**Files Details**

1. **DatabaseSchema.txt** - Contains the schema for the database, including the structure of tables and relationships between entities.
2. **Dockerfile** - Used for building a Docker image for the application. It defines the environment in which the application runs, including dependencies.
3. **Problem Statement.docx** - Describes the problem that this application aims to solve.
4. **README.md** - A markdown file providing an overview of the project, including setup instructions and details on how to use the application.
5. **database.py** - Contains the database setup and configuration. It defines the get\_db function that provides a database session for dependency injection.
6. **docker-compose.yml** - Defines the Docker services required for the application, including the database and any other dependencies.
7. **llmmodelsetup.py** - Contains functions related to language model operations, such as generating text summaries using a pre-trained T5 model.
8. **main.py** - The main FastAPI application file that defines all the API endpoints.
9. **mlmodel.py** - Contains the machine learning model used for generating book recommendations based on genre and rating.
10. **models.py** - Defines the ORM models (e.g., Book, Review) for interacting with the database.
11. **requirements.txt** - Lists all Python dependencies required to run the application.
12. **schemas.py** - Defines Pydantic schemas (e.g., BookCreate, ReviewCreate) for request validation and serialization.

**API Endpoints**

**1. Create Book**

* **URL**: /books
* **Method**: POST
* **Request Body**:
  + BookCreate schema
* **Response**:
  + Book object represents the newly created book.
* **Description**: Creates a new book entry in the database.

**2. Get All Books**

* **URL**: /books
* **Method**: GET
* **Response**:
  + List of all Book objects in the database.
* **Description**: Retrieves all books.

**3. Get Book by ID**

* **URL**: /books/{id}
* **Method**: GET
* **Parameters**:
  + id: Integer representing the book ID.
* **Response**:
  + Book object with the specified ID.
* **Description**: Retrieves a single book by its ID.

**4. Update Book**

* **URL**: /books/{id}
* **Method**: PUT
* **Parameters**:
  + id: Integer representing the book ID.
* **Request Body**:
  + BookCreate schema with updated data.
* **Response**:
  + Updated Book object.
* **Description**: Updates an existing book entry.

**5. Delete Book**

* **URL**: /books/{id}
* **Method**: DELETE
* **Parameters**:
  + id: Integer representing the book ID.
* **Response**:
  + A message confirming the deletion.
* **Description**: Deletes a book entry by ID.

**6. Create Review**

* **URL**: /books/{id}/reviews
* **Method**: POST
* **Parameters**:
  + id: Integer representing the book ID.
* **Request Body**:
  + ReviewCreate schema.
* **Response**:
  + Review object representing the newly created review.
* **Description**: Adds a review to a specified book.

**7. Get Reviews for a Book**

* **URL**: /books/{id}/reviews
* **Method**: GET
* **Parameters**:
  + id: Integer representing the book ID.
* **Response**:
  + List of Review objects for the specified book.
* **Description**: Retrieves all reviews associated with a specified book.

**8. Summarize Text**

* **URL**: /summarize
* **Method**: POST
* **Request Body**:
  + text: String representing the text to summarize.
* **Response**:
  + A dictionary containing the generated summary.
* **Description**: Generates a summary of the provided text using a pre-trained T5 model.

**9. Get Book Recommendations**

* **URL**: /recommendations
* **Method**: GET
* **Parameters**:
  + genre: String representing the book genre.
  + average\_rating: Float representing the average rating of the book.
* **Response**:
  + A dictionary with the recommendation result.
* **Description**: Provides a recommendation for a book based on its genre and average rating.

**Dependencies**

* **FastAPI** - For building the web API.
* **SQLAlchemy** - ORM for interacting with the database.
* **Pydantic** - For data validation and serialization.
* **Docker** - For containerizing the application.
* **Machine Learning Model** - For generating book recommendations based on genre and rating.

**Database Structure**

Refer to DatabaseSchema.txt for a detailed schema layout, including table names, columns, and relationships.

**Machine Learning Model**

The recommendation system is defined in mlmodel.py, which contains a model that provides book recommendations based on genre and rating. The model takes inputs for genre and average rating and outputs a recommendation.

**Summary Generation**

llmmodelsetup.py contains the function generate\_summary, which uses a pre-trained T5 model to generate summaries of text provided to the /summarize endpoint.

**Error Handling**

* The application uses HTTPException from FastAPI to handle errors gracefully.
* Errors such as invalid data inputs or missing records result in appropriate HTTP status codes and messages.

**Setup and Installation**

1. **Clone the Repository**:

git clone <repository-url>

1. **Install Dependencies**:
   * Install Python dependencies:

pip install -r requirements.txt

1. **Run with Docker**:
   * Build and run the Docker containers:

docker-compose up --build

1. **Access the Application**:
   * The FastAPI application will be available at http://localhost:<port>, where <port> is defined in the Docker setup.