Comparison of FastAPI, Django, and Flask

Feature	FastAPI	Django	Flask
Framework Type	Modern, asynchronous web framework	Full-stack web framework	Lightweight micro-framework
Performance	High performance, asynchronous support	Moderate performance, synchronous	High performance, but synchronous
Ease of Use	Easy to use, intuitive syntax	Steeper learning curve	Simple and easy to get started
Use Case	APIs, microservices, asynchronous applications	Large-scale applications, rapid development	Simple web applications, microservices
Built-in Features	Minimal, focuses on API creation	Comes with ORM, admin panel, authentication	Minimal, requires extensions for added features
Asynchronous Support	Yes, built-in	No, requires additional libraries	No, requires additional libraries
ORM	None by default (Can use SQLAlchemy)	Built-in ORM (Django ORM)	None by default (Can use SQLAlchemy)
Admin Interface	No	Built-in admin interface	No
Template Engine	Jinja2	Django Templating Engine	Jinja2
Routing	Declarative routing	URL routing based on	Declarative routing

		configuration	
Data Validation	Built-in with Pydantic	Custom validators and serializers	None by default (Can use WTForms)
Security	Built-in support for OAuth2, JWT	Built-in support for authentication, CSRF, etc.	Minimal, requires extensions for added security
Documentation	Automatic interactive API documentation	Manual documentation	Manual documentation
Community Support	Growing, but smaller compared to Django and Flask	Large, well-established community	Large, well-established community
Third-party Libraries	Growing ecosystem	Vast ecosystem with many plugins	Vast ecosystem with many plugins
Deployment	ASGI, supports Uvicorn, Hypercorn	WSGI, supports Gunicorn, uWSGI	WSGI, supports Gunicorn, uWSGI
Learning Curve	Moderate	Steep	Low
Flexibility	High, allows for custom setups	Moderate, many features are built-in	High, allows for custom setups
Scalability	High, designed for scalability	Moderate, good for monolithic applications	High, good for microservices architecture