**//Home work program**

<!DOCTYPE html>

<html>

<head>

<title>User Input Validation</title>

</head>

<body>

<?php

$name = $email = $phone = "";

$nameErr = $emailErr = $phoneErr = "";

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Validate Name

$name = test\_input($\_POST["name"]);

if (empty($name)) {

$nameErr = "Name is required";

}

// Validate Email

$email = test\_input($\_POST["email"]);

if (empty($email)) {

$emailErr = "Email is required";

} elseif (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

$emailErr = "Invalid email format";

}

// Validate Phone

$phone = test\_input($\_POST["phone"]);

if (empty($phone)) {

$phoneErr = "Phone number is required";

} elseif (!preg\_match("/^[0-9]{10}$/", $phone)) {

$phoneErr = "Invalid phone number format";

}

}

function test\_input($data) {

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

?>

<h2>User Input Validation</h2>

<form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>">

Name: <input type="text" name="name" value="<?php echo $name; ?>">

<span style="color: red;"><?php echo $nameErr; ?></span>

<br><br>

Email: <input type="text" name="email" value="<?php echo $email; ?>">

<span style="color: red;"><?php echo $emailErr; ?></span>

<br><br>

Phone: <input type="text" name="phone" value="<?php echo $phone; ?>">

<span style="color: red;"><?php echo $phoneErr; ?></span>

<br><br>

<input type="submit" name="submit" value="Submit">

</form>

</body>

</html>

This program uses the htmlspecialchars() function to prevent cross-site scripting (XSS) attacks. The test\_input() function is created to perform basic input sanitization by removing leading and trailing whitespaces, and backslashes. The email validation is done using filter\_var() with the FILTER\_VALIDATE\_EMAIL filter, and the phone number is validated using a simple regular expression. The error messages are displayed in red if there are validation errors.

------------------------------------------------------------------------------------------------------------------------------------------

The terms "DBMS" (Database Management System) and "RDBMS" (Relational Database Management System) are related but have distinct meanings. Here are the key differences between the two:

**Database Management System (DBMS):**

1. **Definition:**
   * **DBMS:** A general term that refers to a software system designed to manage and facilitate the storage, retrieval, modification, and organization of data in a database.
   * **RDBMS:** A specific type of DBMS that is based on the relational model, where data is organized in tables with rows and columns.
2. **Data Model:**
   * **DBMS:** Can support various data models, including hierarchical, network, object-oriented, and more.
   * **RDBMS:** Specifically follows the relational data model, which organizes data into tables with predefined relationships.
3. **Schema:**
   * **DBMS:** May or may not enforce a schema for data organization. Schemas are more flexible and can be dynamic.
   * **RDBMS:** Enforces a fixed schema where the structure of tables and relationships is defined in advance.
4. **Flexibility:**
   * **DBMS:** Offers more flexibility in terms of data organization and relationships.
   * **RDBMS:** Imposes a structured, tabular format for data representation, which provides a high level of data integrity but may be less flexible for certain types of data.

**Relational Database Management System (RDBMS):**

1. **Data Structure:**
   * **DBMS:** Can handle various data structures, including hierarchical and network structures.
   * **RDBMS:** Specifically uses tables (relations) with predefined columns and data types. Each row in a table represents a record, and relationships between tables are defined using keys.
2. **Query Language:**
   * **DBMS:** May have its own query language or use standard SQL for data manipulation.
   * **RDBMS:** Typically uses SQL (Structured Query Language) as the standard language for querying and managing the relational database.
3. **Normalization:**
   * **DBMS:** May or may not enforce normalization rules for data storage.
   * **RDBMS:** Emphasizes normalization principles to eliminate data redundancy and maintain data integrity.
4. **Examples:**
   * **DBMS:** MongoDB, CouchDB, Berkeley DB.
   * **RDBMS:** MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server.

In summary, while all RDBMS systems are DBMS, not all DBMS systems are strictly RDBMS. RDBMS is a subset of DBMS that specifically adheres to the relational data model principles. The choice between a general-purpose DBMS and an RDBMS depends on the specific requirements of a project, including data structure, flexibility, and the need for standardized query language.

# Introduction

### What is MYSQL?

MySQL is a Database Management System.

### What is a Database Management System?

The software system that enables users to define, create, maintain and control access to the database is called a Database Management System. It is a software for storing and retrieving users' data while considering appropriate security measures.  
Examples of DBMS include MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database, and Microsoft Access.+

### Use of MYSQL

* It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL.
* Provides powerful mechanisms for ensuring only authorised users have entry to the database server.
* It has features such as - Data Protection, Scalability on Demand, High Efficiency, Excellent Workflow Control, and Lower Total Ownership Cost.

# Connecting to MySQL

In order to access the data in the MYSQL database, first we need to connect to the database. Now, there are two types of approaches when connecting to the MySQL server.

1. MySQLi Object-Oriented Method
2. MySQLi Procedural Method

### Connecting to Database using the MySQLi Object-Oriented Method

**<?php**

$servername = "localhost";

$username = "username";

$password = "password";

// Creating a connection

$conn = new mysqli($servername, $username, $password);

// Checking the connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

echo "Connected successfully";

**?>**

### Connecting to Database using the MySQLi Procedural Method

**<?php**

$servername = "localhost";

$username = "username";

$password = "password";

// Create a connection

$conn = mysqli\_connect($servername, $username, $password);

// Check the connection

if (!$conn) {

die("Connection failed: " . mysqli\_connect\_error());

}

echo "Connected successfully";

**?>**

### 

### Closing the Connection

We can close the connection with the following code.

//MySQLi Object-Oriented

$conn->close();

//MySQLi Procedural

mysqli\_close($conn);

# Creating a MySQL Database

We can use the CREATE DATABASE query inside a PHP script to create a new database in MySQL.

There are also two methods by which we can do it.

### MySQLi Object-oriented approach

**<?php**

$servername = "localhost";

$username = "username";

$password = "password";

// Create a connection

$conn = new mysqli($servername, $username, $password);

// Check the connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Create a database query

$sql = "CREATE DATABASE cwhDB";

if ($conn->query($sql) === TRUE) {

echo "Database created successfully";

} else {

echo "Error creating database: " . $conn->error;

}

$conn->close();

**?>**

### MySQLi Procedural approach

**<?php**

$servername = "localhost";

$username = "username";

$password = "password";

// Create a connection

$conn = mysqli\_connect($servername, $username, $password);

// Check the connection

if (!$conn) {

die("Connection failed: " . mysqli\_connect\_error());

}

// Create a database

$sql = "CREATE DATABASE cwhDB";

if (mysqli\_query($conn, $sql)) {

echo "Database created successfully";

} else {

echo "Error creating database: " . mysqli\_error($conn);

}

mysqli\_close($conn);

**?>**

# Creating a Table in MySQL using php | PHP Tutorial

#### Introduction

In this tutorial we will learn to create a table in MySQL using phpMyAdmin. We will use the same script, which was used in the last tutorial and modify it.

#### Creating a table

We will create a new function called $database for creating the table in the specified database. Now we will create another function called $sql and we will write the specified sql query to make a table(Note we don't need to add the database name in the query as we have already added it on $database function). Now to check if the table was created successfully or not we need to use an "if-else" statement and the code should look like this.

// Create a table in the db (Here Table Name is phptrip )

$sql = "CREATE TABLE `phptrip` ( `sno` INT(6) NOT NULL AUTO\_INCREMENT , `name` VARCHAR(12) NOT NULL , `dest` VARCHAR(6) NOT NULL , PRIMARY KEY (`sno`))";

$result = mysqli\_query($conn, $sql);

// Check for the table creation success

if($result){

echo "The table was created successfully!<br>";

}

else{

echo "The table was not created successfully because of this error ---> ". mysqli\_error($conn);

}

I hope this has cleared your doubts. for better understanding watch the complete video.

#### Code as described/written in the video

**<?php**

// Connecting to the Database

$servername = "localhost";

$username = "root";

$password = "";

$database = "dbharry";

// Create a connection

$conn = mysqli\_connect($servername, $username, $password, $database);

// Die if connection was not successful

if (!$conn){

die("Sorry we failed to connect: ". mysqli\_connect\_error());

}

else{

echo "Connection was successful<br>";

}

// Create a table in the db

$sql = "CREATE TABLE `phptrip` ( `sno` INT(6) NOT NULL AUTO\_INCREMENT , `name` VARCHAR(12) NOT NULL , `dest` VARCHAR(6) NOT NULL , PRIMARY KEY (`sno`))";

$result = mysqli\_query($conn, $sql);

// Check for the table creation success

if($result){

echo "The table was created successfully!<br>";

}

else{

echo "The table was not created successfully because of this error ---> ". mysqli\_error($conn);

}

**?>**

# Insert Data Into MySQL Using MySQLi using php

#### Introduction

In this tutorial we will learn to insert data into MySQL database using PHP. We will use MySQLi.

#### Inserting Data

Let's open our favourite code editor VSCode and start coding. We are going to connec to the database using the following code.

// Connecting to the Database

$servername = "localhost";

$username = "root";

$password = "";

$database = "dbharry";

Now we will try to establish a connection to the database using a simple if-else method.

// Create a connection

$conn = mysqli\_connect($servername, $username, $password, $database);

// Die if connection was not successful

if (!$conn){

die("Sorry we failed to connect: ". mysqli\_connect\_error());

}

else{

echo "Connection was successful<br>";

}

Now we need to run the sql query. We can copy sql query from the console as described in the video or we can write our own sql query. We just need to create a function to run the sql query. We can also add variables that can be inserted to the table using this method. We just need to create a variable function and then we need to use those variable function on the sql query in order to run. Then we can also use the if-else condition to check if the data was intserted or not. Like this :

// Variables to be inserted into the table

$name = "Vikrant";

$destination = "Russia";

// Sql query to be executed

$sql = "INSERT INTO `phptrip` (`name`, `dest`) VALUES ('$name', '$destination')";

$result = mysqli\_query($conn, $sql);

// Add a new trip to the Trip table in the database

if($result){

echo "The record has been inserted successfully successfully!<br>";

}

else{

echo "The record was not inserted successfully because of this error ---> ". mysqli\_error($conn);

}

**<?php**

// Connecting to the Database

$servername = "localhost";

$username = "root";

$password = "";

$database = "dbharry";

// Create a connection

$conn = mysqli\_connect($servername, $username, $password, $database);

// Die if connection was not successful

if (!$conn){

die("Sorry we failed to connect: ". mysqli\_connect\_error());

}

else{

echo "Connection was successful<br>";

}

// Variables to be inserted into the table

$name = "Vikrant";

$destination = "Russia";

// Sql query to be executed

$sql = "INSERT INTO `phptrip` (`name`, `dest`) VALUES ('$name', '$destination')";

$result = mysqli\_query($conn, $sql);

// Add a new trip to the Trip table in the database

if($result){

echo "The record has been inserted successfully successfully!<br>";

}

else{

echo "The record was not inserted successfully because of this error ---> ". mysqli\_error($conn);

}

**?>**