Perfect 👍
Below is a **start-to-finish, production-style task list** for your **Core Django LMS**.
Follow this **step by step**, and you'll finish with a **portfolio-grade project** that smoothly transitions to DRF later.

✅ LMS – Detailed Development Task List (Core Django)

> **Rule:** Do not jump ahead.
> Complete each phase fully before moving to the next.

🟢 PHASE 0 — Planning & Setup

0.1 Project Planning

- Read PRD fully
- Finalize feature scope (no extra features yet)
- Draw rough ER diagram (paper is fine)
- Decide app names

0.2 Environment Setup

- Create virtual environment
- Install Django
- Start Django project `lms`
- Create `.gitignore`
- Initialize Git repository
- Create initial `README.md`

🟢 PHASE 1 — Project Structure & Core Configuration

1.1 Create Apps

- Create apps:
    - `users`
    - `courses`
    - `lessons`
    - `enrollments`
    - `quizzes`
    - `dashboard`
- Register apps in `INSTALLED_APPS`

1.2 Core Settings

- Configure templates directory
- Configure static files
- Configure media files
- Add custom context processors (if needed)

🟢 PHASE 2 — User Management & Authentication

2.1 Custom User Model

- Extend `AbstractUser`
- Add `role` field (student / instructor / admin)
- Update `AUTH_USER_MODEL`
- Run migrations

2.2 Authentication System

- User registration
- Login & logout
- Role-based redirects
- Secure password handling

2.3 Permissions & Access Control

- Custom decorators / mixins
- Instructor-only access
- Student-only access
- Admin overrides

🟢 PHASE 3 — Course Management

3.1 Course Models

- Create `Course` model
- Add fields:
  - title
  - description
  - category
  - thumbnail
  - instructor
  - status (Draft / Published)
- Add timestamps
- Run migrations

3.2 Course CRUD

- Instructor can create course
- Instructor can edit course
- Instructor can delete course
- Validate ownership

3.3 Course Visibility

- Only published courses visible to students
- Draft courses visible only to instructors

🟢 PHASE 4 — Modules & Lessons

4.1 Module Model

- Create `Module` model
- Link to course
- Ordering support

4.2 Lesson Model

- Create `Lesson` model
- Fields:
  - title
  - content
  - video (optional)
  - module
  - order
- Secure lesson access

4.3 Lesson Flow

- Sequential lesson navigation
- Disable skipping (optional)
- Lesson detail view

🟢 PHASE 5 — Enrollment System

5.1 Enrollment Model

- Create `Enrollment` model
- Prevent duplicate enrollments
- Timestamp enrollment

5.2 Enrollment Logic

- Enroll button
- Enrollment validation
- Restrict lesson access to enrolled users

🟢 PHASE 6 — Progress Tracking

6.1 Lesson Progress Model

- Create `LessonProgress` model
- Track completion per user

6.2 Progress Logic

- Mark lesson as completed
- Auto-calculate progress
- Prevent double completion

6.3 Progress UI

- Progress bar per course
- Completed lesson indicator

🟢 PHASE 7 — Quiz & Assessment System

7.1 Quiz Models

- Create `Quiz` model
- Create `Question` model
- Support MCQ format

7.2 Quiz Logic

- Quiz attempt form
- Auto-evaluation
- Store scores
- One attempt per user

7.3 Quiz Access Rules

- Quiz unlocks after lesson completion
- Prevent reattempt

🟢 PHASE 8 — Dashboards

8.1 Student Dashboard

- Enrolled courses list
- Progress overview
- Quiz scores

8.2 Instructor Dashboard

- Created courses list
- Enrolled students count
- Completion statistics

🟢 PHASE 9 — Admin & Optimization

9.1 Admin Panel

- Register all models
- Custom admin views
- Filters & search

9.2 Query Optimization

- Use `select_related`
- Use `prefetch_related`
- Reduce N+1 queries

🟢 PHASE 10 — UI & UX Polish

10.1 Templates

- Base layout
- Navigation bar
- Flash messages
- Error pages

10.2 Forms & Validation

- Server-side validation
- User-friendly error messages

🟢 PHASE 11 — Testing & Stability

11.1 Manual Testing

- Student flow testing
- Instructor flow testing
- Permission testing

11.2 Django Tests (Optional but Strong)

- Model tests
- View tests
- Permission tests

🟢 PHASE 12 — Documentation & Portfolio Prep

12.1 Documentation

- Update `README.md`
- Add project overview
- Add setup instructions

- Add screenshots
- Add future roadmap

12.2 Final Cleanup

- Remove unused code
- Check `.gitignore`
- Run `makemigrations` & `migrate`
- Run server cleanly

🟢 PHASE 13 — Ready for DRF Transition

13.1 DRF Preparation

- Identify serializers mapping
- Identify API endpoints
- Separate business logic from views

🎯 Final Outcome

After completing this:

- You will **fully understand Django**
- DRF will feel **natural**
- Your portfolio will contain a **real backend system**

🚀 Next Step?

If you want, I can:

- Break **PHASE 1 into day-by-day tasks**
- Create **model code templates**
- Act as a **daily development guide**
- Prepare a **DRF roadmap**

Just tell me how you want to proceed 👌