Introduction

The FastAPI Blog API is designed to manage blog posts and comments efficiently using FastAPI and MongoDB. The application enables users to perform CRUD (Create, Read, Update, Delete) operations while leveraging MongoDB's query capabilities.

This document outlines the design decisions, implementation choices, and API documentation to help understand the system's architecture and functionality.

Design Decisions

Choice of FastAPI as the Web Framework

- **Performance:** FastAPI is an asynchronous web framework built on Starlette and Pydantic, making it faster than Flask or Django.
- **Automatic Documentation:** OpenAPI and ReDoc documentation are automatically generated.
- **Built-in Validation:** Uses Pydantic models to validate request/response data.

Choice of MongoDB as the Database

- **NoSQL Flexibility:** Since blog posts and comments have varying structures, a document-based store like MongoDB is more suitable than a relational database.
- **Scalability:** MongoDB supports horizontal scaling, making it ideal for large-scale applications.

Data Model

The database follows a two-collection model:

Posts Collection (posts)

- _id: Unique identifier
- title: Blog post title
- content: Blog post content
- created_at: Timestamp
- updated_at: Timestamp

Comments Collection (comments)

- _id: Unique identifier
- post_id: Reference to a blog post
- text: Comment text
- created_at: Timestamp

- **Validation using Pydantic** to prevent injection attacks.

Implementation Choices

🔹 **API Development**

- Asynchronous operations are used for MongoDB interactions (async def).
- CRUD operations implemented for both blog posts and comments.
- MongoDB Query Operators used for filtering and sorting.

🔹 **Technologies Used**

- **Backend:** FastAPI
- **Database:** MongoDB
- **ORM:** Motor (Async MongoDB Driver)
- **Testing:** Pytest
- **Dependency Management:** uv or pip

🔹 **Folder Structure**

```
/blog-api
├── main.py            # FastAPI entry point
├── db.py              # Database connection
├── models.py          # Pydantic models
├── routes.py          # API endpoints
├── tests/             # Unit tests
│   ├── test_routes.py     # Test file
├── requirements.txt       # Dependencies
├── README.md          # Project documentation
```

API Documentation (User Guide)

Base URL

http://127.0.0.1:8000

Blog Post Endpoints

Method	Endpoint	Description
POST	/posts/	Create a new blog post

Method	Endpoint	Description
GET	/posts/	Get all blog posts
GET	/posts/{id}	Get a post by ID
PUT	/posts/{id}	Update a post's title/content
DELETE	/posts/{id}	Delete a post

Comments Endpoints

Method	Endpoint	Description
POST	/comments/	Add a comment to a post
GET	/comments/{post_id}	Get comments for a post
PUT	/comments/{id}	Update a comment's text
DELETE	/comments/{id}	Delete a comment

API Documentation (Swagger UI)

After running the FastAPI server, API documentation can be accessed at:

- **Swagger UI:** http://127.0.0.1:8000/docs
- **ReDoc UI:** http://127.0.0.1:8000/redoc

Testing

- **Pytest** is used to validate API functionality.
- Run tests using:

  pytest tests -v

Conclusion

This project efficiently manages blog posts and comments using FastAPI and MongoDB, providing a scalable, async-first, and high-performance solution.