

Perfect 

Below is a **clear, recruiter-ready PRD** for a **Core Django Learning Management System (LMS)**.

You can treat this as your **single source of truth** while building the project.

---



# Product Requirements Document (PRD)

## Project: Learning Management System (LMS)

**Tech Stack:** Django (Core), SQLite → PostgreSQL (optional later)

**Purpose:** Portfolio project to showcase backend fundamentals before learning DRF

---

## 1. Project Overview

The Learning Management System (LMS) is a web application that allows instructors to create and manage courses while students can enroll, learn, track progress, and complete assessments.

This project focuses on:

- Clean backend architecture
  - Proper Django fundamentals
  - Real-world business logic
  - Scalability for future API (DRF) conversion
- 

## 2. Problem Statement

Existing beginner Django projects (blogs, TODO apps) fail to demonstrate:

- Complex relationships
- Role-based permissions
- Structured workflows
- Business logic depth

This LMS solves that by simulating a **real educational platform** with multiple user roles and workflows.

---

### 3. User Roles & Permissions

#### 3.1 Student

- Register & login
- Browse courses
- Enroll in courses
- View lessons
- Mark lessons as completed
- Attempt quizzes
- View progress

#### 3.2 Instructor

- Create and manage courses
- Add modules and lessons
- Upload content (text/video)
- Create quizzes
- View enrolled students
- Track student progress

#### 3.3 Admin

- Manage users
  - Approve instructors (optional)
  - Manage all courses
  - Platform oversight
- 

### 4. Core Features

#### 4.1 Authentication & Authorization

- User registration & login
- Role-based access control
- Custom permissions
- Secure session handling

---

## **4.2 Course Management**

- Course creation (Instructor)
  - Course fields:
    - Title
    - Description
    - Category
    - Thumbnail image
    - Status (Draft / Published)
  - Only published courses visible to students
- 

## **4.3 Course Structure**

- Course → Modules → Lessons
  - Lessons can be:
    - Text-based
    - Video-based (URL or file)
  - Sequential lesson flow
- 

## **4.4 Enrollment System**

- Students can enroll in courses
  - Prevent duplicate enrollments
  - Restrict lesson access to enrolled users only
- 

## **4.5 Progress Tracking**

- Track lesson completion per student
  - Calculate course completion percentage
  - Display progress in dashboards
- 

## **4.6 Quiz & Assessment System**

- MCQ-based quizzes
- Auto-evaluation

- Score storage
  - Quiz tied to lessons or modules
- 

## 4.7 Dashboards

### Student Dashboard

- Enrolled courses
- Progress tracking
- Completed quizzes

### Instructor Dashboard

- Courses created
  - Enrolled students
  - Completion statistics
- 

## 5. Data Models (High Level)

### User

- username
- email
- role (student / instructor / admin)

### Course

- title
- description
- instructor (FK)
- category
- status
- created\_at

### Module

- title
- course (FK)

### Lesson

- title
- content
- module (FK)
- order

## Enrollment

- student (FK)
- course (FK)
- enrolled\_at

## LessonProgress

- student (FK)
- lesson (FK)
- completed (Boolean)

## Quiz

- lesson (FK)
- total\_marks

## Question

- quiz (FK)
- text
- options
- correct\_answer

---

## 6. Application Structure

Ims/

```
|   └── users/  
|   └── courses/  
|   └── enrollments/  
|   └── lessons/  
|   └── quizzes/  
|   └── dashboard/  
|   └── templates/  
|   └── static/
```

This modular structure reflects **industry-level Django projects**.

---

## 7. Business Rules

- Only instructors can create courses
  - Students must enroll before accessing lessons
  - Draft courses are invisible to students
  - Quiz scores are immutable after submission
  - Progress updates only after lesson completion
- 

## 8. Out of Scope (For Now)

- Payments
- Certificates
- Live classes
- Email notifications
- Mobile frontend

(You can mention these as **future improvements** in README)

---

## 9. Non-Functional Requirements

- Secure authentication
  - Clean error handling
  - Form validation
  - Pagination
  - Optimized queries
  - Readable, well-documented code
- 

## 10. Future Roadmap (DRF Transition)

This LMS will later be:

- Converted into REST APIs using DRF
- Used by React / Mobile frontend
- Secured with JWT authentication

---

## 11. Success Criteria (Portfolio-Ready)

Your project is **complete** when:

- All roles work correctly
  - Permissions are enforced
  - Code is modular and clean
  - README explains architecture
  - App runs locally without errors
- 

## Why This PRD Matters

If a recruiter opens your repo and sees:

- This PRD
- Clean Django code
- Proper structure

You instantly stand out from **90% of entry-level candidates**.

---

## Next Steps (Tell Me What You Want)

I can:

1. Convert this PRD into **development tasks**
2. Create **ER diagram**
3. Guide you **module-by-module**
4. Prepare **DRF migration plan**

Just say the word 