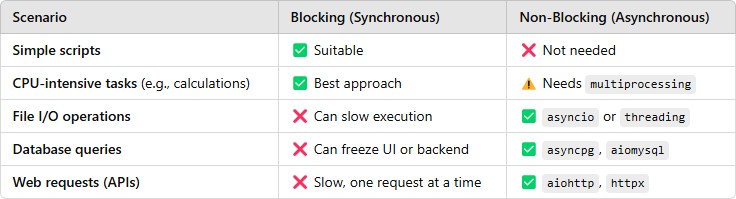
# What is FastAPI?

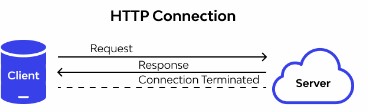
* FastAPI is a high-performance web framework for building APIs with Python. It is built on Starlette and Pydantic, supporting async programming and automatic API documentation.

# Starlette

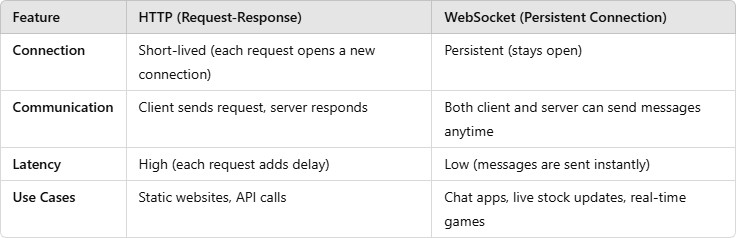
* FastAPI is built on Starlette, meaning all FastAPI applications inherit Starlette's features, such as:
* Asynchronous request handling
* WebSockets support
* Background tasks
* Middleware support
* CORS (Cross-Origin Resource Sharing) handling
* Blocking vs Non Blocking
* In programming, blocking and non-blocking code refer to how a program handles operations that take time, such as I/O operations, network requests, or database queries.



* Python functions run synchronously by default
* Each operation must complete before the next one starts This can cause delays, especially when dealing with:
  + I/O-bound tasks (Database queries, API requests)
  + Network operations (HTTP requests)
  + File handling (Reading/writing large files)
* Async capability allows the program to continue executing other tasks while waiting for an operation to complete
* Depending on purpose there are two types of API communication models:
  + HTTP
  + WebSocket







## CORS (Cross-Origin Resource Sharing)

* Security feature in web browsers that restricts web pages from making requests to a different domain than the one that served the page

Objectives

* Prevents unauthorized scripts from accessing APIs from different origins
* Ensures APIs can be accessed only by trusted clients

### Malicious Websites Stealing Your Data (CSRF - Cross-Site Request Forgery)

* Without CORS, a fake website could send a request to your API pretending to be you
* If you were logged in, it could steal sensitive data like account info, transactions, or personal details.

### Unauthorized Access to Your API

* If an attacker hosts a malicious website that tries to access your API, CORS blocks it.
* This ensures that only trusted domains (your frontend) can make requests to your API.

### Preventing Data Leaks

* If an API allows requests from any website, a hacker can scrape your data and use it for fraud.- CORS helps limit access to only trusted websites

# Background Tasks (FastAPI)

* Sending emails after user registration
* Processing large files or reports
* Logging and saving analytics data
* Calling an external API without making the user wait

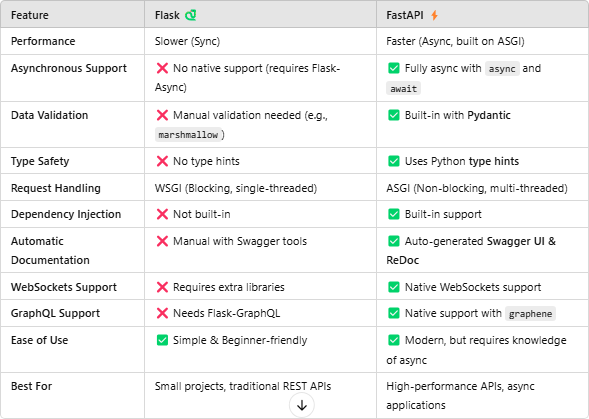
# What is Pydantic?

* Python library used for data validation and serialization.
* Ensures that data matches the expected structure and type, preventing errors in API requests.
* Built-in with FastAPI (No need for extra setup)
* Validates JSON data automatically
* Converts types (e.g., string to int)
* API documentation provides structured details about:
  + API endpoints
  + Parameters
  + Responses
* FastAPI automatically generates Swagger and ReDoc documentation.
* API documentation provides structured details about:
  + API endpoints
  + Parameters
  + Responses
* FastAPI automatically generates Swagger and ReDoc documentation.

# Tools for API Documentation

* Swagger UI (auto-generated by FastAPI)
* ReDoc
* Postman
* Redocly

# Flask vs FastAPI



FastAPI - Path Parameters

* Path parameters allow dynamic values in the URL.
* Example: /items/{item\_id}

# FastAPI - Query Parameters

* Query parameters are used in URLs with '?'.
* Example: /search/?name=Item&limit=10

# FastAPI - Response Model

* Response models define structured API responses using Pydantic models.

# FastAPI - Data Validation

* FastAPI uses Pydantic to automatically validate request data types.

# FastAPI - Dependency Injection

* FastAPI provides built-in dependency injection for reusable logic across routes.

# FastAPI - Database Connectivity

* FastAPI supports SQL (SQLAlchemy) and NoSQL (MongoDB with Motor) databases.

# SQL vs NoSQL

* SQL: Relational, fixed schema, best for structured data.
* NoSQL: Flexible schema, scalable, best for unstructured data.

# What is SQLAlchemy and How to

Use It?

* SQLAlchemy is an ORM for SQL databases, allowing interaction using Python objects.

# What is Motor and How to Use It?

* Motor is an async MongoDB driver, used for non-blocking database operations.

# What is Uvicorn?

* Uvicorn is an ASGI web server used to run FastAPI applications efficiently.

## Advance Python Programming

Concepts

* Isolated environment where you can install packages and dependencies without affecting the system-wide Python installation

✅ Avoid conflicts between different Python projects

✅ Use different package versions for different projects

✅ Keep the global Python environment clean

✅ Easily share dependencies using requirements.txt

* Create a Virtual Environment:

python -m venv venv

(New folder named venv/ is created, containing an isolated Python setup)

* Activate the Virtual Environment: venv\Scripts\activate

(To get inside the virtual environment with venv prompt)

* Install Dependencies Inside the Virtual Environment pip install fastapi uvicorn

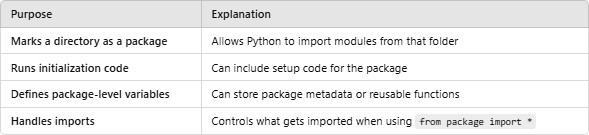
(installed packages will be available only inside virtual environment)

* + The init .py file is a special file in Python packages that tells Python to treat a directory as a package.
  + Without init .py → Python won't recognize the folder

as a package.

* + With init .py → You can import modules from that

package.



* + System-level variables that store configuration settings without exposing them in code:
    - Sensitive information
    - System-wide settings

|  |
| --- |
| **Sensitive Information** |
| Storing API Keys |
| Database Credentials |
| App Mode (Prod/Test) |
|  |
|  |

|  |
| --- |
| **System-wide Settings** |
| Timezone |
| Proxy Routing |
| Global llimits for APIs (eg. timeout) |
| Policy Enforcement |
|  |

* + Environment variables store system-level configurations
  + Improves security, portability, and flexibility
  + Can be accessed using os.getenv()
  + Use .env files with python-dotenv for easier management

Python initialization (

init

.py)

