# Introduction

## Overview

This document describes the architecture and design of Car Parking Application. Build on open-source libraries using Spring Boot, Java the application is used to perform booking for customer car parking

## Purpose of the System

The application is created with a simple service API that will allow a client to manage their parking slot allocation. Parking slots may be queried and managed – allocated, amended or unallocated with below

**Features**

* A car park uploading many parking slots.
* A slot could be acquired for a X number of hours, where X >= 1h and X <= 4h
* When the acquired time period elapses, it should be possible to allocate the slot again

**Scenarios**

1. User enters a car park and requests a free slot for a number of hours.
2. User chooses to re-allocate their current slot for a number of hours.
3. User leaves car park before the allocated time has passed.

**Technical stack**

* The solution is written in Java & Spring Boot, Schedular, JPA
* The backend uses H2 in memory
* Used a TDD approach – Spring Boot Controller MockMVC, Spring Service Mock & DataJPA
* Expose API as REST resources

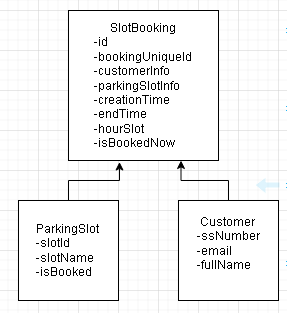
**Simplifications**

* Persistence is held in memory
* No authentication and authorization used
* Handled future reservations.
* No log or history is created

# System Design

There are three models involved Customer,ParkingSlot and SlotBooking so the below model is generated.

## Model Design



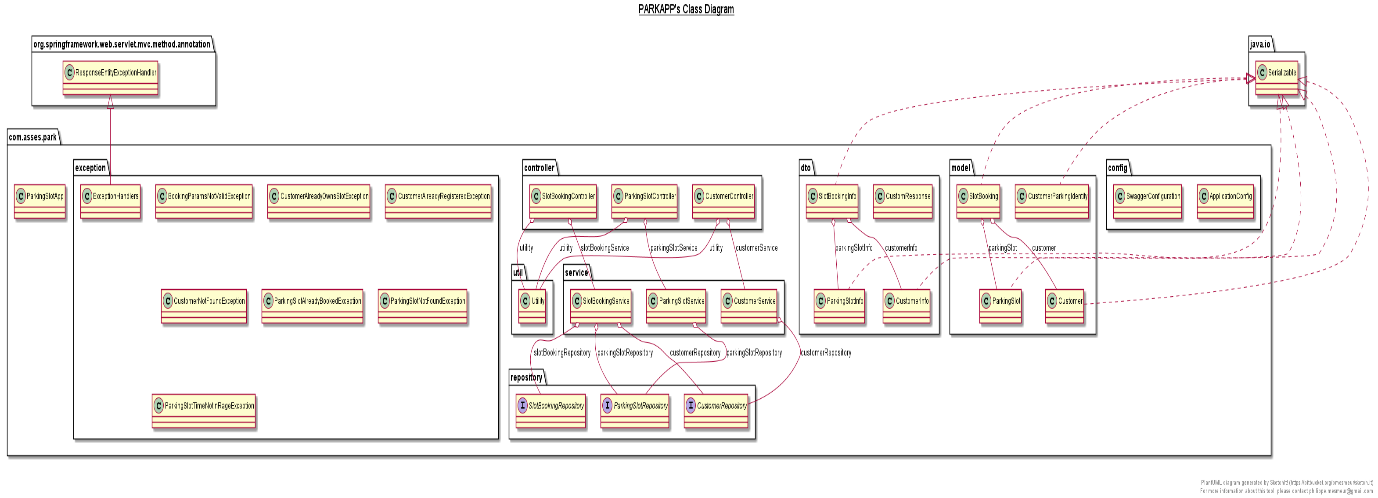
## ER Model

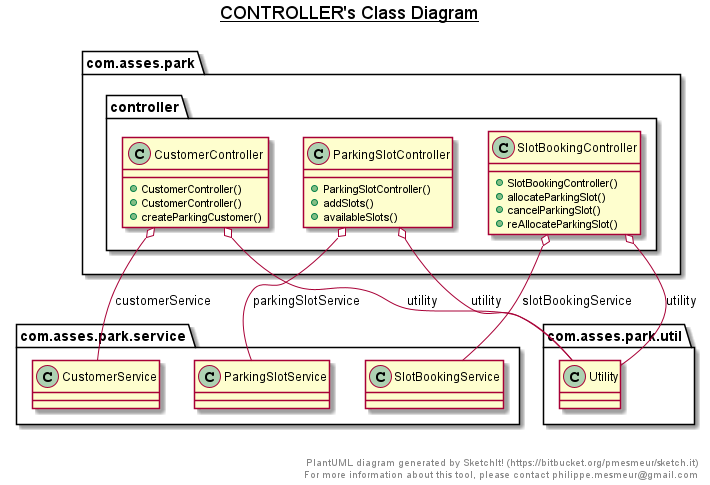
SlotBooking becomes the primary entity and the parkingSlot and customer are dependent entities hence the below ER model is designed and implemented.

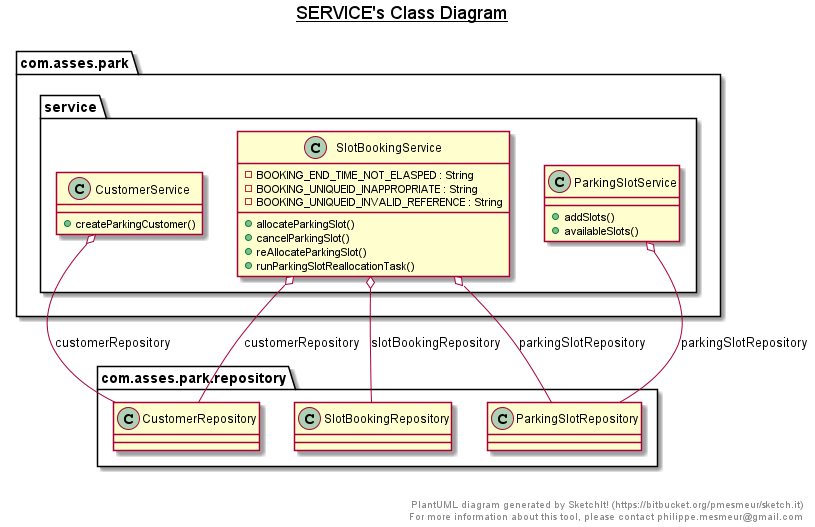


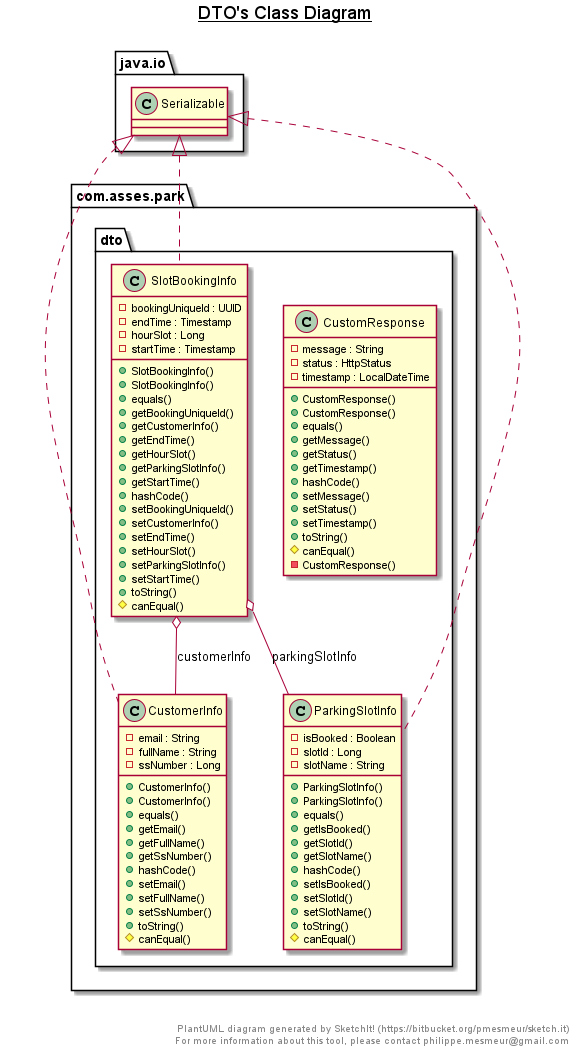
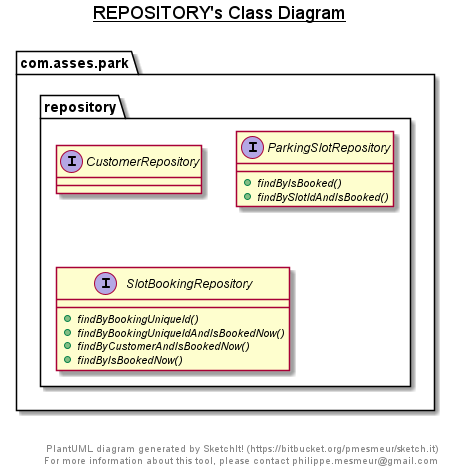
## Class Design

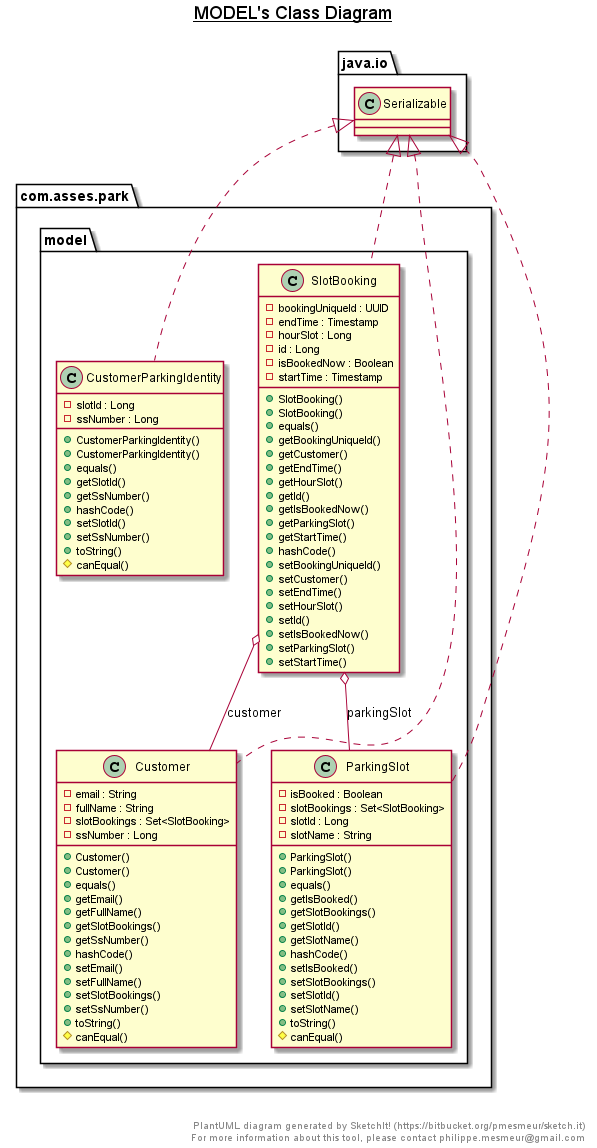
Class design of this system completely involves using typical REST API and uses Spring Boot to serve the purpose. The below are the classes used in the application.











## Sequence Design

The flow of the application for customer creation, adding slots, listing available parking slots are listed below

**Customer Creation Flow**



**Parking Slot Addition Flow**



**Listing Available Parking Slots Flow**

****

The flow of the application for allocateParkingSlot, reAllocateParkingSlot, cancelParkingSlot and runParkingSlotReallocationTask are attached as PDF since the sequence flow is more.



## Unit Test Cases

The application is designed to cover the unit test cases for all the three modules of Controller,Service and Repository for all three moles Customer,ParkingSlot and SlotBooking

As a TDD approach the application uses SpringRunner for JUnit and below are UTC technologies induvial parts of above specified three modules,

1. Spring Boot Controller MockMVC
2. Spring Service Mockito
3. DataJPA Test

# Application Usage

**3.1 Starting the application**

Note: Kindly install Java and Maven before proceeding to execute the application

Once the application is built using maven, we can find ParkApp-1.0-SNAPSHOT.jar inside target directory relative to application where its present now.

Then start the application using command in shell or command prompt,

Java -jar ParkApp-1.0-SNAPSHOT.jar

Now application gets started and ready to usage.

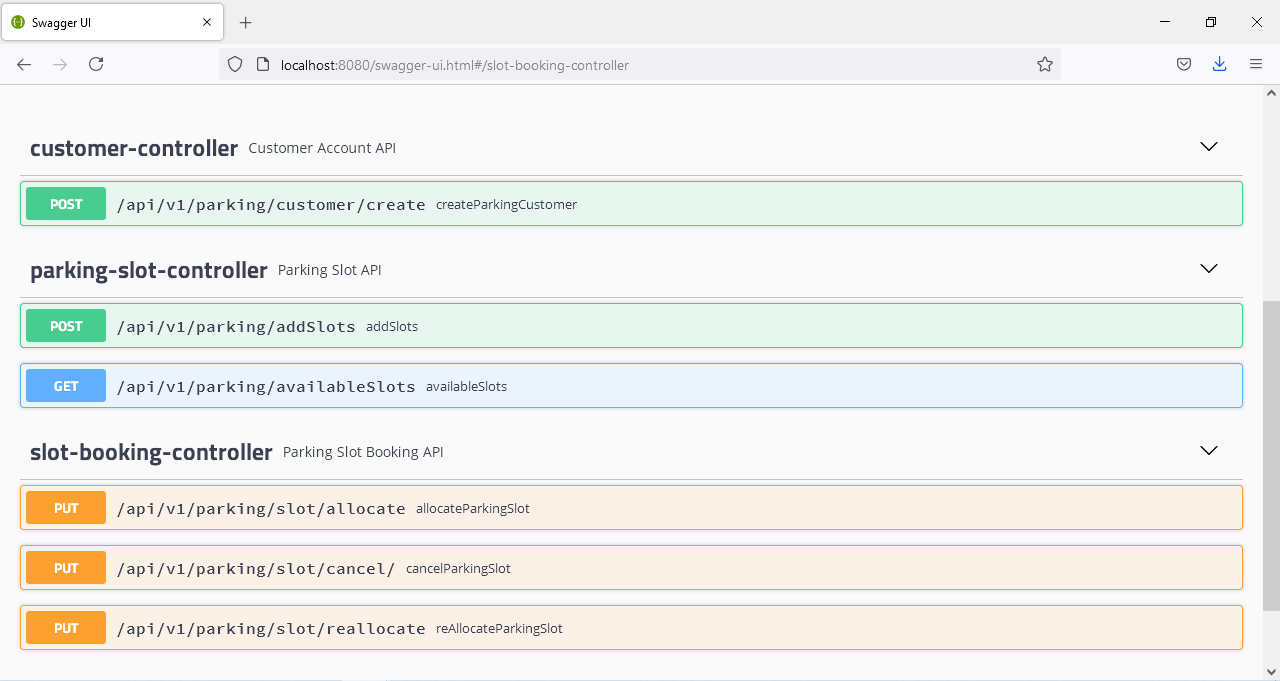
**End Points of API**

The application has below API which are exposed as REST API

|  |  |  |
| --- | --- | --- |
| **SNo** | **URI** | **Purpose** |
| 1 | <http://localhost:8080/api/v1/parking/customer/create> | To create customer |
| 2 | <http://localhost:8080/api/v1/parking/addSlots> | To add parking slots |
| 3 | <http://localhost:8080/api/v1/parking/availableSlots> | To get all parking slots |
| 4 | <http://localhost:8080/api/v1/parking/slot/allocate> | To allocate parking slots for a customer |
| 5 | <http://localhost:8080/api/v1/parking/slot/reallocate> | To reallocate parking slots for a customer |
| 6 | [http://localhost:8080/api/v1/parking/slot/cancel/?bookingUniqueId=<<uuid>>](http://localhost:8080/api/v1/parking/slot/cancel/?bookingUniqueId=%3c%3cuuid%3e%3e) | To cancel a booked parking slot |

**Swagger API Documentation**

The application also has built in swagger which provides detailed usage of API

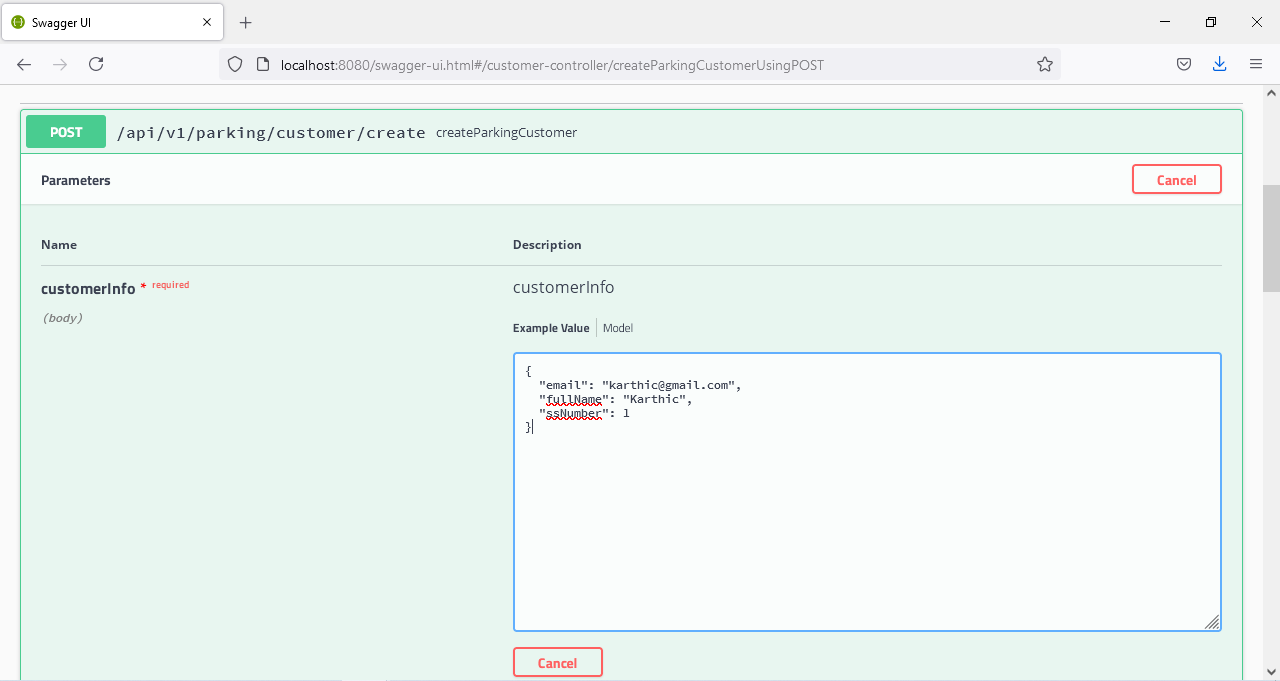


**Note**: For usage of APIs below use the exact JSON requests which are valid, if inappropriate values are passed process do not gets succeed.

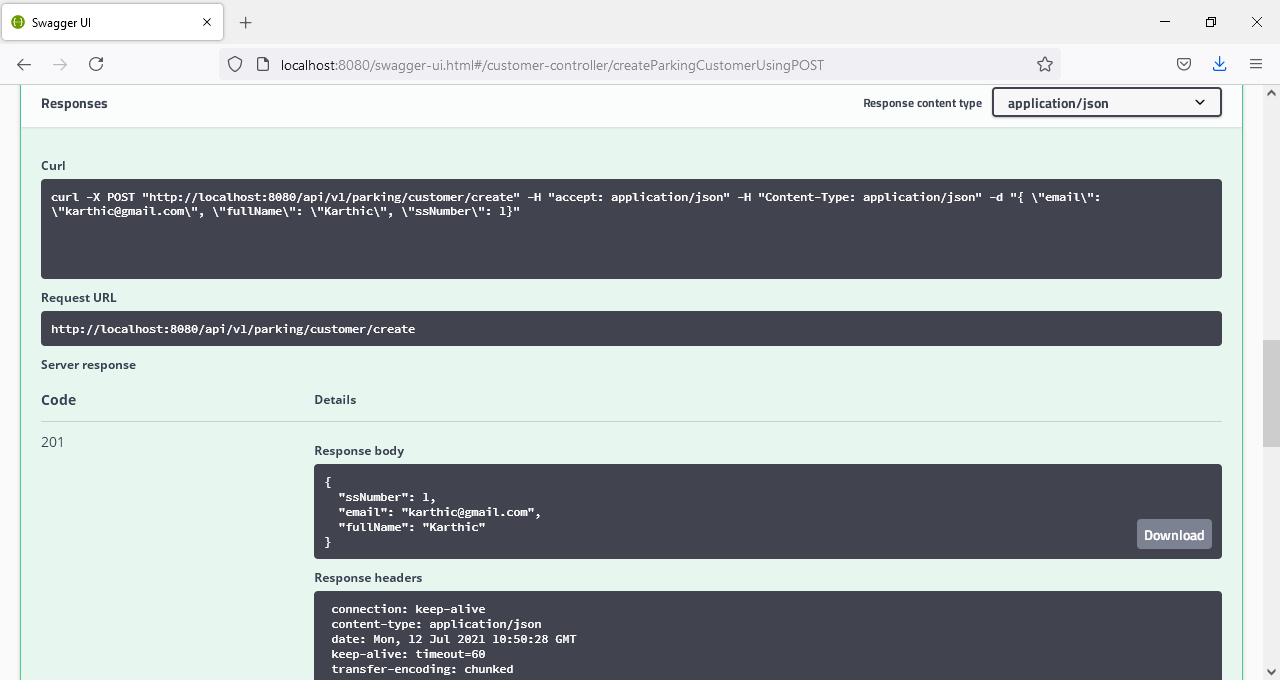
**1. Create User API**

Create user API can be used with below data in swagger and press execute button and the user gets resisted with application see in below request & response.

Request:



Response:

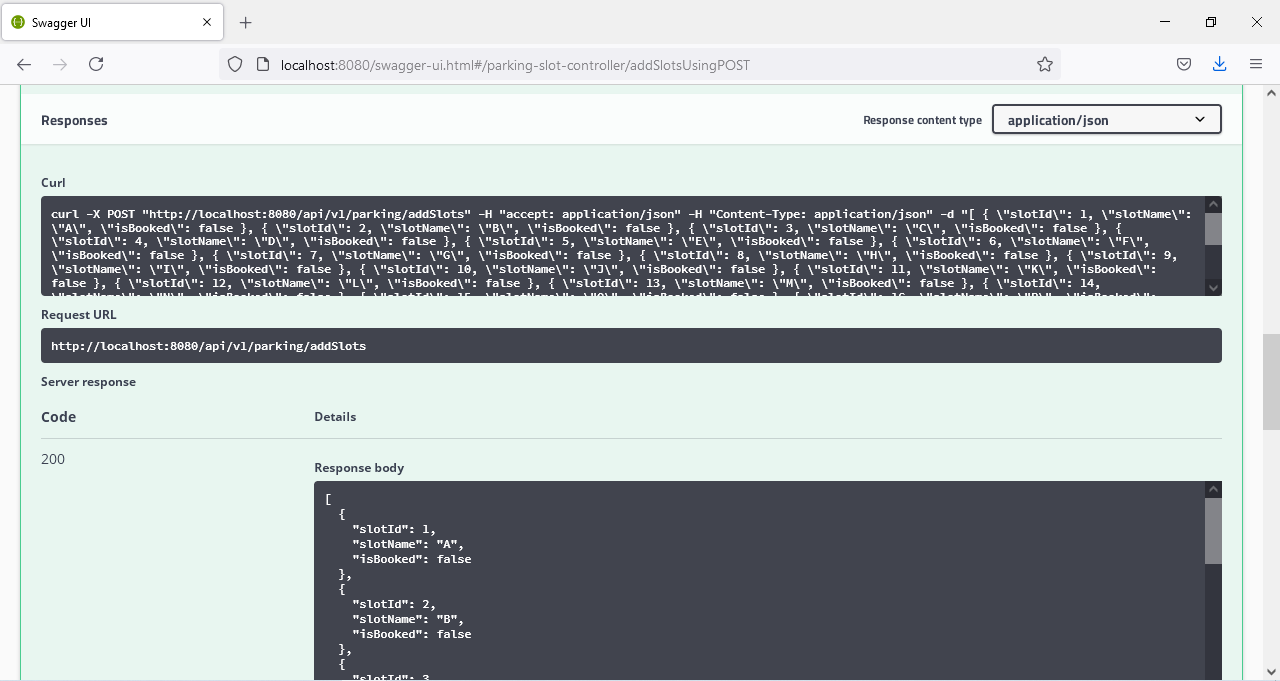


**2. addSlots API**

Add Slots API can be used with below data in swagger and press execute button and the parking slots gets added to application see in below request & response.

Request:



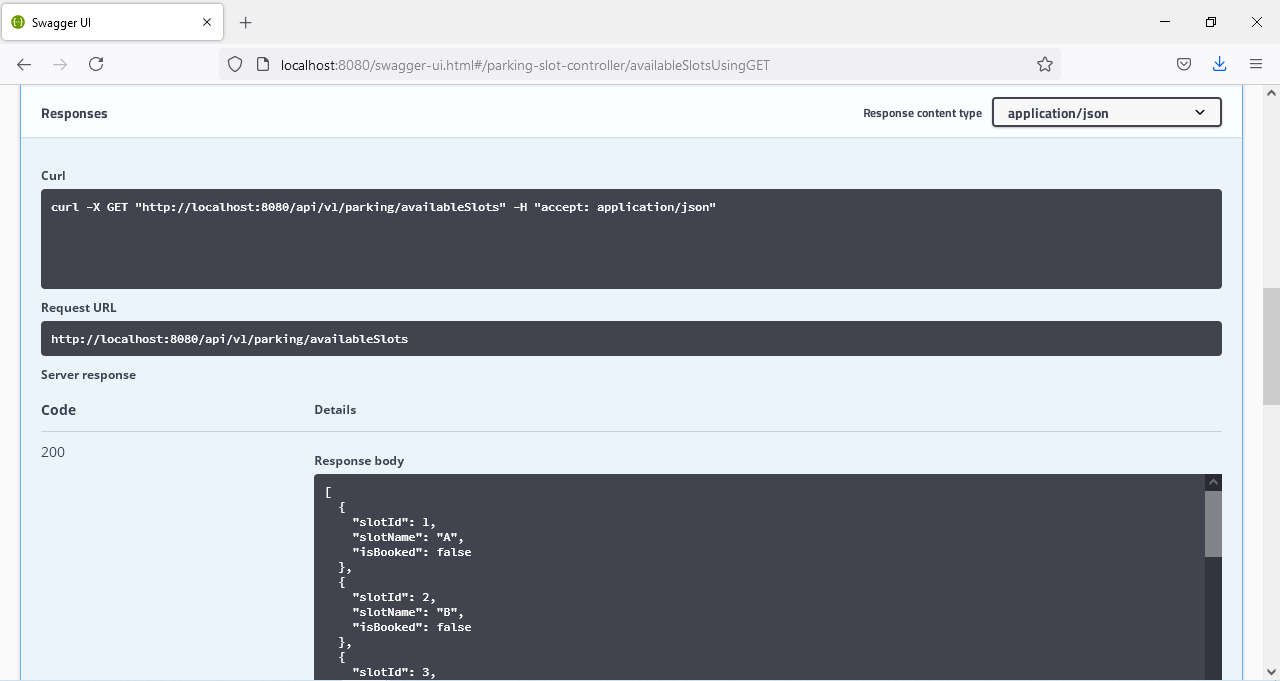
Response:

**3. availableSlots API**

Add Slots API can be used with below data in swagger and press execute button and the parking slots which are added earlier to application see in below request & response.

Request:

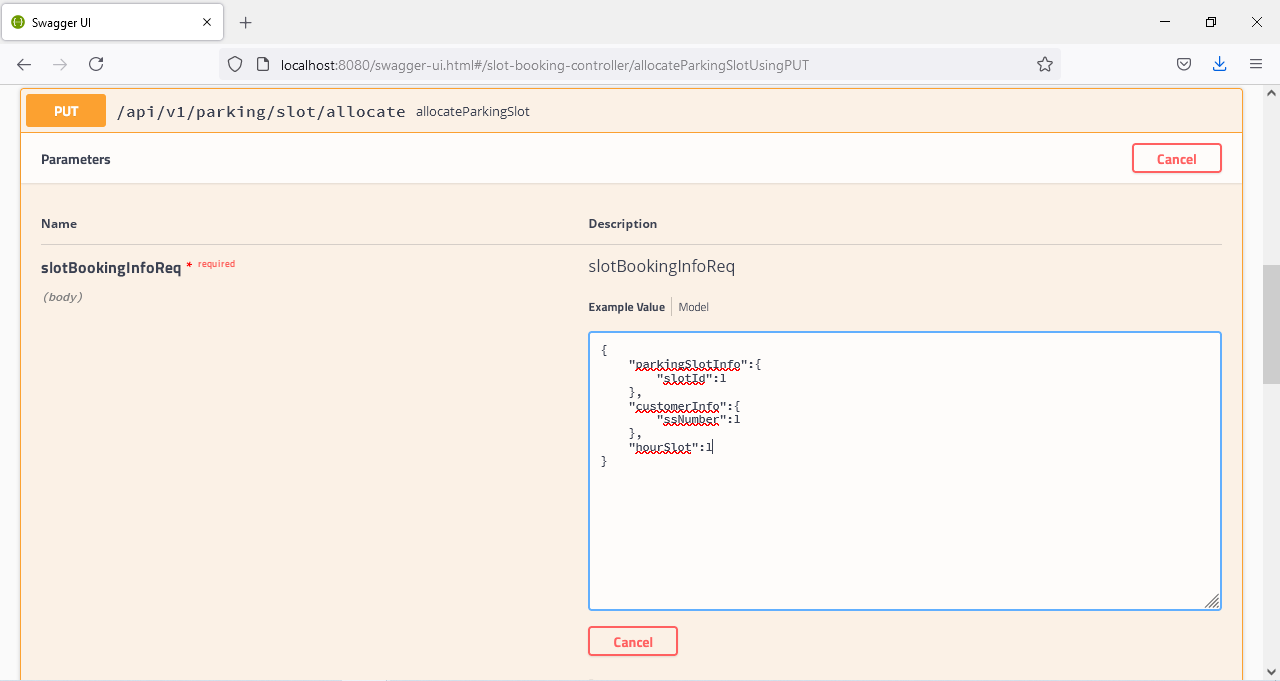


Response:

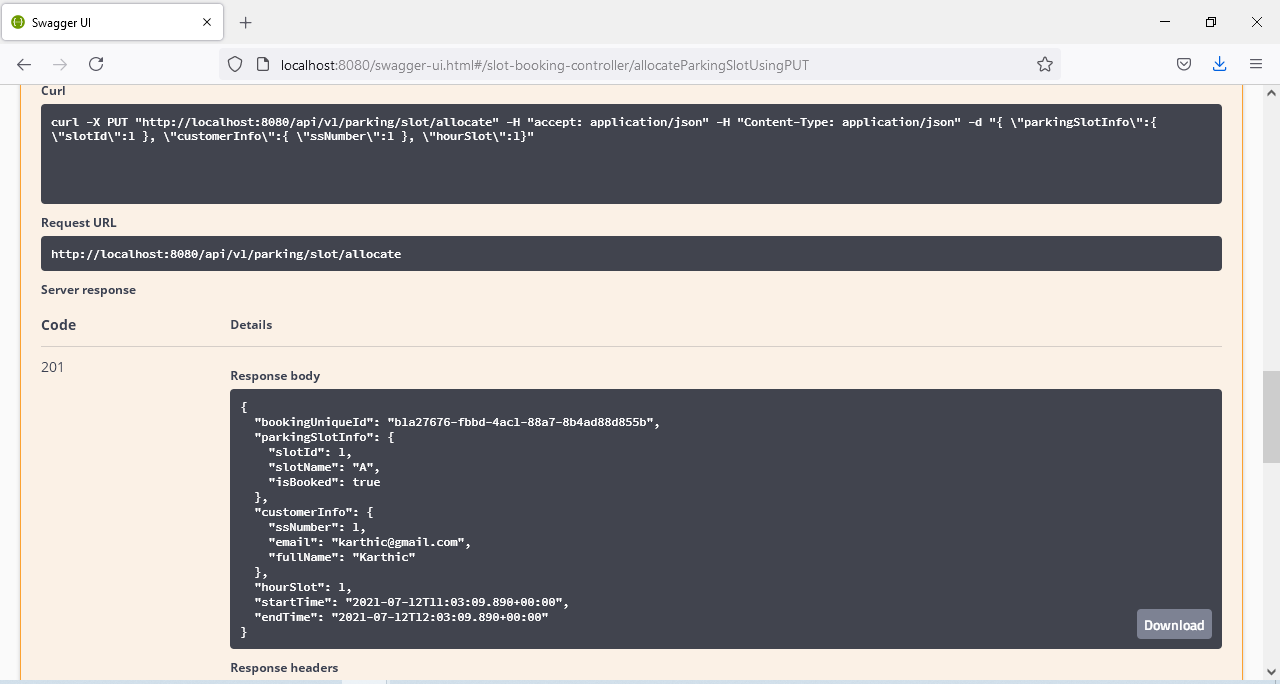
**4. allocate API**

Allocate API can be used with below data in swagger and press execute button and the specified parking slot for user with the time duration gets booked to application, see in below request & response.

Request:



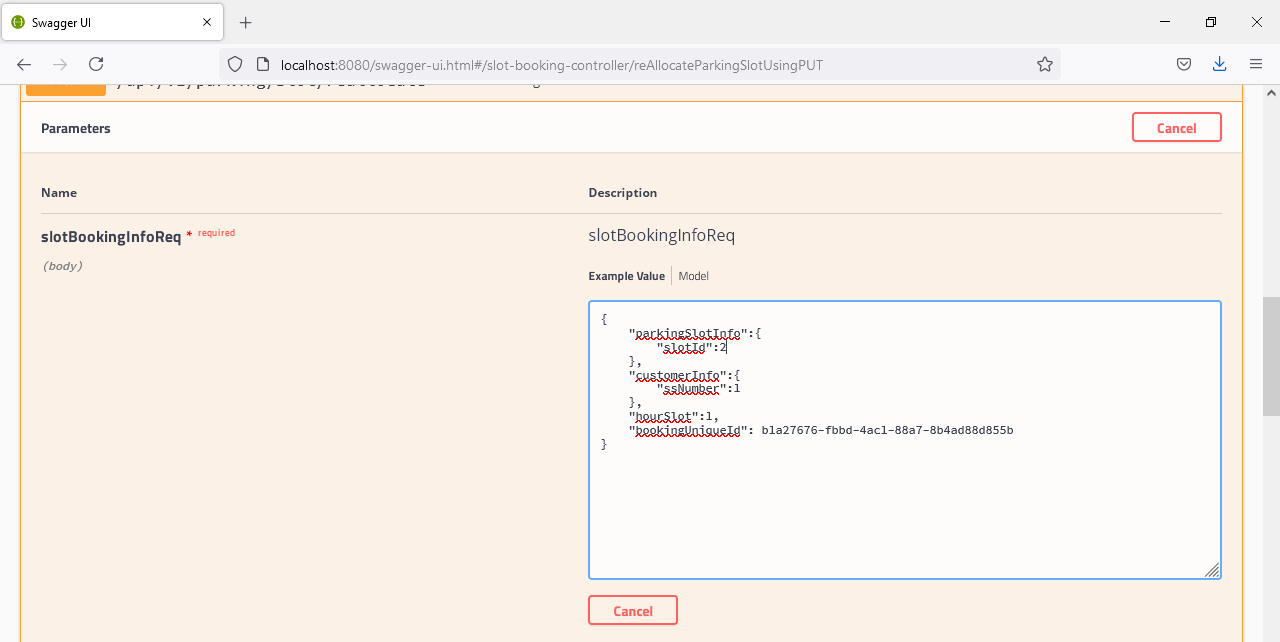
Response:



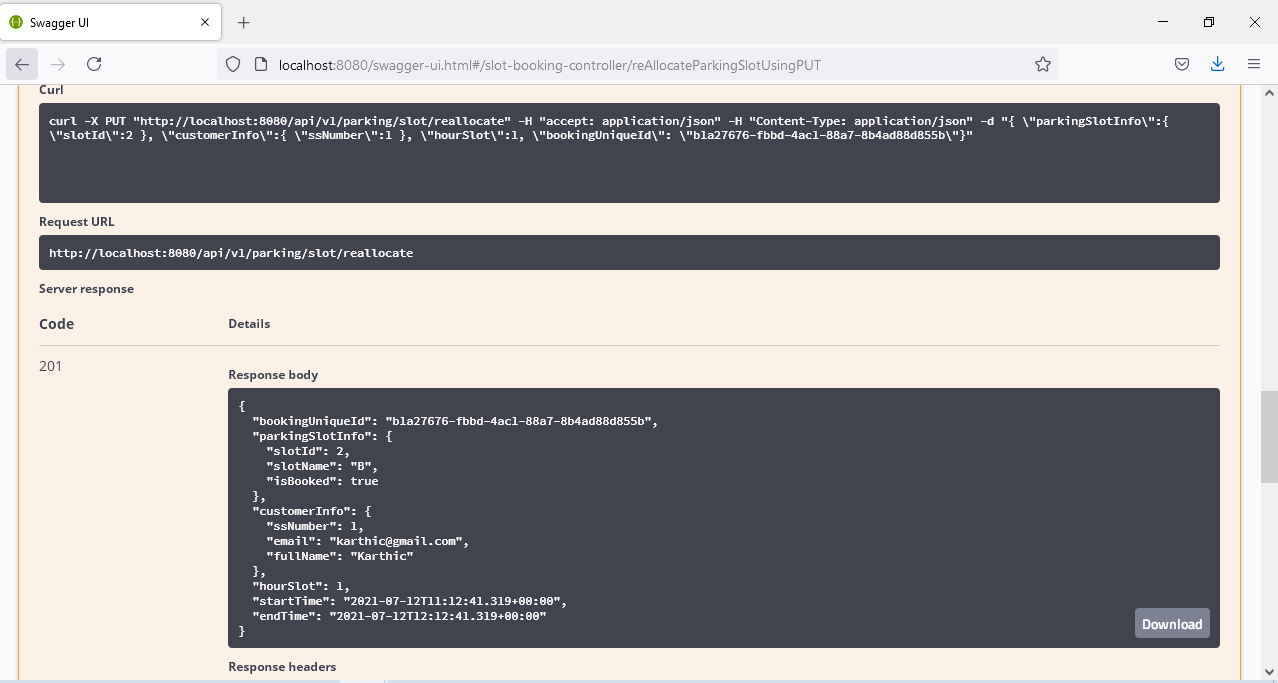
**5. reallocate API**

Reallocate API can be used with below data in swagger and press execute button and the specified parking slot for registered user with the time duration gets reallocated to application, see in below request & response.

Request:



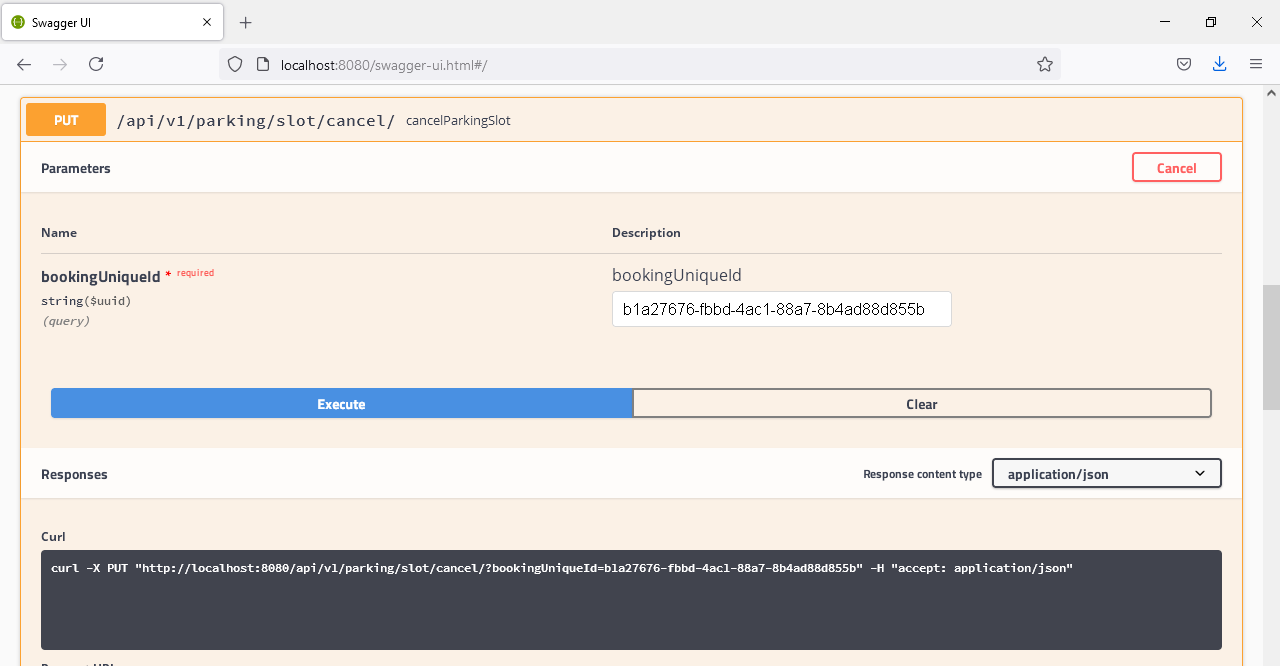
Response:

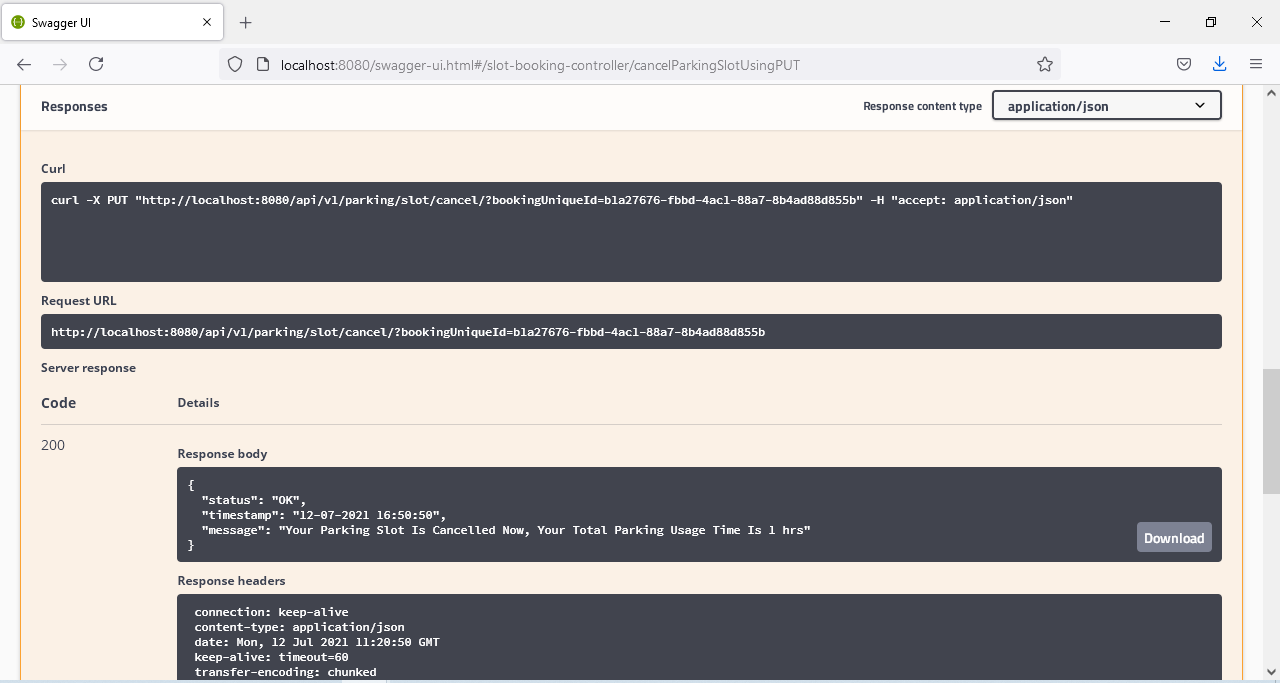


**6. cancel API**

Cancel API can be used with below data in swagger and press execute button and the specified bookingUniqueId gets disassociated from booking and cancel booking from application, see in below request & response.

Request:



Response:

Also the postman collection is attached below, if needs to be accessed via Postman tool,

