Problem Statement:

You are tasked with creating an intelligent book management system using Python, a locally running Llama3 generative AI model, and AWS cloud infrastructure. The system should allow users to add, retrieve, update, and delete books from a PostgreSQL database, generate summaries for books using the Llama3 model, and provide book recommendations based on user preferences. Additionally, the system should manage user reviews and generate rating and review summaries for books. The system should be accessible via a RESTful API and deployed on AWS.

Requirements:

1. **Database Setup:**
   - Use PostgreSQL to store book information.
   - Create a `books` table with the following fields: `id`, `title`, `author`, `genre`, `year_published`, `summary`.
   - Create a `reviews` table with the following fields: `id`, `book_id` (foreign key referencing `books`), `user_id`, `review_text`, `rating`.
2. **Llama3 Model Integration:**
   - Set up a locally running Llama3 generative AI model to generate summaries for books based on their content.
   - Integrate the Llama3 model to generate summaries for new book entries and review summaries for each book.
3. **Machine Learning Model:**
   - Develop a machine learning model to recommend books based on the `genre` and average `rating` fields.
   - Train the model on a sample dataset of books (you can use any open dataset or generate synthetic data).
4. **RESTful API:**
   - Develop a RESTful API with the following endpoints:
     - `POST /books`: Add a new book.
     - `GET /books`: Retrieve all books.
     - `GET /books/<id>`: Retrieve a specific book by its ID.
     - `PUT /books/<id>`: Update a book's information by its ID.
     - `DELETE /books/<id>`: Delete a book by its ID.
     - `POST /books/<id>/reviews`: Add a review for a book.
     - `GET /books/<id>/reviews`: Retrieve all reviews for a book.
     - `GET /books/<id>/summary`: Get a summary and aggregated rating for a book.
     - `GET /recommendations`: Get book recommendations based on user preferences.
     - `POST /generate-summary`: Generate a summary for a given book content.
5. **Asynchronous Programming:**
   - Implement asynchronous operations for database interactions and AI model predictions using `sqlalchemy[asyncio]` and `asyncpg`.
6. **AWS Deployment:**
   - Deploy the application on AWS using services such as EC2, Lambda, or ECS.
   - Ensure the database is hosted on AWS RDS.
   - Use AWS S3 for storing any model files if necessary.

    - Set up a CI/CD pipeline for automatic deployment.
7. **Authentication and Security:**
    - Implement basic authentication for the API.
    - Ensure secure communication with the database and API endpoints.

Bonus:

- Implement caching for the book recommendations using AWS ElastiCache.
- Add unit and integration tests for the API endpoints and AI model.
- Use AWS SageMaker for deploying and managing the machine learning model.

Instructions:

1. **Database Schema:**
    - Define the schema for the `books` and `reviews` tables in PostgreSQL.
2. **Llama3 Integration:**
    - Set up a local instance of the Llama3 model for generating book and review summaries.
    - Develop an endpoint to generate summaries for new book entries and review summaries.
3. **Model Training:**
    - Train a machine learning model for book recommendations and save the model for inference.
4. **API Development:**
    - Develop the RESTful API using FastAPI or Flask.
    - Implement the necessary asynchronous operations for database and AI model interactions.
5. **AWS Deployment:**
    - Set up an AWS account and use AWS services to deploy your application.
    - Ensure the API is accessible over the internet and securely connected to the PostgreSQL database.
6. **Testing and Documentation:**
    - Write unit tests for your API endpoints.
    - Document your API using Swagger or similar tools.
    - Provide clear instructions on how to set up and run your application.

Deliverables:

- Source code for the application, including the Llama3 integration, machine learning model, API, and database schema.
- Documentation on how to deploy and use the application.
- A link to the deployed application on AWS.

Evaluation Criteria:

- Correctness and efficiency of the solution.
- Proper use of asynchronous programming in Python.
- Effective implementation of Llama3 for summarization.
- Effective implementation of machine learning for recommendations.
- Quality of the RESTful API and adherence to REST principles.

- Successful deployment on AWS with proper configuration and security measures.
- Comprehensive testing and documentation.