# Django Learning Roadmap (Fresh Project per Phase)

Tailored for: Bharat — Python + Flask background • Frontend (HTML/CSS/JS) • MySQL

Approach: Build a new Django project folder for each phase so setup becomes second nature. Each phase focuses on a small, testable goal with a concrete deliverable.

## Contents

# Prerequisites & Setup

- Python 3.10+ installed; create a dedicated virtualenv per project (one folder = one venv).
- Django installed via pip; verify with `django-admin --version`.
- Git initialized in each folder; commit at the end of every milestone.
- Editor setup: format on save (Black), linting (ruff/pylance), Django extension for templates.
- Database: start with SQLite; switch to MySQL/PostgreSQL when practicing migrations and relations.

# Learning Strategy

- New folder for every phase to practice bootstrapping repeatedly.
- Keep a short README.md in each folder: goal, commands, gotchas.
- After finishing a phase, write a 5–10 line retrospective: what you learned, what still confuses you.
- Prefer Class-Based Views (CBVs) once you understand Function-Based Views (FBVs).
- Use the official docs as the primary source; build something small immediately after reading.

# Phase-by-Phase Roadmap (Fresh Project per Phase)

## Phase 1 — Basics Project (folder: django_basics/)

- Goal: internalize Django project/app structure & MTV pattern.
- Features: simple Notes app — fields: title, content, created_at.
- Tasks: startproject + startapp, URLs, views (FBV), templates, static files, CRUD, migrations.
- Deliverable: minimal CRUD web app with list/detail/create/update/delete.

## Phase 2 — Forms & Auth (folder: django_auth/)

- Goal: server-side validation & built-in auth.
- Features: user signup, login/logout, password reset (email console backend), profile page.
- Tasks: Django Forms vs ModelForms, messages framework, login-required on CRUD; associate Note.owner with User.
- Deliverable: authenticated CRUD with per-user data isolation.

## Phase 3 — Admin & Middleware (folder: django_admin/)

- Goal: manage data via Django admin and understand request/response cycle.
- Features: at least two related models; admin customization (list_display, search, filters, inlines).
- Tasks: write one middleware (request timing or audit log), add signals for post_save auditing.
- Deliverable: admin-driven data dashboard + middleware log.

## Phase 4 — REST API with DRF (folder: django_api/)

- Goal: design clean API surfaces with DRF.
- Features: API for Notes (list/retrieve/create/update/delete).
- Tasks: Serializers, ViewSets, Routers; permission classes; throttling; pagination.
- Deliverable: browsable API working in the DRF UI; curl/Postman collection committed.

## Phase 5 — API Auth (JWT) (folder: django_api_auth/)

- Goal: secure APIs with JWT using Simple JWT.
- Tasks: install & configure Simple JWT; obtain/refresh/verify endpoints; protect ViewSets with JWTAuthentication.
- Deliverable: Postman or HTTPie scripts demonstrating token flow & authorized requests.

## Phase 6 — Deployment Essentials (folder: django_deploy/)

- Goal: practice production settings safely.
- Tasks: settings split (base/dev/prod), environment variables, Django deployment checklist; static with WhiteNoise; logging & email errors.

- Deliverable: production build locally (DEBUG=False) with collectstatic + security headers verified.

## Phase 7 — Async & Realtime (optional) (folder: django_channels/)

- Goal: add WebSockets with Channels.
- Tasks: ASGI config, channels, channel layers (e.g., in-memory for practice), basic chat/notifications.
- Deliverable: small realtime feature (e.g., live activity feed).

## Phase 8 — Tasks & Performance (optional) (folder: django_tasks/)

- Goal: background work & caching.
- Tasks: Celery + a broker (e.g., Redis) for one periodic task; per-view or per-template caching; select_related/prefetch_related usage.
- Deliverable: scheduled task demo + measured response-time improvement.

# Bridge Tasks: Mapping Flask → Django

- Routing: Flask blueprints ■ Django urls.py + app namespaces.
- Templates: Jinja2 ■ Django Template Language (blocks, extends, filters).
- DB layer: SQLAlchemy ■ Django ORM (QuerySet API, migrations).
- Forms: Flask-WTF ■ Django Forms/ModelForms + messages.
- Auth: Flask-Login ■ django.contrib.auth (User, permissions).
- REST: Flask-RESTful ■ DRF (serializers, viewsets, routers, permissions).
- Static: Flask send_from_directory ■ collectstatic + WhiteNoise/CDN.

# Weekly Pacing Plan (8 Weeks)

### *Week 1*

- Phase 1 (django_basics): CRUD notes; push to Git; README with commands.

### *Week 2*

- Phase 2 (django_auth): Signup/login/reset; per-user notes; messages.

### *Week 3*

- Phase 3 (django_admin): Admin polish; middleware & signals; audit log.

### *Week 4*

- Phase 4 (django_api): Serializers/ViewSets/Routers; permissions; pagination.

### *Week 5*

- Phase 5 (jwt): Simple JWT; protect endpoints; Postman/HTTPie flows.

### *Week 6*

- Phase 6 (deploy): Settings split, DEBUG=False, collectstatic with WhiteNoise; logging.

### *Week 7*

- Phase 7 (channels, optional): WebSocket demo; live notifications.

### *Week 8*

- Phase 8 (tasks/caching, optional): Celery beat; caching; ORM optimization.

# Capstone Idea

- Build a personal Knowledge Base: web UI + DRF API + JWT + Admin management.
- Features: markdown notes, tags, full-text search (simple icontains), favorites, recent activity.
- Extras: background task to send daily 'review notes' email; optional realtime 'someone edited' indicator using Channels.
- Deliverables: production-ready repo with README, .env.example, Makefile and Postman collection.

# Quality Checklist (apply to every phase before 'Done')

- All CRUD paths work; 404/403 handled; friendly messages.
- Auth rules enforced; anonymous users cannot access protected views.
- Model relations correct; migrations created and applied cleanly.
- N+1 queries eliminated where obvious; added select_related/prefetch_related if needed.
- Static assets load; templates extend a base and use blocks.
- README shows setup, run, and key commands; .env.example included where relevant.
- At least 5 meaningful Git commits with clear messages.

# References (Free & Official)

- Django Docs: https://docs.djangoproject.com/
- Django Tutorial – Polls: https://docs.djangoproject.com/en/stable/intro/tutorial01/
- Django Deployment Checklist: https://docs.djangoproject.com/en/stable/howto/deployment/checklist/
- How to deploy Django: https://docs.djangoproject.com/en/stable/howto/deployment/
- Django REST Framework: https://www.django-rest-framework.org/
- DRF Authentication Guide: https://www.django-rest-framework.org/api-guide/authentication/
- Simple JWT docs: https://django-rest-framework-simplejwt.readthedocs.io/
- WhiteNoise docs: https://whitenoise.readthedocs.io/
- Django Channels docs: https://channels.readthedocs.io/
- MDN Django Tutorial (Local Library): https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Django

All links are free to access. Use official docs as the primary learning source.