- Question 1: Explain how FastAPI handles asynchronous requests and its

benefits over synchronous code in Python.

- Question 2: Describe how dependency injection works in FastAPI and give an

example of its practical use.

- Question 3: Code walkthrough

**Explain how FastAPI handles asynchronous requests and its benefits over synchronous code in Python.**

FastAPI uses the async/await keywords to create coroutines which are used to handle asynchronous requests; this is known as concurrency in a non blocking manner.

The main advantage of using async/await is the execution of a certain block of I/O intensive operation such as a REST API call or a DB call can be paused added to the event loop while we are waiting for the response and the processing of some other task can be done in the meantime. This leads to a much better utilisation of resources and leads to better performance.

However this is true only for I/O bound tasks if awaiting is done for CPU intensive tasks it might in fact reduce performance in certain cases and you should be utilising multiprocessing in such cases.

The main benefits of asynchronous code over synchronous code is that it's much more performant when it comes to requests with time consuming I/O operations as it handles it concurrently which also means more requests can be handled at the same time efficiently.

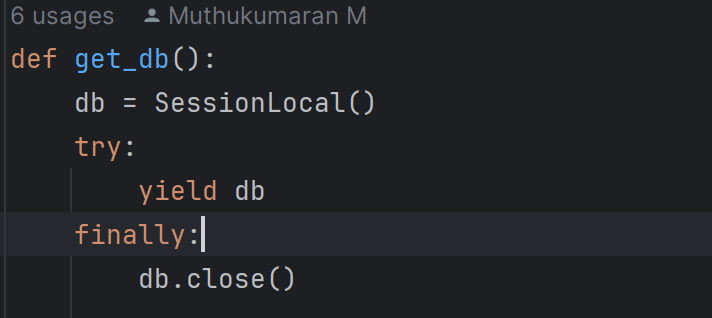
**Describe how dependency injection works in FastAPI and give an**

**example of its practical use.**

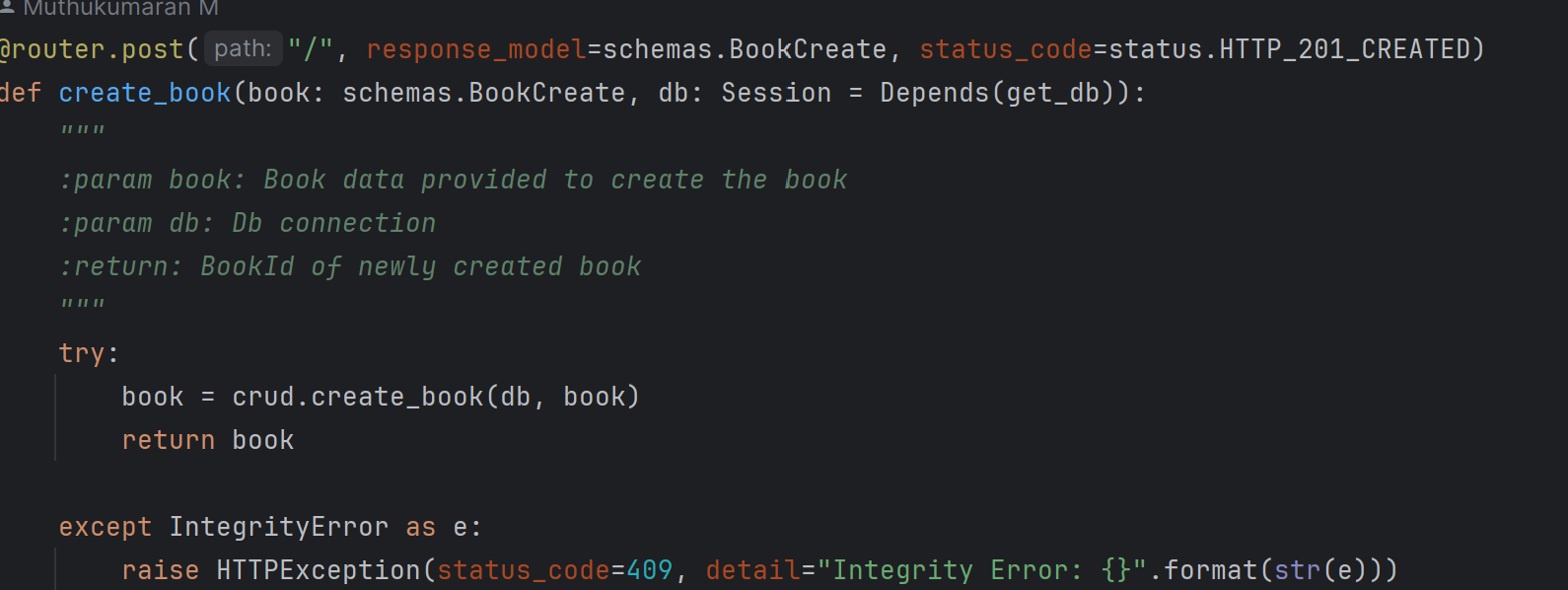
In the Fastapi app I created and shared as part of the assignment. I am passing the database connection which is a dependency required to process the incoming requests hitting any of the endpoints.

I pass this dependency as a function parameter which FastAPI automatically injects when processing that particular request, this is known as dependency injection.

FastAPI manages the dependency by itself initialising the dependency and cleaning it up as needed in the app.The dependency was defined as a function which has to be executed to create the database connection and close it when the request function is done.



Above is snippet of the database connection from the assignment which a dependency



Above is a snippet of an endpoint to which the dependency is passed as function parameter to create a record in the db using the connection.

**Code walkthrough**

The application created is divided into several files.

main.py : Entry point of the application

database.py: Has some basic configurations related to sqlite and db\_connection dependency.

crud.py: Defined a separate module with the ORM queries used by the endpoints in this file.

models.py: Defined the models and relationships using SQLAlchemy which is used to define the tables in the sqlite db.I have added various constraints and indexed the title column for performance reason as it would be frequently queried.

schemas.py: Defined the models using Pydantic for serialising/deserializing and validations for all the fields being used.

test\_main.py:Wrote unit test using Pytest/HTTPX to test the endpoints.Can be executed using the CLI command *pytest*.Tests are very basic but have around 90% coverage.

routers/book.py: I have defined all the required endpoints in this particular file related to books and reviewing them. I contemplated having reviews as a separate service but decided to nest it inside the book service itself.

I have done basic error handling for cases such as querying books using invalid IDs, adding a book which has already been created.

Mocked the process of sending emails after a review is submitted to the reviewer. Ideally I should have used something like Celery, defined an SMTP server or used a mail service.But due to time constraints I am just writing it to a text file.

I am requesting the email id as part of request but ideally I should have defined a user model with the id and fetch from that model. Also using that model to automatically name the reviewer and for authentication and authorization purposes as well.So that review cannot be modified by anyone else.

You can test out the endpoints at 127.0.01:8000/docs which has all the required details to consume and interact with them.