Event Booking API Documentation

API Documentation for Event Booking API (Laravel with Sanctum)

Approach:

This project is a Laravel-based RESTful API for managing events, attendees, and bookings. It enforces clean code practices, database integrity, and validation rules to ensure a robust booking system. Laravel Sanctum is used for token-based authentication to protect event management routes, while attendee registration and bookings remain open.

Key Components:

- ***Models:*** `Event`, `Attendee`, `Booking`, and `User`
- ***Controllers:*** Separated for API and view logic (`EventController`, `AttendeeController`, `BookingController`, `AuthController`)
- ***Authentication:*** Laravel Sanctum
- ***Testing:*** PHPUnit feature tests for all API operations

Setup Steps:

1. Clone the Repository
   git clone https://github.com/your-username/ bookingapi.git
   cd bookingapi
2. Install Dependencies
   composer install
   npm install
3. Setup Environment
   cp .env.example .env
   php artisan key:generate
4. Configure Database
   Update your `.env` file with your database credentials.
   ```
   DB_DATABASE=booking
   DB_USERNAME=root
   DB_PASSWORD=
   ```
5. Install Sanctum
   composer require laravel/sanctum
   php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"
   php artisan migrate

6. Run the App
   php artisan serve
   Visit: `http://localhost:8000`
7. Run Tests
   php artisan test
8. Hosting with Local Server (XAMPP/WAMP)
   If the application is not running, ensure that your local development server is started:
   - For **XAMPP**, start Apache and MySQL, then place the Laravel project folder in the `htdocs` directory.
   - For **WAMP**, start the server and place the project in the `www` directory.
   Then navigate to the app in the browser, e.g.:
   http://localhost/bookingApi/public
   Ensure `.env` database configuration matches your MySQL setup, and that migrations are executed.

Base URL: http://127.0.0.1:8000/api

Authentication

Register

POST /register

```
{
    "name": "Peter Parker",
    "email": "peterparket@marvel.com",
    "password": "marvel123"
}
```

Response:

```
{
    "user": { ... },
    "token": "access_token_here"
}
```

Login

POST /login

```
{
    "email": "john@example.com",
    "password": "password"
}
```

Response:

```
{
    "user": { ... },
    "token": "access_token_here"
}
```

Event Booking API Documentation

Use the token in headers:

Authorization: Bearer {token}

1. **Events (Authenticated Only)**

- List Events
  GET /events
- Create Event
  POST /events

```
{
      "name": "Event 1",
      "description": "This is testing Event",
      "date": "2025-05-15",
      "capacity": 1000,
      "country": "India"
}
```

- Get Event
  GET /events/{id}
- Update Event
  PUT /events/{id}

```
{
      "name": "Updated Name"
}
```

- Delete Event
  DELETE /events/{id}

2. **Attendees (Public)**

- Register Attendee
  POST /attendees

```
{
      "name": "Steve Rogers",
      "email": "steverogers@marvel.com"
}
```

- Update Attendee
  PUT /attendees/{id}

```
{
      "name": "Caption America"
}
```

- Delete Attendee
  DELETE /attendees/{id}

3. **Bookings (Public)**

- Create Booking
  POST /bookings

```
{
      "event_id": 1,
      "attendee_id": 1
}
```

- Delete Booking
  DELETE /bookings/{id}

➢ Validations:
Prevents:
- Duplicate booking
- Overbooking (capacity exceeded)

➢ Errors
- 401 Unauthorized: Missing or invalid token
- 422 Unprocessable Entity: Validation or logic error (e.g. duplicate booking)
- 404 Not Found: Resource does not exist

➢ Covers:
- Events (CRUD)
- Attendees (CRUD)
- Bookings (create/delete + validations)

Postman API Documentation:

To interact with and test this API via Postman:

1. **Setup:**
   i) Open Postman
   ii) Create a new **Collection** named `Event Booking API`
   iii) Add the following requests to the collection:
2. **Auth:**
   - `POST /api/register` – Register user
   - `POST /api/login` – Login and receive Bearer Token
3. **Events (require token)**
   - `GET /api/events` – List events
   - `POST /api/events` – Create event
   - `GET /api/events/{id}` – Get single event
   - `PUT /api/events/{id}` – Update event
   - `DELETE /api/events/{id}` – Delete event

4. **Attendees**
   - `POST /api/attendees` – Register attendee
   - `PUT /api/attendees/{id}` – Update attendee
   - `DELETE /api/attendees/{id}` – Delete attendee
5. **Bookings**
   - `POST /api/bookings` – Create booking
   - `DELETE /api/bookings/{id}` – Cancel booking

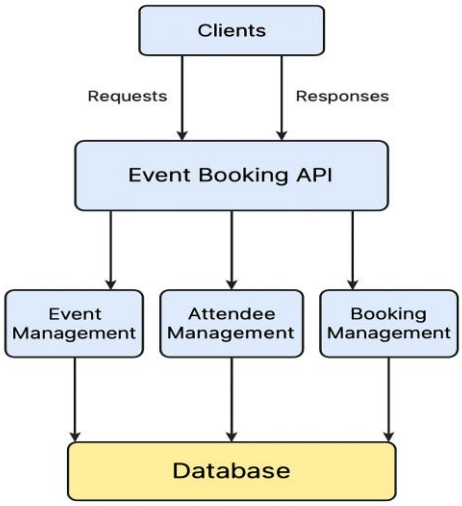
Assumptions:

- Events must have unique name-date combinations (implicitly handled by validation logic).
- Attendees can register and book events without authentication.
- Authenticated users (API consumers) can manage events.
- Booking is allowed only if the event has available capacity and hasn't already been booked by the attendee.
- XAMPP or WAMP server is activated.

Notes:

1. Include Authorization: Bearer {token} for protected routes
2. Always use API **Base URL** at the start of endpoint. I have attached Postman collection for your reference in the repository.

Architecture Diagram:



Event Booking API