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 is a software framework that provides a basic structure for building web applications using PHP. It offers a standardized way to build and deploy applications, promoting code reuse and adhering to best practices, ultimately streamlining the development process.

Accelerate Development

Frameworks provide pre-built modules and functions, reducing the need to write code from scratch. This significantly speeds up the development process.

Structured Codebase

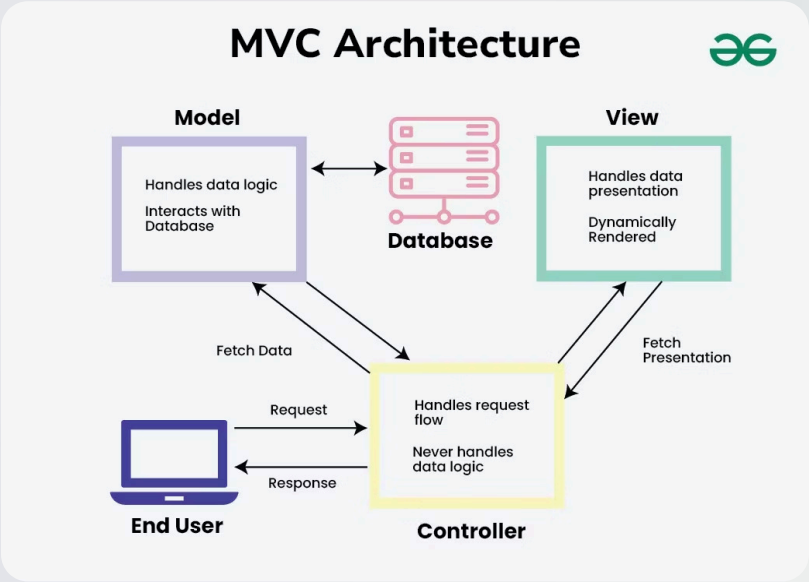
They enforce architectural patterns (like MVC), making the codebase organized, consistent, and easier to manage, especially in large projects with multiple developers.

Enhanced Security

Frameworks often come with built-in security features, protecting against common web vulnerabilities like XSS, CSRF, and SQL injection, making applications more robust.

Easier Maintenance & Scaling

Standardized code and clear structures simplify debugging, updates, and the addition of new features, ensuring long-term maintainability and scalability.



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.

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

```
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.

Launch Laravel Server

```
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
```

1 Navigate to Project

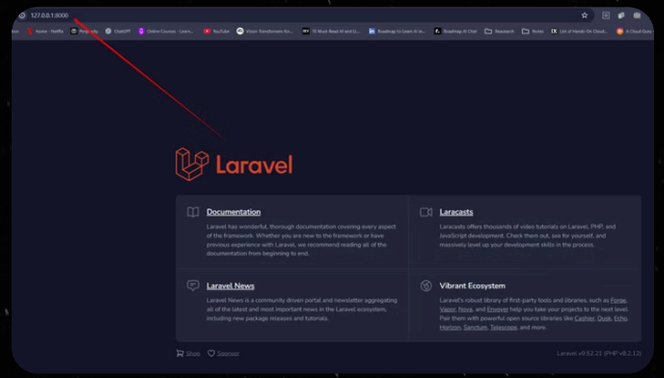
```
cd C:\projects\student-crud-laravel
```

2 Start Server

```
php artisan serve
```

3 Access Application

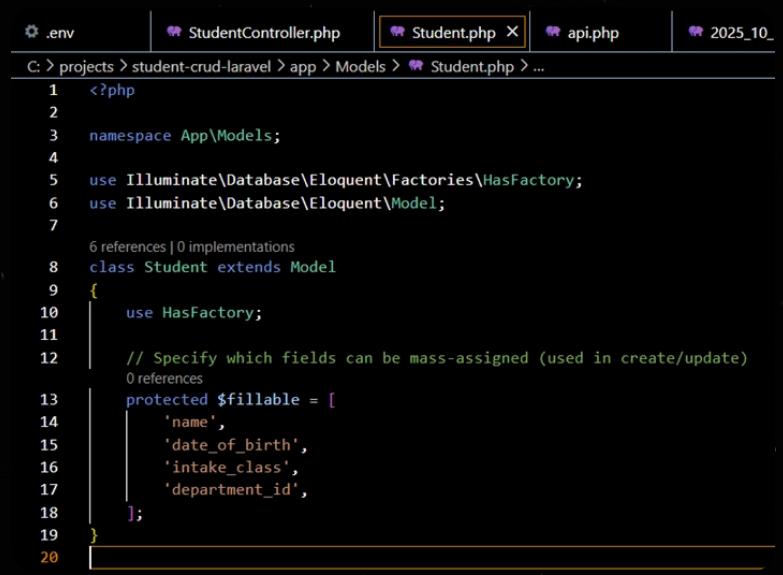
Visit http://127.0.0.1:8000 to see Laravel welcome page



Building the CRUD Components

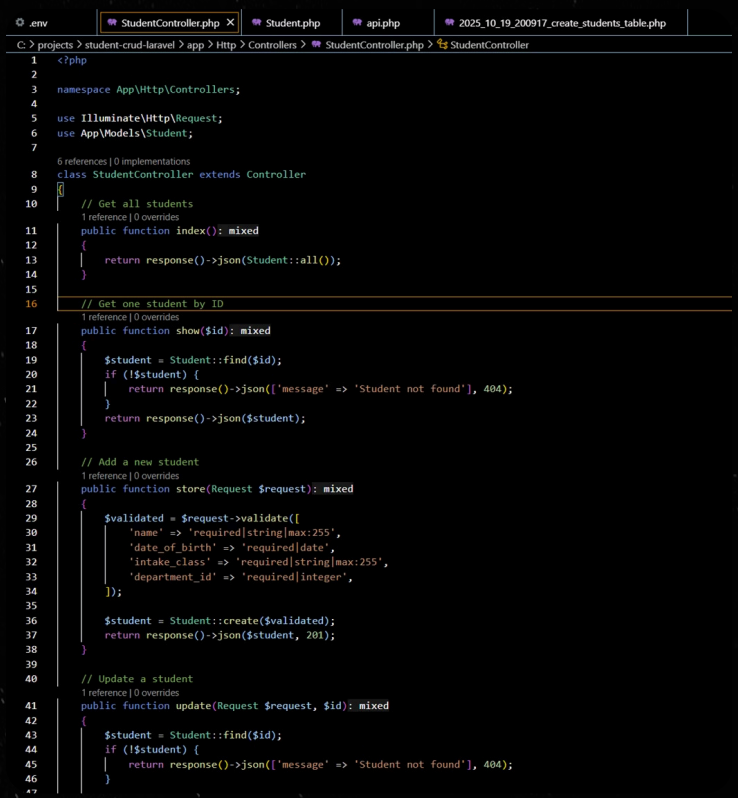
Student Model

Defines database structure with `$fillable` fields: name, date_of_birth, intake_class, department_id



Student Controller

Contains 5 methods: index(), show(), store(), update(), destroy() for complete CRUD operations



API Routes Configuration

GET /api/students

List all students

GET /api/students/{id}

Show single student

POST /api/students

Create new student

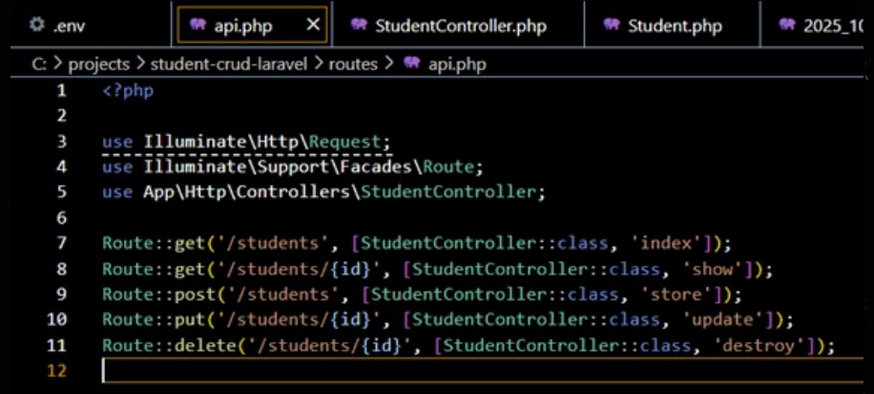
PUT /api/students/{id}

Update student

DELETE /api/students/{id}

Delete student

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



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

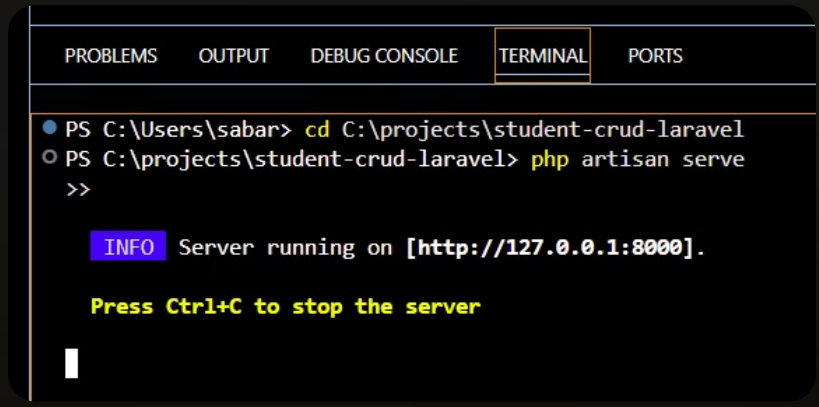
Verify Path

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.



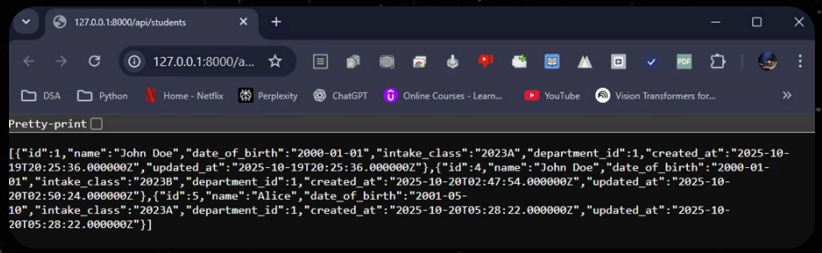
Accessing the Application

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.



Testing Routes

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

```
php artisan route:list
```

This command displays all registered routes, methods, URIs, and corresponding controller actions.

```
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**

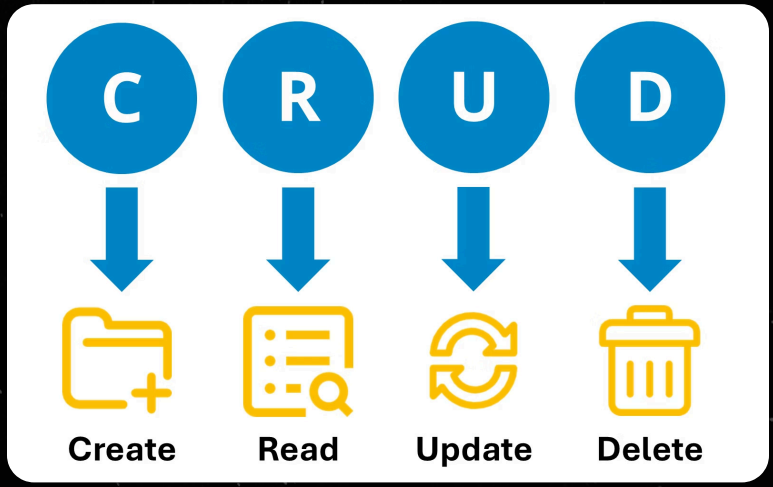
   Fetch all: `Student::all()` or single: `Student::find(1)`.
   View at http://127.0.0.1:8000/api/students

3. **UPDATE**

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

4. **DELETE**

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



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.

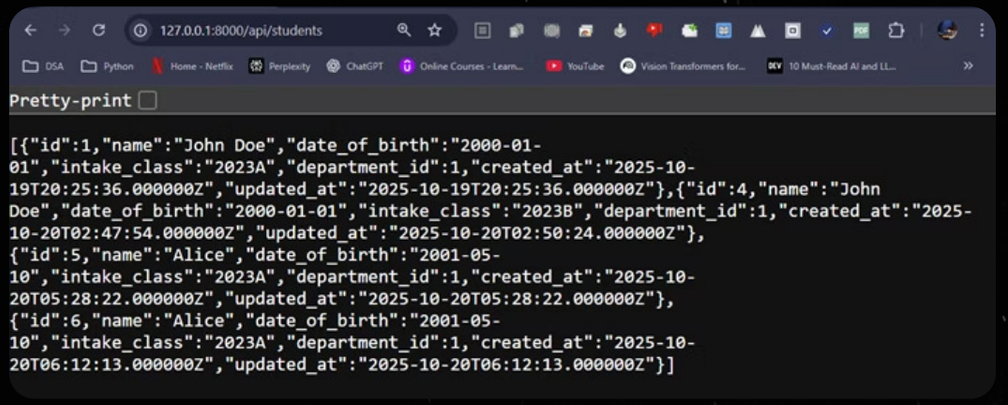
3

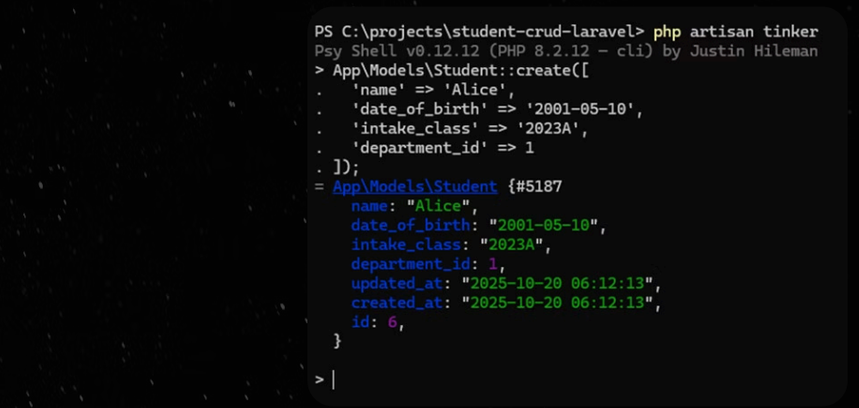
Verify in Browser

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**





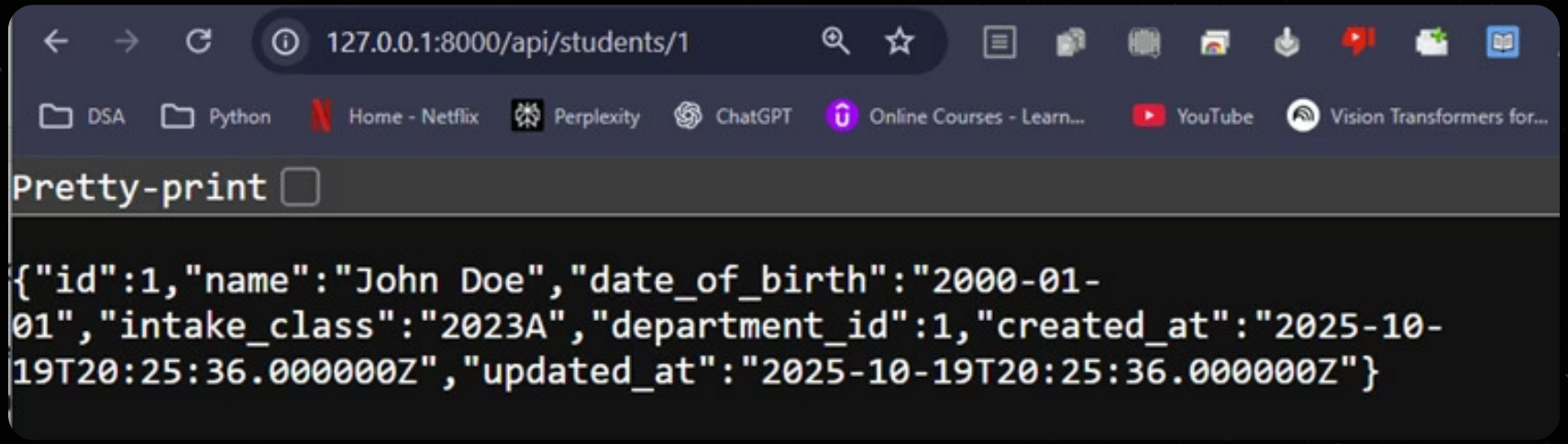
READ Operation

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.



UPDATE Operation

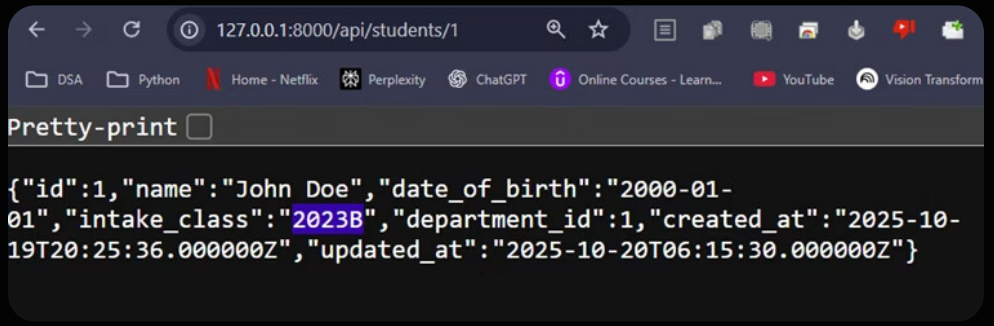
Modify Data

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.



DELETE Operation

1

Execute Delete

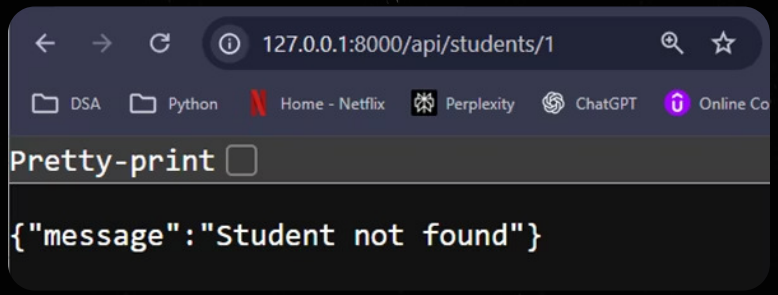
Run `Student::find(1)->delete()` in Tinker to remove the record.

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

2

Confirm Removal

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



Demonstrate CRUD (Using Tinker Only)

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

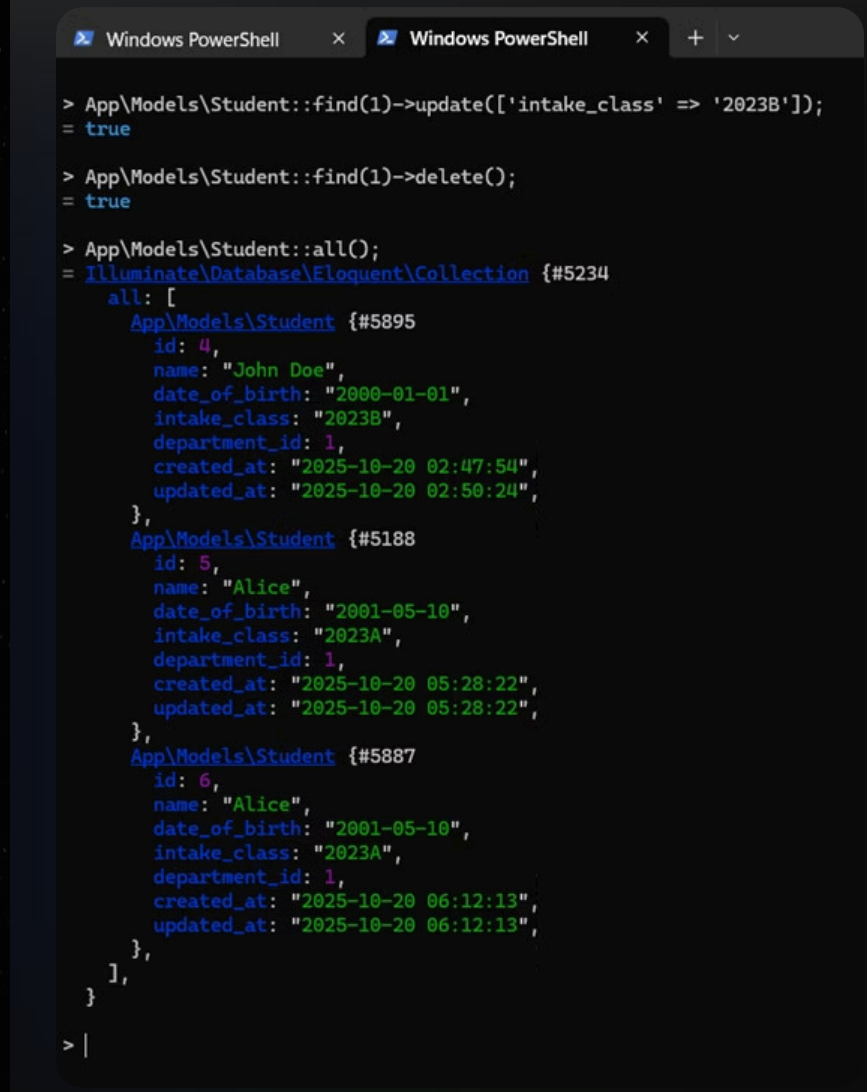
Run:
App\Models\Student::find(1)->update(['intake_class' => '2023B']);

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

DELETE: Remove a student record

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

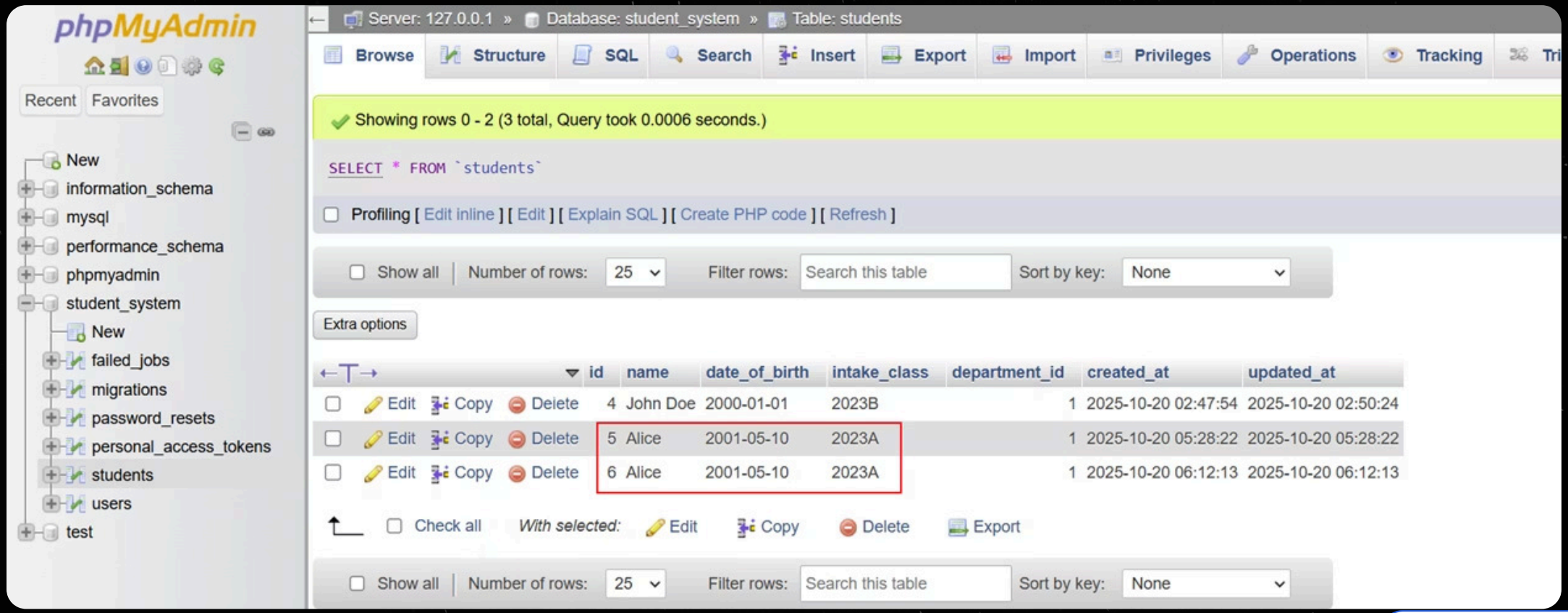
>>> App\Models\Student::find(1);



Database Verification

All CRUD operations reflect live in phpMyAdmin under the **student_system** database and **students** table.

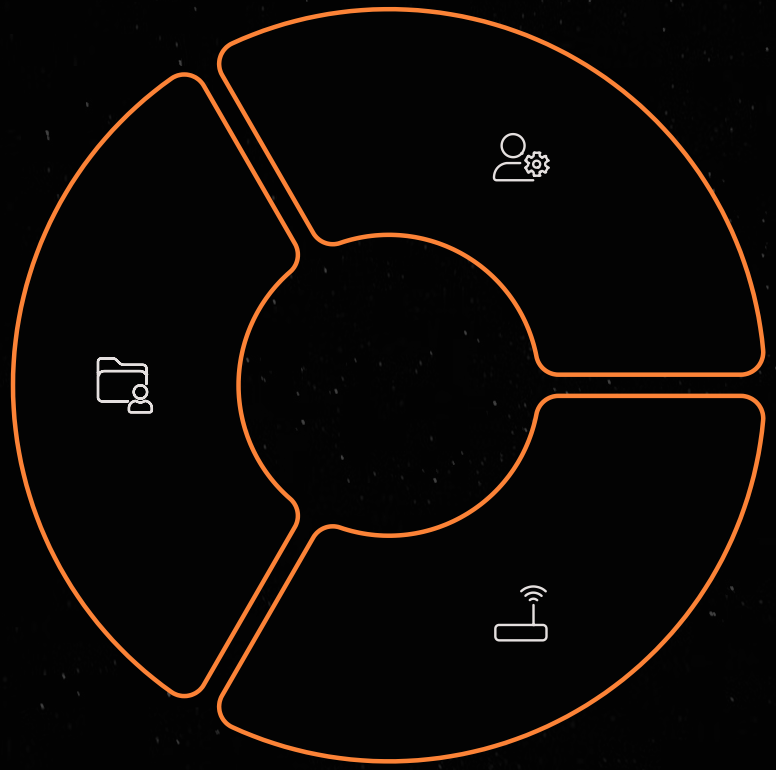
Direct database inspection confirms data integrity and successful API operations.



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.

🧭 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 builds or updates tables automatically.

☁️ 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.

GitHub

sabs-27/student-crud-laravel

https://github.com/sabs-27/student-crud-laravel.git