**BookBazaar - Library Management and Review System**

**Table of Contents**

**1. Introduction**

- Project Goals

- Technologies Used

**2. Installation and Setup**

- Python Dependencies

- SQLite Setup

- MongoDB Setup

- Apache Configuration

**3. Running the Application**

- Starting the Flask Application

- Accessing the Application

**4. API Documentation**

- Overview of API Endpoints

- Detailed Endpoint Documentation

- Example API Calls

**5. Postman Testing**

- Importing Postman Collection

- Testing API Endpoints

**6. Troubleshooting**

- Common Issues and Fixes

**7. Conclusion**

---

**1. Introduction**

**Project Goals**

The **BookBazaar - Library Management and Review System** is designed to:

- Manage a library of books using SQLite.

- Allow users to add, update, and delete reviews for books using MongoDB.

- Provide a RESTful API for interacting with the system.

**Technologies Used**

- **Backend:** Flask (Python)

- **Database:** SQLite (for book management), MongoDB (for reviews)

- **API Testing:** Postman, Web Server

- **Deployment:** Apache

**2. Installation and Setup**

**Python Dependencies**

1. Install Python 3.11 or later.

2. Install the required Python packages:

A screenshot of a computer

Description automatically generated

**SQLite Setup**

1. Create a SQLite database named `bookbazaar.db`.

2. Create the (Users, Authors, Books) tables:

A screen shot of a computer screen

Description automatically generated

3. Insert sample data:

A computer screen shot of a black screen

Description automatically generated

**MongoDB Setup**

1. Install MongoDB and start the MongoDB server.

A black rectangle with blue text

Description automatically generated

2. Create a database named `bookbazaar\_reviews`.

A screenshot of a computer screen

Description automatically generated

3. Create a collection named `reviews`.

A screenshot of a computer program

Description automatically generated

**Apache Configuration (Optional)**

Deploying the application using Apache:

1. Install Apache and mod\_wsgi.

* Link: <https://httpd.apache.org/docs/2.4/platform/windows.html>

2. Configure the Apache virtual host to serve the Flask application.

3. Restart Apache to apply the changes.

**3. Running the Application**

**Starting the Flask Application**

1. Navigate to the project directory.

2. Run the Flask application:

A screenshot of a computer

Description automatically generated

3. The application will start at `http://127.0.0.1:5000`.

A screen shot of a computer

Description automatically generated

**Accessing the Application**

- Use a web browser or API testing tool (e.g., Postman) to interact with the API.

**(Look at Report File)**

**4. API Documentation**

**Overview of API Endpoints**

| HTTP Method | Endpoint | Description |

|------------------|---------------------------------------|-----------------------------------------------|

| GET | `/books/<id>/reviews` | Get all reviews for a book |

| POST | `/books/<id>/reviews` | Add a new review for a book |

| PUT | `/reviews/<review\_id>` | Update an existing review |

| DELETE | `/reviews/<review\_id>` | Delete an existing review |

Hint: Add (<http://127.0.0.1:5000/> Or http:// localhost:5000/ ) Before Endpoint

**Detailed Endpoint Documentation**

**GET /books/<id>/reviews**

- **Description**: Retrieve all reviews for a specific book.

- **Request**:

- **URL:** `http://127.0.0.1:5000/books/1/reviews`

- **Method:** `GET`

- **Response**: json

A computer screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

**POST /books/<id>/reviews**

- **Description**: Add a new review for a specific book.

- **Request**:

- URL: `http://127.0.0.1:5000/books/1/reviews`

- Method: `POST`

- **Body (JSON):**

A screenshot of a computer screen

Description automatically generated

- \*\*Response\*\*: json

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**PUT /reviews/<review\_id>**

- **Description**: Update an existing review.

- **Request**:

- **URL**: `http://127.0.0.1:5000/reviews/6785095d1e3ffbdb680e6967`

- **Method**: `PUT`

- **Body (JSON)**: json

A black rectangular object with blue and red text

Description automatically generated

- **Response**: json

A black rectangle with blue text

Description automatically generated

A screenshot of a computer

Description automatically generated

**DELETE /reviews/<review\_id>**

- **Description**: Delete an existing review.

- **Request**:

- **URL**: `http://127.0.0.1:5000/reviews/6785095d1e3ffbdb680e6967`

- **Method**: `DELETE`

- **Response**: json

A black rectangle with blue text

Description automatically generated

A screenshot of a computer

Description automatically generated

**5. Postman Testing**

**Importing Postman Collection**

1. Download the Postman collection JSON file for this project.

2. Open Postman and import the collection.

**Testing API Endpoints**

1. Use the imported collection to test all API endpoints.

2. Modify the request parameters (e.g., `book\_id`, `review\_id`) as needed.

**6. Troubleshooting**

**Common Issues and Fixes**

**1. SQLite Error: No such table:**

- Ensure the `Books` table exists in the `bookbazaar.db` database.

- Run the `CREATE TABLE` script if the table is missing.

**2. MongoDB Connection Error:**

- Ensure the MongoDB server is running.

- Verify the connection URI in the Flask application.

**3. Flask Application Not Starting:**

- Check for syntax errors in the Python code.

- Ensure all dependencies are installed.

**7. Conclusion**

The **BookBazaar - Library Management and Review System** provides a robust API for managing books and reviews. With clear documentation and examples, new developers can easily set up and use the system.

**Attachments**

- **Postman Collection**: `Sprints.postman\_collection.json`

- **SQLite Database**: `bookbazaar.db`

- **Flask Application**: `app.py`, `app2.py`