LEARNING GEN AI

- BY SANJANA MCS

FAST API

Introduction to FastAPI

What is FastAPI?

FastAPI is a modern, high-performance web framework for building APIs with Python. It is designed to be **fast**, **easy to use**, and **efficient**, making it a great choice for beginners and professionals alike.

FastAPI is based on **Python type hints** and uses **asynchronous programming** to make API responses faster. It is often used to create web applications, data processing services, and machine learning model APIs.

Why Use FastAPI?

FastAPI is popular because:

1. **Blazing Fast**: It is one of the fastest Python frameworks, as it is built on **Starlette** and **Pydantic**.
2. **Easy to Learn**: Simple and clean code, similar to Flask but more powerful.
3. **Built-in Data Validation**: Uses Python type hints to automatically check request data.
4. **Asynchronous Support**: Can handle many requests at once without slowing down.
5. **Interactive API Documentation**: Automatically generates Swagger UI and ReDoc for easy testing.
6. **Great for Machine Learning & AI**: Easily deploy AI/ML models as APIs.

Components of Fastapi

FastAPI is a modern web framework for building APIs with Python. It has several key components:

1. **Routes (Endpoints)** – Define API paths using `@app.get()`, `@app.post()`, etc.
2. **Pydantic Models** – Used for request validation and response serialization.
3. **Dependency Injection** – Helps manage reusable dependencies like database sessions.
4. **Middleware** – Intercepts requests/responses for logging, authentication, etc.
5. **Background Tasks** – Allows running tasks asynchronously without blocking the main request.
6. **Exception Handling** – Manages errors gracefully with custom responses.
7. **Automatic Documentation** – Generates OpenAPI and Swagger docs automatically.

Report on CRUD Operations in FastAPI

CRUD stands for **Create, Read, Update, and Delete**—the core operations performed on a database. In FastAPI, these operations are implemented using HTTP methods such as **POST, GET, PUT, and DELETE**.

1. Create (POST)

Used to insert new records into the database. The client sends data in JSON format, and the server validates and stores it.

2. Read (GET)

Retrieves data from the database. It can fetch all records or a specific record based on parameters.

3. Update (PUT/PATCH)

Modifies existing records. The **PUT** method updates the entire resource, while **PATCH** modifies only specific fields.

4. Delete (DELETE)

Removes records from the database using unique identifiers like an ID.

Example Use Case

Consider a **Bug Tracking System** where:

- A developer **creates** a new bug report using `POST /bugs`.
- The QA team **retrieves** bug details using `GET /bugs/{id}`.
- A developer **updates** the bug status using `PUT /bugs/{id}`.
- An admin **deletes** resolved bugs using `DELETE /bugs/{id}`.

Few Important points to note on:

FastAPI vs Other Frameworks

Feature	FastAPI	Flask	Django REST	Express (Node.js)
Speed	✅ Very Fast	❌ Slow	❌ Slow	✅ Fast
Async Support	✅ Yes	❌ No	❌ No	✅ Yes
Type Validation	✅ Yes (Pydantic)	❌ No	✅ Yes	❌ No
Built-in Docs	✅ Yes	❌ No	❌ No	❌ No
Best For	APIs & ML Models	Small Apps	Large Apps	Web Apps

Conclusion

FastAPI is an excellent choice for building APIs because it is **fast, simple, and powerful**. It is widely used in **backend development, machine learning, and cloud applications**.

Key Takeaways

- FastAPI is the fastest Python API framework.
- Automatically generates API documentation.
- Uses Python type hints for validation.
- Perfect for modern, async-based applications.