Student CRUD API using Laravel Framework

Presented by: Venkat

Course: Backend Web Development

Instructor: Dr. Sheik Anik

What is a PHP Framework?

A PHP framework provides a foundational structure for building web applications with PHP. It standardizes development, promotes code reuse, and streamlines the process by adhering to best practices.

Accelerate Development

Pre-built modules and functions reduce the need for writing code from scratch, significantly speeding up development.

Structured Codebase

Enforces architectural patterns like MVC, creating organized, consistent, and manageable code, especially for large projects.

Enhanced Security

Built-in security features protect against common vulnerabilities like XSS, CSRF, and SQL injection, making applications more robust.

Easier Maintenance & Scaling

Standardized code simplifies debugging, updates, and adding new features, ensuring long-term maintainability and scalability.

Accelerate Development

Laravel's comprehensive set of pre-built modules and functions means developers don't have to write everything from scratch, allowing for rapid application creation.

> "Less coding, faster results."

❌ Traditional PHP (Example)

```
$conn = mysqli_connect('localhost', 'root', '',
'student_system');
$result = mysqli_query($conn, "SELECT * FROM
students");
```

✅ Laravel Way (Example)

```
$students = Student::all();
```

- **Significant Time Savings**

  Automated tasks and built-in features dramatically cut down development hours.

- **Enhanced Code Quality**

  Utilizing well-tested framework components inherently reduces potential bugs and errors.

- **Focus on Core Logic**

  Developers can dedicate more time to implementing unique business requirements and features.

Structured Codebase

Every developer instantly knows where to look and what to change thanks to clear conventions.

Laravel enforces the **MVC architecture** (Model–View–Controller) pattern, keeping your codebase organized, consistent, and easy to maintain.

Example: Folder Organization

```
app/
  ├── Http/Controllers/StudentController.php
  ├── Models/Student.php
routes/
  └── api.php
resources/
  └── views/
```

Organized

Clear separation of application logic, data handling, and user interface elements.

Consistent

A standard structure is maintained across all Laravel applications, aiding developer onboarding.

Manageable

Simplifies debugging, updates, and collaborative team development on larger projects.

🔒 Enhanced Security

Laravel provides robust, built-in security features that protect your applications from common web vulnerabilities

XSS (Cross-Site Scripting)

Prevents attackers from injecting malicious scripts into web pages viewed by other users.

CSRF (Cross-Site Request Forgery)

Protects against unauthorized commands transmitted from a trusted user's browser.

SQL Injection

Safeguards your database from malicious SQL queries that could expose sensitive data.

Without Laravel

⚠️ ❌ Without Laravel (Vulnerable PHP)

```
// User input comes directly from the URL
$id = $_GET['id'];
$query = "SELECT * FROM students WHERE id = $id";
$result = mysqli_query($conn, $query);
```

🔓 If a hacker visits `?id=1 OR 1=1` in the URL, all student records get exposed in an SQL Injection Attack!

🛡️ ✅ With Laravel (Safe by Default)

```
// Laravel uses prepared statements automatically
$student = Student::find($id);
```

💡 **What Happens Behind the Scenes:**

- Eloquent ORM automatically binds variables safely.
- No raw SQL concatenation, preventing injection.
- User input is sanitized before database interaction.

Easier Maintenance & Scaling

Laravel's design principles, including its standardized structure and modular architecture, significantly simplify the ongoing maintenance and future scaling of web applications.

Simple to Debug

Consistent patterns and clear error reporting allow developers to quickly identify and resolve issues, minimizing downtime.

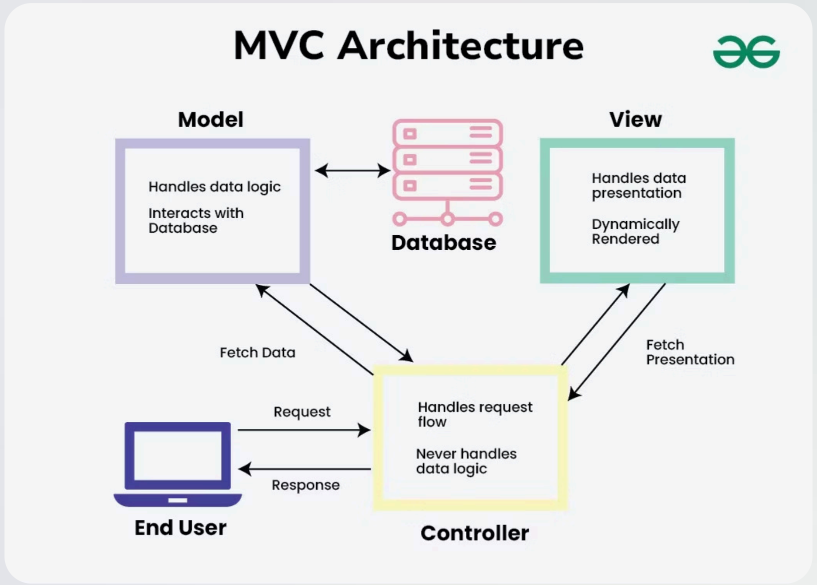
Easy to Update

Adherence to best practices and semantic versioning makes upgrading Laravel versions and dependencies smooth and predictable.

Quick to Extend

Modular components and a rich ecosystem enable seamless integration of new features and functionalities as needs evolve.

This foundational stability ensures that your application remains robust, adaptable, and cost-effective as it grows and evolves over time.



Understanding Laravel's MVC Architecture

Model

Handles database operations and data structure

View

Returns output—JSON data for the API

Controller

Manages application logic and request handling

Laravel is a modern PHP framework based on the MVC architecture, providing elegant syntax and powerful tools for web development.

Composer — Laravel’s Dependency Manager

Composer installs, updates, and manages all the external packages

💻 Basic Commands

```
# Install Laravel project
composer create-project laravel/laravel student-crud-laravel

# Install dependencies
composer install

# Update all packages
composer update
```

No manual setup required — Composer does the heavy lifting, ensuring your project is ready to run with minimal fuss.

> Without Composer, Laravel wouldn’t be plug-and-play — Composer makes Laravel possible
>
> 📦 Automatically installs Laravel’s required packages
>
> 🔄 Keeps dependencies up to date
>
> 🧩 Manages libraries
>
> 🧱 Ensures your project stays consistent across all systems

Framework Setup

01

Install Dependencies

Install XAMPP for PHP and MySQL, and Composer as Laravel's dependency manager

02

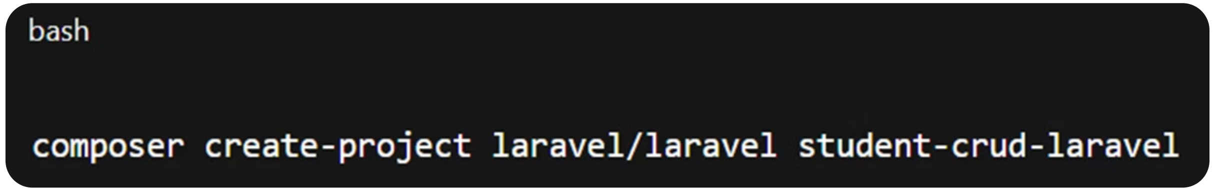
Create Laravel Project

```
composer create-project laravel/laravel student-crud-laravel
```

03

Configure Database

Connect to MySQL database `student_system` using the .env file

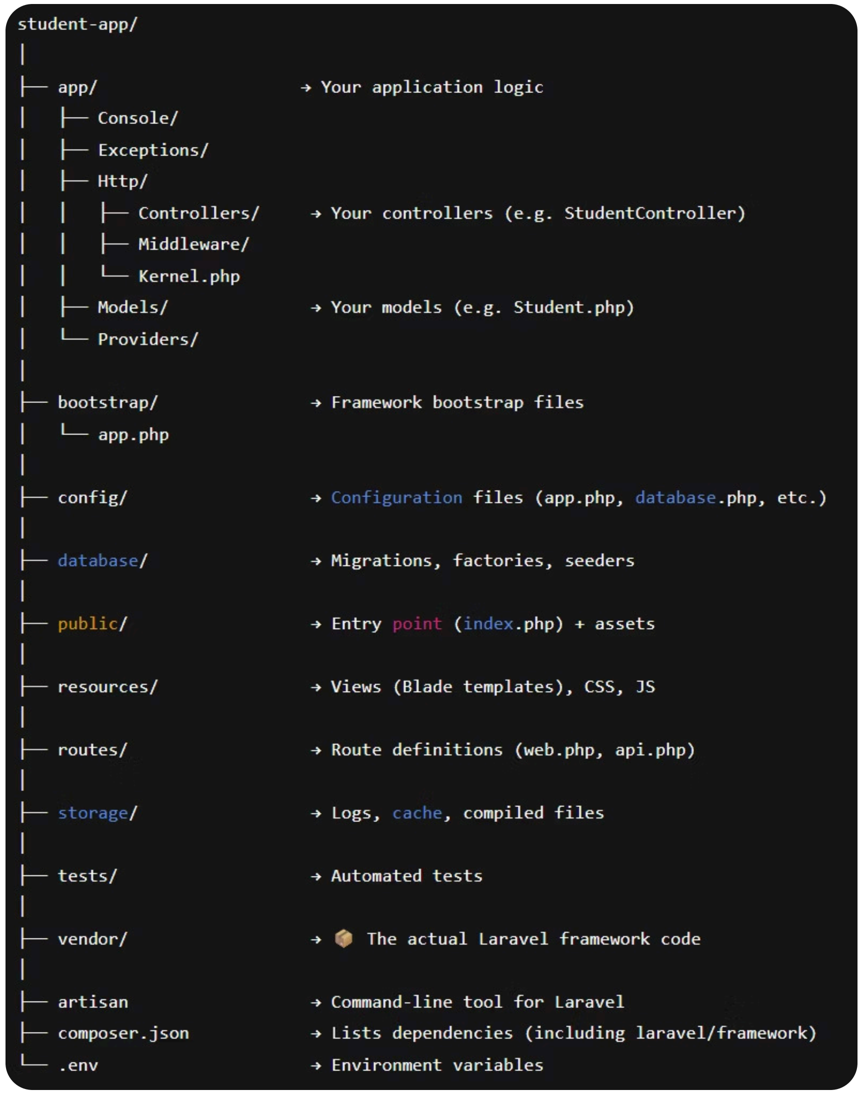


bash

```bash
composer create-project laravel/laravel student-crud-laravel
```

Laravel Project Structure (Pure Form)

Here's what a **fresh Laravel installation** looks like before you write any code:



```
student-app/
│
├── app/                    → Your application logic
│   ├── Console/
│   ├── Exceptions/
│   ├── Http/
│   │   ├── Controllers/     → Your controllers (e.g. StudentController)
│   │   ├── Middleware/
│   │   └── Kernel.php
│   ├── Models/              → Your models (e.g. Student.php)
│   └── Providers/
│
├── bootstrap/               → Framework bootstrap files
│   └── app.php
│
├── config/                  → Configuration files (app.php, database.php, etc.)
│
├── database/                → Migrations, factories, seeders
│
├── public/                  → Entry point (index.php) + assets
│
├── resources/               → Views (Blade templates), CSS, JS
│
├── routes/                  → Route definitions (web.php, api.php)
│
├── storage/                 → Logs, cache, compiled files
│
├── tests/                   → Automated tests
│
├── vendor/                  → 📦 The actual Laravel framework code
│
├── artisan                  → Command-line tool for Laravel
├── composer.json            → Lists dependencies (including laravel/framework)
└── .env                     → Environment variables
```

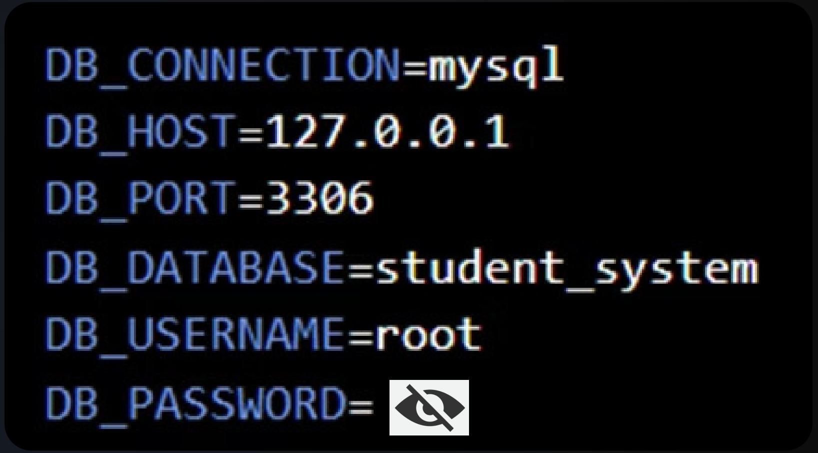
Pure Laravel Code Lives in

```
vendor/laravel/framework/
```

If you open that directory, you'll see Laravel in its **raw form**.

Example:

```
vendor/laravel/framework/src/Illuminate/
```



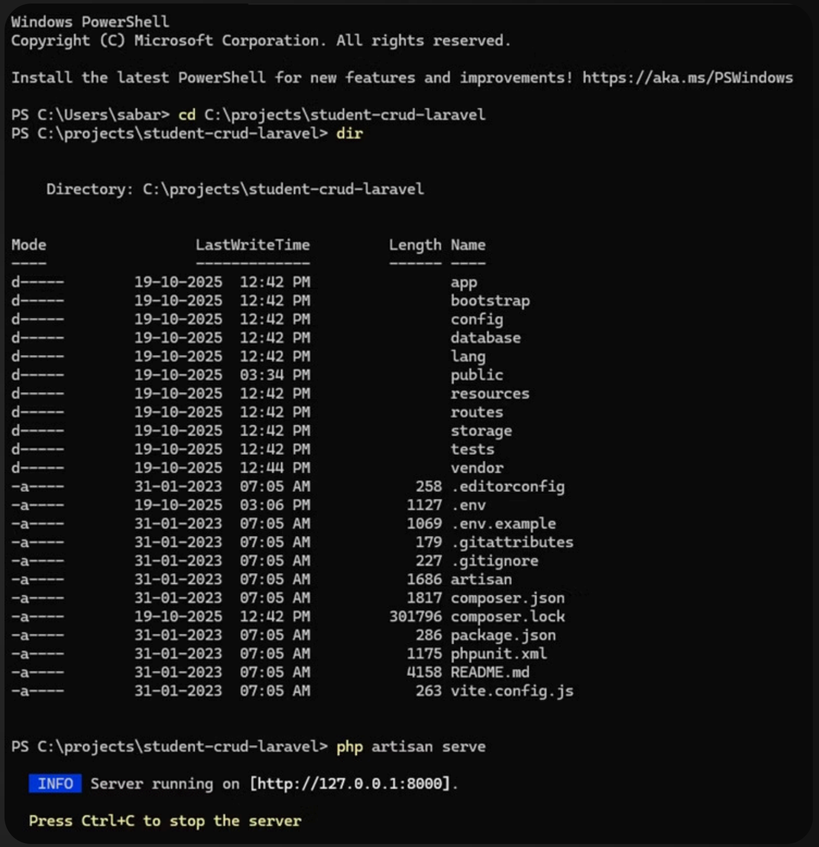
```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=student_system
DB_USERNAME=root
DB_PASSWORD=
```

Database Configuration

.env File Settings

```
DB_DATABASE=student_system
DB_USERNAME=root
DB_PASSWORD=
```

The .env file stores environment-specific configuration, connecting Laravel to the MySQL database.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\sabar> cd C:\projects\student-crud-laravel
PS C:\projects\student-crud-laravel> dir


    Directory: C:\projects\student-crud-laravel


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        19-10-2025  12:42 PM                app
d-----        19-10-2025  12:42 PM                bootstrap
d-----        19-10-2025  12:42 PM                config
d-----        19-10-2025  12:42 PM                database
d-----        19-10-2025  12:42 PM                lang
d-----        19-10-2025  03:34 PM                public
d-----        19-10-2025  12:42 PM                resources
d-----        19-10-2025  12:42 PM                routes
d-----        19-10-2025  12:42 PM                storage
d-----        19-10-2025  12:42 PM                tests
d-----        19-10-2025  12:44 PM                vendor
-a----        31-01-2023  07:05 AM            258 .editorconfig
-a----        19-10-2025  03:06 PM           1127 .env
-a----        31-01-2023  07:05 AM           1069 .env.example
-a----        31-01-2023  07:05 AM            179 .gitattributes
-a----        31-01-2023  07:05 AM            227 .gitignore
-a----        31-01-2023  07:05 AM           1686 artisan
-a----        31-01-2023  07:05 AM           1817 composer.json
-a----        19-10-2025  12:42 PM         301796 composer.lock
-a----        31-01-2023  07:05 AM            286 package.json
-a----        31-01-2023  07:05 AM           1175 phpunit.xml
-a----        31-01-2023  07:05 AM           4158 README.md
-a----        31-01-2023  07:05 AM            263 vite.config.js


PS C:\projects\student-crud-laravel> php artisan serve

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

  Press Ctrl+C to stop the server
```

Launch Laravel Server

1

Navigate to Project

```
cd C:\projects\student-crud-laravel
```

2

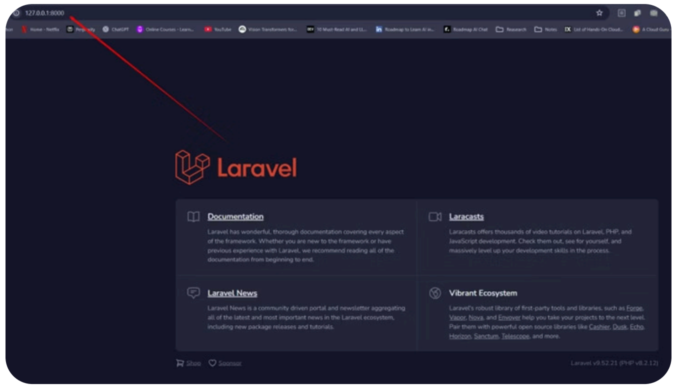
Start Server

```
php artisan serve
```

3

Access Application

Visit http://127.0.0.1:8000 to see Laravel welcome page



Documentation

Laravel has wonderful, thorough documentation covering every aspect of the framework. Whether you are new to the framework or have previous experience with Laravel, we recommend reading all of the documentation from beginning to end.

Laracasts

Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process.

Laravel News

Laravel News is a community driven portal and newsletter aggregating all of the latest and most important news in the Laravel ecosystem, including new package releases and tutorials.

Vibrant Ecosystem

Laravel's robust library of first-party tools and libraries, such as Forge, Vapor, Nova, and Envoyer help you take your projects to the next level. Pair them with powerful open source libraries like Cashier, Dusk, Echo, Horizon, Sanctum, Telescope, and more.

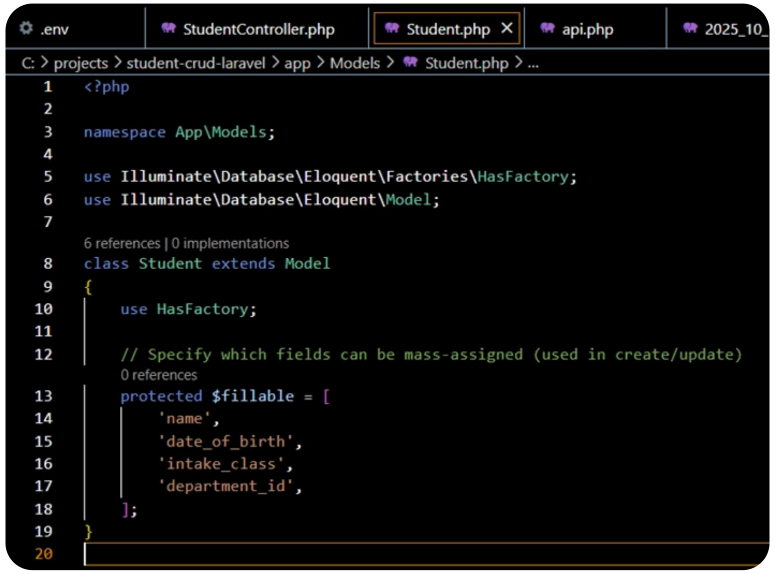
Shop Sponsor

Laravel v9.52.21 (PHP v8.2.12)

Building the CRUD Components

🧩Model — `Student.php`

Defines **database structure** and specifies which fields can be mass-assigned (safe for insert/update).

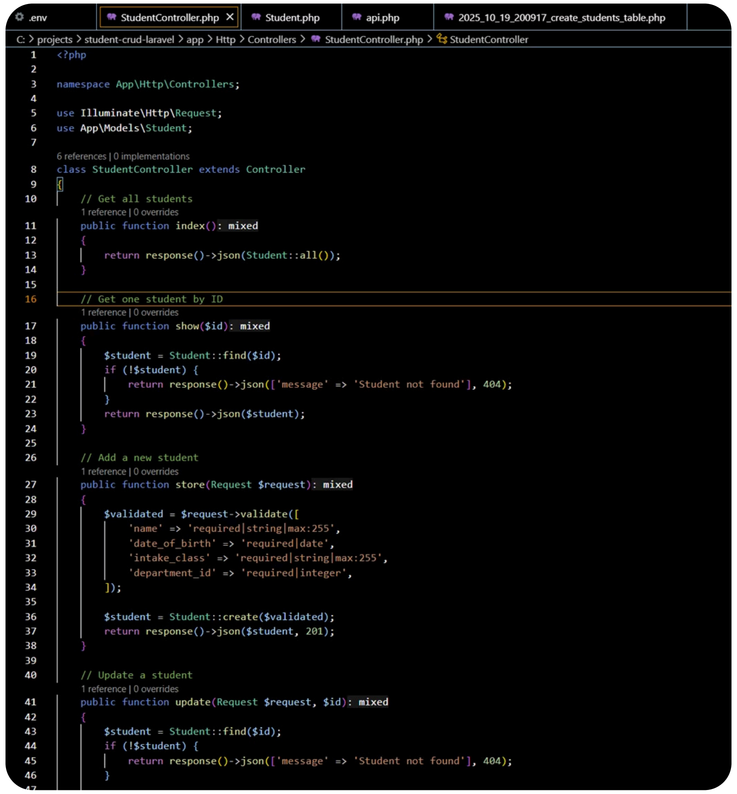

.env
StudentController.php
Student.php
api.php
2025_10_
C: > projects > student-crud-laravel > app > Models > Student.php > ...
6 references | 0 implementations
0 references

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Student extends Model
{
    use HasFactory;

    // Specify which fields can be mass-assigned (used in create/update)
    protected $fillable = [
        'name',
        'date_of_birth',
        'intake_class',
        'department_id',
    ];
}
```

⚙️Controller — `StudentController.php`

Implements **all CRUD actions**

(Create, Read, Update, Delete).


.env
StudentController.php
Student.php
api.php
2025_10_19_200917_create_students_table.php
C: > projects > student-crud-laravel > app > Http > Controllers > StudentController.php > StudentController
6 references | 0 implementations
1 reference | 0 overrides

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Student;

class StudentController extends Controller
{
    // Get all students
    public function index(): mixed
    {
        return response()->json(Student::all());
    }

    // Get one student by ID
    public function show($id): mixed
    {
        $student = Student::find($id);
        if (!$student) {
            return response()->json(['message' => 'Student not found'], 404);
        }
        return response()->json($student);
    }

    // Add a new student
    public function store(Request $request): mixed
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'date_of_birth' => 'required|date',
            'intake_class' => 'required|string|max:255',
            'department_id' => 'required|integer',
        ]);

        $student = Student::create($validated);
        return response()->json($student, 201);
    }

    // Update a student
    public function update(Request $request, $id): mixed
    {
        $student = Student::find($id);
        if (!$student) {
            return response()->json(['message' => 'Student not found'], 404);
        }
```

API Routes Configuration

GET /api/students

List all students

GET /api/students/{id}

Show single student

POST /api/students

Create new student

PUT /api/students/{id}

Update student

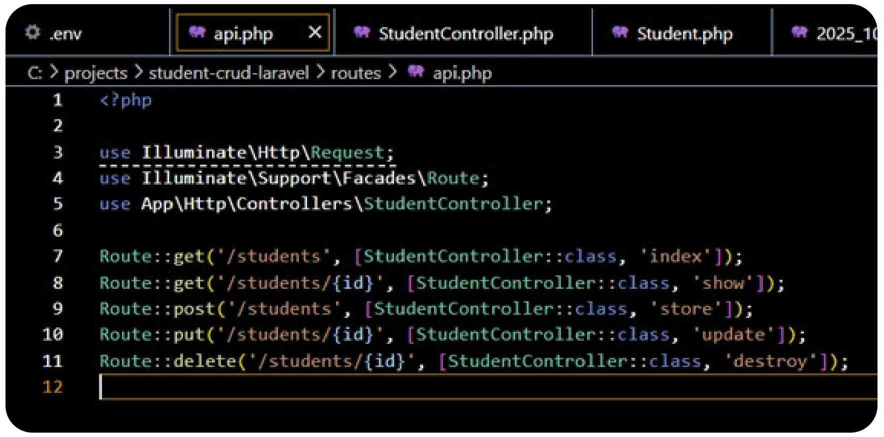
DELETE /api/students/{id}

Delete student

Routes connect URLs to controller methods. Verify with: php artisan route:list

🌐 **API Routes —** routes/api.php

```
Route::apiResource('students', StudentController::class);
```



```php
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\StudentController;

Route::get('/students', [StudentController::class, 'index']);
Route::get('/students/{id}', [StudentController::class, 'show']);
Route::post('/students', [StudentController::class, 'store']);
Route::put('/students/{id}', [StudentController::class, 'update']);
Route::delete('/students/{id}', [StudentController::class, 'destroy']);
```

✅ **Result:**

Laravel instantly generates **all 5 CRUD routes**!

No need to write each manually.

Laravel Student CRUD API

A complete demonstration of building and testing a Laravel CRUD API using VS Code, Tinker, and browser-based verification.

Opening the Terminal

01

Launch Terminal

Press Ctrl+` in VS Code to open the integrated terminal.

02

Verify Path

Ensure terminal shows C:\projects\student-crud-laravel>.

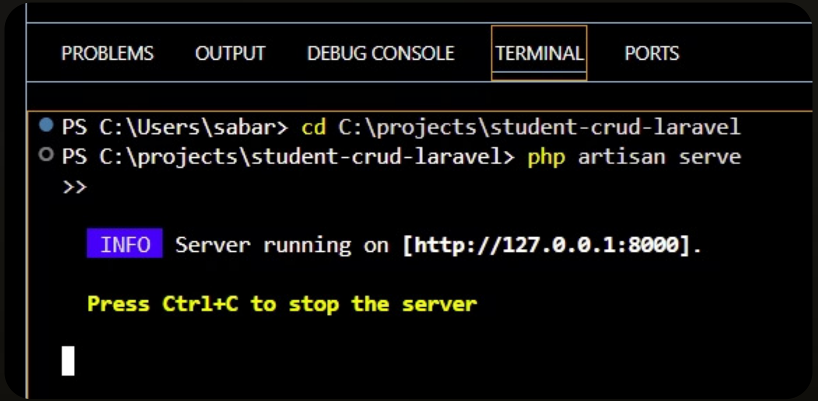
If not, run

cd C:\projects\student-crud-laravel.

03

Start Server

Execute php artisan serve to launch the development server.

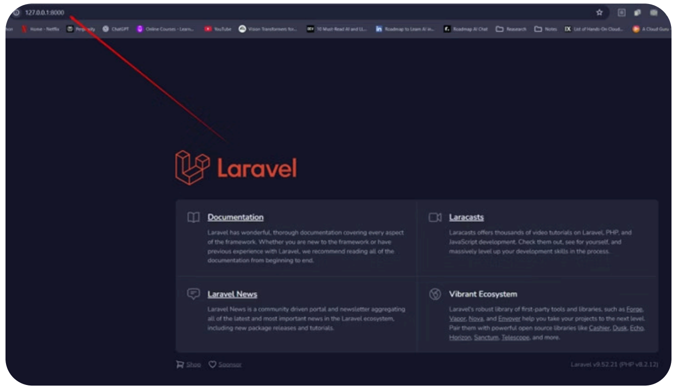


```
PS C:\Users\sabar> cd C:\projects\student-crud-laravel
PS C:\projects\student-crud-laravel> php artisan serve
>>

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

   Press Ctrl+C to stop the server
```

Accessing the Application

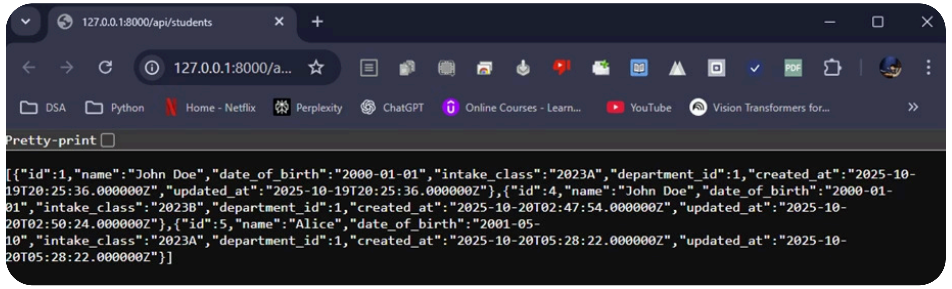
Server URL

Open the URL displayed in terminal, typically `http://127.0.0.1:8000`.



API Endpoint

Navigate to `http://127.0.0.1:8000/api/students` to view JSON response in browser.


127.0.0.1:8000/api/students
127.0.0.1:8000/a...
DSA
Python
Home - Netflix
Perplexity
ChatGPT
Online Courses - Learn...
YouTube
Vision Transformers for...
Pretty-print

```
[{"id":1,"name":"John Doe","date_of_birth":"2000-01-01","intake_class":"2023A","department_id":1,"created_at":"2025-10-
19T20:25:36.000000Z","updated_at":"2025-10-19T20:25:36.000000Z"},{"id":4,"name":"John Doe","date_of_birth":"2000-01-
01","intake_class":"2023B","department_id":1,"created_at":"2025-10-20T02:47:54.000000Z","updated_at":"2025-10-
20T02:50:24.000000Z"},{"id":5,"name":"Alice","date_of_birth":"2001-05-
10","intake_class":"2023A","department_id":1,"created_at":"2025-10-20T05:28:22.000000Z","updated_at":"2025-10-
20T05:28:22.000000Z"}]
```

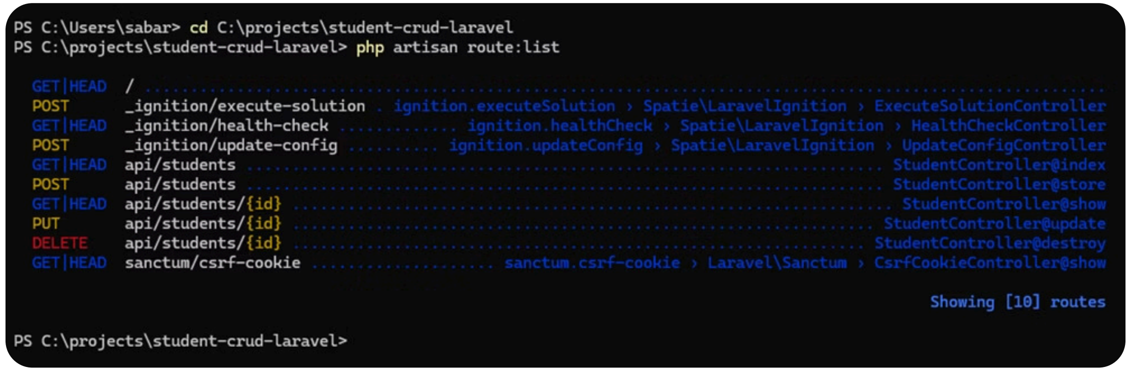
Testing Routes

Confirm API endpoints using Laravel's built-in route list command.

```
php artisan route:list
```

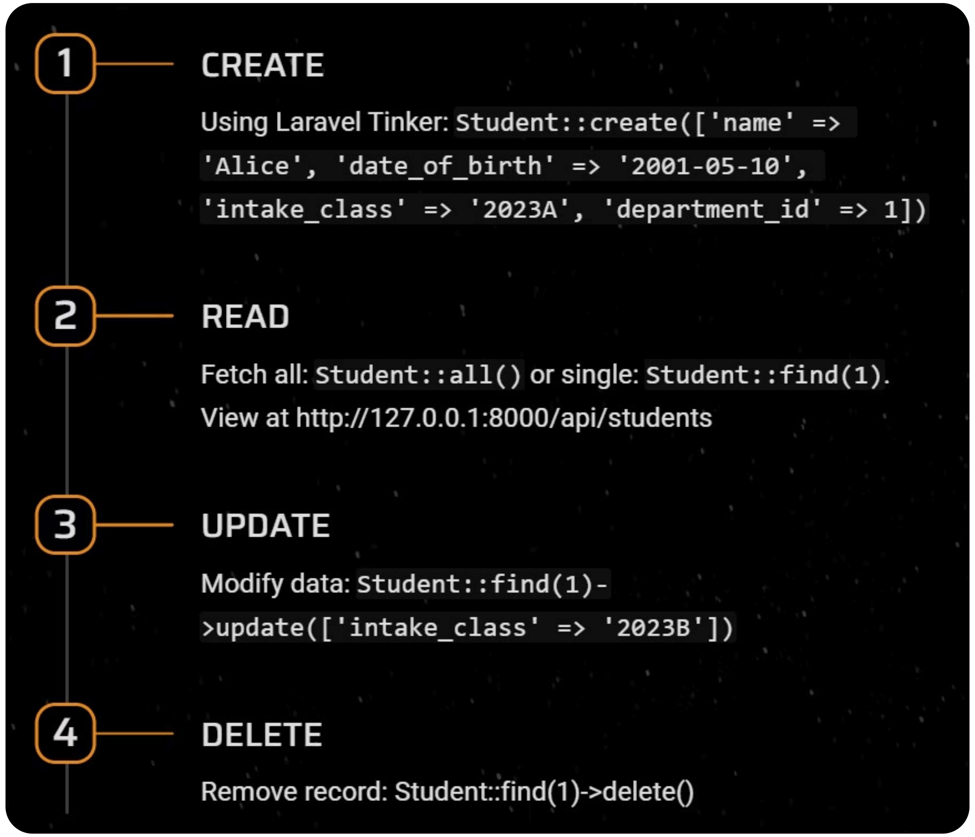
Displays all registered routes, HTTP methods, and their controller actions.

✅ Ensures your CRUD endpoints are active and working.



```
PS C:\Users\sabar> cd C:\projects\student-crud-laravel
PS C:\projects\student-crud-laravel> php artisan route:list

  GET|HEAD  / ...................................................................................................
  POST      _ignition/execute-solution . ignition.executeSolution › Spatie\LaravelIgnition › ExecuteSolutionController
  GET|HEAD  _ignition/health-check ............ ignition.healthCheck › Spatie\LaravelIgnition › HealthCheckController
  POST      _ignition/update-config ......... ignition.updateConfig › Spatie\LaravelIgnition › UpdateConfigController
  GET|HEAD  api/students ...................................................................... StudentController@index
  POST      api/students ...................................................................... StudentController@store
  GET|HEAD  api/students/{id} .................................................................. StudentController@show
  PUT       api/students/{id} ................................................................ StudentController@update
  DELETE    api/students/{id} ............................................................... StudentController@destroy
  GET|HEAD  sanctum/csrf-cookie ................... sanctum.csrf-cookie › Laravel\Sanctum › CsrfCookieController@show

                                                                                                    Showing [10] routes

PS C:\projects\student-crud-laravel>
```

CRUD Operations in Action



1 CREATE

Using Laravel Tinker: `Student::create(['name' => 'Alice', 'date_of_birth' => '2001-05-10', 'intake_class' => '2023A', 'department_id' => 1])`

2 READ

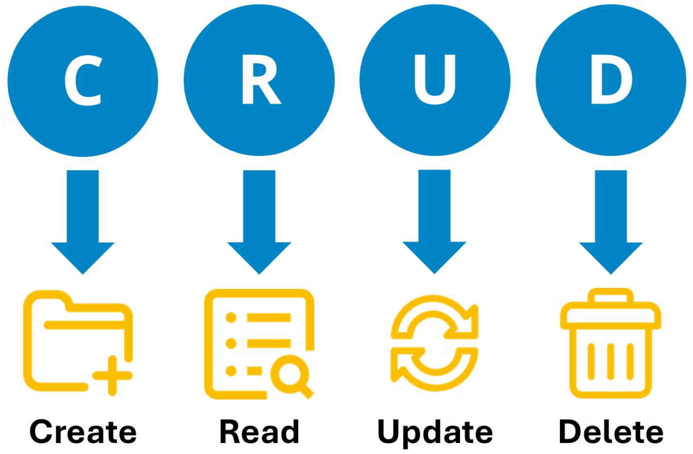
Fetch all: `Student::all()` or single: `Student::find(1)`.
View at http://127.0.0.1:8000/api/students

3 UPDATE

Modify data: `Student::find(1)->update(['intake_class' => '2023B'])`

4 DELETE

Remove record: Student::find(1)->delete()



Demonstrate CRUD (Using Tinker)

CREATE: Add a new student record

```
Run: php artisan tinker

App\Models\Student::create([
  'name' => 'Alice',
  'date_of_birth' => '2001-05-10',
  'intake_class' => '2023A',
  'department_id' => 1
]);
```

READ: Retrieve student records

Read All Students

```
>>> App\Models\Student::all();
```

Read a Single Student

```
>>> App\Models\Student::find(1);
```

UPDATE: Modify a student record

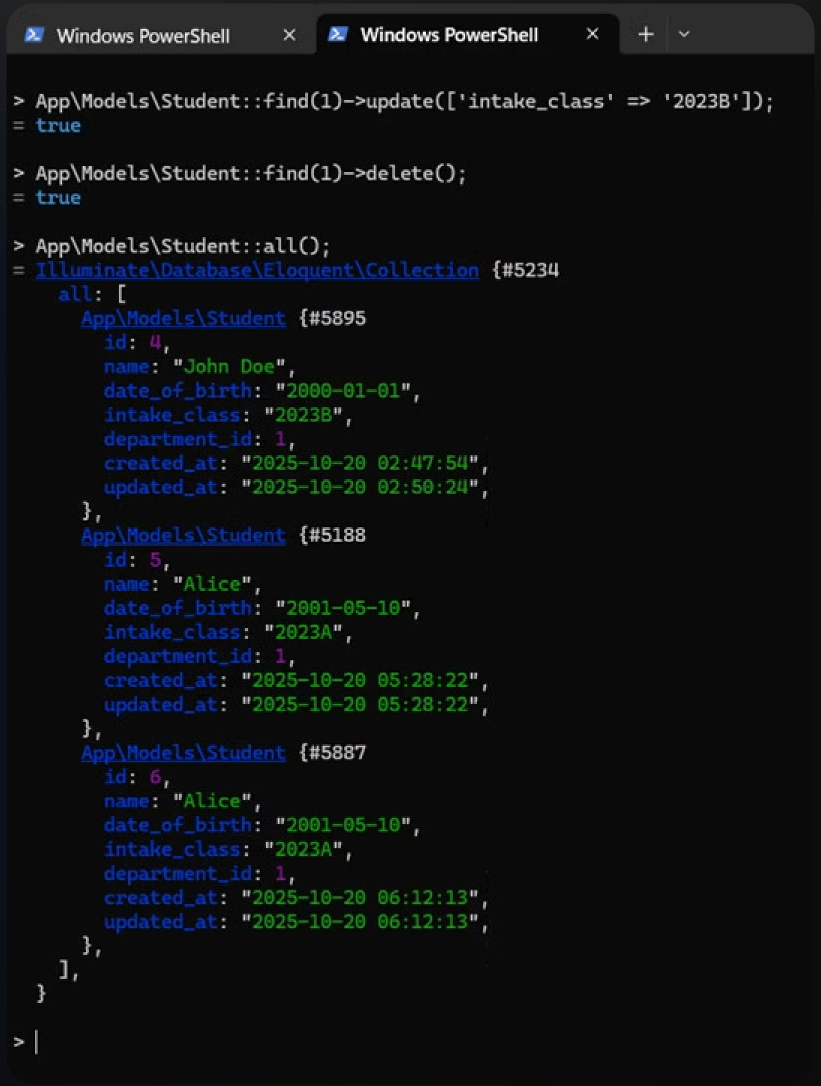
Run:
App\Models\Student::find(1)->update(['intake_class' => '2023B']);

```
>>> App\Models\Student::find(1);
```

DELETE: Remove a student record

Run: App\Models\Student::find(1)->delete();

```
>>> App\Models\Student::find(1);
```



```
> App\Models\Student::find(1)->update(['intake_class' => '2023B']);
= true

> App\Models\Student::find(1)->delete();
= true

> App\Models\Student::all();
= Illuminate\Database\Eloquent\Collection {#5234
    all: [
      App\Models\Student {#5895
        id: 4,
        name: "John Doe",
        date_of_birth: "2000-01-01",
        intake_class: "2023B",
        department_id: 1,
        created_at: "2025-10-20 02:47:54",
        updated_at: "2025-10-20 02:50:24",
      },
      App\Models\Student {#5188
        id: 5,
        name: "Alice",
        date_of_birth: "2001-05-10",
        intake_class: "2023A",
        department_id: 1,
        created_at: "2025-10-20 05:28:22",
        updated_at: "2025-10-20 05:28:22",
      },
      App\Models\Student {#5887
        id: 6,
        name: "Alice",
        date_of_birth: "2001-05-10",
        intake_class: "2023A",
        department_id: 1,
        created_at: "2025-10-20 06:12:13",
        updated_at: "2025-10-20 06:12:13",
      },
    ],
  }

> |
```

CREATE Operation

1

Launch Tinker

Run `php artisan tinker` to open Laravel's interactive shell.

2

Insert Record

Execute create command with student details: name, date_of_birth, intake_class, and department_id.

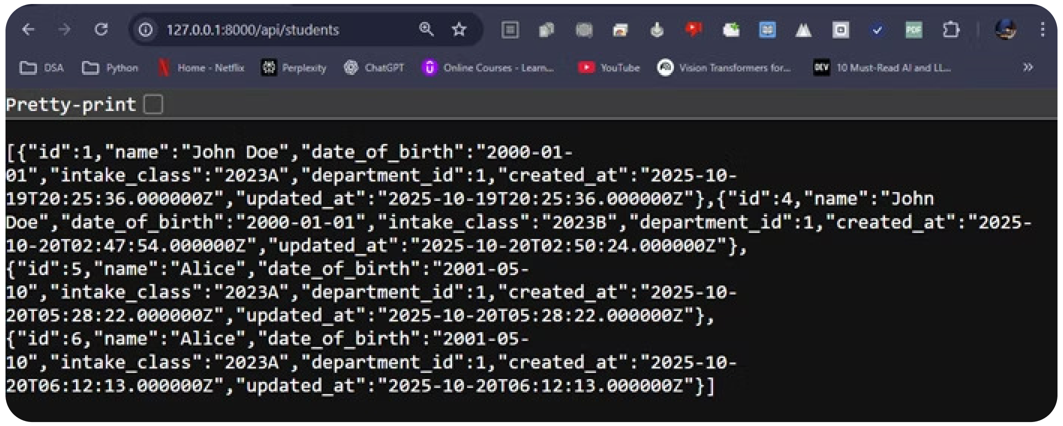
3

Verify in Browser

Visit `/api/students` to see the newly created record in JSON format.

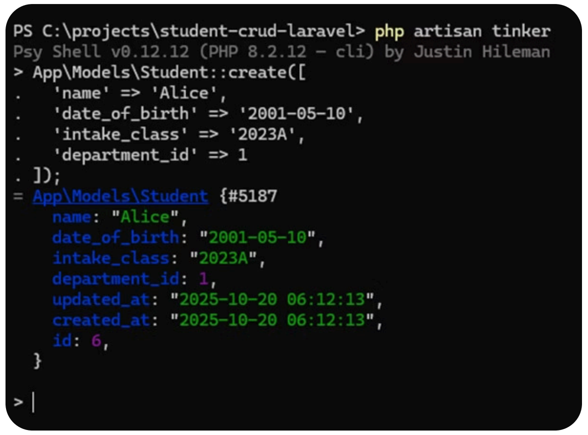
This data is now inserted into my database. Let's check it in the browser.

🖥️ **Go to: http://127.0.0.1:8000/api/students**


127.0.0.1:8000/api/students
DSA
Python
Home - Netflix
Perplexity
ChatGPT
Online Courses - Learn...
YouTube
Vision Transformers for...
10 Must-Read AI and LL...

Pretty-print ☐

```
[{"id":1,"name":"John Doe","date_of_birth":"2000-01-
01","intake_class":"2023A","department_id":1,"created_at":"2025-10-
19T20:25:36.000000Z","updated_at":"2025-10-19T20:25:36.000000Z"},{"id":4,"name":"John
Doe","date_of_birth":"2000-01-01","intake_class":"2023B","department_id":1,"created_at":"2025-
10-20T02:47:54.000000Z","updated_at":"2025-10-20T02:50:24.000000Z"},
{"id":5,"name":"Alice","date_of_birth":"2001-05-
10","intake_class":"2023A","department_id":1,"created_at":"2025-10-
20T05:28:22.000000Z","updated_at":"2025-10-20T05:28:22.000000Z"},
{"id":6,"name":"Alice","date_of_birth":"2001-05-
10","intake_class":"2023A","department_id":1,"created_at":"2025-10-
20T06:12:13.000000Z","updated_at":"2025-10-20T06:12:13.000000Z"}]
```



```
PS C:\projects\student-crud-laravel> php artisan tinker
Psy Shell v0.12.12 (PHP 8.2.12 — cli) by Justin Hileman
> App\Models\Student::create([
.   'name' => 'Alice',
.   'date_of_birth' => '2001-05-10',
.   'intake_class' => '2023A',
.   'department_id' => 1
. ]);
= App\Models\Student {#5187
    name: "Alice",
    date_of_birth: "2001-05-10",
    intake_class: "2023A",
    department_id: 1,
    updated_at: "2025-10-20 06:12:13",
    created_at: "2025-10-20 06:12:13",
    id: 6,
  }

> |
```

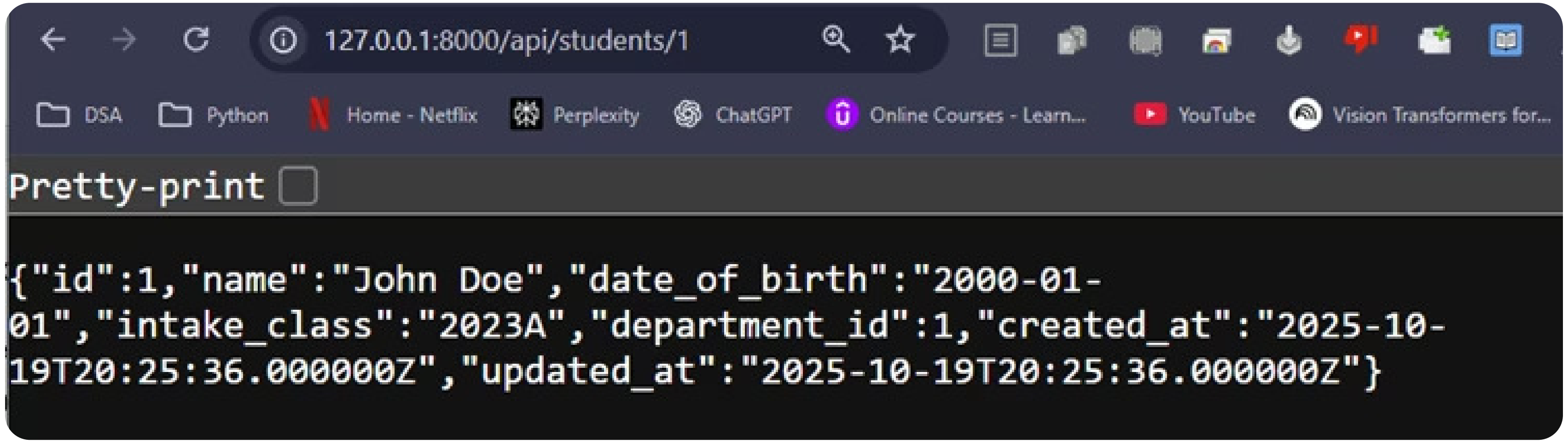
READ Operation

All Students

Access http://127.0.0.1:8000/api/students to retrieve all student records.

Single Student

Use http://127.0.0.1:8000/api/students/1 to fetch a specific student by ID.

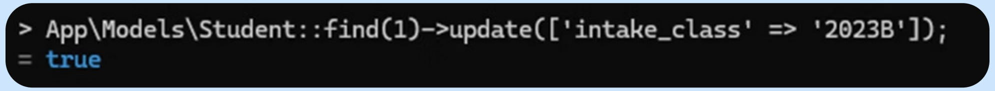

127.0.0.1:8000/api/students/1
DSA
Python
Home - Netflix
Perplexity
ChatGPT
Online Courses - Learn...
YouTube
Vision Transformers for...
Pretty-print

```
{"id":1,"name":"John Doe","date_of_birth":"2000-01-
01","intake_class":"2023A","department_id":1,"created_at":"2025-10-
19T20:25:36.000000Z","updated_at":"2025-10-19T20:25:36.000000Z"}
```

UPDATE Operation

Modify Data

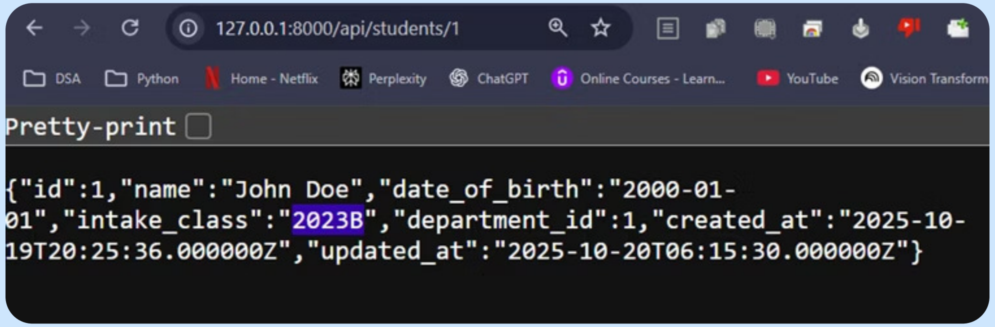
Use Tinker to update student record: `Student::find(1)->update(['intake_class' => '2023B'])`



```
> App\Models\Student::find(1)->update(['intake_class' => '2023B']);
= true
```

Verify Changes

Refresh browser to see updated data reflected in the JSON response.



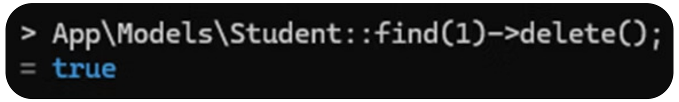
```
{"id":1,"name":"John Doe","date_of_birth":"2000-01-
01","intake_class":"2023B","department_id":1,"created_at":"2025-10-
19T20:25:36.000000Z","updated_at":"2025-10-20T06:15:30.000000Z"}
```

DELETE Operation

1

Execute Delete

Run `Student::find(1)->delete()` in Tinker t

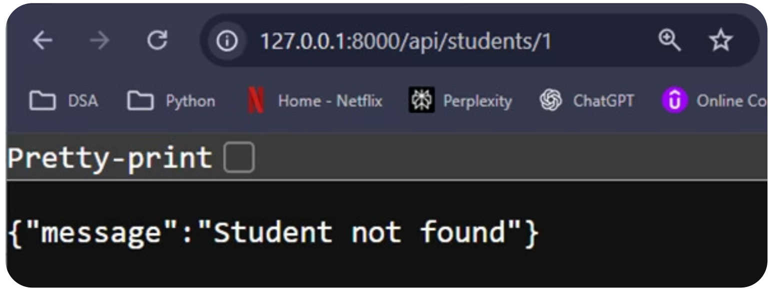


```
> App\Models\Student::find(1)->delete();
= true
```

2

Confirm Removal

Refresh browser endpoint—record no longer appears in JSON output.



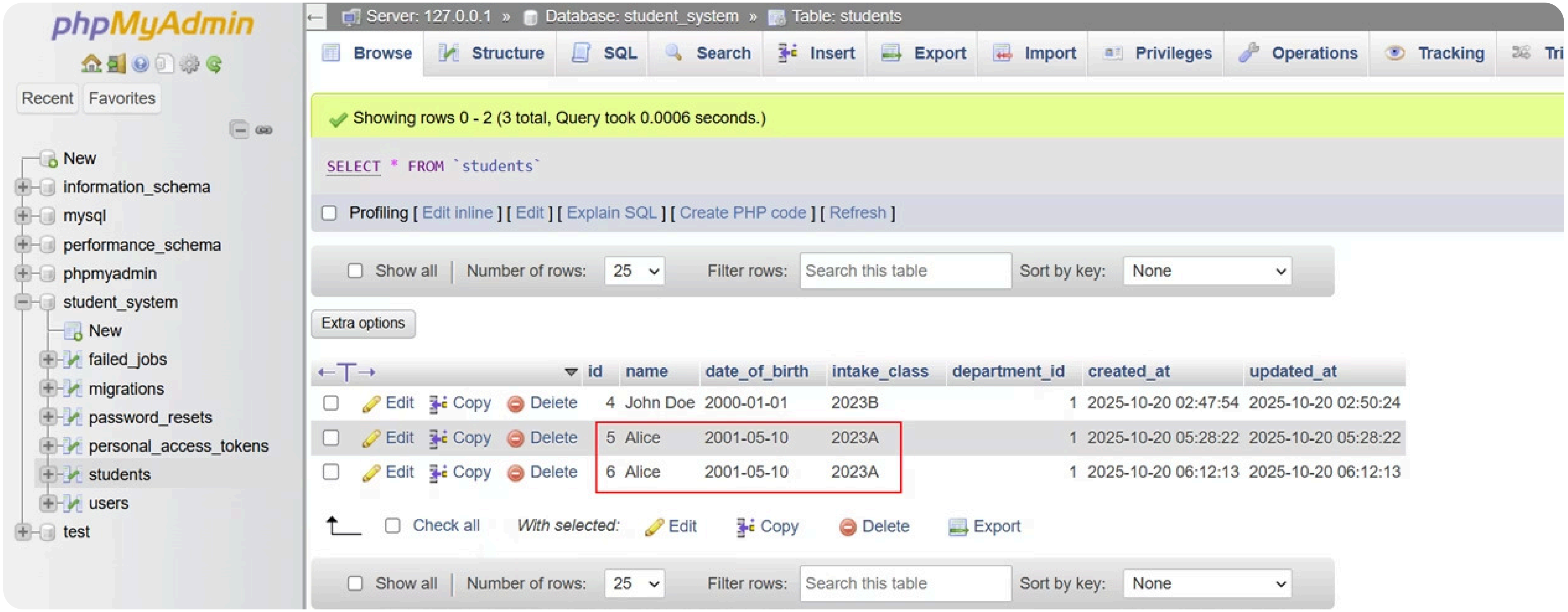
```
{"message":"Student not found"}
```

Database Verification

All CRUD operations reflect live in phpMyAdmin

http://localhost/phpmyadmin

Direct database inspection confirms data integrity and successful API operations.

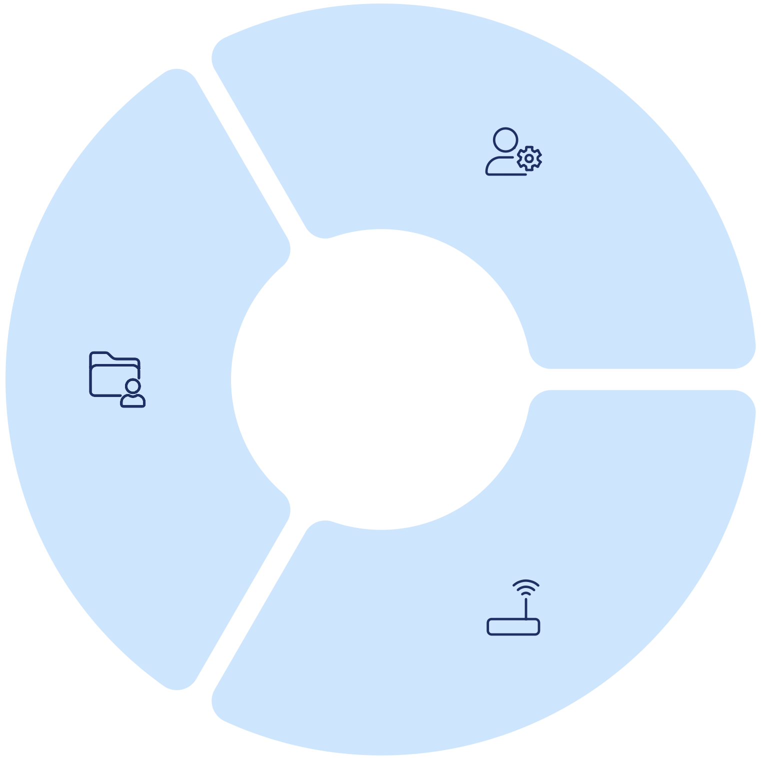


Showing rows 0 - 2 (3 total, Query took 0.0006 seconds.)

```
SELECT * FROM `students`
```

	id	name	date_of_birth	intake_class	department_id	created_at	updated_at
Edit Copy Delete	4	John Doe	2000-01-01	2023B	1	2025-10-20 02:47:54	2025-10-20 02:50:24
Edit Copy Delete	5	Alice	2001-05-10	2023A	1	2025-10-20 05:28:22	2025-10-20 05:28:22
Edit Copy Delete	6	Alice	2001-05-10	2023A	1	2025-10-20 06:12:13	2025-10-20 06:12:13

MVC Architecture in Action



Model

Student model interacts with the database, defining structure and relationships.

Controller

StudentController contains business logic and handles API request processing.

Routes

API routes define endpoints and map HTTP requests to controller methods.

MVC Components in Detail

1

🧱 Model

Defines the data structure and interacts with the database via Eloquent ORM.

2

⚙️ Controller

Contains the application logic, receives requests, calls the Model, and prepares data.

3

🌐 Routes

Define API endpoints and link incoming HTTP requests to specific controller actions.

Student Model

File: app/Models/Student.php

```
class Student extends Model {
    protected $fillable = [
        'name',
        'date_of_birth',
        'intake_class',
        'department_id'
    ];
}
```

✅ Handles all database communication.

StudentController Index Method

File: app/Http/Controllers/StudentController.php

```
public function index() {
    $students = Student::all();
    return response()-
>json($students);
}
```

✅ Acts as the brain of the MVC pattern.

API Route Definition

File: routes/api.php

```
Route::get(
    '/students',
    [StudentController::class,
'index']
);
```

✅ Keeps URLs clean and logical.

🧩 Create and set up the Student model step-by-step

```
php artisan make:model Student -m
```

✅ This command creates:

- app/Models/Student.php → the model file
- database/migrations/xxxx_xx_xx_create_students_table.php → a migration file for the da

1. **Edit the model file (**app/Models/Student.php**)**
   Add the following code inside it:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Student extends Model
{
    use HasFactory;

    protected $fillable = [
        'name',
        'date_of_birth',
        'intake_class',
        'department_id',
    ];
}
```

1. **Edit the migration file** (database/migrations/...create_students_table.php)
   Add columns for the fields:

```
public function up(): void
{
    Schema::create('students', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->date('date_of_birth');
        $table->string('intake_class');
        $table->foreignId('department_id')->constrained()->onDelete('cascade');
        $table->timestamps();
    });
}
```

1. **Run the migration to create the table:**

```
php artisan migrate
```

🎓 StudentController Implementation Guide (Step-by-Step Instructions)

1️⃣ Create the Controller

Open your terminal and run:

```
php artisan make:controller StudentController
```

This will create a new file at:

```
app/Http/Controllers/StudentController.php
```

2️⃣ Import Dependencies

At the top of the controller file, add:

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

These imports allow access to HTTP request handling and the `Student` model.

3️⃣ Define the Controller Class

Inside the file, define:

```
class StudentController extends Controller
```

This makes the controller inherit core Laravel controller functionality.

4️⃣ Create CRUD Methods

🟢 A. Get All Students

Purpose: Retrieve a list of all students from the database.

```
public function index()
{
    return response()->json(Student::all());
}
```

- Uses `Student::all()` to fetch all student records.
- Returns them as a JSON response.

🟡 B. Get One Student by ID

Purpose: Fetch details of a specific student using their ID.

```
public function show($id)
{
    $student = Student::find($id);
    if (!$student) {
        return response()->json(['message' => 'Student not
found'], 404);
    }
    return response()->json($student);
}
```

- Uses `find()` to locate a record by ID.
- Returns a **404 error** if not found.

🟠 C. Add a New Student

Purpose: Insert a new student record into the database.

```
public function store(Request $request)
{
    $validated = $request->validate([
        'name' => 'required|string|max:255',
        'date_of_birth' => 'required|date',
        'intake_class' => 'required|string|max:255',
        'department_id' => 'required|integer',
    ]);

    $student = Student::create($validated);
    return response()->json($student, 201);
}
```

- Uses Laravel validation to ensure correct data input.
- Creates a student record using `Student::create()`.

✅ **Note:** Ensure the `Student` model includes:

```
protected $fillable = ['name', 'date_of_birth', 'intake_class', 'department_id'];
```

🔵 D. Update a Student

Purpose: Modify an existing student record.

```
public function update(Request $request, $id)
{
    $student = Student::find($id);
    if (!$student) {
        return response()->json(['message' => 'Student not found'], 404);
    }

    $student->update($request->all());
    return response()->json($student);
}
```

- Checks if the student exists.
- Updates the record using the provided request data.

🔴 E. Delete a Student

Purpose: Remove a student record from the database.

```
public function destroy($id)
{
    $student = Student::find($id);
    if (!$student) {
        return response()->json(['message' => 'Student not found'], 404);
    }

    $student->delete();
    return response()->json(['message' => 'Student deleted successfully']);
}
```

- Deletes the record if found.
- Returns a success message.

5️⃣ Define Routes

Open:

```
routes/api.php
```

Add:

```
use App\Http\Controllers\StudentController;

Route::apiResource('students', StudentController::class);
```

This automatically creates the following API routes:

Method	URI	Action
GET	`/students`	index
GET	`/students/{id}`	show
POST	`/students`	store
PUT/PATCH	`/students/{id}`	update
DELETE	`/students/{id}`	destroy

6️⃣ Test in Console / API Client

Example commands: curl

Get all students

curl http://localhost:8000/api/students

Get a single student

curl http://localhost:8000/api/students/1

Add a new student

curl -X POST http://localhost:8000/api/students
-H "Content-Type: application/json"
-d '{"name":"John Doe","date_of_birth":"2005-04-12","intake_class":"Form 2","department_id":1}'

Example commands: Tinker

Open Tinker

php artisan tinker

Get all students

App\Models\Student::all();

Get a single student

App\Models\Student::find(1);

Add a new student

App\Models\Student::create(['name' => 'John Doe', 'date_of_birth' => '2005-04-12', 'intake_class' => 'Form 2', 'department_id' => 1,]);

Laravel vs Traditional PHP

🧭 1. Routing

Without Laravel:

```
if ($_SERVER['REQUEST_URI'] == '/hello') {
    echo "Hello PHP!";
}
```

With Laravel:

```
Route::get('/hello', function () {
    return 'Hello Laravel!';
});
```

👉 **Laravel handles routing neatly, no messy conditionals.**

💾 2. Database (Eloquent ORM)

Without Laravel:

```
$conn = mysqli_connect('localhost', 'root', '', 'test');
$result = mysqli_query($conn, "SELECT * FROM users WHERE
id=1");
$user = mysqli_fetch_assoc($result);
echo $user['name'];
```

With Laravel:

```
$user = User::find(1);
echo $user->name;
```

👉 **No SQL — just object-style database access.**

🎨 3. Template (View)

Without Laravel:

```
<h1>Hello <?php echo $name; ?>
</h1>
```

With Laravel (Blade):

```
<h1>Hello, {{ $name }}</h1>
```

👉 Blade is cleaner and prevents XSS automatically.

⚙️ 4. Controller Creation

Without Laravel: Manually create a file and write all logic yourself.

With Laravel:

```
php artisan make:controller
PostController
```

👉 One command — ready-to-use controller file.

🗃️ 5. Migration (Database Setup)

Without Laravel: Manually run SQL commands in phpMyAdmin.

With Laravel:

```
php artisan migrate
```

👉 Laravel **automatically builds or updates database tables** based on your migration files

6. Laravel vs. Other PHP Frameworks

Choosing the right framework is crucial for your project's success. Here's a comparison of Laravel with other popular PHP frameworks:

PHP Frameworks	**⚡ Laravel**	**⚙️ CodeIgniter**	**🪶 Slim**	**🧱 Yii**	**🏛️ Symfony**
Framework Type	Modern MVC	Simple MVC	Micro-framework	Full-featured MVC	Enterprise MVC
Built-in ORM	✅ Eloquent	❌	❌	✅	✅ Doctrine
Templating Engine	✅ Blade	❌	❌	✅	✅ Twig
Security Features	✅ Strong	❌ Limited	❌ Lacks	✅ Strong	✅ Highly secure
CLI Tooling	✅ Artisan	❌ Manual	❌	✅ Gii Code Gen.	❌ Complex setup
Learning Curve	Moderate	Easy	Easy	Steeper	Complex
Best Use Case	Modern Full-stack Apps	Small to Medium Projects	REST APIs, Microservices	Enterprise, Data-driven	Large-scale Enterprise

👉 Laravel offers a comprehensive suite for modern web development, balancing features with ease of use.

Summary

Framework	**Strength**	**Ideal Use**
⚡ Laravel	Most balanced (easy, powerful, secure)	Full-stack web apps
⚙️ CodeIgniter	Lightweight & simple	Small projects
🪶 Slim	Fast & minimal	REST APIs
🧱 Yii	Secure & robust	Enterprise apps
🏛️ Symfony	Enterprise-grade & modular	Large corporate systems

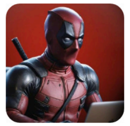
☁️ Deployment (GitHub Prep)

Before uploading your Laravel project to GitHub, ensure you remove sensitive or unnecessary files:

- `vendor/` → too large; can be reinstalled with `composer install`.
- `.env` → contains sensitive info (DB passwords, keys).
- `storage/logs/` → unnecessary local logs; not needed in repo.

👉 Keep your repository clean and secure by excluding these files.

sabs-27/**student-crud-laravel2**



2 Contributors   0 Issues   0 Stars   0 Forks

GitHub

GitHub - sabs-27/student-crud-laravel2

Contribute to sabs-27/student-crud-laravel2 development by creating an account on GitHub.