Lumay, Alyza Shane C.

IT3R5

Performance Innovative Task:

Building a Full-Stack To-Do List Application with FastAPI Documentation

DRF vs FastAPI: Comparison

Feature	Django REST Framework (DRF)	FastAPI
Framework Base	Built on Django (batteries-included web framework)	Built on Starlette (lightweight ASGI framework)
Speed	Slower (sync-based by default)	Very fast (async support out of the box)
Type Hints / Validation	Optional, uses Django forms/serializers	Mandatory, uses Python type hints & Pydantic
Learning Curve	Steeper, especially for beginners	Easier for those familiar with Python type hints
Admin Panel	Built-in Django admin	No built-in admin panel
ORM	Django ORM (mature, feature-rich)	Can use SQLAlchemy or others (flexible but manual)
Auto Docs	Basic (via third-party like drf-yasg or coreapi)	Swagger UI & Redoc auto generated by default
Community & Ecosystem	Large and mature	Growing rapidly, but smaller than DRF
Best Use Cases	Enterprise-level projects, CMS, complex admin apps	Fast APIs, microservices, modern async apps

Advantages

DRF

- Mature ecosystem with lots of extensions
- Integrated with Django's ORM and admin
- Great for projects with complex models & authentication needs

FastAPI

- Much faster performance (async I/O)
- Cleaner code with automatic data validation (Pydantic)
- Instant interactive API docs (Swagger & ReDoc)
- Ideal for microservices and modern web apps

Disadvantages

DRF

- Slower response time due to synchronous nature
- Verbose code with serializers and views
- Less intuitive docs compared to FastAPI

FastAPI

- No built-in admin or user system
- Smaller ecosystem for advanced features
- Requires manual setup for things DRF does out of the box (auth, admin, etc.)