Alpenlux

**The Hotel Management System**

**Preface :**

This is the documentation for our new Hotel – **Alpenlux** . This document explains each component in detail and also a technical documentation for the complete implementation of the application.

**Introduction:**

The StayInn is a hotel management backend system that takes care of providing APIs for creating, updating, fetching and deleting the different entities like Categories, Rooms, Bookings and CheckIns. In the next section, we can dive deeper into each entity and the mappings between them.

**Technical Configurations:**

This application is completely a backend solution offering various REST APIs which can be used for all the operations required for the complete hotel management. The storage used is MongoDb. The Hotel Management Suite will be the right word to describe it.

**Database Design:**

The database design is the fundamental aspect and a properly designed database marks the completion of the strong foundation of the application. The database design used in this application is as follows:

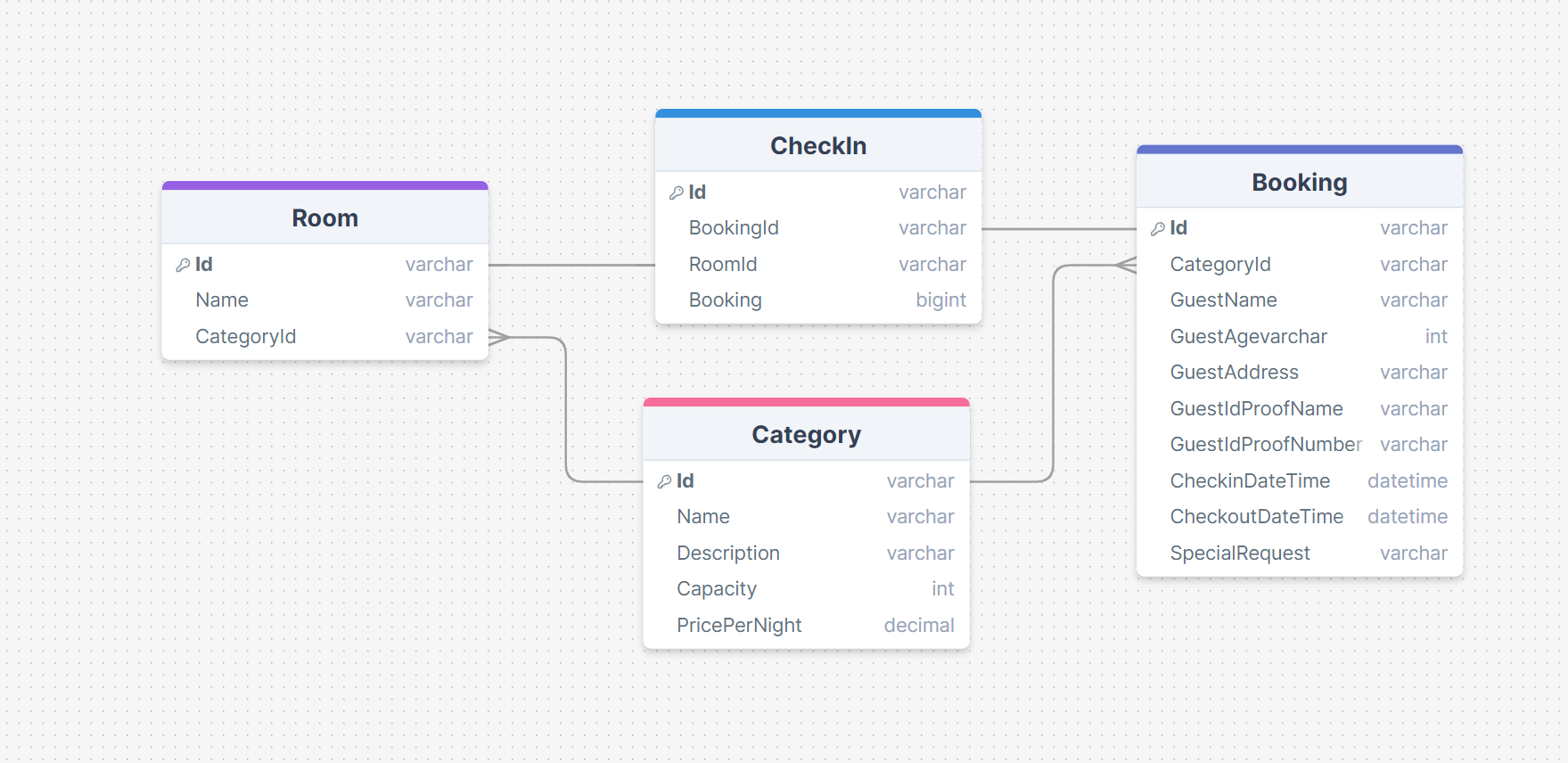


Fig 1.0 – Database Design

There are multiple ways the database components can be designed for this hotel management suite. Even more entities can be added to this design. However, I chose this design since the database should be very minimal and easy to understand for this task.

**Category:**

The Category entity is the parent for all the components since the overall category/type of rooms are stored in this table/container/collection. Since we are using MongoDb, the storing entities will be called as collections going further.

Different types of categories –

* Single
* Twin
* Queen
* Studio
* Rooms with view
* Junior Suite
* Suites
* Presidential Suite

**Room:**

As the name suggests, the room entity stores the different rooms available. Each room will have a categoryId associated with the entity which marks which room falls under which category. For example,

* S-101 – Single
* T-001 – Twin Room
* PS-001 – Presidential Suite

**Booking:**

The booking entity is responsible for storing the bookings made by guests. The booking can be made either by the guest using any frontend website developed based on this backend application and deployed to cloud for public usage or directly by the hotel front desk by using the frontend in hotel server.

The booking entity will have the guest details like name, age, address etc., and CategoryId for marking which booking is made for which category of room, based on which the hotel staff can assign the room for that booking using CheckIn API.

**CheckIn:**

The check-in entity stores the check-in of each booking. This check-in entity marks whether the booking has been successfully checked in or not. This entity contains bookingId, roomId which corresponds to the respective Id column of booking collection and Id column of room collection.

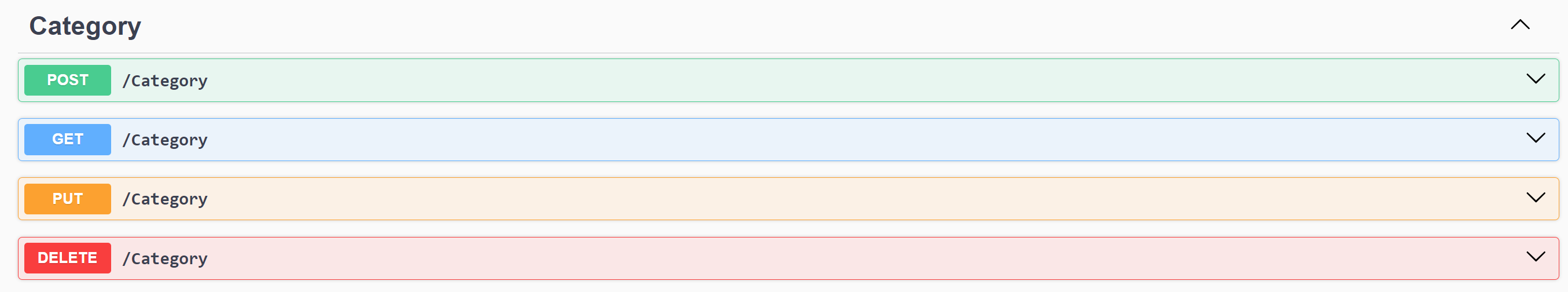
**APIs – Application Programming Interfaces:**

The APIs or the application programming interfaces gives the access to database collections for maintaining the functionality of the application.

This application contains many APIs which corresponds to different actions. Each API is explained in detail below:

**Category:**

The APIs under category section give all the essential CRUD operations like Create, Read, Update and Delete. As discussed in the previous section, this API is responsible for doing the CRUD operations on category entity.



**Room:**

The APIs under room section give all the essential CRUD operations like Create, Read, Update and Delete. As discussed in the previous section, this API is responsible for doing the CRUD operations on room entity.

A group of colored lines

Description automatically generated

**Booking:**

The APIs under booking section give all the essential CRUD operations like Create, Read, Update and Delete. As discussed in the previous section, this API is responsible for doing the CRUD operations on room entity.

A close-up of a colorful line

Description automatically generated

**CheckIn:**

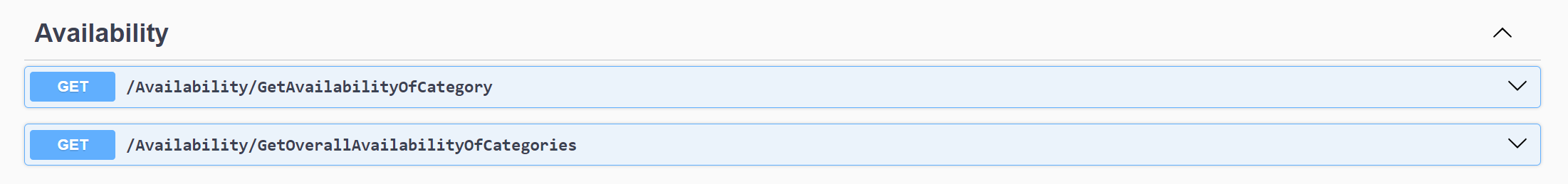
The APIs under checkin section give all the essential CRUD operations like Create, Read, Update and Delete. As discussed in the previous section, this API is responsible for doing the CRUD operations on checkin entity.

A group of colorful rectangular objects

Description automatically generated with medium confidence

**Availability:**

This section is very important for this application since the APIs under availability gives the exact availability of each room and also based on categories.



* **GetAvailabilityOfCategory**

This API requires CategoryId which can taken from the category api, checkin and checkout dates are given as the request and it responds with the number of available rooms in that category.

**Get CategoryId**

**A screenshot of a computer

Description automatically generated**

**Actual Availability API A screenshot of a computer

Description automatically generated**

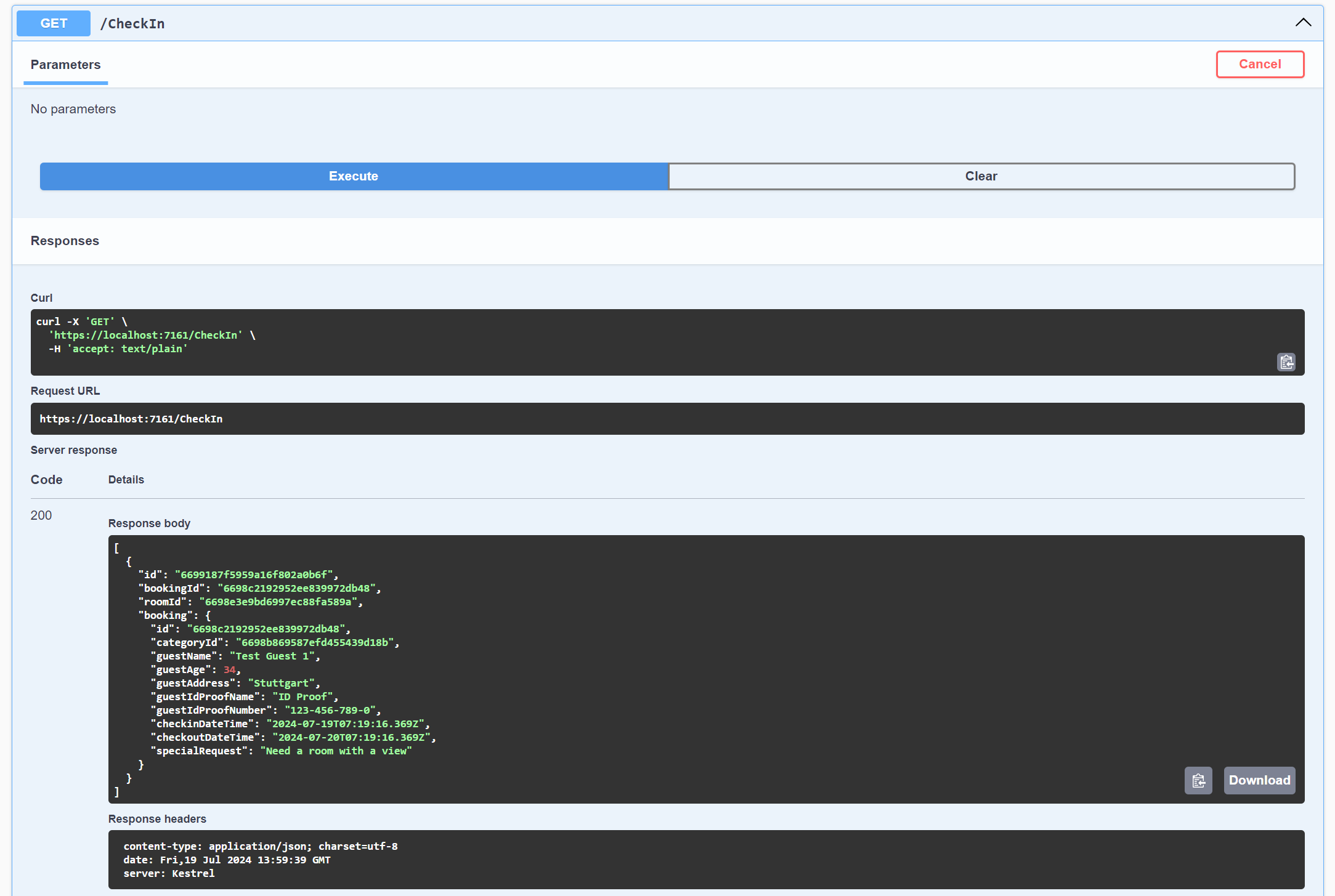
* **GetAvailabilityOfCategories**

This API requires only the checkin and checkout dates as the request and it responds with the number of available rooms in each of the category.

**A screenshot of a computer

Description automatically generated**

Even though there are two single rooms available in the hotel, only one is shown as available since the other has been checked in for that particular date.



When we use some other dates for the availability API, then two single rooms will be returned. Thus the room management, bookings and check in managements are designed.

There are plenty of improvements can be made to the application. I have many ideas for it, Unit Tests also and due to time constraints and multiple outages in Azure during this week, I was sharing my time with my current company project and this. Hence I am able to complete till this part.

Even with all this, this API is functionable and can be integrated with UI for a full fledged experience.