Advanced FastAPI Interview Questions & Answers

1. Q: **What makes FastAPI different from Flask or Django REST Framework?**
A: FastAPI is asynchronous, type-hinted, and built on Starlette & Pydantic, providing faster performance, automatic validation, and auto-generated Swagger/OpenAPI docs.

2. Q: **How does FastAPI handle request validation?**
A: It uses Pydantic models to enforce type validation, ensuring data integrity for query params, body, and headers automatically.

3. Q: **How can you implement JWT authentication in FastAPI?**
A: Using python-jose or PyJWT for token generation/verification, combined with OAuth2PasswordBearer dependency in FastAPI.

4. Q: **What is dependency injection in FastAPI?**
A: It allows you to define reusable dependencies (like DB sessions, auth checks) and inject them into path operations using the Depends() function.

5. Q: **How do you handle database connections in FastAPI?**
A: Using SQLAlchemy or Tortoise ORM with SessionLocal and Depends for dependency injection. Typically managed with lifespan events.

6. Q: **How does FastAPI support asynchronous programming?**
A: It supports async def functions natively, leveraging Python's asyncio, Starlette, and ASGI servers like Uvicorn.

7. Q: **What are Background Tasks in FastAPI?**
A: Tasks executed after returning a response using BackgroundTasks class, e.g., sending emails or logging.

8. Q: **How do you use Middleware in FastAPI?**
A: Middleware is added with @app.middleware("http"), wrapping requests and responses for logging, CORS, authentication, etc.

9. Q: **How does FastAPI generate API documentation automatically?**
A: It uses OpenAPI and JSON Schema from type hints and Pydantic models to build Swagger UI and ReDoc endpoints.

10. Q: **How do you structure a large FastAPI project?**
A: Split into modules like routers/, schemas/, models/, services/, core/, and use APIRouter for modular routes.

11. Q: **How do you handle errors globally in FastAPI?**
A: Using @app.exception_handler decorators or creating custom exception handlers with HTTPException.

12. Q: **Can FastAPI handle WebSockets?**
A: Yes, using @app.websocket with async methods. Useful for chat apps, live updates, etc.

13. Q: **How can you secure an API endpoint in FastAPI?**
A: Using OAuth2 scopes, JWT authentication, dependency injection, and HTTPS.

14. Q: **What are response models in FastAPI?**
A: Pydantic models used to control and validate the response structure with response_model parameter.

15. Q: **How do you handle environment variables in FastAPI?**
A: Using pydantic.BaseSettings or python-dotenv for configuration management.

16. Q: **How do you test FastAPI applications?**
A: Using TestClient from Starlette with pytest for unit tests and integration tests.

17. Q: **How do you enable CORS in FastAPI?**
A: With CORSMiddleware from Starlette.

18. Q: **How does FastAPI support file uploads?**
A: Using File and UploadFile parameters for handling file streams.

19. Q: **How do you implement rate limiting in FastAPI?**
A: By adding middleware or external libraries like slowapi.

20. Q: **How do you serve static files in FastAPI?**
A: Using app.mount("/static", StaticFiles(directory="static"), name="static").

21. Q: **Can FastAPI support GraphQL?**
A: Yes, using integrations like graphene or strawberry-graphql.

22. Q: **What is Depends in FastAPI?**
A: A way to inject dependencies (DB sessions, authentication, etc.) into routes automatically.

23. Q: **How do you implement pagination in FastAPI?**
A: By using query parameters (limit, offset) and applying them in DB queries.

24. Q: **How do you deploy a FastAPI app?**
A: Commonly with Uvicorn/Gunicorn + Nginx, Docker containers, or cloud platforms (AWS, GCP, Azure).

25. Q: **How does FastAPI ensure type safety?**
A: Through Python type hints and Pydantic validation at runtime.

26. Q: **How do you handle dependency overrides in FastAPI?**

A: Using app.dependency_overrides for testing or environment-specific changes.

27. Q: **How do you stream large responses in FastAPI?**
A: Using StreamingResponse for large files or data chunks.

28. Q: **How do you document security schemes in FastAPI?**
A: With OAuth2PasswordBearer, Security dependencies, and OpenAPI configuration.

29. Q: **How does FastAPI support database migrations?**
A: By integrating with Alembic when using SQLAlchemy.

30. Q: **How do you handle multiple routers in FastAPI?**
A: Using APIRouter and including them in app.include_router().

31. Q: **How do you log requests and responses in FastAPI?**
A: Using middleware or Python's logging module.

32. Q: **What is lifespan in FastAPI?**
A: It defines startup/shutdown events for initializing DB connections, caches, etc.

33. Q: **How do you manage background jobs in FastAPI beyond BackgroundTasks?**
A: Using Celery, RQ, or APScheduler for distributed task queues.

34. Q: **How do you handle security headers in FastAPI?**
A: With SecurityMiddleware or by manually setting headers in responses.

35. Q: **How do you integrate FastAPI with Redis?**
A: Using libraries like aioredis for caching, session management, or pub/sub.

36. Q: **How do you use WebSockets for real-time notifications?**
A: By maintaining connections in memory/Redis and sending events via @app.websocket.

37. Q: **How do you handle multipart/form-data requests in FastAPI?**
A: With Form and UploadFile for file + form data handling.

38. Q: **How do you improve performance of a FastAPI app?**
A: Async DB queries, connection pooling, caching, and optimizing middleware.

39. Q: **How do you secure sensitive data in FastAPI?**
A: Environment variables, hashing passwords (bcrypt), and encrypted storage.

40. Q: **What are common production issues with FastAPI?**
A: Blocking I/O in async code, memory leaks due to unclosed sessions, improper error handling, and CORS misconfigurations.