# Web Technology-PHP Laravel Todo Doc. for basic CRUD Op.

First of all, we should install all the content below for beginning of this task.

**Step 1: Install Required Software**

**1.Node.js**

**2.Composer**

**3.XAMPP**

**4.7zip**

**5.Terminal**

**6.Visual studio code**

**Step 2: Edit Environment Variable**

**Add New Path**

**1.Php (C:\xampp\php)**

**2.Nodejs (C:\Program Files\nodejs)**

**Step 3: Verify Setup**

**By opening windows PowerShell**

**1.Node –version**

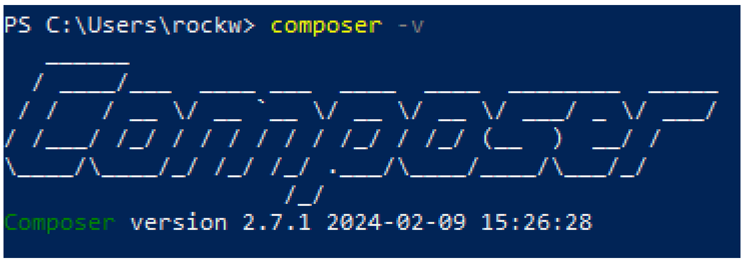
***You'll be able to view the following output***

**PS C:\Users\rockw> node --version**

**v20.11.1**

**2.Composer –v**

***You'll be able to view the following output***



```
PS C:\Users\rockw> composer -v
   ______
  / ____/___  ____ ___  ____  ____  ________  _____
 / /   / __ \/ __ `__ \/ __ \/ __ \/ ___/ _ \/ ___/
/ /___/ /_/ / / / / / / /_/ / /_/ (__  )  __/ /
\____/\____/_/ /_/ /_/ .___/\____/____/\___/_/
                    /_/
Composer version 2.7.1 2024-02-09 15:26:28
```

**3.Php -v**

```
PS C:\Users\rockw> php -v
PHP 8.2.12 (cli) (built: Oct 24 2023 21:15:15) (ZTS Visual C++ 2019 x64)
```

Copyright (c) The PHP Group

Zend Engine v4.2.12, Copyright (c) Zend Technologies

**Step 4: install Laravel project**

**1.composer create-project laravel/Laravel your-project-name**

**2.cd your-project-name**

**3.Php artisan serve**

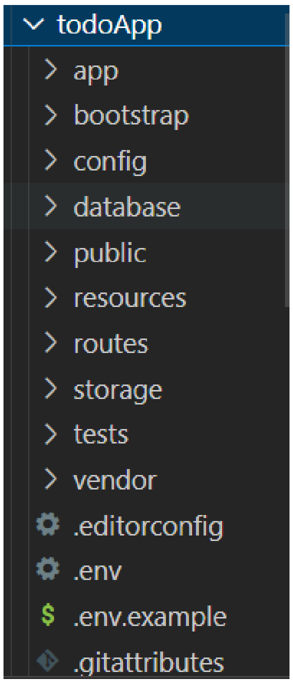
**4.For running other than port 800**

***Php artisan serve –port=8001***

***Now, let's start a project by moving to your appropriate folder and installing the project using following command below.***

**Step 1:** composer create-project laravel/laravel todoApp

Once the project is successfully created you can observe many folders and files like this.



**Step 2: cd todoApp**

**Step 3: php artisan serve**

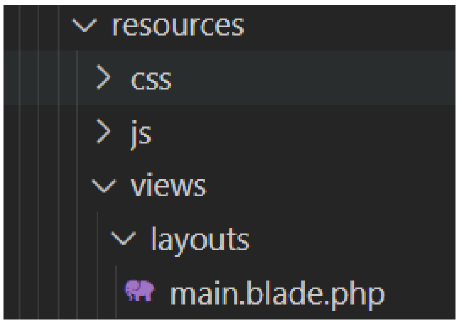
**INFO  Server running on [http://127.0.0.1:8000]**

**You can now view on the browser to render the welcome page.**

After this follow the folder **resources->views**, you can observe the file welcome.blade.php inside it. This is the file which you are viewing in the web browser in which the server is running.

***[*Note: In the xampp application, your apache server should be in running state.]***

Now, inside a views folder, make a layout folder and create a file ***main.blade.php***

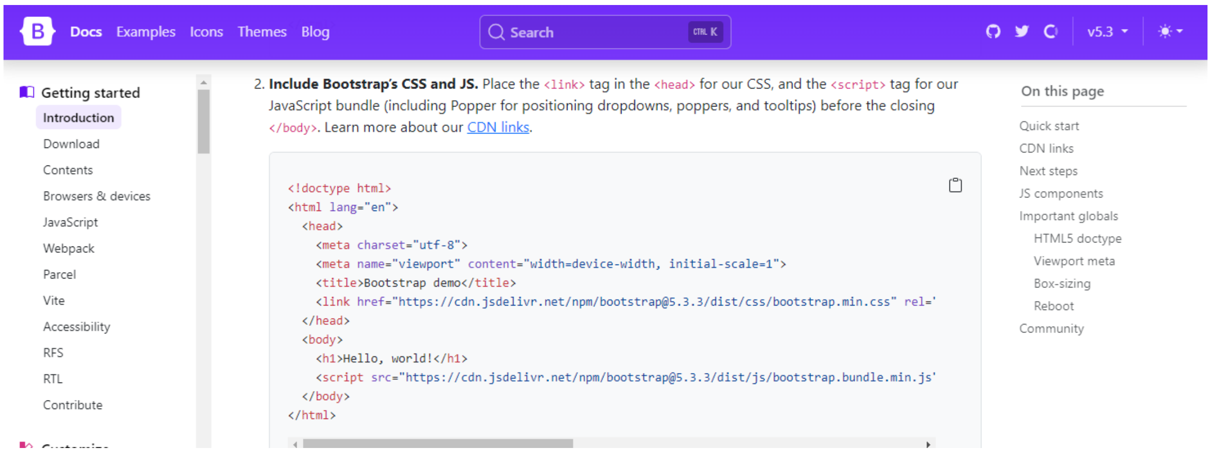


The content inside this file is

**<u>main.blade.php</u>**

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    @stack('head') <!--this helps to create a stack-->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min
.js" integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
  </head>
  <body>
    <div class="bg-dark">
        <div class="container py-3"> <!-- container with padding 3 -->
            <div class="h1 text-white">Todo List App</div>
        </div>
    </div>
    @yield('main-section')
  </body>
</html>
```

Here, we can use directly use a bootstrap CDN link via bootstrap website(*[https://getbootstrap.com/docs/5.3/getting-started/introduction/](https://getbootstrap.com/docs/5.3/getting-started/introduction/)*).



2. **Include Bootstrap's CSS and JS.** Place the `<link>` tag in the `<head>` for our CSS, and the `<script>` tag for our JavaScript bundle (including Popper for positioning dropdowns, poppers, and tooltips) before the closing `</body>`. Learn more about our CDN links.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel='
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js'
  </body>
</html>
```

Now, create further files. welcome.blade.php, create.blade.php, update.blade.php. The information in each files are.

## <u>welcome.blade.php</u>

```
@extends('\layouts.main') <!-- Extending layout\main -->

@push('head')<!—we have created a stack and pushing this header information-->
<title>Todo List App</title>
@endpush

@section('main-section')
<div class="container">
    <div class="d-flex justify-content-between align-items-center my-5">
      <!—flex-direction view with Margin 5-->
        <div class="h2">All Todos</div>
        <a href="{{route("todo.create")}}" class="btn btn-primary btn-lg">Add 
Todo</a>
    </div>
    <!-- {{print_r($todos)}} -->
    <table class="table table-stripped table-dark">
        <tr>
            <th>Task Name</th>
            <th>Description</th>
            <th>Due date</th>
            <th>Action</th>
        </tr>
        @foreach($todos as $todo)
            <tr valign="middle">
                <td>{{$todo->name}}</td>
                <td>{{$todo->work}}</td>
                <td>{{$todo->duedate}}</td>
                <td>
                    <a href="{{route("todo.edit",$todo->id)}}" class="btn btn-
success btn-sm">Update</a>
                    <a href="{{route("todo.delete",$todo->id)}}" class="btn
btn-danger btn-sm">Delete</a>
                </td>
            </tr>
        @endforeach
    </table>
</div>

@endsection
```

**<u>create.blade.php</u>**

```
@extends('\layouts.main')
@push('head')
<title>Add Todo </title>
@endpush

@section('main-section')
<div class="container">
    <div class="d-flex justify-content-between align-items-center my-5"> <!--
Margin 5-->
        <div class="h2">Add Todo</div>
        <a href="{{route("todo.home")}}" class="btn btn-primary btn-
lg">Back</a>
    </div>
    <!-- {{print_r($errors->all())}} -->
    <div class="card">
        <div class="card-body">
            <form action="{{route("todo.store")}}" method="post">
                @csrf
                <label for="" class="form-label mt-4">Task Name</label><!--
mt-4 = margin 4 -->
                <input type="text" name="name" class = "form-control" id="">
                    <div class="text-danger">
                        @error('name')
                            {{$message}}
                        @enderror
                    </div>
                <label for="" class="form-label mt-4">Description</label>
                <input type="text" name="work" class = "form-control" id="">
                <div class="text-danger">
                        @error('work')
                            {{$message}}
                        @enderror
                    </div>
                <label for="" class="form-label mt-4">Due Date</label>
                <input type="date" name="duedate" class = "form-control"
id="">
                <div class="text-danger">
                        @error('duedate')
                            {{$message}}
                        @enderror
                    </div>
                <button class="btn btn-primary btn-lg mt-4">Add Todo</button>
            </form>
        </div>
    </div>
</div>

@endsection
```

## update.blade.php

```
@extends('\layouts.main')
@push('head')
<title>Update Todo</title>
@endpush

@section('main-section')
<div class="container">
    <div class="d-flex justify-content-between align-items-center my-5"> <!--
Margin 5-->
        <div class="h2">Update Todo</div>
        <a href="{{route("todo.home")}}" class="btn btn-primary btn-
lg">Back</a>
    </div>
    <!-- {{print_r($todo)}} -->
    <!-- {{print_r($errors->all())}} -->
    <div class="card">
        <div class="card-body">
            <form action="{{route("todo.updateData")}}" method="post">
                @csrf
                <label for="" class="form-label mt-4">Task Name</label><!--
mt-4 = margin 4 -->
                <input type="text" name="name" class = "form-control" id=""
value="{{$todo->name}}">
                <label for="" class="form-label mt-4">Description</label>
                <input type="text" name="work" class = "form-control" id=""
value="{{$todo->work}}">
                <label for="" class="form-label mt-4">Due Date</label>
                <input type="date" name="duedate" class = "form-control" id=""
value="{{$todo->duedate}}">
                <input type="number" name="id" value="{{$todo->id}}" hidden>
                <button class="btn btn-primary btn-lg mt-4">Update
Todo</button>
            </form>
        </div>
    </div>
</div>

@endsection
```

Again, goto .env file and edit the following information about database.

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

Choose database by following a ***[http://localhost/phpmyadmin](http://localhost/phpmyadmin)*** and creating a database. In this case, **todoApp** is my database name.

In folder **database->migrations**, you can see the **create_todos_table** schema add the fields, name, work and duedate like this.

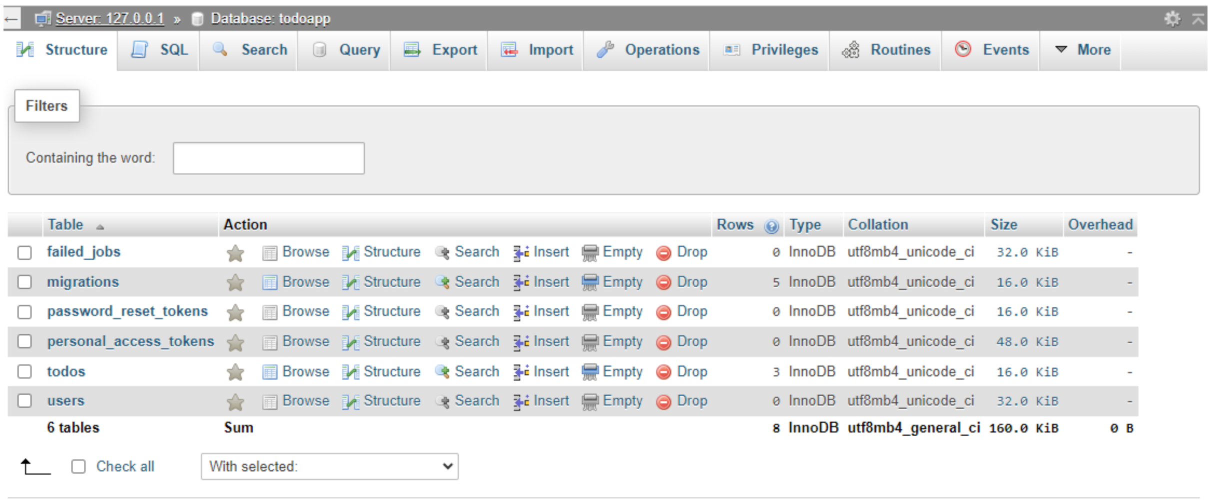
```
    public function up(): void
    {
        Schema::create('todos', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('work');
            $table->string('duedate');
            $table->timestamps();
        });
    }
```

After that, run this command to create a physical table in the database selected.

**php artisan make:model todos –migration**

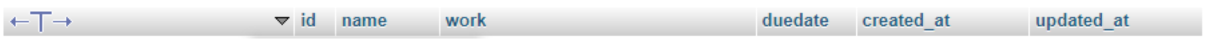
The table can be viewed as like this below.

# Web Technology-PHP Laravel Todo Doc. for basic CRUD Op.



| Table | Action | Rows | Type | Collation | Size | Overhead |
|---|---|---|---|---|---|---|
| failed_jobs | Browse Structure Search Insert Empty Drop | 0 | InnoDB | utf8mb4_unicode_ci | 32.0 KiB | - |
| migrations | Browse Structure Search Insert Empty Drop | 5 | InnoDB | utf8mb4_unicode_ci | 16.0 KiB | - |
| password_reset_tokens | Browse Structure Search Insert Empty Drop | 0 | InnoDB | utf8mb4_unicode_ci | 16.0 KiB | - |
| personal_access_tokens | Browse Structure Search Insert Empty Drop | 0 | InnoDB | utf8mb4_unicode_ci | 48.0 KiB | - |
| todos | Browse Structure Search Insert Empty Drop | 3 | InnoDB | utf8mb4_unicode_ci | 16.0 KiB | - |
| users | Browse Structure Search Insert Empty Drop | 0 | InnoDB | utf8mb4_unicode_ci | 32.0 KiB | - |
| 6 tables | Sum | 8 | InnoDB | utf8mb4_general_ci | 160.0 KiB | 0 B |

And the table field of todos can be viewed as:



| ←T→ | id | name | work | duedate | created_at | updated_at |
|---|---|---|---|---|---|---|

After that, make a todocontroller using the following commands.

**php artisan make:controller todosController**

In, **App->models-><u>todos.php</u>**

Add protected variables like this.

```
class todos extends Model
{
    use HasFactory;
    protected $table='todos';
    protected $primary_key='id';
}
```

Now, in controller section.

**App->http->controllers-><u>todosController.php</u>**

Following information should be contained, one for default view of todo page, another for adding todo, another for editing the field value and finally updating to database, and lastly deleting any unwanted todo.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\todos; // using the todos model for database operations.
```

```
class todosController extends Controller
{
    public function index(){
        $todos=todos::all();
        $data=compact('todos');
        return view("welcome")->with($data);
    }

    public function store(Request $request){
        // print_r($request->all());
        $request->validate(
            [
                'name'=>'required',
                'work'=>'required',
                'duedate'=>'required'
            ]
            );
            // echo " fields are vaidated";
            $todo = new todos;
            $todo->name=$request['name'];
            $todo->work=$request['work'];
            $todo->duedate=$request['duedate'];
            $todo->save();
            // if($todo->save())
            //     echo "Record inserted successfully";
            return redirect(route("todo.home"));
        }

    public function delete($id){
        todos::find($id)->delete();
        return redirect(route("todo.home"));
    }

    public function edit($id){
        $todo=todos::find($id);
        $data=compact('todo');
        return view("update")->with($data);
    }

    public function updateData(Request $request){
        $request->validate(
            [
                'name'=>'required',
                'work'=>'required',
                'duedate'=>'required'
            ]
            );
            $id = $request['id'];
            $todo=todos::find($id);

            $todo->name=$request['name'];
            $todo->work=$request['work'];
            $todo->duedate=$request['duedate'];
            $todo->save();
            return redirect(route("todo.home"));
    }
}
```

Now, everything should be working finally only if you have routed each view and controller significantly with model in control. For this you have to set up route for each.

For this, visit **routes->web.php**

And, add these appropriate paths for each request from views.

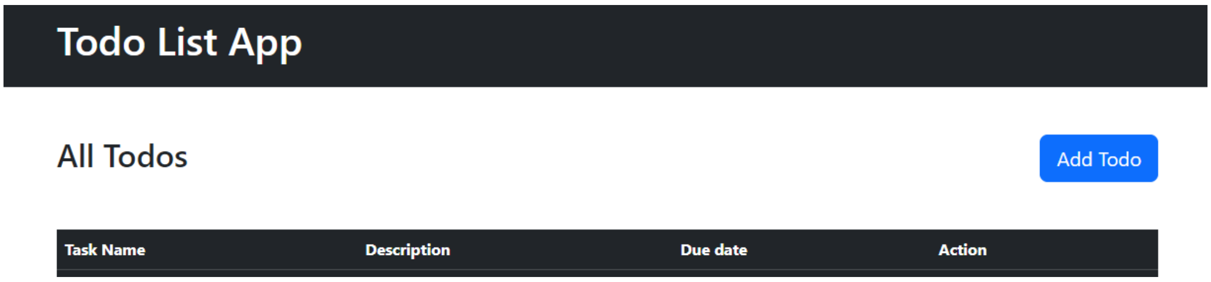
```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\todosController;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/


//homepage todo route
Route::get('/',[todosController::class, 'index'])->name("todo.home");

//create todo route
Route::get('/create', function () {
    return view('create');
})->name("todo.create");

//edit todo route
Route::get('/edit/{id}',[todosController::class,'edit'])->name("todo.edit");
```

```
//update todo route
Route::post('/update', [todosController::class,'updateData'])-
>name("todo.updateData");

//store todo route
Route::post('/create', [todosController::class,'store'])->name("todo.store");

//delete toto route
Route::get('/delete/{id}', [todosController::class,'delete'])-
>name("todo.delete");
```

Now, write command ***php artisan serve*** to run the application.

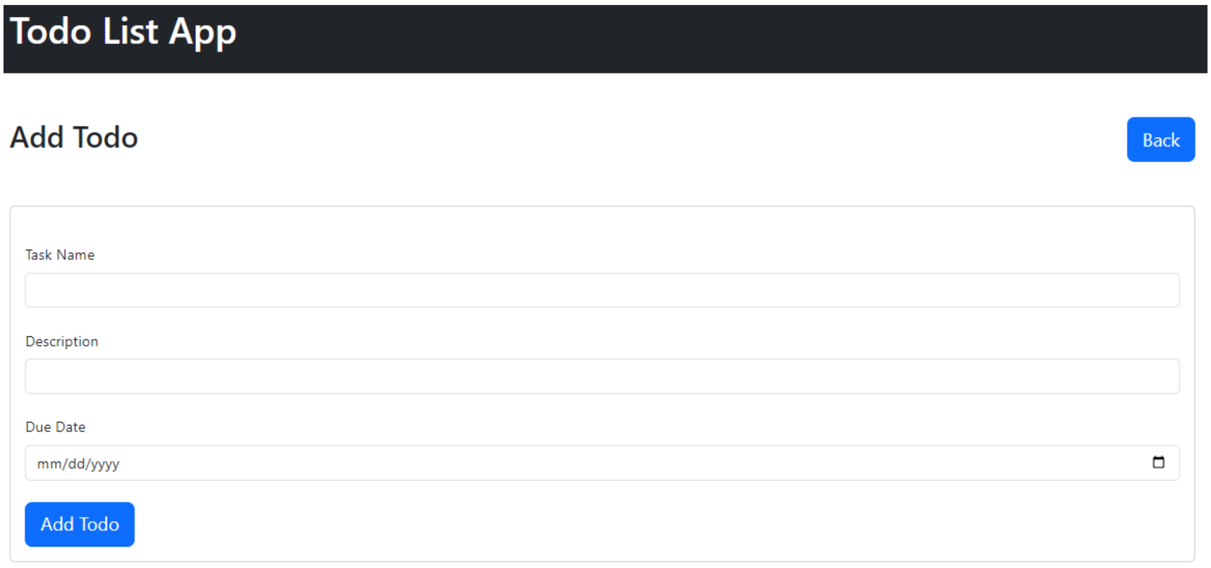
You can observe the following output homepage.

Initially, there are no todos created so following is observed.



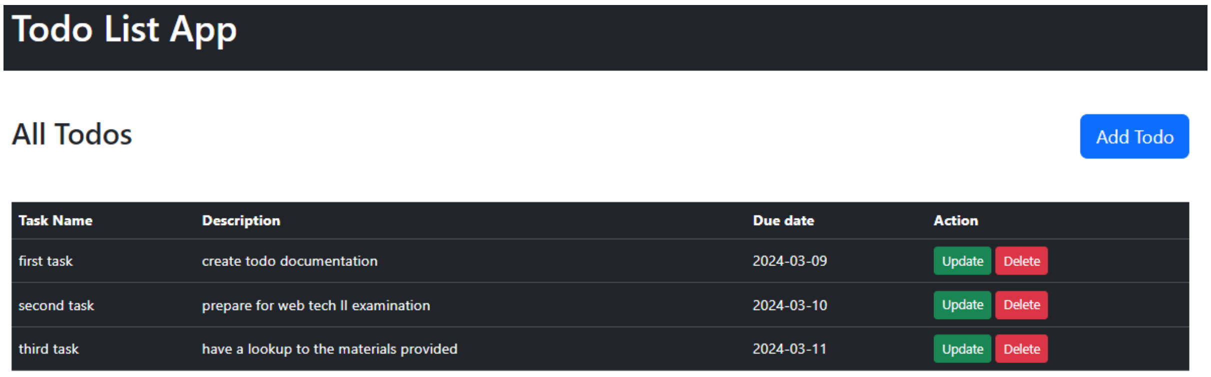
| Task Name | Description | Due date | Action |
|---|---|---|---|

Now, by clicking on Add Todo button we can create a todos.



Lets, create some todos as follows. These created todos can be listed in index/home page of todos as like this.

# Web Technology-PHP Laravel Todo Doc. for basic CRUD Op.

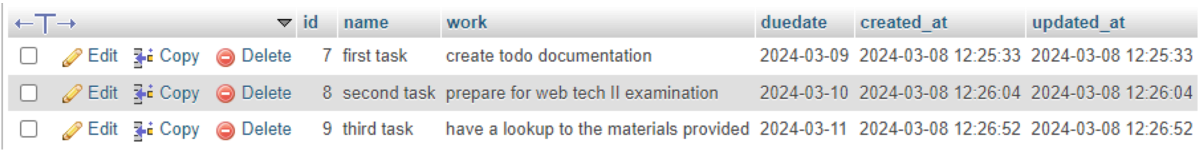


Todo List App

All Todos

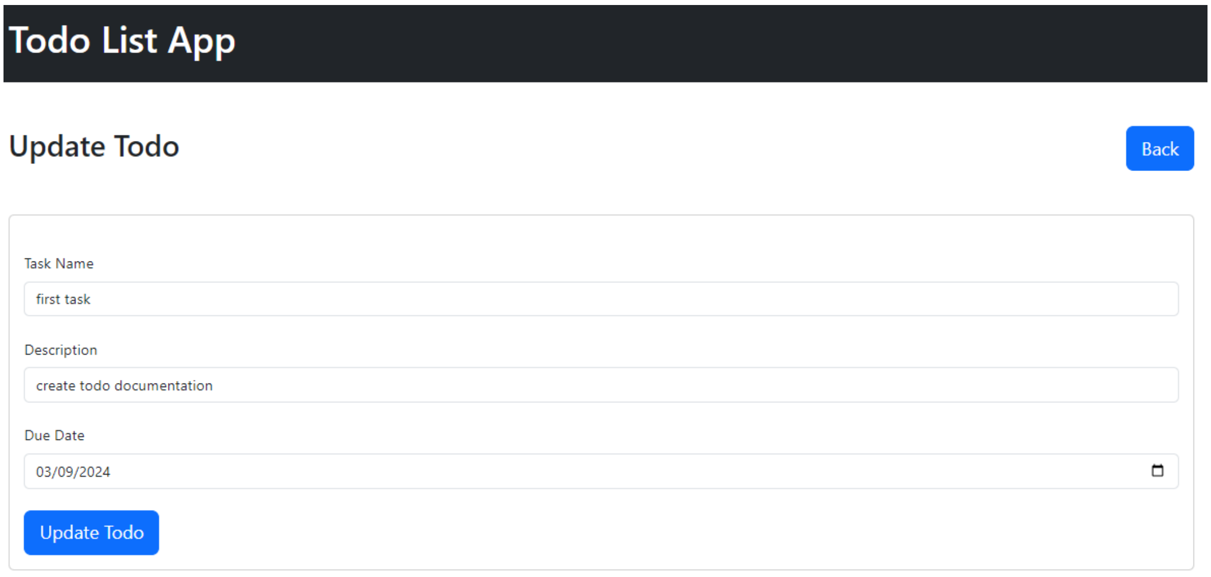
| Task Name | Description | Due date | Action |
|---|---|---|---|
| first task | create todo documentation | 2024-03-09 | Update Delete |
| second task | prepare for web tech II examination | 2024-03-10 | Update Delete |
| third task | have a lookup to the materials provided | 2024-03-11 | Update Delete |

Also, these can be seen on database as follow.



| ←T→ | id | name | work | duedate | created_at | updated_at |
|---|---|---|---|---|---|---|
| Edit Copy Delete | 7 | first task | create todo documentation | 2024-03-09 | 2024-03-08 12:25:33 | 2024-03-08 12:25:33 |
| Edit Copy Delete | 8 | second task | prepare for web tech II examination | 2024-03-10 | 2024-03-08 12:26:04 | 2024-03-08 12:26:04 |
| Edit Copy Delete | 9 | third task | have a lookup to the materials provided | 2024-03-11 | 2024-03-08 12:26:52 | 2024-03-08 12:26:52 |

We can now click on a update todo to update any todo list and modify fields.



Todo List App

Update Todo

Task Name
first task

Description
create todo documentation

Due Date
03/09/2024

Likewise, we can do a delete operation with just a click on a delete button.

From these all, a basic CRUD operation in Todo application can be demonstrated using PHP Laravel and MySql database server.