FastAPI Interview Questions & Answers

■ Basic Level (Freshers / 1 year experience)

Q: What is FastAPI?
A: FastAPI is a modern, fast (high-performance) web framework for building APIs with Python. It is built on Starlette (for web handling) and Pydantic (for data validation). It uses Python type hints for automatic validation and documentation.

Q: What are the advantages of FastAPI?
A: • Very fast (comparable to Node.js & Go)
• Automatic API documentation using Swagger UI & ReDoc
• Built-in data validation using Pydantic
• Supports asynchronous programming with async/await

Q: How do you create a simple FastAPI application?
A: from fastapi import FastAPI
app = FastAPI()

@app.get('/')
def home():
return {'message': 'Hello, FastAPI!'}

Run using: uvicorn main:app --reload

Q: What is Uvicorn in FastAPI?
A: Uvicorn is an ASGI server used to run FastAPI applications. It handles concurrent requests and provides high performance.

Q: How do you handle query parameters in FastAPI?
A: @app.get('/items/')
def read_item(skip: int = 0, limit: int = 10):
return {'skip': skip, 'limit': limit}

■ Intermediate Level (2+ Years Experience)

Q: What are dependencies in FastAPI?
A: Dependencies are a way to inject reusable logic (like DB connections, authentication, logging) into path operations.
Example:
from fastapi import Depends

def get_token(token: str):
return token

@app.get('/secure-data/')
def secure_data(token: str = Depends(get_token)):
return {'token': token}

Q: How does FastAPI handle request validation?

A: FastAPI uses Pydantic models for validation.
Example:

```
from pydantic import BaseModel

class User(BaseModel):
name: str
age: int

@app.post('/users/')
def create_user(user: User):
return user
```

If invalid data is passed, FastAPI automatically returns a 422 Unprocessable Entity error.

Q: What's the difference between synchronous and asynchronous endpoints in FastAPI?
A: • Synchronous (def) → Blocks until the request is complete
• Asynchronous (async def) → Allows concurrent request handling

```
@app.get('/sync')
def sync_api():
return {'msg': 'Synchronous'}

@app.get('/async')
async def async_api():
return {'msg': 'Asynchronous'}
```

Q: How do you handle authentication in FastAPI?
A: Using OAuth2PasswordBearer or JWT tokens.
Example:

```
from fastapi.security import OAuth2PasswordBearer

oauth2_scheme = OAuth2PasswordBearer(tokenUrl='token')

@app.get('/users/me')
def read_users_me(token: str = Depends(oauth2_scheme)):
return {'token': token}
```

Q: How do you document APIs in FastAPI?
A: FastAPI automatically generates documentation at:
• http://127.0.0.1:8000/docs (Swagger UI)
• http://127.0.0.1:8000/redoc (ReDoc UI)