## **🧭 Етапи розробки EduCore — покроковий план**

### **📁 1. Підготовка середовища**

✅ Термін: 0.5 дня  
 **Завдання:**

* Встановити Docker, Node.js, Composer, Git
* Ініціалізувати Git-репозиторій
* Створити docker-compose.yml (MySQL + Laravel + Vue)
* Налаштувати базову структуру frontend/ і backend/

### **🧱 2. Проєктування бази даних**

✅ Термін: 0.5 дня  
 **Завдання:**

* ER-діаграма
* Створення Laravel міграцій для:  
  + users, courses, sections, lessons, tests, test\_answers, results, certificates
* Сідери для демо-даних
* Laravel моделі з відношеннями (hasMany, belongsTo)

### **🔐 3. Аутентифікація та авторизація**

✅ Термін: 1 день  
 **Завдання:**

* Laravel Sanctum API login/register/logout
* Middleware для ролей (student, instructor, admin)
* Vue: форми логіну/реєстрації
* Збереження токена в localStorage

### **🧑‍🎓 4. Ролі та особисті кабінети**

✅ Термін: 1 день  
 **Завдання:**

* Vue router з захистом маршрутів
* Панель студента: список курсів, прогрес
* Панель викладача: CRUD курсів
* Панель адміністратора: статистика, модерація

### **📚 5. Курси, уроки, тести (CRUD)**

✅ Термін: 2–3 дні  
 **Завдання:**

* Backend:  
  + Створення, оновлення, видалення курсів
  + Додавання секцій та уроків
  + Додавання тестів до уроку
* Frontend:  
  + Форма додавання курсу з уроками
  + Програвання відео через <video> або YouTube
  + Форма проходження тесту
* Збереження результатів тестів

### **📜 6. Генерація сертифіката**

✅ Термін: 0.5 дня  
 **Завдання:**

* Laravel DOMPDF PDF-генератор
* Шаблон сертифіката
* Збереження сертифікатів в /storage
* Завантаження PDF для студента

### **🌘 7. Темна тема та UI**

✅ Термін: 0.5–1 день  
 **Завдання:**

* Підключити TailwindCSS
* Темна/світла тема — перемикач
* Зберігати тему в localStorage
* Основна стилізація інтерфейсу (адаптивно)

### **📈 8. Прогрес та аналітика**

✅ Термін: 1 день  
 **Завдання:**

* Зберігати перегляд кожного уроку
* Вираховувати % завершення курсу
* Побудова графіку з Chart.js на фронті
* Адміністративна аналітика (кількість студентів, завершення курсів)

### **💬 9. Коментарі до уроків**

✅ Термін: 1 день  
 **Завдання:**

* Таблиця lesson\_comments
* Форма коментування під уроком
* Відображення списку коментарів
* Модерація для адміністратора

### **🤖 10. Telegram-бот: нагадування**

✅ Термін: 1 день  
 **Завдання:**

* Реєстрація username студента
* Laravel команда для schedule:run
* Відправка повідомлень студентам, які не проходили курси > 3 днів
* Використати Telegram Bot API

### **🧪 11. Тестування**

✅ Термін: 1 день  
 **Завдання:**

* Laravel Feature/Unit тести (Auth, Courses)
* Vue тестування компонентів (опціонально)
* Тестування API вручну (Postman)

### **🚀 12. CI/CD + Деплой (опційно)**

✅ Термін: 0.5–1 день  
 **Завдання:**

* Netlify для фронтенду
* Railway/Render/Heroku для Laravel API
* GitHub Actions або GitHub Pages (для auto-deploy)

### **🧾 13. Презентація для захисту**

✅ Термін: 1 день  
 **Завдання:**

* Скріншоти/відео з demo
* Готове live demo
* Слайди:  
  + Мета
  + Архітектура
  + Технології
  + Демонстрація
  + Унікальні фічі
  + Висновки

## **🗓️ Загальна оцінка часу:**

| **Етап** | **Час** |
| --- | --- |
| Підготовка + Docker | 0.5 дня |
| БД + ER-діаграма | 0.5 дня |
| Auth + Ролі | 1 день |
| Курси + Тести + Відео | 3 дні |
| Сертифікати + UI | 1 день |
| Telegram-бот + Коментарі | 2 дні |
| Тестування + Презентація | 2 дні |
| **Разом** | **~10 днів активної розробки** |

# **⚙️ План розробки коду для бекенда та фронтенда**

## **🔧 1. Backend (Laravel) – покроково**

### **🔹 1.1. Створення проєкту та початкова конфігурація**

bash

КопироватьРедактировать

laravel new backend

cd backend

composer require laravel/sanctum

php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"

php artisan migrate

* Увімкнути api middleware з Sanctum
* Налаштувати .env для бази

### **🔹 1.2. Моделі та міграції**

bash

КопироватьРедактировать

php artisan make:model Course -m

php artisan make:model Section -m

php artisan make:model Lesson -m

php artisan make:model Test -m

php artisan make:model TestAnswer -m

php artisan make:model Result -m

php artisan make:model Certificate -m

php artisan make:model LessonComment -m

Прописати зв’язки (hasMany, belongsTo, belongsToMany, тощо)

#### **🔸 Приклад: Course**

php

КопироватьРедактировать

// app/Models/Course.php

public function sections() {

return $this->hasMany(Section::class);

}

### **🔹 1.3. Аутентифікація з ролями**

* Додати поле role у таблицю users
* Використати middleware:

php

КопироватьРедактировать

// routes/api.php

Route::middleware(['auth:sanctum', 'role:instructor'])->group(function () {

Route::post('/courses', [CourseController::class, 'store']);

});

* Написати AuthController (реєстрація, логін, logout)

### **🔹 1.4. CRUD функціонал**

bash

КопироватьРедактировать

php artisan make:controller CourseController --api

php artisan make:controller LessonController

php artisan make:controller TestController

#### **CRUD для курсів:**

* /api/courses — GET (всі курси)
* /api/courses/{id} — GET (один курс з секціями/уроками)
* /api/courses — POST (додати курс)
* /api/courses/{id} — PUT/DELETE (оновити/видалити)

### **🔹 1.5. Тести та результати**

* Тести (питання, варіанти)
* Збереження відповідей
* Автоматичне підрахування правильності
* Збереження в results таблицю

### **🔹 1.6. Генерація PDF сертифікату**

bash

КопироватьРедактировать

composer require barryvdh/laravel-dompdf

* Генерація шаблону в resources/views/pdf/certificate.blade.php
* Контролер CertificateController@generate
* Збереження PDF та посилання на нього

### **🔹 1.7. Telegram інтеграція**

* Telegram Token з BotFather
* Webhook або Laravel schedule:run
* HTTP POST запит до Telegram API:

php

КопироватьРедактировать

Http::post("https://api.telegram.org/bot{$token}/sendMessage", [

'chat\_id' => $user->telegram\_id,

'text' => 'Привіт! Не забудь пройти курс!'

]);

### **🔹 1.8. Коментарі до уроків**

* Таблиця lesson\_comments (user\_id, lesson\_id, content)
* Створити CommentController
* Маршрути: POST/GET коментарів

## **🖼️ 2. Frontend (Vue 3 + Pinia + Tailwind) – покроково**

### **🔹 2.1. Ініціалізація проєкту**

bash

КопироватьРедактировать

npm create vite@latest frontend --template vue

cd frontend

npm install

npm install axios pinia vue-router tailwindcss postcss autoprefixer

npx tailwindcss init -p

* Налаштувати Tailwind у tailwind.config.js
* Створити структуру:

css

КопироватьРедактировать

src/

├── assets/

├── components/

├── pages/

├── router/

├── store/

└── services/

### **🔹 2.2. Роутінг (Vue Router)**

js

КопироватьРедактировать

// router/index.js

const routes = [

{ path: '/', component: HomePage },

{ path: '/login', component: LoginPage },

{ path: '/dashboard', component: DashboardPage },

{ path: '/course/:id', component: CourseDetail },

];

* Навігаційні гардзи (потрібна авторизація)

### **🔹 2.3. Сторінки для кожної ролі**

* **Гостьові**: Реєстрація, логін, головна
* **Студент**: Мій прогрес, список курсів, сторінка курсу, проходження тесту
* **Викладач**: Створення курсу, уроків, тестів
* **Адмін**: Глобальна статистика

### **🔹 2.4. Авторизація (JWT з Sanctum)**

* Axios інстанс із baseURL + авторизація через Authorization header
* Зберігання токена в localStorage
* Захищені запити

### **🔹 2.5. CRUD курсів та проходження**

* Vue компоненти:  
  + CourseCard.vue
  + CourseForm.vue
  + LessonPlayer.vue
  + TestRunner.vue
* Динамічне рендерення уроків і секцій
* Тести з перевіркою без перезавантаження

### **🔹 2.6. Прогрес та сертифікат**

* Компонент ProgressChart.vue (на Chart.js)
* Сертифікат — кнопка для завантаження PDF

### **🔹 2.7. Коментарі до уроків**

* Компонент LessonComments.vue
* API: GET /api/lessons/{id}/comments, POST /api/comments
* Виведення та форма відправки

### **🔹 2.8. Темна тема**

* Tailwind dark: клас
* Тогл-світч з Pinia (режим зберігається в localStorage)
* Стилізувати всі компоненти з підтримкою dark mode

## **✅ Поради:**

* Для бекенда: зроби apiResource контролери
* Для фронта: компоненти мають бути маленькі та перевикористовувані
* Валидацію роби як на бекенді (FormRequest), так і на фронті (vuelidate або кастомно)
* Підключи Toast для сповіщень (наприклад, Vue Toastification)