Shun Cyrel S. Caseres
BSIT-3R1

Difference between Django REST Framework (DRF) and FastAPI

Django REST Framework (DRF):

Key Features:

- Built on top of Django, which is a full-stack web framework, making it ideal for projects that need both backend and frontend.
- Provides tools for creating APIs, authentication, and permissions with ease.
- Includes a browsable API, which allows for testing and exploring the API directly in the browser.

Advantages:

- Integration with Django's ORM makes database management easier.
- Extensive documentation and community support.
- Built-in authentication, serialization, and permissions.

Disadvantages:

- Slower performance compared to FastAPI, especially for large-scale applications.
- Can feel heavy if you don't need all of Django's features.
- Less flexible compared to FastAPI in terms of handling asynchronous requests.

FastAPI:

Key Features:

- Modern, fast web framework designed for building APIs with Python.
- Fully supports asynchronous programming and is built around Python's type hints for improved development speed and validation.
- Automatic generation of OpenAPI and JSON Schema documentation.

Advantages:

- Very fast performance due to async capabilities, ideal for high-performance applications.

- Automatic generation of documentation via OpenAPI, making it easy to test and explore APIs.
- Lightweight and flexible, with easy integration into existing projects or microservices.

Disadvantages:

- Less built-in functionality compared to Django (e.g., no ORM, so you need to set up your own database integration).
- Smaller community compared to Django, although growing quickly.
- Can be challenging for developers used to synchronous programming due to async concepts.

Challenges Encountered in Development and Deployment

Django REST Framework (DRF):

Development Challenges:

ISSUE: *CORS error when connecting frontend to backend.*

FIX: *Solved by installing* `django-cors-headers`*, adding it to middleware, and allowing frontend origin.*

ISSUE: *React frontend couldn't fetch data initially.*

FIX: *Fixed by verifying correct URL for API requests and enabling CORS.*

Deployment Challenges:

ISSUE: *Render PostgreSQL connection issues when replacing default SQLite.*

FIX: *Solved by using* `dj-database-url` *to properly parse the Render-provided connection string.*

ISSUE: *React frontend couldn't connect to deployed API.*

FIX: *Double-checked backend URL and CORS settings; tested endpoints directly using Postman.*

ISSUE: Difficulty creating a superuser in deployment.

FIX: Created a createsuperuser.py script to programmatically create a superuser, and added it to the **start command** of the deployment. This ensured automatic superuser creation without manual input.

FastAPI:

Development Challenges:

ISSUE: *CORS errors when connecting frontend to backend.*

FIX: *Solved by adding* CORSMiddleware *in main.py and allowing Vite's dev server origin.*

Deployment Challenges:

ISSUE: *Render deployment crash due to invalid* *check_same_thread* *config.*

FIX: *Fixed by removing* check_same_thread *(only needed for SQLite).*

ISSUE: *Frontend failed to fetch data from deployed backend.*

FIX: *Fixed by ensuring correct backend URL and CORS setup.*