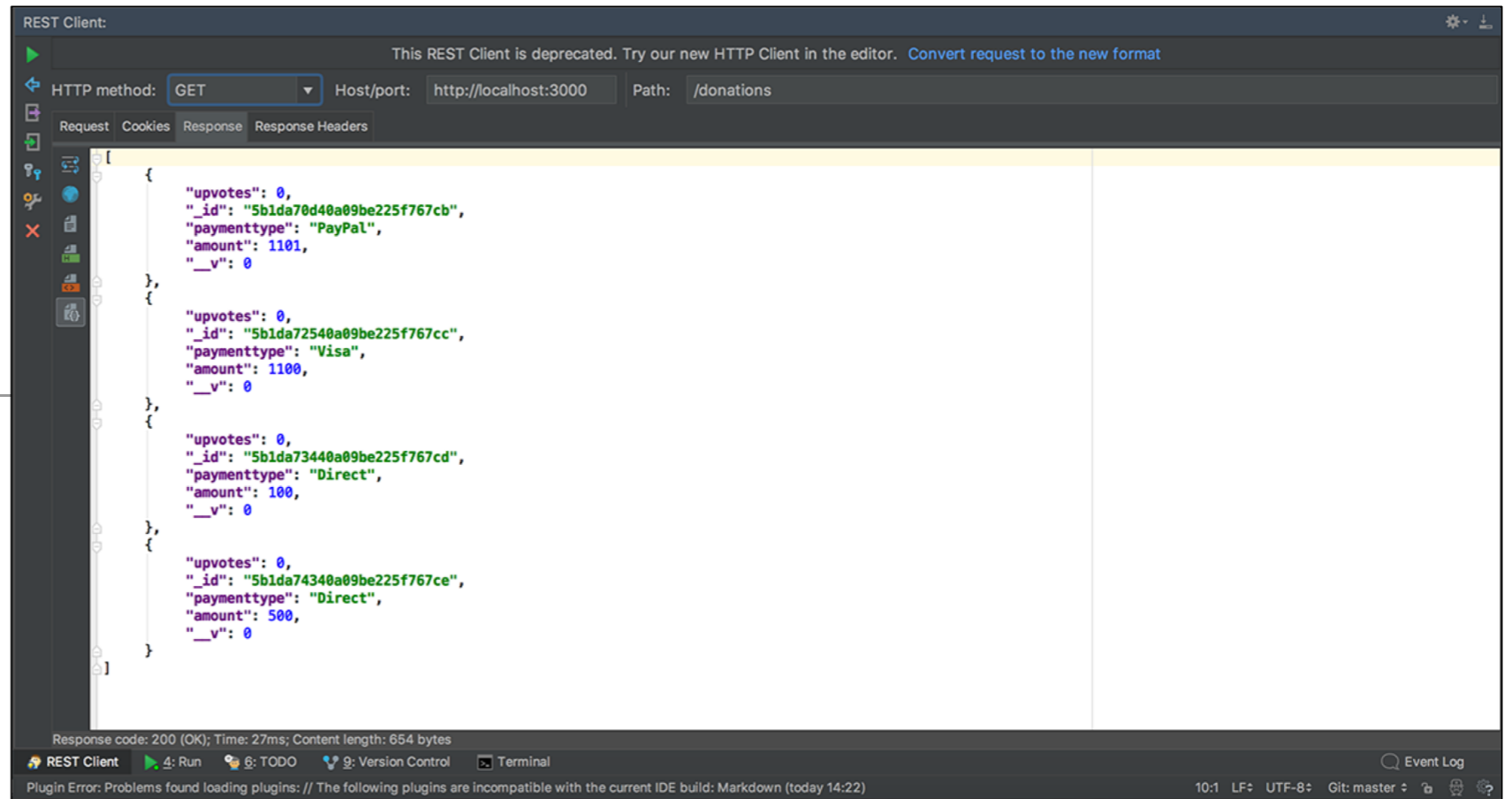


# Assignment 1

30% of Overall Grade



The screenshot shows a REST Client interface with the following details:

- HTTP method:** GET
- Host/port:** http://localhost:3000
- Path:** /donations
- Response:** A JSON array of four donation objects.

```
[{"upvotes": 0, "_id": "5b1da70d40a09be225f767cb", "paymenttype": "PayPal", "amount": 1101, "__v": 0}, {"upvotes": 0, "_id": "5b1da72540a09be225f767cc", "paymenttype": "Visa", "amount": 1100, "__v": 0}, {"upvotes": 0, "_id": "5b1da73440a09be225f767cd", "paymenttype": "Direct", "amount": 100, "__v": 0}, {"upvotes": 0, "_id": "5b1da74340a09be225f767ce", "paymenttype": "Direct", "amount": 500, "__v": 0}]
```

Response code: 200 (OK); Time: 27ms; Content length: 654 bytes

REST Client | Run | TODO | Version Control | Terminal | Event Log

Plugin Error: Problems found loading plugins: // The following plugins are incompatible with the current IDE build: Markdown (today 14:22)

10:1 | LF | UTF-8 | Git: master

# Options

---

Work on your own app, exhibiting similar level of complexity/feature density as covered in the 1<sup>st</sup> part of the Semester Case Study - Donation.

# Case Study - Donation

---

- A Node Web Server to manage donations made to 'Homers Presidential Campaign'.
- App Features (all via RESTful API)
  - POST a payment type and donation amount in JSON format
  - GET a list of donation amounts and types
  - GET an individual donation using an ID
  - DELETE an individual donation using and ID
  - Upvote a donation via PUT request
- Persistence via MongoDB deployed to Heroku

# POST – Request & Response

The image displays two screenshots of a REST Client interface. The top screenshot shows the configuration for a POST request to `http://localhost:3000/donations`. The request headers are `Accept: application/json`, `Cache-Control: no-cache`, and `Content-Type: application/json`. A dialog box titled "Specify the text to send:" is open, showing the JSON payload: `{"id":0,"paymenttype":"Direct","amount":500,"upvotes":0}`. The bottom screenshot shows the response received: `{"message":"Donation Added!"}`. Both the request configuration and the response are highlighted with red boxes.

REST Client

46 `console.log('ID: ' + req.para`

▶ HTTP method: POST Host/port: `http://localhost:3000` Path: `/donations`

Request Cookies Response Response Headers

Headers Request Param

Accept: application/json  
Cache-Control: no-cache  
Content-Type: application/json

Specify the text to send:

1 `{"id":0,"paymenttype":"Direct","amount":500,"upvotes":0}`

REST Client

46 `console.log('ID: ' + req.para`

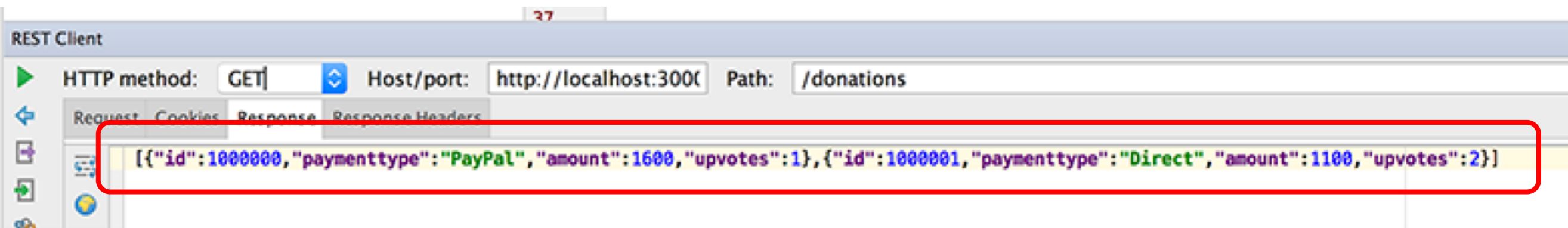
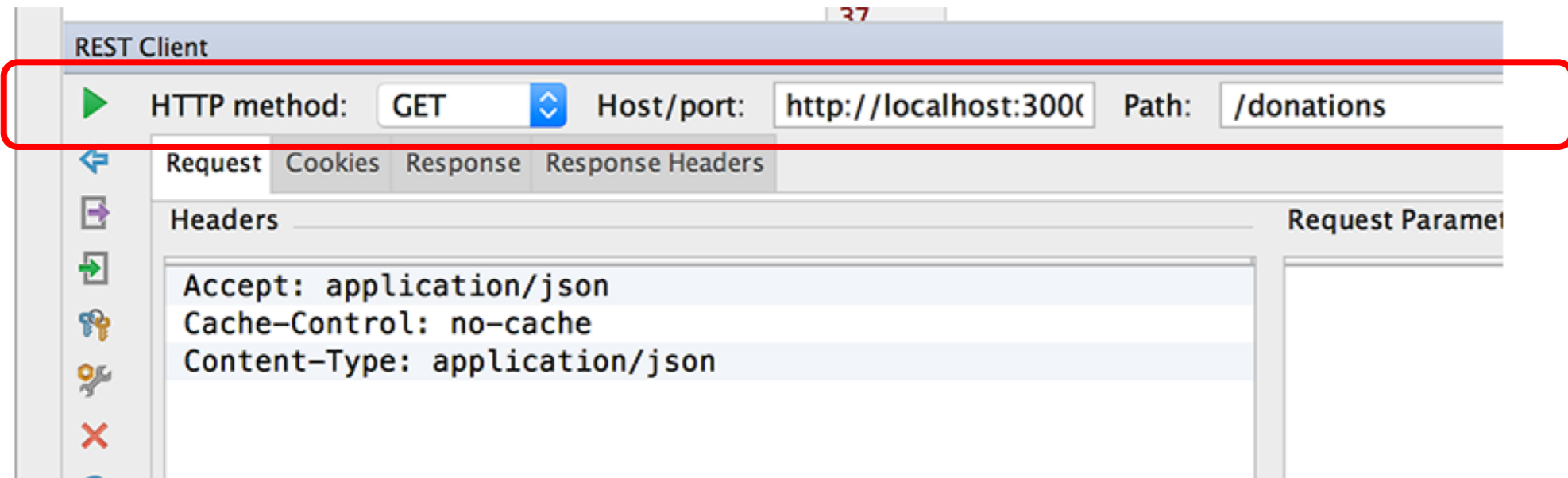
▶ HTTP method: POST Host/port: `http://localhost:3000` Path: `/donations`

Request Cookies Response Response Headers

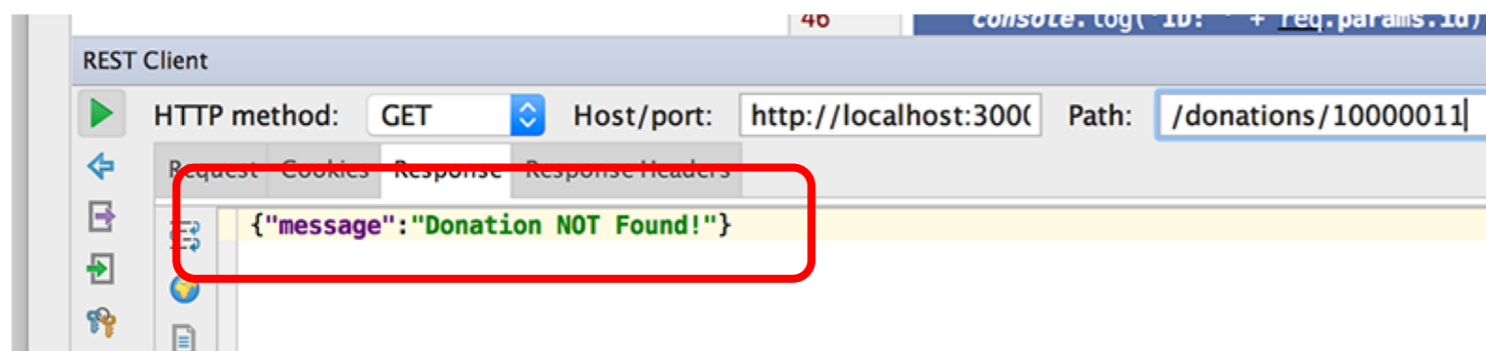
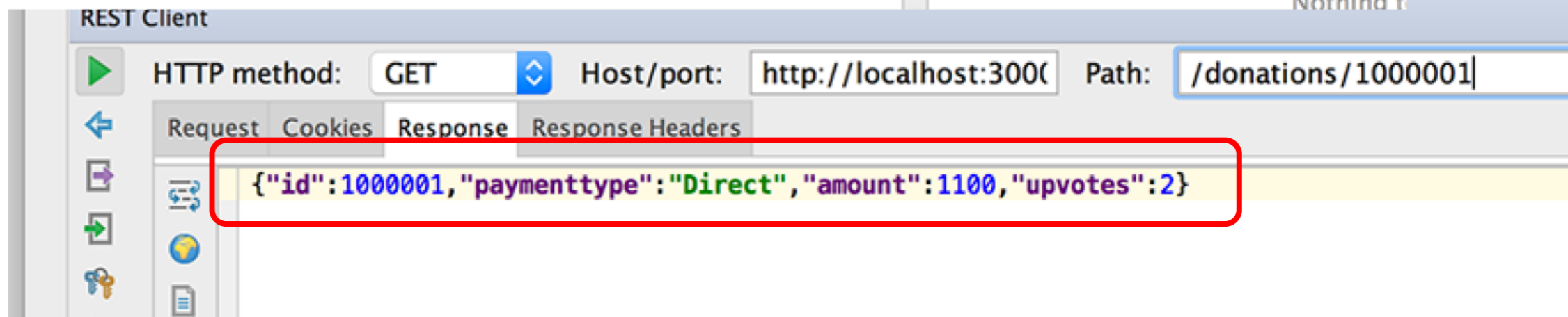
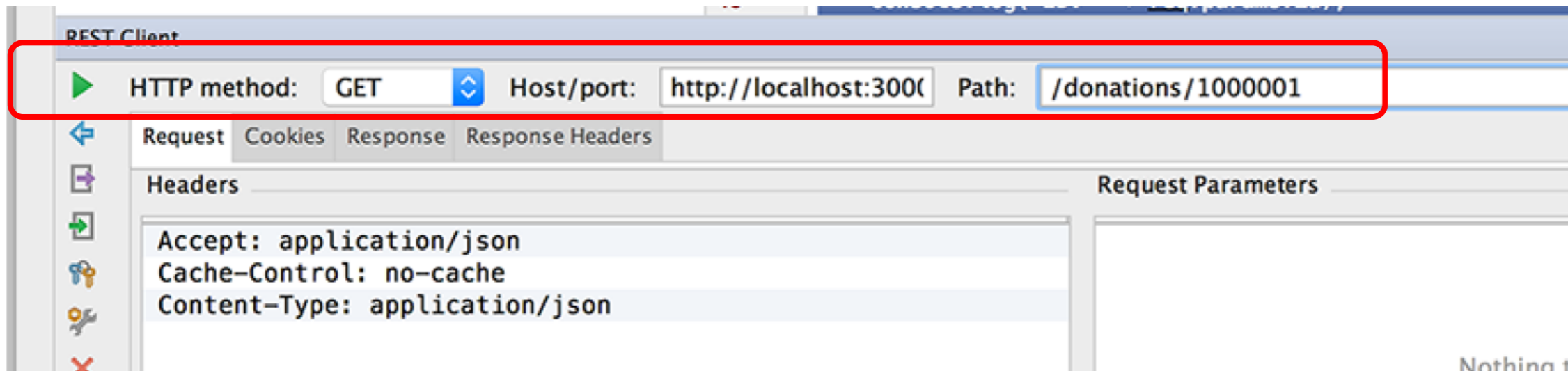
`{"message":"Donation Added!"}`

Cancel OK

# GET (1) – Request & Response



# GET (2) – Request & Response



# DELETE – Request & Response

REST Client

HTTP method: DELETE Host/port: http://localhost:3000 Path: /donations/100001

Request Cookies Response Response Headers

Headers

```
Accept: application/json
Cache-Control: no-cache
Content-Type: application/json
```

Request Parameters

action(err, req, res, next)

REST Client

HTTP method: DELETE Host/port: http://localhost:3000 Path: /donations/100001

Request Cookies Response Response Headers

```
{"message": "Donation Deleted!"}
```

REST Client

HTTP method: GET Host/port: http://localhost:3000 Path: /donations

Request Cookies Response Response Headers

```
[
  {
    "id": 1000000,
    "paymenttype": "PayPal",
    "amount": 1600,
    "upvotes": 1
  },
  {
    "id": 1000001,
    "paymenttype": "Direct",
    "amount": 1100,
    "upvotes": 2
  },
  {
    "id": 941345,
    "paymenttype": "Direct",
    "amount": 500,
    "upvotes": 1
  }
]
```

Response code: 200 (OK); Time: 66ms; Content length: 191 bytes

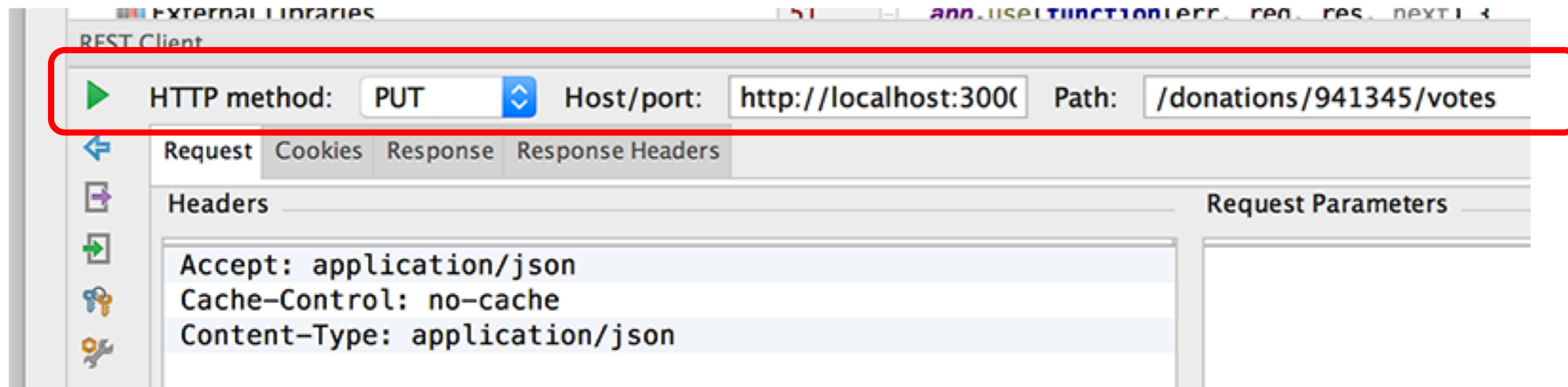
REST Client

HTTP method: GET Host/port: http://localhost:3000 Path: /donations

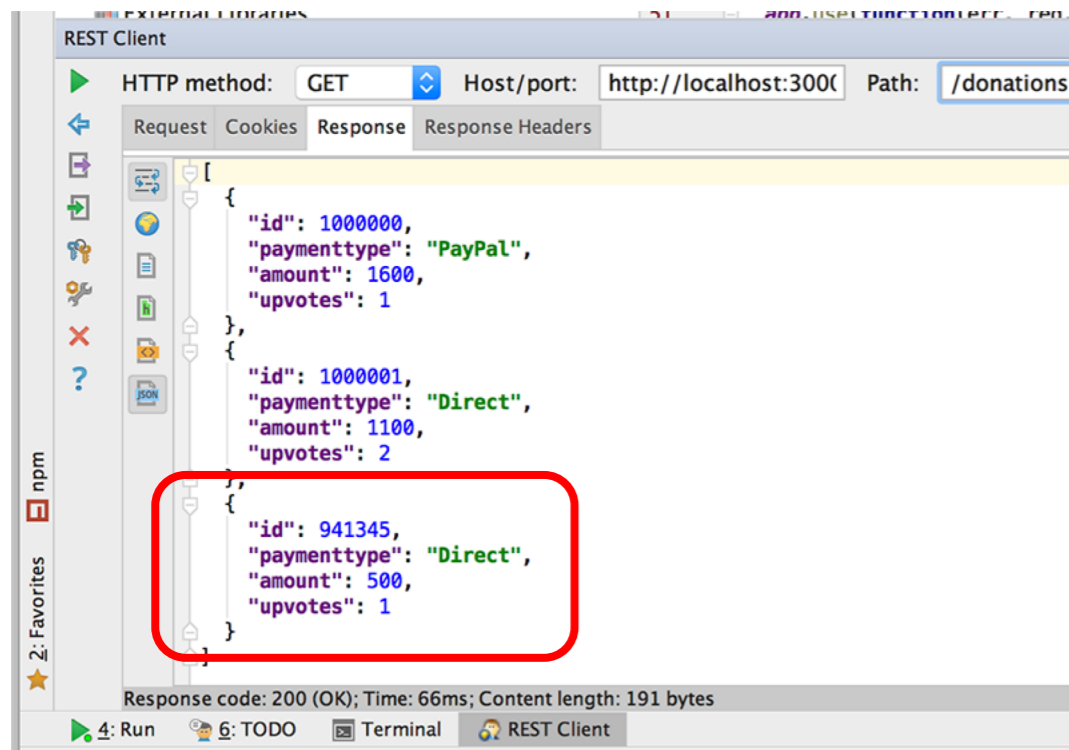
Request Cookies Response Response Headers

```
[
  {
    "id": 1000000,
    "paymenttype": "PayPal",
    "amount": 1600,
    "upvotes": 1
  },
  {
    "id": 941345,
    "paymenttype": "Direct",
    "amount": 500,
    "upvotes": 1
  }
]
```

# PUT – Request & Response



- Adds 1 to 'upvotes'





# Assignment Rubric for Assignment 1

| Standard                            | CRUD Node Server<br>[70%]   | Model<br>[10%]                                 | Persistence<br>[10%]                           | DX<br>(Developer eXperience)<br>[10%]         |
|-------------------------------------|---|--|--|---|
| Baseline                            | > 2 GET routes  | 1 Basic Model                                  | Basic JS Persistence                           | Data Validation                               |
| Good<br><b>Pass line</b>            | 2 GET routes<br>1 POST route<br>1 PUT route<br>1 DELETE route         | 1 Complex Model of different types             | MongoDB Persistence                            | Adherence to JS Best Practices eg SoC, Design |
| Very Good                           | > 3 GET routes<br>> 2 POST route<br>> 2 PUT route<br>> 2 DELETE route | 2 Complex Models with Schema                   | MongoDB Persistence with Schema                | Automated Testing (models)                    |
| Excellent/<br>Outstanding<br>(70%+) | Additional Features included, eg fuzzy searches, authentication etc.  | > 3 Models with Schema & related to each other | Advanced Features eg. deployed, authentication | Repo Usage, git etc.                          |

# README file

---

Include a brief README file (max 2 - 3 pages):

- Name and Student ID.
- Brief description of functionality.
- Persistence approach adopted i.e. what's persisted and where.
- Git approach adopted and link to git project / access.
- DX approach adopted.
- **References**

# Submitting Project Deliverables

---

Submit zip of project via Moodle dropbox. This zip should also include:

- the README file and
- full source code of your web project
- Youtube link to video (5 – 10 mins MAX) of Server Testing

Give read access to your lecturer to your GitHub / BitBucket repos. GitHub and BitBucket ids are:

- **ddrohan.**

# Questions?

---



mongoose



mongoDB