# API Documentation

For Installing all the required Libraries to run this API , follow the steps mentioned in the Readme file in GIT repository.

Once server is Running , testing can be done .

Assumptions Made for this API Design:

Frontend will be sending the HTTP POST request . Body will contain all the parameters required . So Parameter checking is omitted from this development.

API’s available:

1. Add Spaceship: POST Request

<http://127.0.0.1:8000/spaceship/AddSpaceship>

Body:

{"id":91081,"name":"First mars","model":"spaceX","city":"Sydney","planet":"Mars","status":"operational"}

Status is Optional and all other parameters are Mandatory

1. Add Location: POST Request

<http://127.0.0.1:8000/spaceship/Addlocation>

Body:

{"id":1,"city":"Sydney","planet":"Earth"}

1. Remove Spaceship: POST Request

<http://127.0.0.1:8000/spaceship/DelSpaceship>

Body:

{"id":91081}

1. Remove Location: POST Request

<http://127.0.0.1:8000/spaceship/DelLocation>

Body:

{"id":2}

1. Update Status: POST Request

<http://127.0.0.1:8000/spaceship/UpdateStatus>

Body:

{"id":91081,"status":"commissioned"}

1. Travel: POST Request

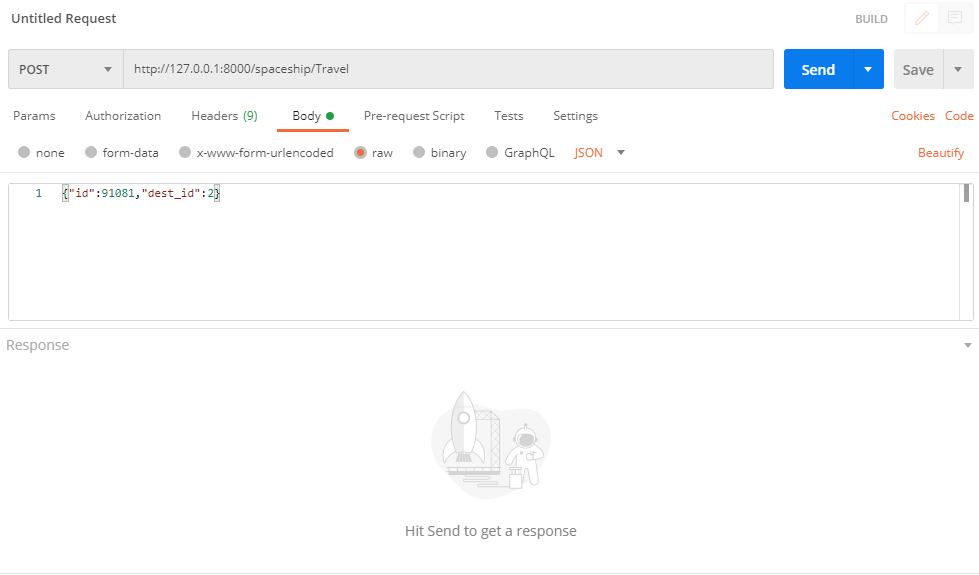
<http://127.0.0.1:8000/spaceship/Travel>

Body:

{"id":91081,"dest\_id":1}

POSTMAN Application need to be downloaded to test this API .From <https://www.postman.com/product/api-client/>

Sample Request in Postman:



Use Case Scenario’s:

1. Add Spaceship:

In this usecase we need to provide Spaceship ID, Name, Model, City and Planet . ID should be unique . ID can only be related to single City and Planet . But City and Planet can have Maximum of 10 Spaceships. If City and Planet location is not available in Location database , then Internally we are creating a location record and proceeding with Add Spaceship usecase.

1. Add Location:

In this usecase we need to provide Location ID,City and Planet . ID should be unique . ID can only be related to single City and Planet . Capacity is default 10 . If Add spaceship is executed then capacity is reduced by 1

1. Del Location :

In this usecase if we delete the Location , then all the spaceship’s related to this Location will be Deleted.

1. Del Spaceship:

In this Usecase if we delete the Spaceship, then we need to increase the capacity of Location by ‘1’.

1. Update Status:

We can Update the Status of Spaceship by providing Spaceship ID and Status .

1. Travel:

In this scenario we need Spaceship Id and Destination Location ID . We need to see first whether we can accommodate this spaceship in the Destination Location. Once it travelled we need to Increase the Capacity of Origin location by 1 and decrease the Capacity of Destination location by 1.

Customized Errors:

|  |  |
| --- | --- |
| Error Code | Error Message |
| 200 | Success |
| 303 | Spaceship Location is Full |
| 306 | Spaceship or Destination Port Doesn’t exists |
| 309 | Spaceship status is not Operational |
| 308 | Destination Location Capacity is Insufficient |
| 305 | Location Doesn’t exist |

Database Design:

Created 2 Models in Django . Spaceship and Location . Spaceship will have Location as Foreign Key.

Spaceship contains: ID, Name, Model, Location, Status (IF Status is not sent it will be Default None)

Location Contains: ID, City, Location and Capacity (Capacity default is 10).