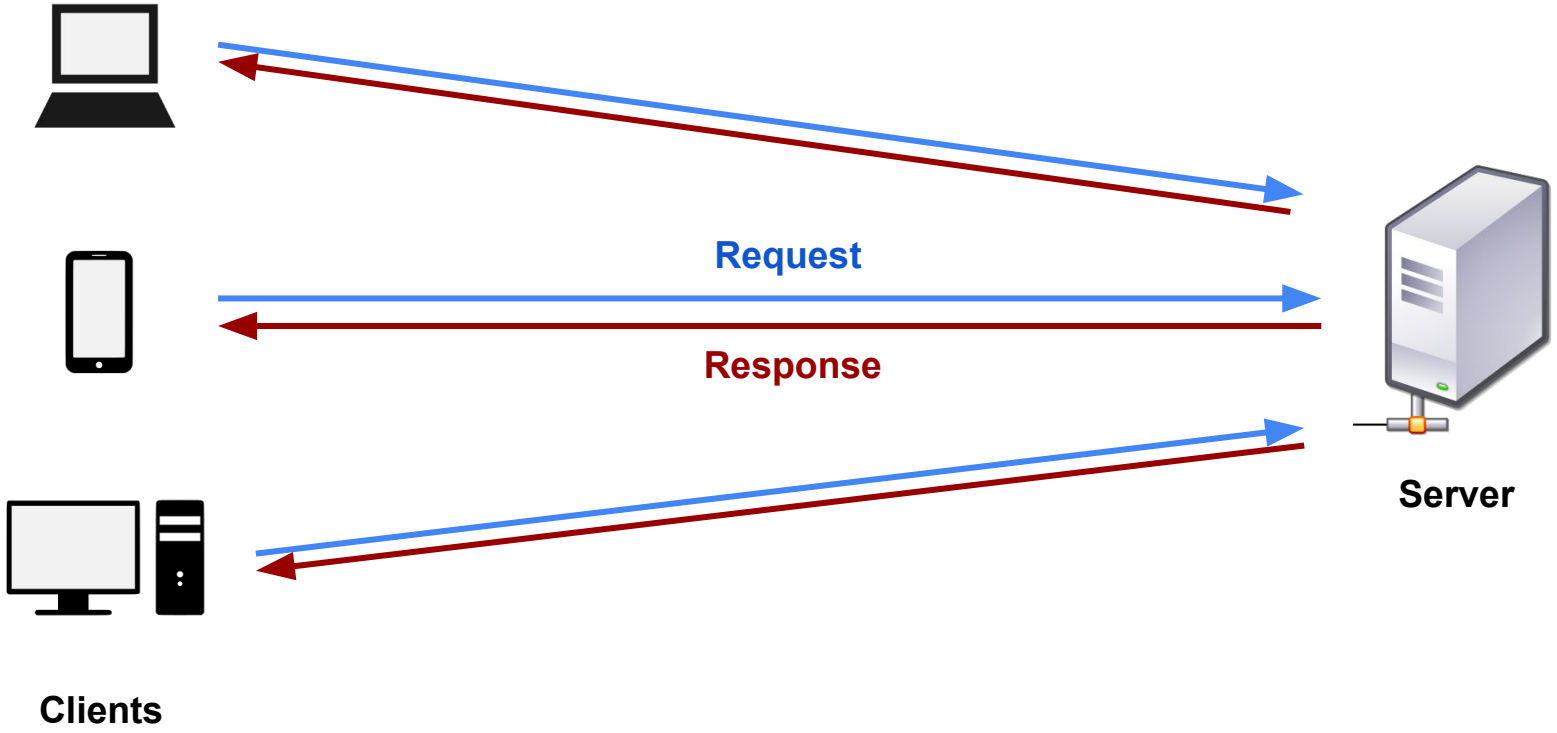
RESTful APIs

Recitation 4

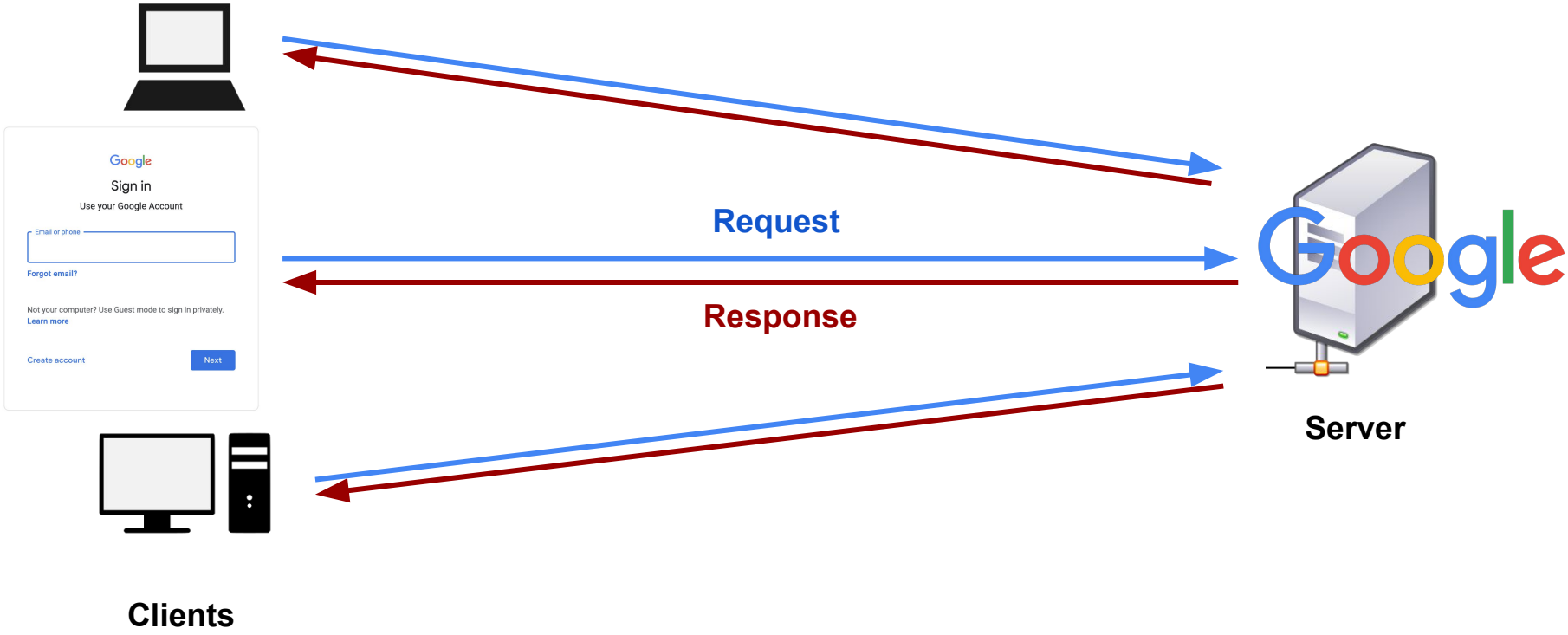
Plan for Today

1. Client-Server Model in Web
2. HTTP
3. RESTful API
4. Exercise

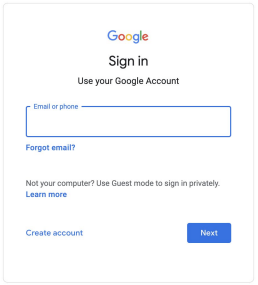
Client-Server Model in Web

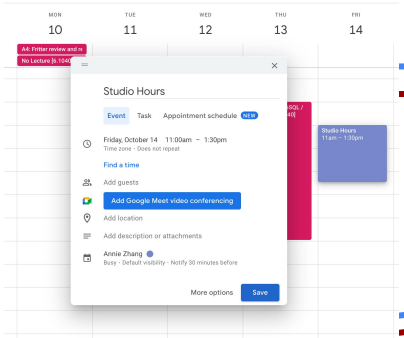


Client-Server Model in Web



HTTP (Hypertext Transfer Protocol)

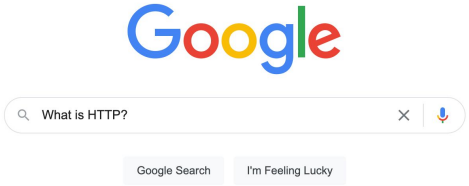




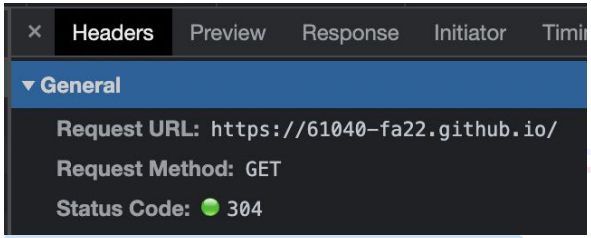
HTTP Request

HTTP Response

Server

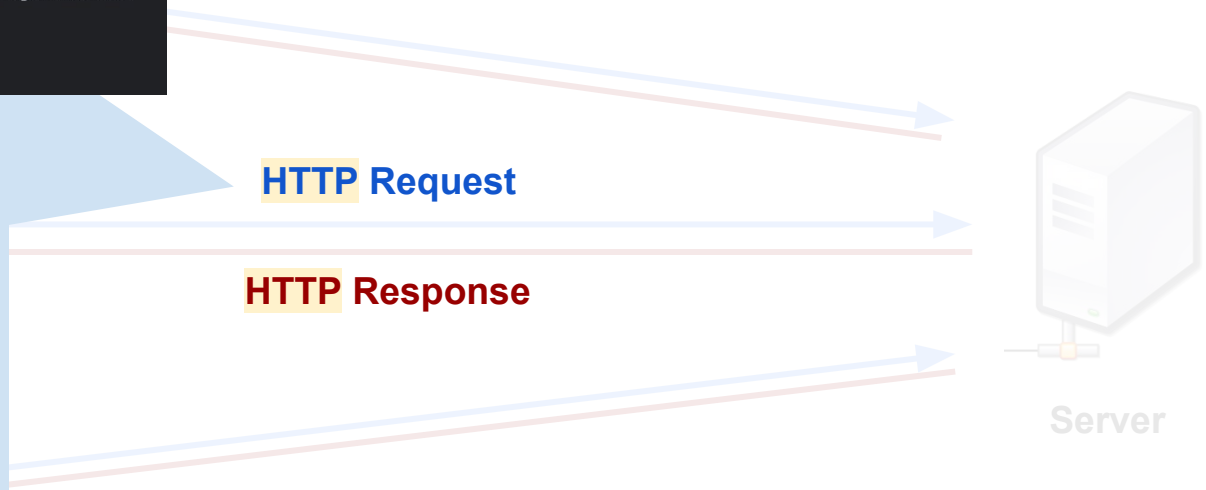


HTTP (Hypertext Transfer Protocol)



HTTP request

1. **URL**
2. **Method**
   - GET
   - POST
   - PUT / PATCH
   - DELETE



HTTP (Hypertext Transfer Protocol)

HTTP request

1. **URL**
2. **Method**
   - GET
   - POST
   - PUT / PATCH
   - DELETE

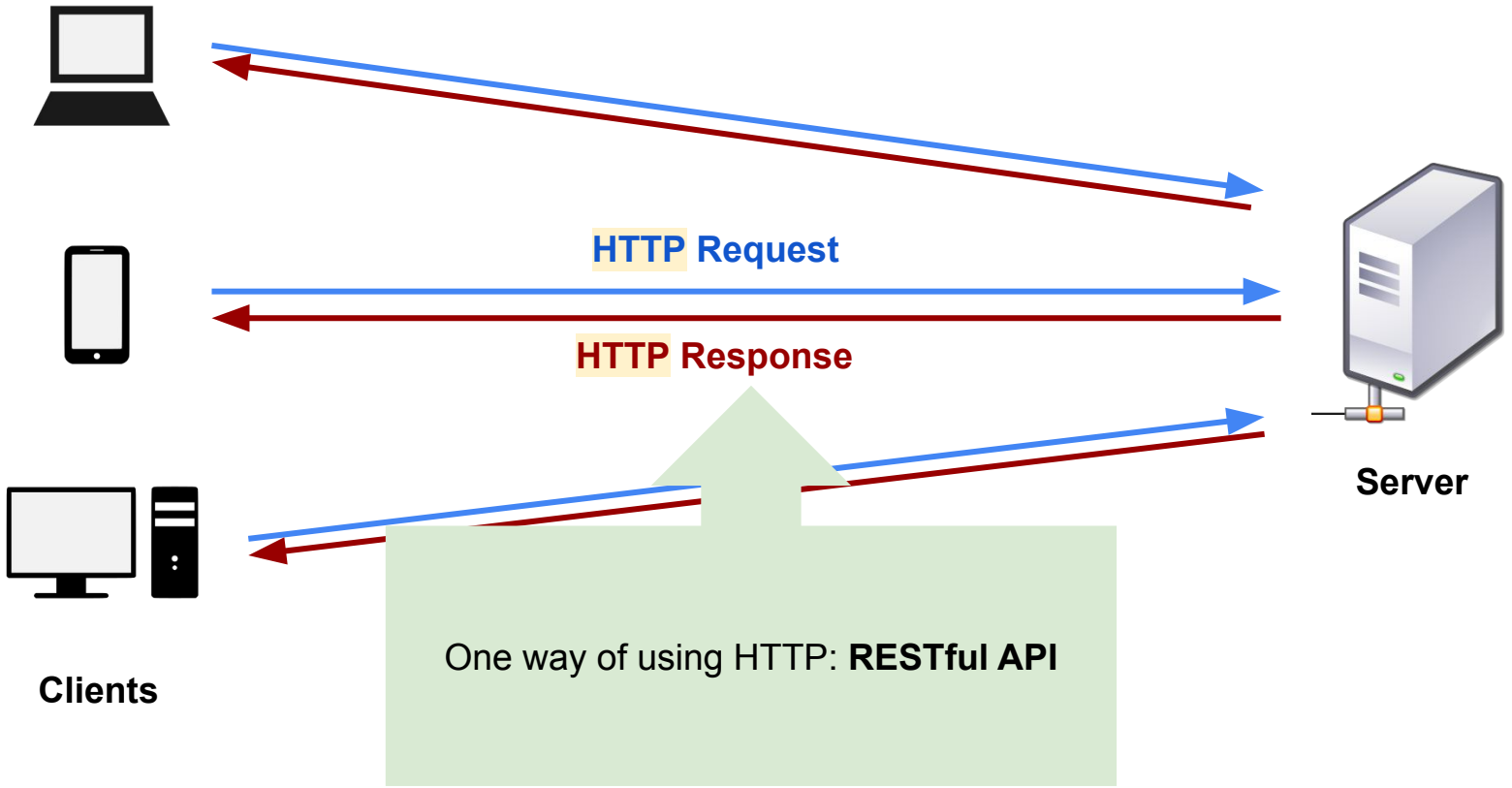
HTTP Request

HTTP Response

HTTP response status codes

e.g., 404 error

https://httpstatusdogs.com


HTTP Request
HTTP Response
Server
Clients
One way of using HTTP: **RESTful API**

RESTful API

RESTful API is an API that conforms to the constraints of REST architectural style

a set of definitions and protocols for building and integrating application software

an architectural standard dictating the structure of an HTTP request for more consistent, reliable client-server communication

“Applying verbs to nouns”

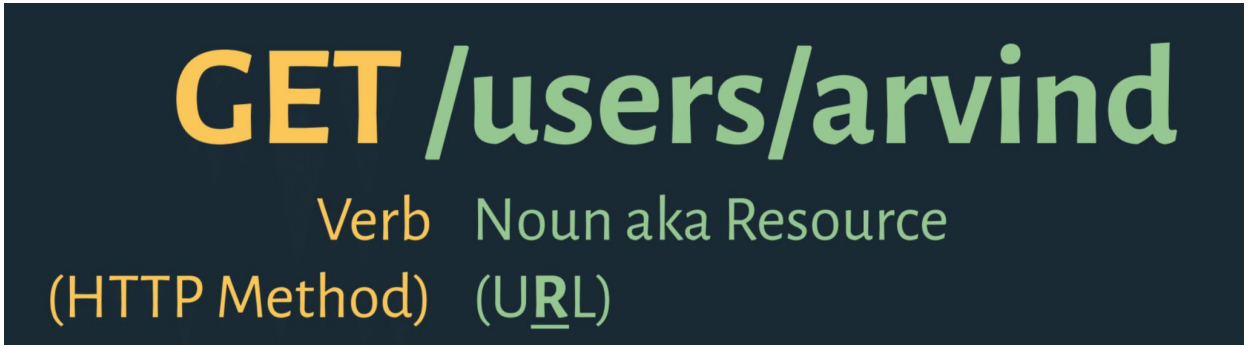
RESTful API

Why use RESTful API?
→ Early web APIs were poorly designed.

/find_users?name=arvind&action=getInformation
/shopping_cart?action=update_qty&user=123
/postComment?entryID=853&text=...

- **Not easily discoverable**: what goes in the path, what goes in the query parameters?
- **Inconsistent**: APIs could be internally inconsistent. Different APIs might have different path/parameter conventions
- **Difficult to maintain/extend**

RESTful API

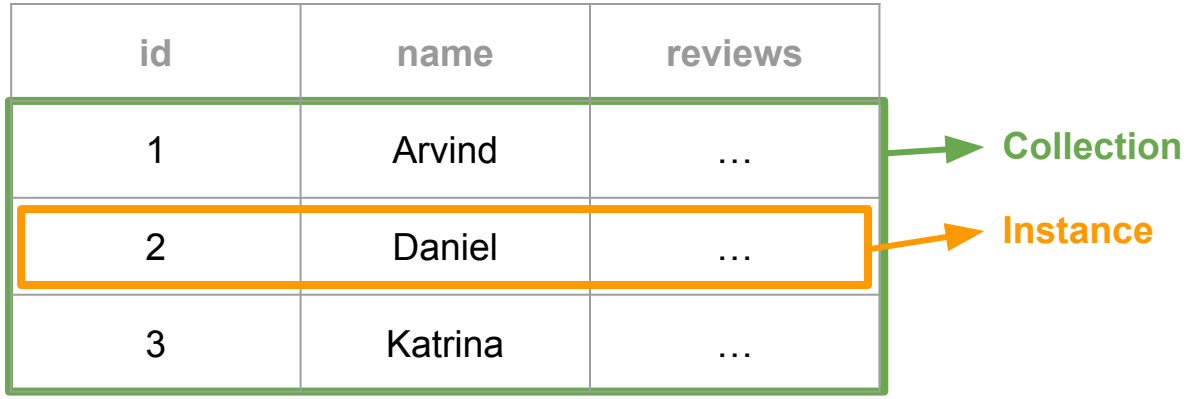


“Applying verbs to nouns”

Imagine you have this database…

A resource of "*professors*"

id	name	reviews
1	Arvind	…
2	Daniel	…
3	Katrina	…



Nouns are Resources

URLs identify a **representation** of a resource.
Use path hierarchies to imply **structure**.

Collections

- /profs
- /profs/reviews
- /profs/arvind/reviews

Instances

- /profs/arvind
- /profs/arvind/reviews/5

Verbs take action on resources

The four basic functions of persistent data is **CRUD.**
These functions map to HTTP methods.

Create	**POST**
Read	**GET**
Update	**PUT / PATCH**
Delete	**DELETE**

What's up with PUT vs PATCH?

- PUT: Overwrite the pre-existing item
- PATCH: Make a partial edit to the item

Verbs take action on resources

Why bother? Why not just use POST for everything?

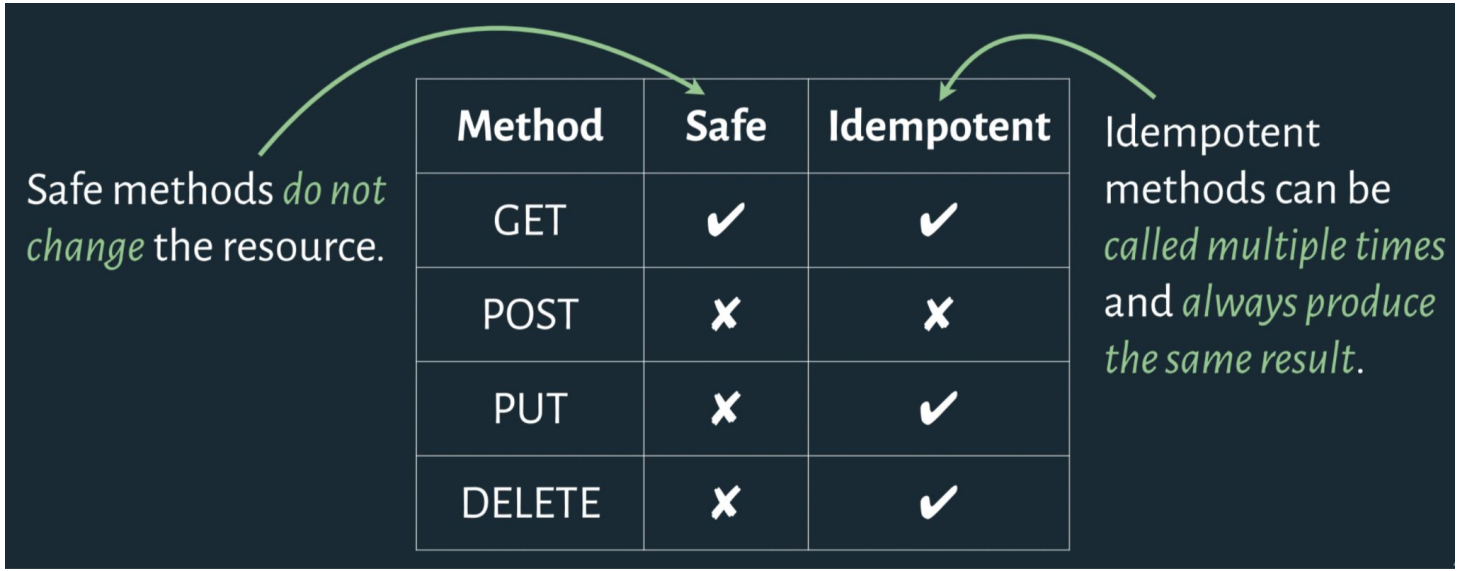
→ Methods carry different semantics and can be applied to the same noun

```
GET /profs/arvind/reviews – Get all reviews for Arvind
POST /profs/arvind/reviews – Create a new review for Arvind
PUT /profs/arvind/reviews/4 – Update review #4 for Arvind
DELETE /profs/arvind/reviews/5 – Delete review #5 for Arvind
```

Verbs take action on resources

Why bother? Why not just use POST for everything?

→ Method semantics make it easier to reason about data safety



Method	Safe	Idempotent
GET	✔	✔
POST	✘	✘
PUT	✘	✔
DELETE	✘	✔

Verbs take action on resources

What about non-CRUD actions?

→ Instead of calling an action, create (or delete) a resource

- Instead of **“login”,** **create a “session”**
- Instead of **“logout”,** **delete the “session”**

Conceptual Exercise: Rewrite the bad URLs!

We saw these earlier:

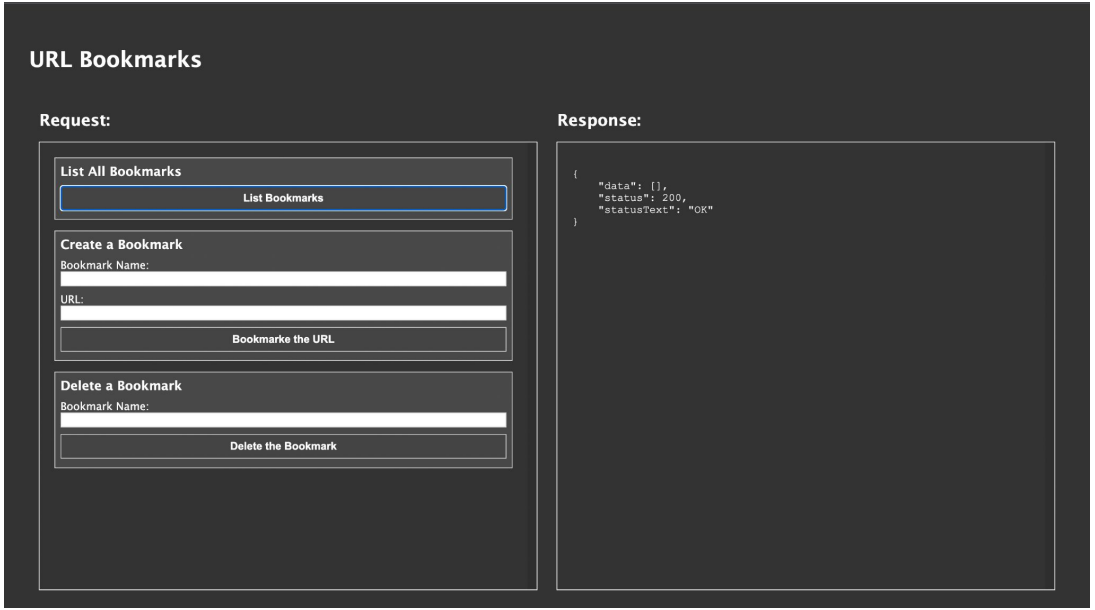
- /find_users?name=arvind&action=getInformation
- /shopping_cart?action=update_qty&user=123
- /postComment?entryID=853&text=...

What would they look like if they were RESTful?

- GET /users/arvind
- PATCH /carts/123
- POST /comment
    - with a body {entryId: 853, text: ‘hello world’}

Exercise

Clone the repo: https://github.com/61040-fa22/recitation-restful




```
/**
 * List all Bookmarks.
 *
 * @name GET /api/bookmarks
 *
 * @return {Bookmark[]} - list of all stored Bookmarks
 */
 router.get('/', (req, res) => {
  res.status(200).json(Bookmarks.findAll()).end();
});
```

Exercise

Clone the repo: https://github.com/61040-fa22/recitation-restful

Finish the todos in routes/bookmarks.js and public/javascripts/services.js

- **Creating a bookmark**
  - TODO 1. Design the RESTful API endpoint (URL) for creating a bookmark
  - TODO 2. Change the "get" to a proper method for creating a bookmark
  - TODO 3. Indicate the proper status code when there already is a bookmark with the same identifier (name)
  - TODO 4. Indicate the proper status code when the server successfully creates the bookmark
- **Deleting a bookmark**
  - TODO 5. Design the RESTful API endpoint (URL) for deleting a bookmark
  - TODO 6. Change the "get" to a proper method for deleting a bookmark