Difference between Django REST Framework (DRF) and FastAPI (highlight key features, advantages, disadvantages)

Django REST Framework (DRF):

- Built atop Django, offering deep integration with its ORM, admin interface, and authentication system.
- Provides robust serialization, viewsets, and a browsable API interface.
- Ideal for full-stack applications requiring comprehensive backend features.

FastAPI:

- Designed for building high-performance APIs with asynchronous support.
- Utilizes Python type hints for automatic data validation and documentation generation.
- Lightweight and suitable for microservices and real-time applications.

Challenges you encountered in development and deployment and how you solved them for both DRF and FastAPI.

Django REST Framework (DRF)

- **Navigating the Framework:** Initially, it was easy to feel overwhelmed without a clear roadmap. DRF offers many features, and without structured guidance, it's challenging to know where to start or what steps to follow. However, once familiar with its structure, the development process becomes more intuitive.
- **Deployment Variability:** Deployment requirements can differ based on user needs. While DRF is versatile, it's particularly well-suited for smaller, static projects. The built-in SQLite database simplifies deployment for such applications.
- **Project Initialization:** Django's project scaffolding is a significant advantage. Running django-admin startproject and python manage.py startapp generates the necessary files and directories, streamlining the setup process.

FastAPI

- **Manual Setup**: Unlike Django, FastAPI doesn't provide a built-in project structure. Developers need to manually create directories and files, such as routers and schemas, which can be time-consuming and requires a good understanding of the application's architecture.
- **Database Management:** Integrating PostgreSQL with FastAPI requires explicit handling of database schemas. Developers must ensure that tables are created or updated appropriately, often using tools like Alembic for migrations. This contrasts with Django's makemigrations and migrate commands, which handle schema changes more seamlessly, especially when using SQLite.
- **Deployment Complexity:** Deploying FastAPI applications can be more involved. Setting up models, defining relationships, and configuring the database require additional effort. In contrast, Django's integration with SQLite allows for quicker deployments, especially for applications with simpler requirements.