

CRUD OPERATION

STEP 1: Create a login form

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f2f2f2;
    }

    .container {
      max-width: 400px;
      margin: 0 auto;
      padding: 20px;
      background-color: #fff;
      border-radius: 5px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

    h2 {
      text-align: center;
      margin-bottom: 20px;
    }

    .form-group {
      margin-bottom: 20px;
    }

    .form-group label {
      display: block;
      font-weight: bold;
      margin-bottom: 5px;
    }

    .form-group input[type="text"],
    .form-group input[type="password"] {
      width: 100%;
      padding: 8px;
      border: 1px solid #ccc;
      border-radius: 4px;
    }

    .form-group button {
```

```

        width: 100%;
        padding: 10px;
        background-color: #4CAF50;
        color: #fff;
        border: none;
        border-radius: 4px;
        cursor: pointer;
    }

    .form-group button:hover {
        background-color: #45a049;
    }
</style>
</head>
<body>
    <div class="container">
        <h2>Login</h2>
        <form>
            <div class="form-group">
                <label for="username">Username:</label>
                <input type="text" id="username" name="name" required>
            </div>
            <div class="form-group">
                <label for="password">Password:</label>
                <input type="password" id="password" name="pass" required>
            </div>
            <div class="form-group">
                <button type="submit">Submit</button>
            </div>
        </form>
    </div>
</body>
</html>

```

STEP 2: Open terminal then create controller
(LoginController)

```
php artisan make:controller LoginController
```

then open controller file then create a function to
access a form.blade.php

```
//access login form
public function index(){
    return view('form');
}
```

STEP 3: Define the route in web.php

a) Use namespace in web.php from controller file

```
use App\Http\Controllers\Logincontroller;
```

b) Define the route to access Controller file

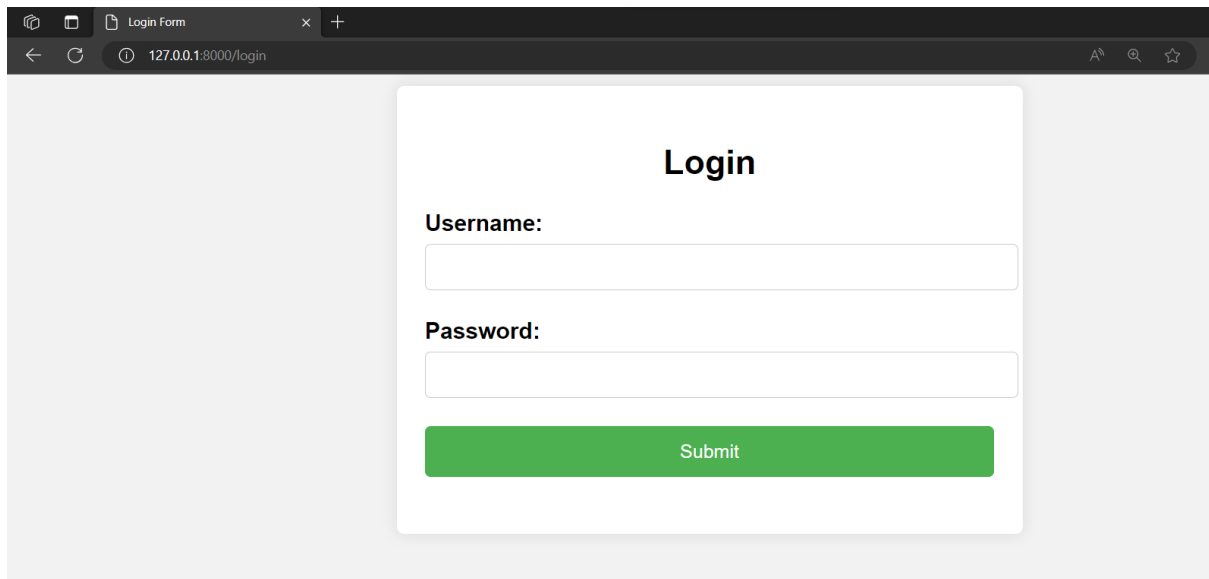
```
//form
Route::get('/login',[LoginController::class,'index']);
```

STEP 4: Access the file on browser

Type command for start Laravel server

```
PS C:\xampp\htdocs\CRUD> php artisan serve
INFO Server running on
[http://127.0.0.1:8000]. Press Ctrl+C to stop the
server
```

Copy the given url then pasted on browser with route name



STEP 5: Use @csrf token (cross-site request forgery) in form.blade.php when we use post method

```
<form action="{{url('/')}}/login" method="post">
    @csrf
```

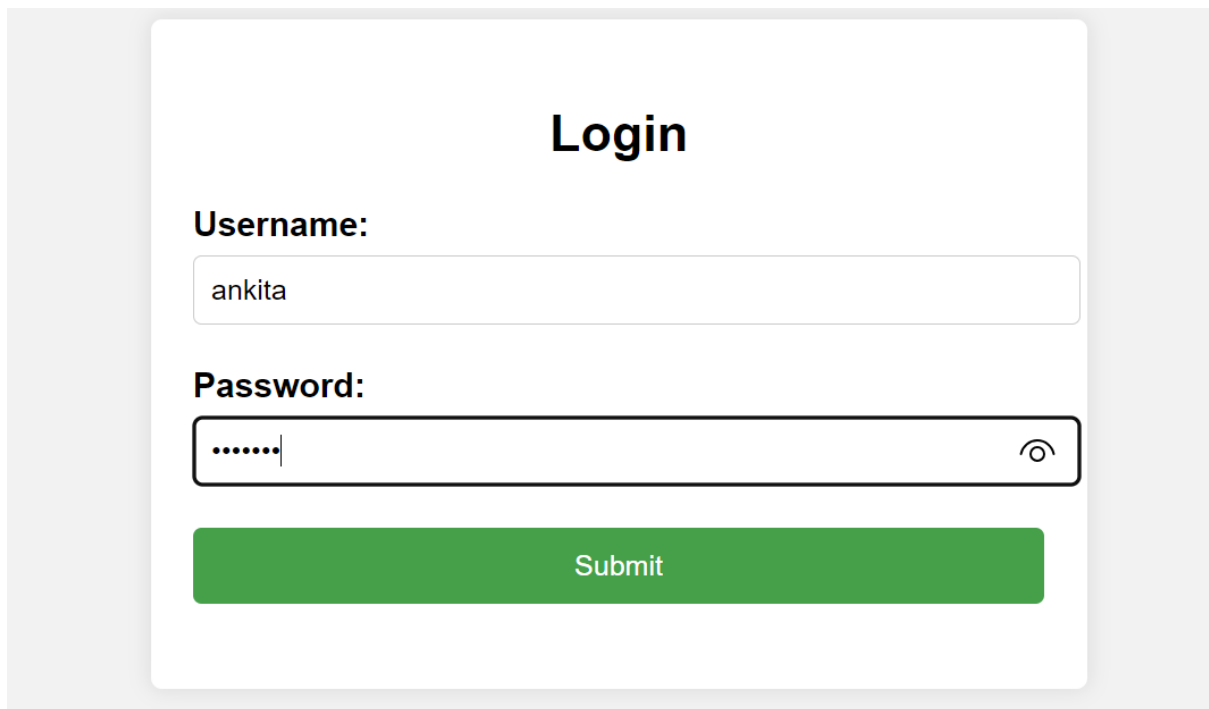
STEP 6: Create a function for show the data after submitted.

```
//show the data
public function register(Request $ab){
    return $ab->all();
}
```

STEP 7: define route in web.php for access the data after submitted through post method

```
//submitted the data
Route::post('/login',[LoginController::class,'register']);
```

STEP 8: Then go to browser then fill the details then click on submit

A login form titled "Login" with a white background and a light gray border. It contains two input fields: "Username:" with the text "ankita" and "Password:" with masked characters ".....". A green "Submit" button is at the bottom. A small eye icon is visible on the right side of the password field.

Login

Username:

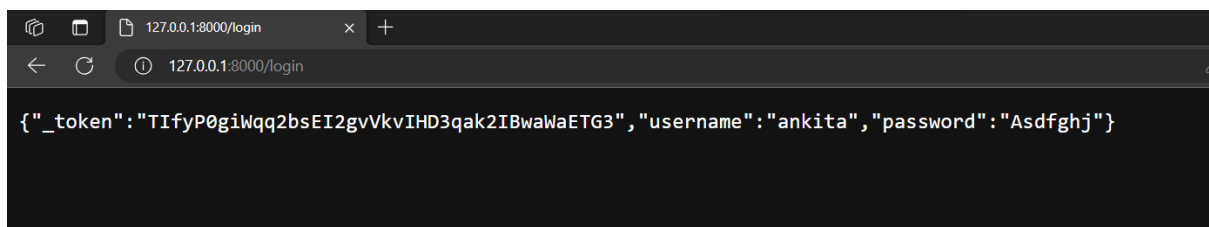
ankita

Password:

.....

Submit

Now see the data will be shown on array format

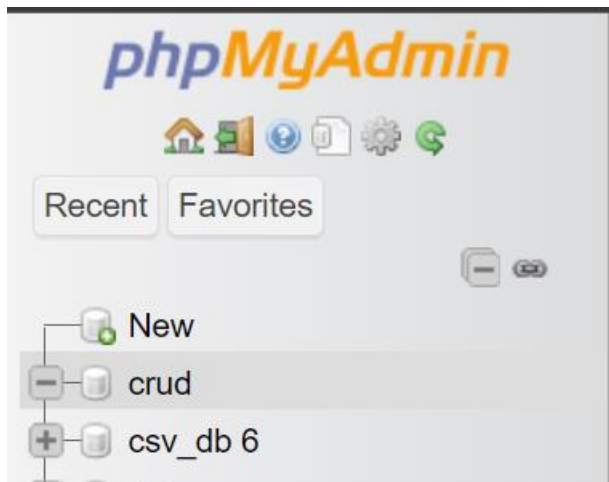
A screenshot of a web browser window. The address bar shows "127.0.0.1:8000/login". The main content area displays a JSON object: {"_token": "TIfyP0giWqq2bsEI2gvVkvIHD3qak2IBwaWaETG3", "username": "ankita", "password": "Asdfghj"}.

```
{"_token": "TIfyP0giWqq2bsEI2gvVkvIHD3qak2IBwaWaETG3", "username": "ankita", "password": "Asdfghj"}
```

Insert the Data

STEP 1: Open xampp then start apache and mysql server

Then go to phpMyAdmin then create a new database



STEP 2: Then open .env file and do some changes here

Mysql port no. and Database Name

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=CRUD
DB_USERNAME=root
DB_PASSWORD=
```

After that run the command php artisan config:cache for save the changes in env file

```
PS C:\xampp\htdocs\CRUD> php artisan config:cache
INFO Configuration cached successfully.
PS C:\xampp\htdocs\CRUD> 
```

STEP 3: create a model and migration

Migration created in → Database → trainees

Model created in → App → Models → Trainee

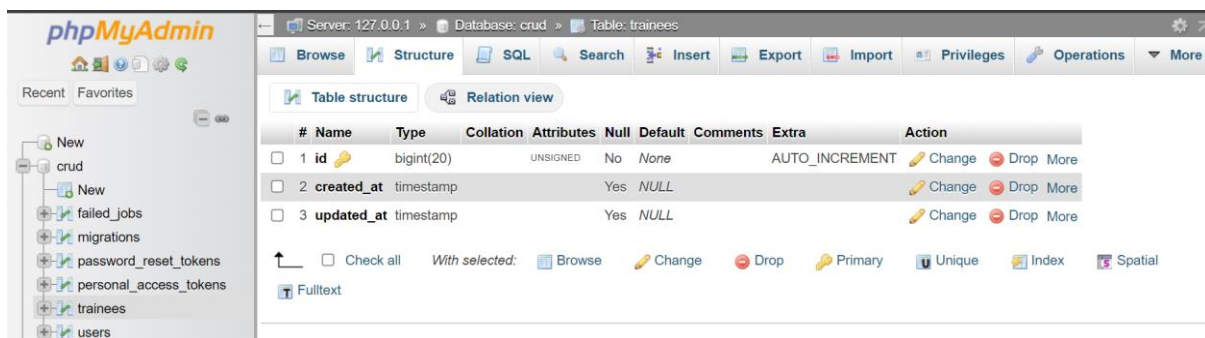
Type the command

```
PS C:\xampp\htdocs\CRUD> php artisan make:model Trainee --migration  
  
INFO Model [C:\xampp\htdocs\CRUD\app\Models\Trainee.php] created successfully.  
  
INFO Migration [C:\xampp\htdocs\CRUD\database\Migrations\2023_06_16_080812_create_trainees_table.php] created successfully.  
  
PS C:\xampp\htdocs\CRUD>
```

STEP 4: Add a field in the create table(migration(trainees))

```
$table->id('Reg_id');  
$table->string('Username');  
$table->string('Password');  
$table->timestamps();
```

Then run migrate for connect to the database



The screenshot shows the phpMyAdmin interface. On the left is a sidebar with a tree view of the database structure, including folders for 'New', 'crud', 'failed_jobs', 'migrations', 'password_reset_tokens', 'personal_access_tokens', 'trainees', and 'users'. The main panel displays the 'Table structure' for the 'trainees' table in the 'crud' database. The table has three columns: 'id' (bigint(20), UNSIGNED, No, None, AUTO_INCREMENT), 'created_at' (timestamp, Yes, NULL), and 'updated_at' (timestamp, Yes, NULL). Each column has a 'Change', 'Drop', and 'More' action link. At the bottom, there are options to 'Check all', 'With selected', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', and 'Spatial'.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)	UNSIGNED	No	None			AUTO_INCREMENT	Change Drop More
2	created_at	timestamp		Yes	NULL				Change Drop More
3	updated_at	timestamp		Yes	NULL				Change Drop More

STEP 5: Then open model and define table name and primary key

```
protected $table="trainees"; //table name  
protected $primaryKey="Reg_id";
```

then use namespace of model in controller file

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Trainee;
```

STEP 6: then open controller file and create a new object for Trainee class

```
$a=new Trainee;

$a->Username=$ab['name'];
$a->Password=$ab['pass'];
$a->save();
```

STEP 7: Check the data is submitted or not

	Reg_id	Username	Password	created_at	updated_at
1	Ankita Shukla	asdfgh		2023-06-16 08:21:39	2023-06-16 08:21:39

Fetch the data

STEP 1: create a new file (view.blade.php)

```
<!DOCTYPE html>
<html>
<head>
    <title>Bootstrap Table Example</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
">
</head>
<body>
    <div class="container">
        <h2>Registered user</h2>
        <a href="{{url('/')}}/login"></a>
        <table class="table">
            <thead>
                <tr>
```



```

        <th>UserNAme</th>
        <th>Password</th>

    </tr>
</thead>
<tbody>
    @foreach ($a as $abc )
        <tr>
            <td>{{$abc->Username}}</td>
            <td>{{$abc->Password}}</td>
        </tr>
    @endforeach

</tbody>
</table>
</div>
</body>
</html>

```

STEP 2: Open a controller file

Redirect to view page after submitted the data

```

//access view page
return redirect('/view');

```

create a function for fetch the data

```

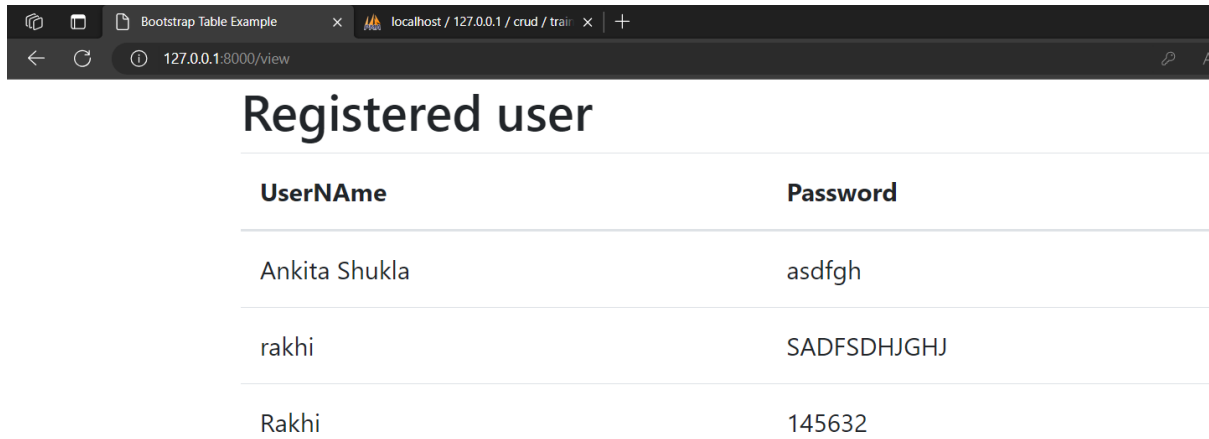
//fetch the data
public function view(){
    $a=Trainee::all();
    $data=compact('a');
    return view('view')->with($data);
}

```

STEP 3: then go to web.php file and define the route for access view function

```
//fetch data
Route::get('/view',[LoginController::class,'view']);
```

STEP 4: check the data is fetch or not



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/view'. The page title is 'Registered user'. Below the title is a table with two columns: 'UserNAme' and 'Password'. The table contains three rows of data.

UserNAme	Password
Ankita Shukla	asdfgh
rakhi	SADFSDHJGHJ
Rakhi	145632

Edit & update

STEP 1: create a file edit.blade.php then create a field you want to edit

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f2f2f2;
    }

    .container {
      max-width: 400px;
      margin: 0 auto;
      padding: 20px;
      background-color: #fff;
      border-radius: 5px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
  </style>
</head>
<body>
  <div class="container">
    <h3>Login Form</h3>
    <form>
      <input type="text" value="Username" />
      <input type="password" value="Password" />
      <input type="submit" value="Login" />
    </form>
  </div>
</body>
</html>
```

```

}

h2 {
  text-align: center;
  margin-bottom: 20px;
}

.form-group {
  margin-bottom: 20px;
}

.form-group label {
  display: block;
  font-weight: bold;
  margin-bottom: 5px;
}

.form-group input[type="text"],
.form-group input[type="password"] {
  width: 100%;
  padding: 8px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.form-group button {
  width: 100%;
  padding: 10px;
  background-color: #4CAF50;
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.form-group button:hover {
  background-color: #45a049;
}
</style>
</head>
<body>
  <div class="container">
    <h2>Login</h2>
    <form action="{{url('update')}}/{{$a->Reg_id}}">
      <div class="form-group">
        <label for="username">Username:</label>
        <input type="text" id="username" name="name" required value="{{$a->Username}}">

```

```

        </div>

        <button type="submit">update</button>
    </div>
</form>
</div>
</body>
</html>

```

STEP 2: create a button on view.blade.php for edit and delete the data.

```

<td>

    <a href="{{url('edit/')}}/{{$abc->Reg_id}}">
        <button class="btn btn-success">EDIT</button>
    </a>
    <a href="{{url('delete/')}}/{{$abc->Reg_id}}">
        <button class="btn btn-danger">delete</button>
    </a>
</td>

```

STEP 3: create a function for edit and update and delete the data in controller file

```

//for edit
public function edit($id){
    $a=Trainee::find($id);
    $da=compact('a');
    return view('edit')->with($da);
}
//for update
public function update(Request $ab, $id){
    $a=Trainee::find($id);
    $a->Username=$ab['name'];
    $a->update();
    //access view page
    return redirect('/view');
}
//delete
public function delete($id){
    Trainee::find($id)->delete();
}

```

```
return redirect()->back();  
}
```

STEP 4: define route for edit and update in web.php

```
//edit  
Route::get('/edit/{id}',[LoginController::class,'edit']);  
  
//update  
Route::put('/update/{id}',[LoginController::class,'update']);  
  
//delete  
Route::get('/delete/{id}',[LoginController::class,'delete']);
```

When we use put method we can mention put method in view.blade.php

```
<form action="{{url('update/')}}/{{$a->Reg_id}}" method="post">  
    @csrf  
    @method('PUT')
```

STEP 5: check the button is working or not

Registered user

UserNAme	Password	Action
Ankita s	asdfgh	EDIT delete

Edit into Ankita Shukla

Registered user

UserNAme	Password	Action
Ankita Shukla	asdfgh	EDIT delete

Click on delete button to delete the data

Registered user		
UserNAme	Password	Action