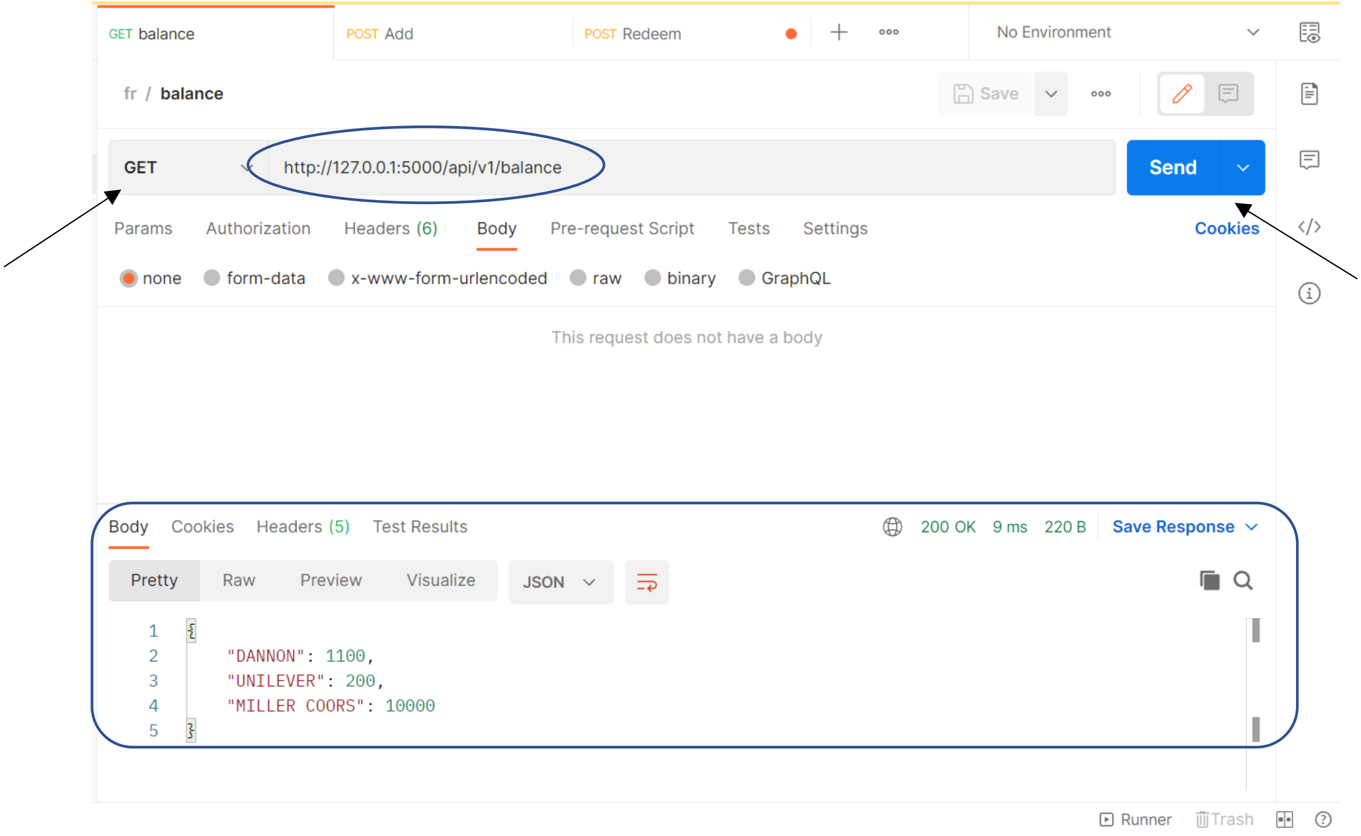
Instructions to test the web services

1. Before testing, make sure to follow How to run commands, to start the webservice.
2. By default, the web service will use port 5000, if user changes the default port or default port is not available, please run webservice on different port, and use the updated port while testing the application.
3. Open postman in your local machine and create a new request (no sign up required, use option skip to main app to avoid sign up)

Method 1: Using Postman API

How to check balance

- Make sure the server is up and running.
- GET: http://127.0.0.1:5000/api/v1/balance
- No data is expected, jus the correct URL and click send.
- The output will show the total balance points against each payer so far.



GET http://127.0.0.1:5000/api/v1/balance

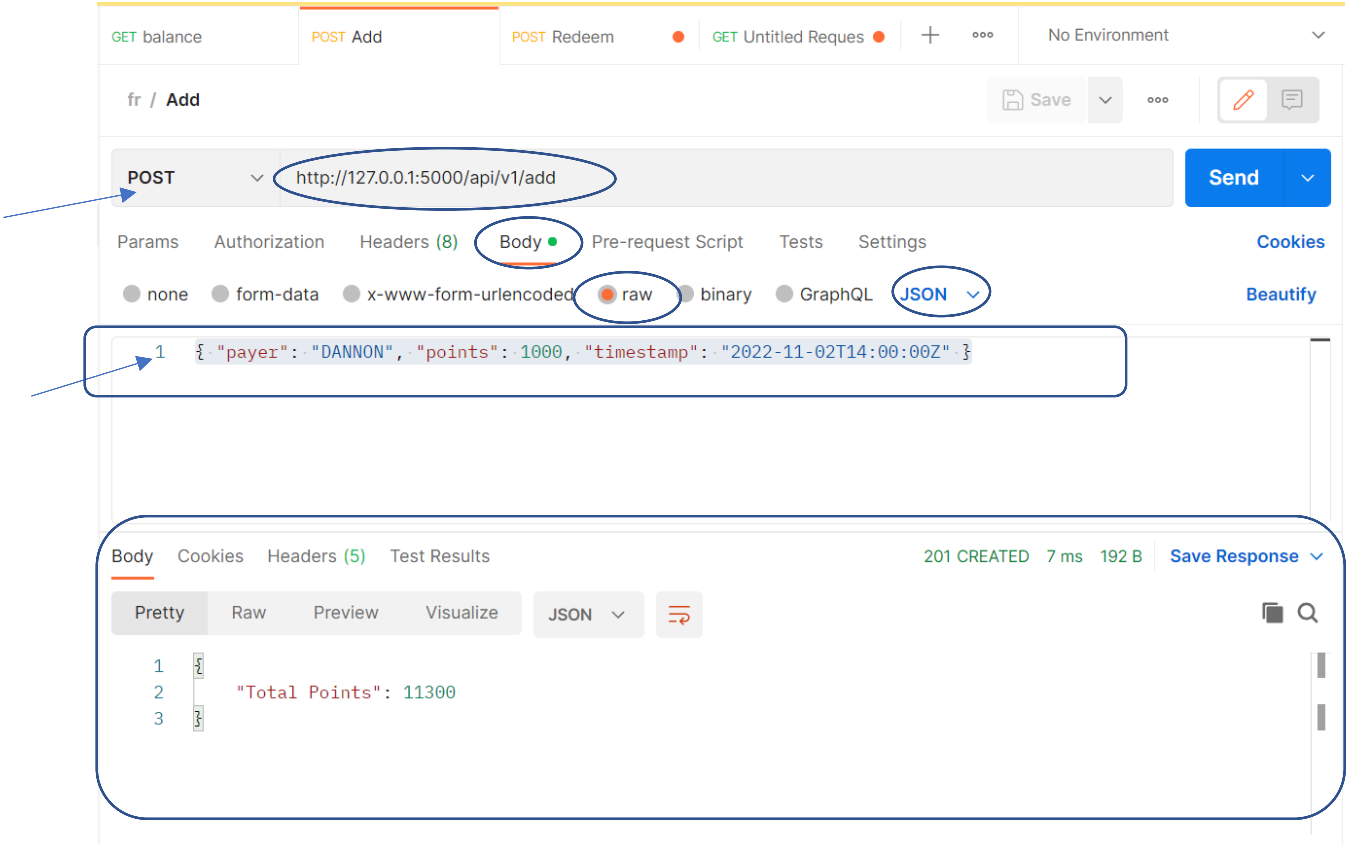
This request does not have a body

200 OK 9 ms 220 B

```
{
    "DANNON": 1100,
    "UNILEVER": 200,
    "MILLER COORS": 10000
}
```

How to Add points:

- POST: http://127.0.0.1:5000/api/v1/add
- This request expects data in json format. Goto ‘body’, choose ‘raw’. From the dropdown select ‘JSON’ in place of ‘Text’.
- Data to be entered: e.g. { "payer": "DANNON", "points": 1000, "timestamp": "2022-11-02T14:00:00Z" }. (one record at a time, as per the requirement)
- Click Send
- The output will show total points so far.



POST http://127.0.0.1:5000/api/v1/add

Send

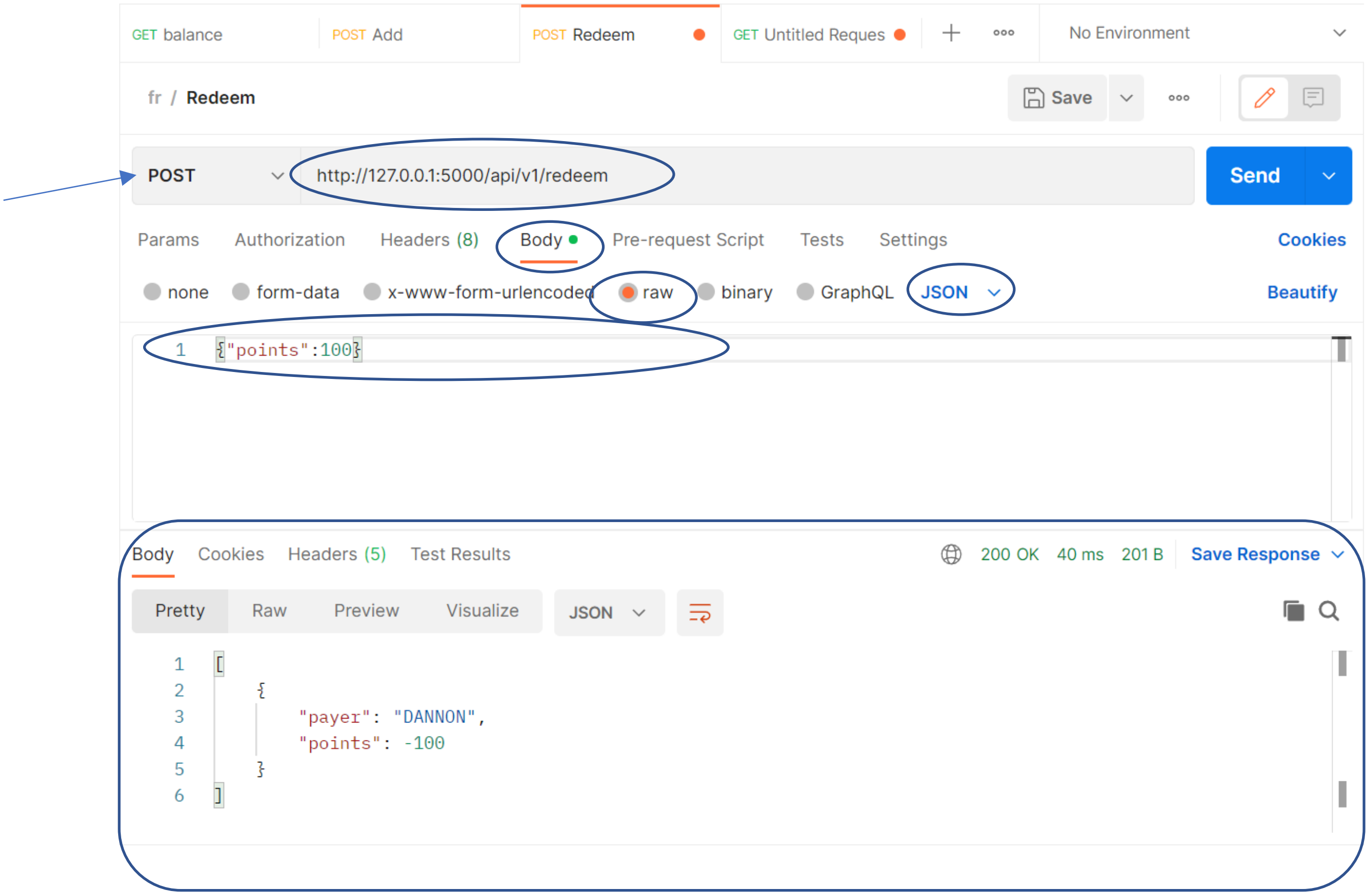
```
{ "payer": "DANNON", "points": 1000, "timestamp": "2022-11-02T14:00:00Z" }
```

201 CREATED 7 ms 192 B

```
{
    "Total Points": 11300
}
```

How to Redeem points:

- POST: http://127.0.0.1:5000/api/v1/redeem
- This request expects data in json format. Goto ‘body’, choose ‘raw’. From the dropdown select ‘JSON’ in place of ‘Text’.
- Data to be entered: e.g. {"points":5000}
- Click Send
- The output will show deducted balance from each payer.



POST http://127.0.0.1:5000/api/v1/redeem

```
{"points":100}
```

200 OK 40 ms 201 B

```
[
    {
        "payer": "DANNON",
        "points": -100
    }
]
```

Method 2: Using test.py script

Using the Postman API there is only 1 problem, if we want to add data in bulk like 100 or 1000 or 10000 records, we have to every time enter the record and hit send. There are two ways to deal with it: (1) in the postman we can write pre-request script which can make the bulk calls. (2) in the test.py script, inside add_data method we can replace the variable data with appropriate list of dictionary, and it will make all the calls.

Inside main function, uncomment the appropriate method (like add_data, balance, redeem) and inside those methods update data as per need and run. The file is already configured to make http requests to server. It will print the server response in console.