it into a container image. This image contains the application and its dependencies in a portable and immutable format.
5. **Container Registry:** Container images are typically stored in a container registry, such as Docker Hub or a private registry. These registries act as repositories for sharing and distributing container images across different environments.

6. **Container Orchestration:** In production environments, containerized applications are often managed by container orchestration platforms like Kubernetes, Docker Swarm, or Amazon ECS. These platforms automate the deployment, scaling, and management of containers across clusters of machines.

Benefits of Dockerization include:

- 1. Portability:** Containers can run on any platform that supports Docker, making it easy to move applications between development, testing, and production environments.
- 2. Isolation:** Containers provide a high level of isolation for applications, ensuring that they do not interfere with each other or with the underlying host system.
- 3. Consistency:** Dockerizing applications helps ensure consistency across different environments, reducing the risk of "it works on my machine" issues.
- 4. Resource Efficiency:** Containers are lightweight and share the host operating system kernel, resulting in faster startup times and lower resource overhead compared to virtual machines.

The Docker Compose Code

This Docker Compose configuration creates a development environment with a web server running PHP, a MySQL database, and phpMyAdmin for database management.

1. First we have declared the version of docker compose
2. Then we have defined the services which gonna contain the containers(which is referenced as a services in the code) we gonna make and all of them will be run and have access to each other
3. We have create a container called www and this container build will be specified in the dockerfile which is in the same directory
4. and the context refers to the directory where the Dockerfile and any other files required for building the Docker image are located and we have declared the dockerfile name in that directory
5. and determined the volumes attribute which is the place of local directory which will have this container run into it
6. and we have declared the ports attributes as the container is an isolated environment so we will access the container from port 80 in the container as a 9000 port on our local machine

```
version: '3'

services:
  # Web server service
  www:
    build:
      context: .
      dockerfile: Dockerfile
    volumes:
      - "./var/www/html" # Mount local directory into container
    ports:
      - "9000:80" # Map container port 80 to host port 9000
```

The DockerFile Code

This Dockerfile sets up a PHP environment with Apache and the `mysqli` extension installed, ready to serve PHP applications that require MySQL database connectivity.

1. Here we have determined the image `php` and the version of it which is `7.4-apache` to make it our base image
2. And we are going to run the **`docker-php-ext-install mysqli`** into the kernel of the container `www` in our example to restart the container and download the `mysqli` so the container work with no errors
3. And we have determined the container port which in all the container in our example is `80`

```
# Use the php:apache base image
FROM php:7.4-apache

# Install the mysqli extension
RUN docker-php-ext-install mysqli
    # Restart Apache to apply changes
    # && service apache2 restart
# Expose port 80
EXPOSE 80
```

1. here we have declared a container called `db` which is a abbreviation to database
2. And we defined the image attribute which gonna pull the image `mysql` and the latest version of it to create this container
3. And in the environment we have determined the `mysql` password and everything in order to access the database server
4. And as above we have made the volumes attribute to mount the volumes in the local machine and to take the `sql` script to create our `sql` in the created container
5. And we have not declared any ports mapping since the we will not access the `db` container from our local machine and just access it from the containers in the services on the default port `80`

```
# MySQL database service
db:
  image: mysql:latest
  environment:
    - MYSQL_DATABASE=students_database
    - MYSQL_USER=hamza12
    - MYSQL_PASSWORD=wh123
    - MYSQL_ALLOW_EMPTY_PASSWORD=1
  volumes:
    - "./db:/docker-entrypoint-initdb.d" # Mount database initialization scripts
```

Here we created the phpmyadmin container which have

1. The image to pull the image with the specified name
2. And ports to map it to our local machine
3. And the environment to determined our local host name and the port of mysql to access it on the default port 3306
4. And we added the depends_on attribute to ensure this container created after the specified container which is db container

```
# phpMyAdmin service
phpmyadmin:
  image: phpmyadmin/phpmyadmin
  ports:
    - "9001:80" # Map container port 80 to host port 9001
  environment:
    - PMA_HOST=db # Set phpMyAdmin host to the database service
    - PMA_PORT=3306 # Set phpMyAdmin port to MySQL port
  depends_on:
    - db # Ensure that the phpMyAdmin service starts after the db service
```