**Code Walkthrough**

**Imports and Setup:**

from fastapi import FastAPI, HTTPException, BackgroundTasks, Depends

from sqlalchemy import create\_engine, Column, Integer, String, ForeignKey

from sqlalchemy.orm import sessionmaker, Session, relationship

from pydantic import BaseModel

from typing import List, Optional

from fastapi.testclient import TestClient

**Import necessary modules and packages:**

- FastAPI: Framework for building APIs.

- HTTPException: Exception to raise when an HTTP error occurs.

- BackgroundTasks: Helper class to run background tasks.

- Depends: Dependency injection mechanism.

- create\_engine: Function to create a database engine.

- Column, Integer, String, ForeignKey: Classes to define database schema.

- sessionmaker: Function to create a session factory.

- relationship: Function to define relationships between database tables.

- BaseModel: Base class for Pydantic models.

- List, Optional: Type hints for lists and optional values.

- TestClient: Test client for API testing.

**Initialize FastAPI and Database**

app = FastAPI()

SQLALCHEMY\_DATABASE\_URL = "sqlite:///./test.db"

engine = create\_engine(SQLALCHEMY\_DATABASE\_URL)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

- Initialize FastAPI application.

- Define the SQLite database URL.

- Create a database engine using SQLAlchemy.

- Create a session factory for database interactions.

**Define Pydantic Models**

class Book(BaseModel):

title: str

author: str

publication\_year: int

class Review(BaseModel):

text\_review: str

rating: int

**Define Pydantic models for data validation:**

- Book: Represents book details.

- Review: Represents review details.

**Define SQLAlchemy Models**

class BookDB(Base):

\_\_tablename\_\_ = "books"

id = Column(Integer, primary\_key=True, index=True)

title = Column(String, index=True)

author = Column(String, index=True)

publication\_year = Column(Integer)

class ReviewDB(Base):

\_\_tablename\_\_ = "reviews"

id = Column(Integer, primary\_key=True, index=True)

text\_review = Column(String)

rating = Column(Integer)

book\_id = Column(Integer, ForeignKey("books.id"))

book = relationship("BookDB", back\_populates="reviews")

**Define SQLAlchemy models for database tables:**

- BookDB: Represents the books table.

- ReviewDB: Represents the reviews table and establishes a relationship with the books table.

**Create Database Tables**

Base.metadata.create\_all(bind=engine)

Create the database tables defined by the SQLAlchemy models.

**Define API Endpoints**

@app.post("/books/", response\_model=Book)

def create\_book(book: Book, db: Session = Depends(get\_db)):

...

@app.post("/books/{book\_id}/reviews/", response\_model=Review)

def create\_review(book\_id: int, review: Review, background\_tasks: BackgroundTasks, db: Session = Depends(get\_db)):

...

@app.get("/books/", response\_model=List[Book])

def get\_books(author: Optional[str] = None, publication\_year: Optional[int] = None, db: Session = Depends(get\_db)):

...

@app.get("/books/{book\_id}/reviews/", response\_model=List[Review])

def get\_book\_reviews(book\_id: int, db: Session = Depends(get\_db)):

...

**Define endpoints for CRUD operations and data retrieval:**

- create\_book: Add a new book to the database.

- create\_review: Add a new review for a book to the database.

- get\_books: Retrieve all books with optional filtering by author or publication year.

- get\_book\_reviews: Retrieve all reviews for a specific book.

**Background Task**

def send\_confirmation\_email(review\_id: int):

...

Define a background task function for sending confirmation emails after a review is posted.

**Test Client and Test Functions**

client = TestClient(app)

def test\_create\_book():

...

def test\_create\_review():

...

def test\_get\_books():

...

def test\_get\_book\_reviews():

...

Define a test client for API testing using FastAPI's TestClient.

Write test functions to test each API endpoint for correctness.