

Intro to Databases

Akshaj Kadaveru

How to navigate through workshops

1. Close out of any unsaved files

JS SingleComment.js ●

JS Feed.js ●

JS App.js ●

JS NewPostInput.js ●

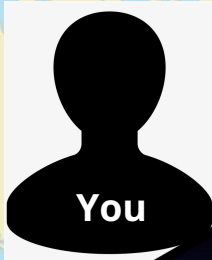
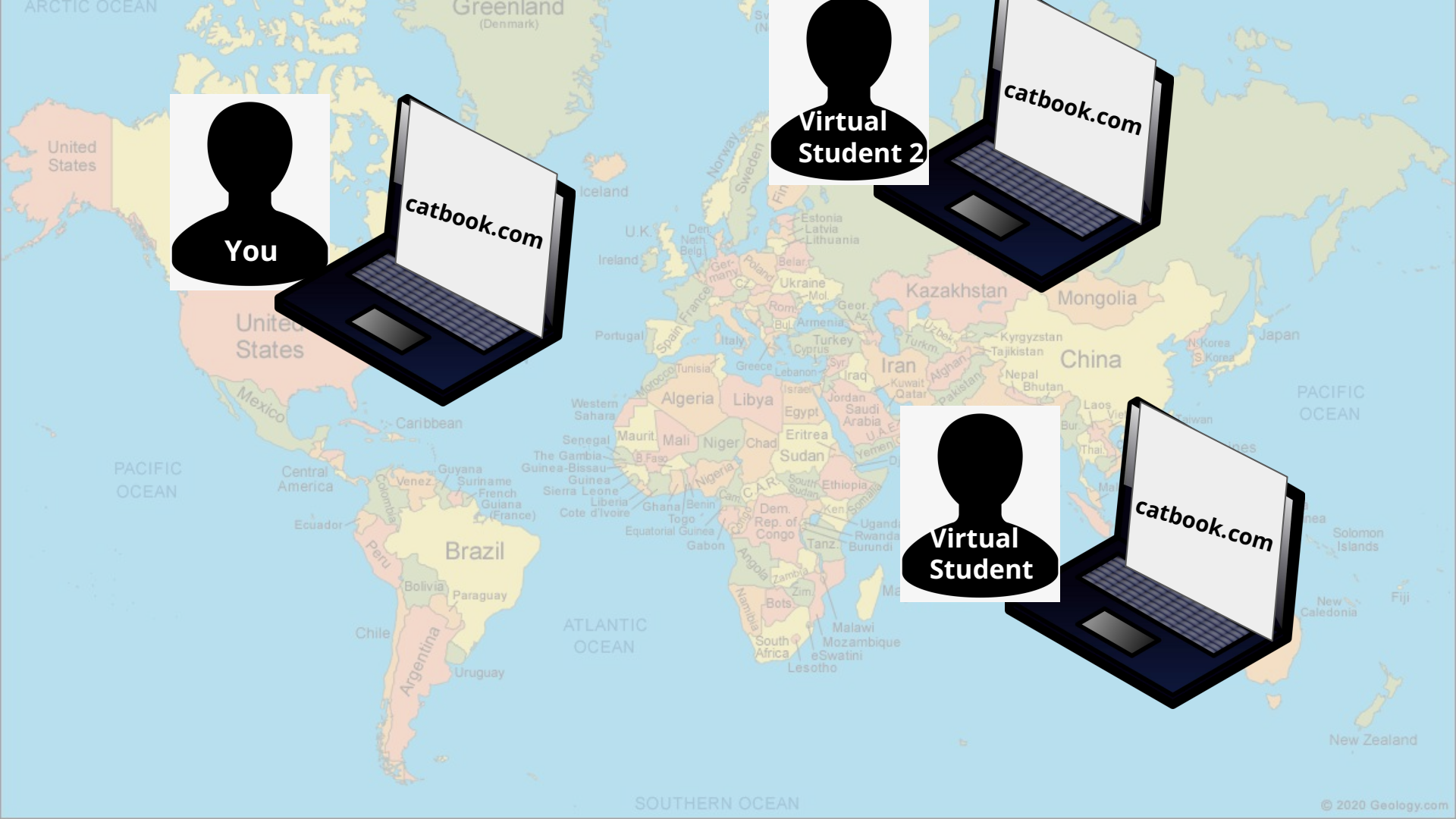
2. **git fetch** to get the latest information
3. **git reset --hard** to clear your changes and revert to the original workshop
4. **npm start** and **npm run hotloader** (in separate terminals)
5. if there's an error, ask **on Piazza** and we will respond **instantly**

Average Response Time:

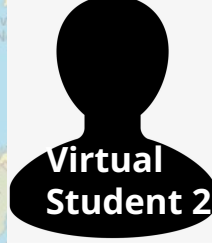
1 min

Websites are Worldwide

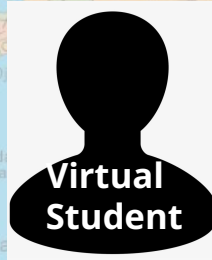




You



Virtual Student 2



Virtual Student



Client

You

catbook.com

Client

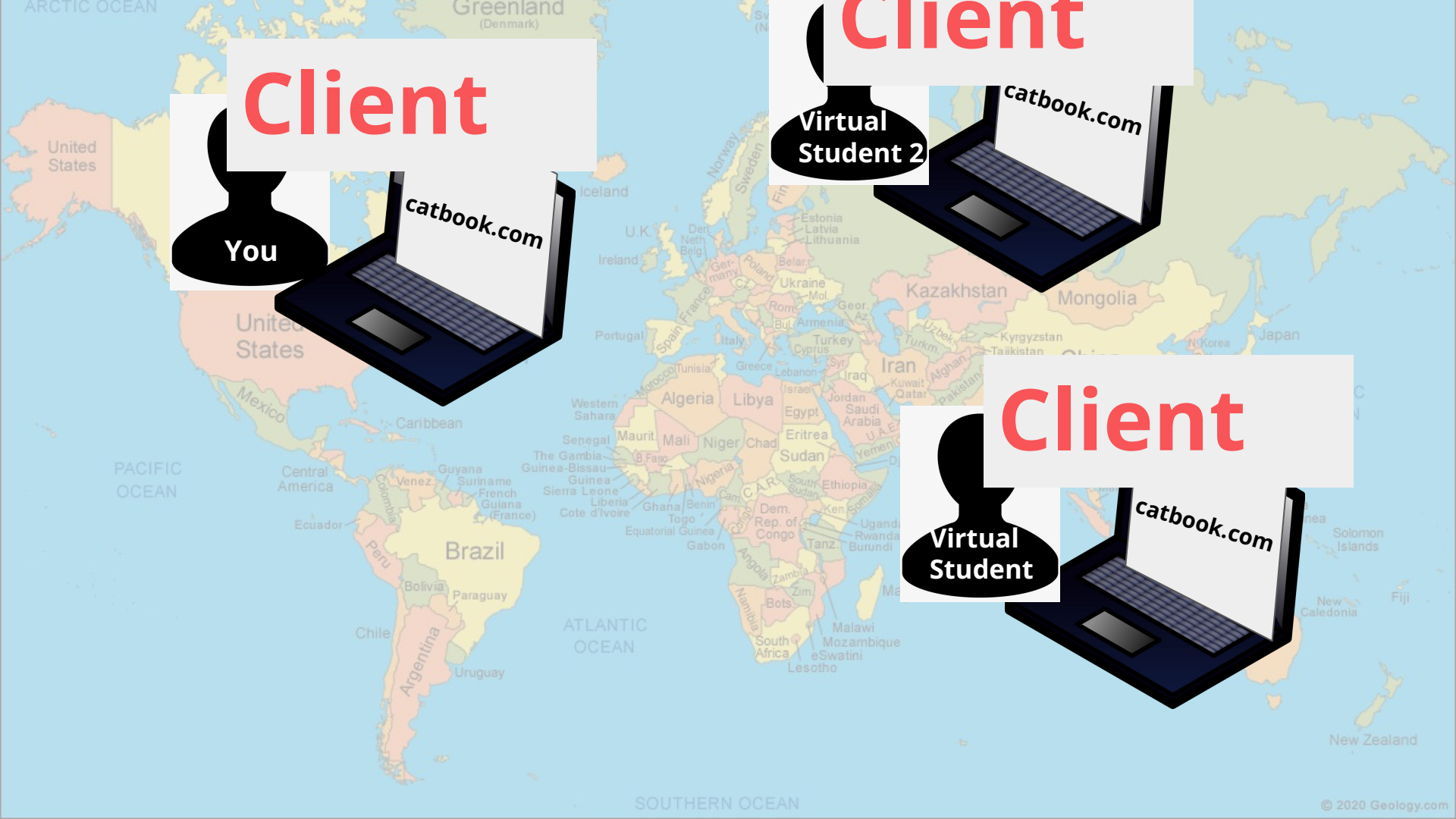
Virtual Student 2

catbook.com

Client

Virtual Student

catbook.com



Client

You

FRONTEND
CODE

Virtual
Student 2

FRONTEND
CODE

Client

Virtual
Student

FRONTEND
CODE

Frontend code:
The code that is in **front** of you

Client

You

```
<App />  
<div>  
<Profile />  
</div>
```

Virtual Student 2

Client

```
<App />  
<div>  
<Profile />  
</div>
```

Client

Virtual Student

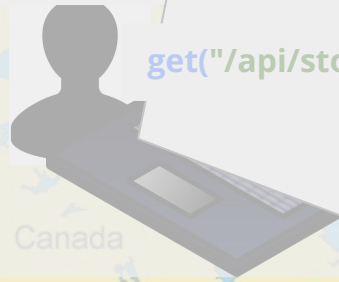
```
<App />  
<div>  
<Profile />  
</div>
```

Frontend code:
The code that is in **front** of you

How do users communicate with each other????

Client

`get("/api/stories")`



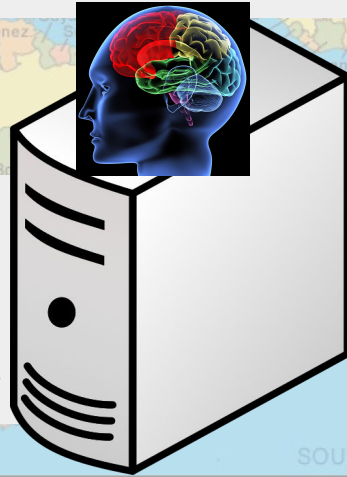
Client

`post("/api/story",
{text: "new story!"})`



Server!! the central hub

Backend code:
The code that is
behind the scenes



Client

`get("/api/stories")`



Server code is...

Instructions on how
to do everything

1. How to get all
stories

2. How to post a
new story

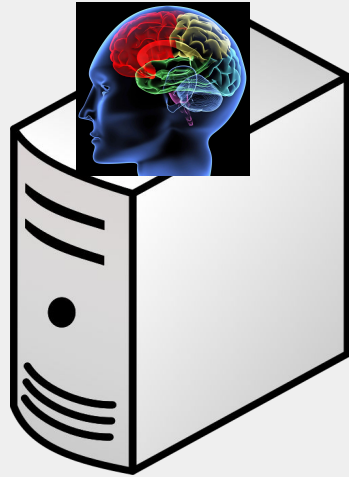
```
router.get("/stories", (req, res) => {  
  // send back all of the stories!  
  res.send(data.stories);  
});  
  
router.post("/story", (req, res) => {  
  // make a new story!  
  const newStory = {  
    _id: data.stories.length,  
    creator_name: MY_NAME,  
    content: req.body.content,  
  };  
  data.stories.push(newStory);  
  res.send(newStory);  
});
```

Our server can respond to five types of requests from the clients:

1. `get("/api/test")`
2. `get("/api/stories")`
3. `get("/api/comment")`
4. `post("/api/story")`
5. `post("/api/comment")`

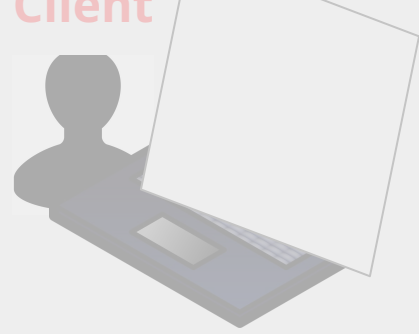
```
router.get("/test", (req, res) => {  
  
});  
  
router.get("/stories", (req, res) => {  
  
});  
  
router.get("/comment", (req, res) => {  
  
});  
  
router.post("/story", (req, res) => {  
  
});  
  
router.post("/comment", (req, res) => {  
  
});
```

Client



Server!!

Client



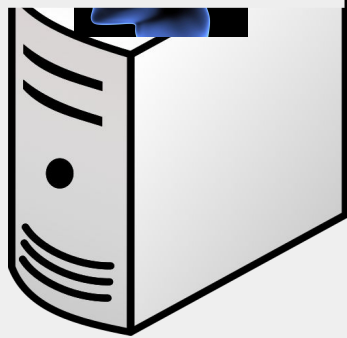
Client



Client

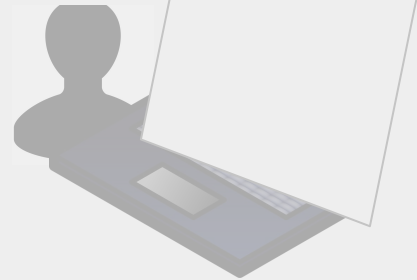


****acquires all the stories****



Server!!

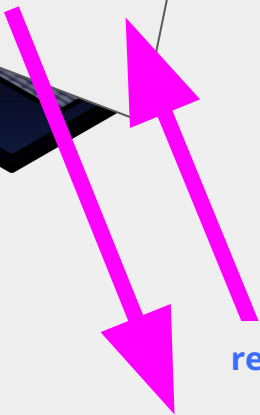
Client



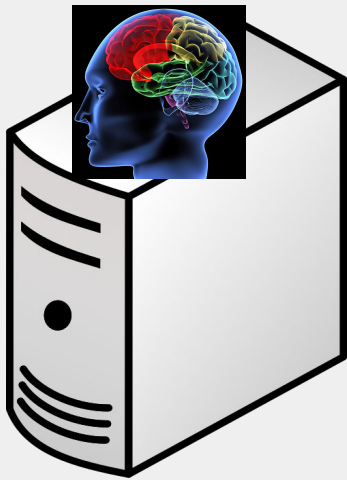
Client



Client

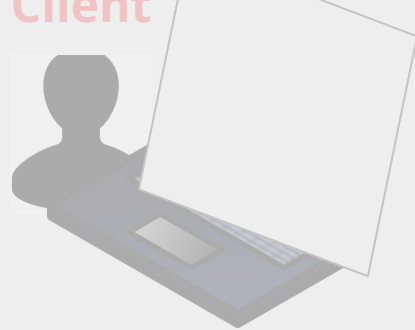


`res.send(allStories)`



Server!!

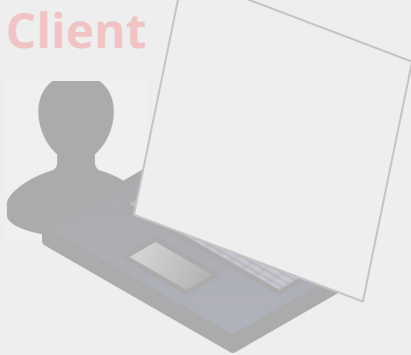
Client



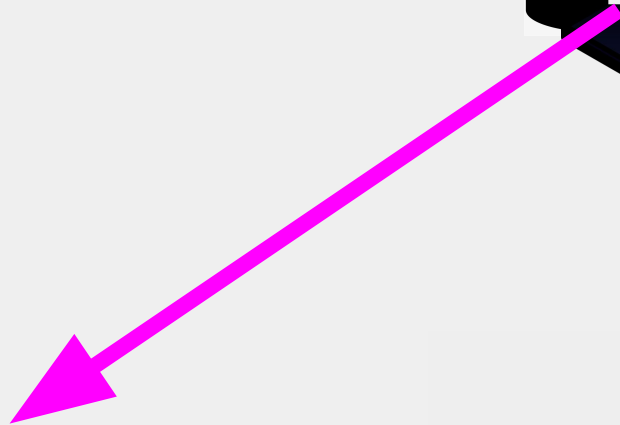
Client



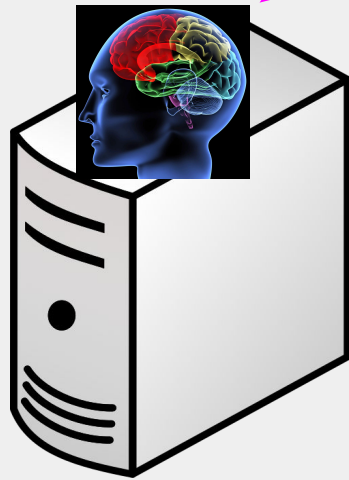
Client



Client



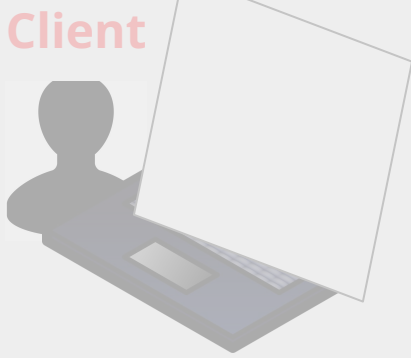
Server!!



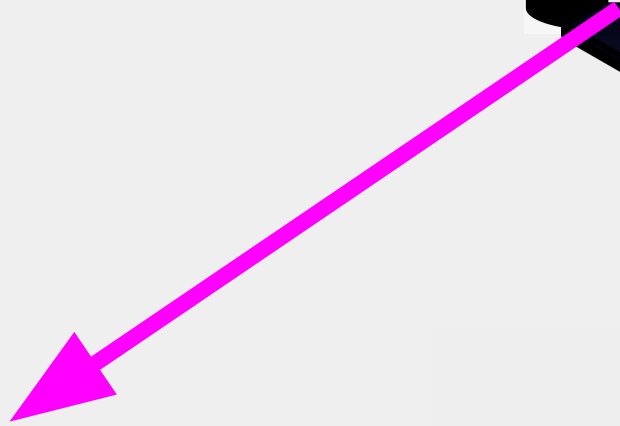
Client



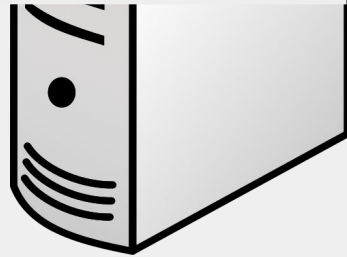
Client



Client



****adds "new story!"
as a new story in
wherever the
stories are stored****

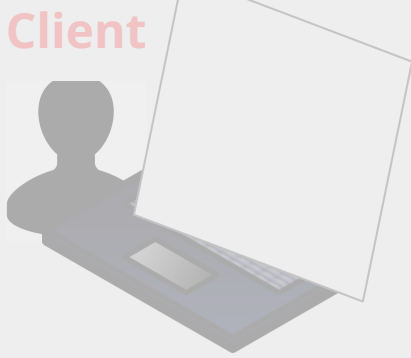


Server!!

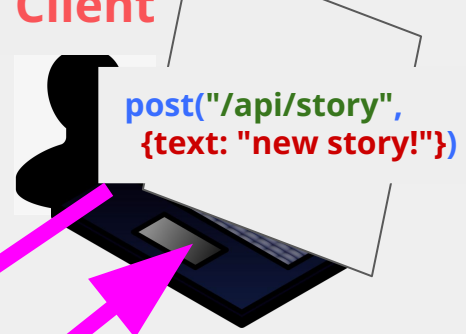
Client



Client



Client

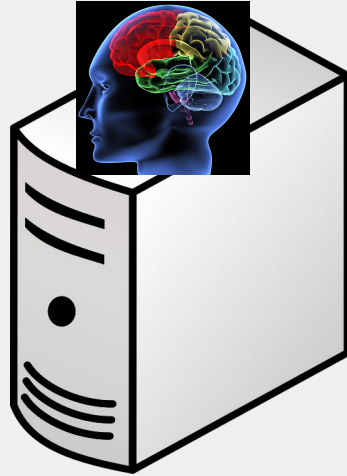


`res.send("done!")`

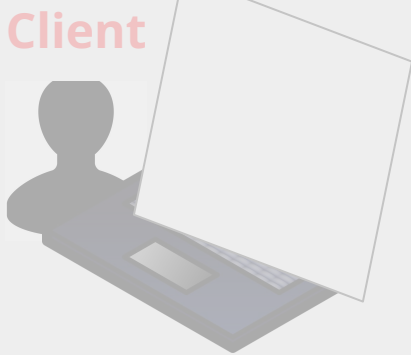
Client



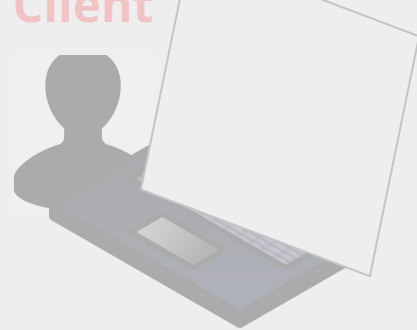
Server!!



Client



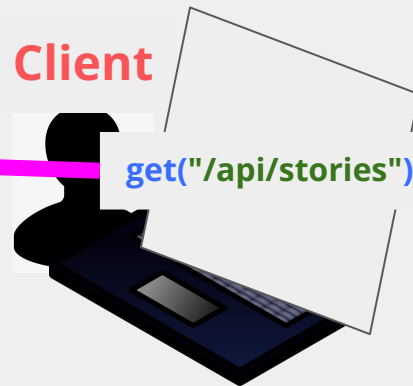
Client



Server!!



Client



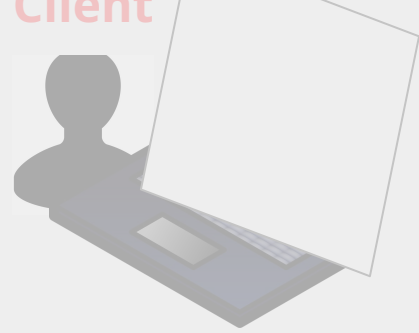
`get("/api/stories")`



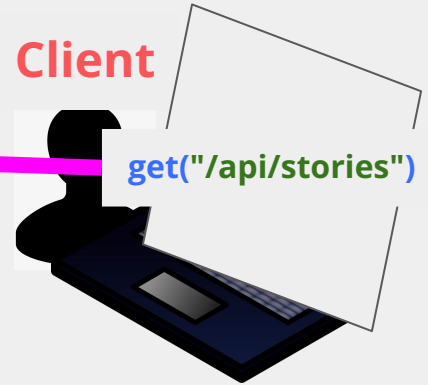
Client



Client



Client



`get("/api/stories")`



****acquires all the stories****

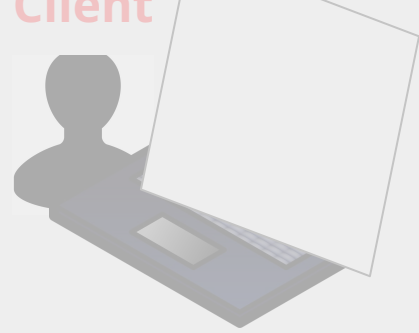


Server!!

Client



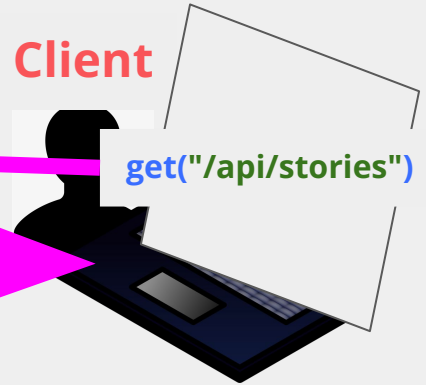
Client



Server!!



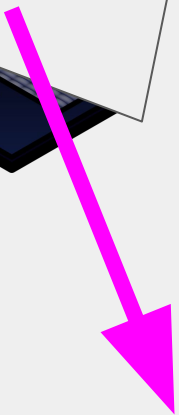
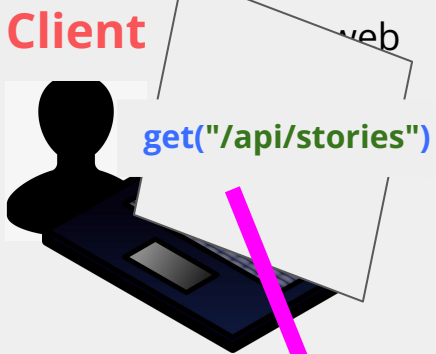
Client



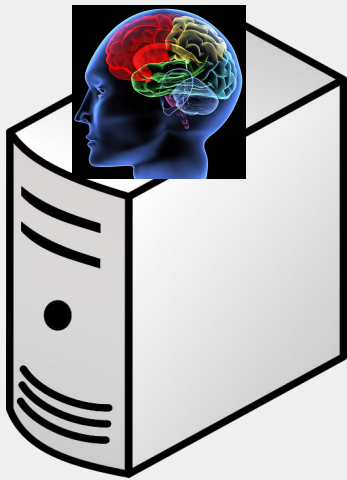
`get("/api/stories")`

`res.send(allStories)`

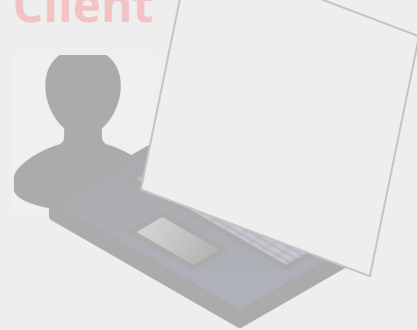
Client



Server!!



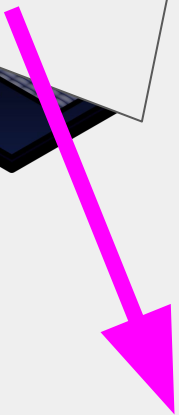
Client



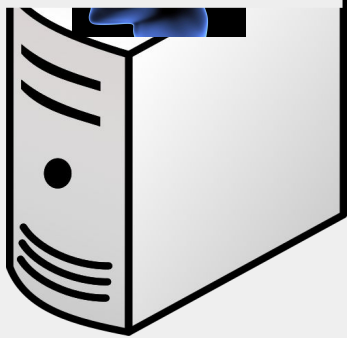
Client



Client

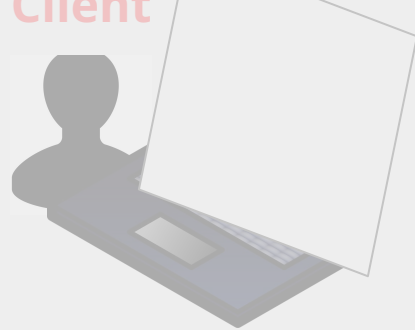


****acquires all the stories****



Server!!

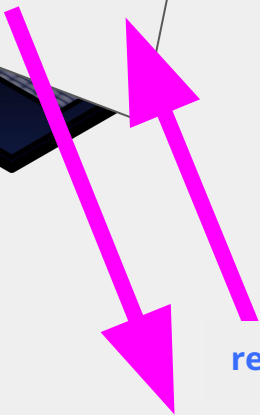
Client



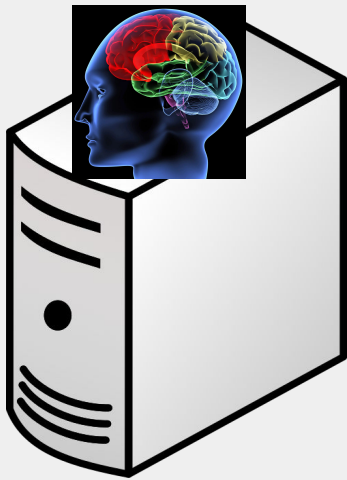
Client



Client

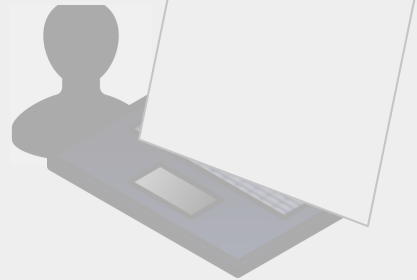


`res.send(allStories)`



Server!!

Client

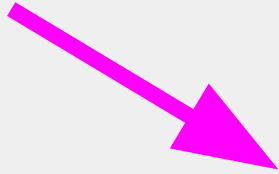


Client



Frontend (React Code)

```
post("/api/comment", { parent: "25", content: "New comment!"})
```

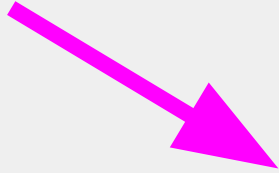


Backend (server code in api.js)

```
router.post("/comment", (req, res) => {  
  const newComment = {  
    parent: req.body.parent,  
    content: req.body.content,  
  };  
  data.comments.push(newComment);  
  res.send(newComment);  
});
```

Frontend (React Code)

```
get("/api/comment", { parent: "25" })
```

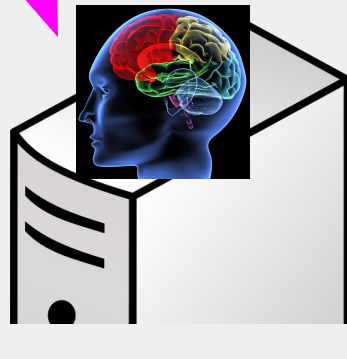
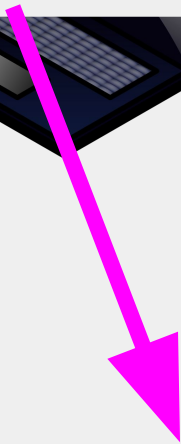
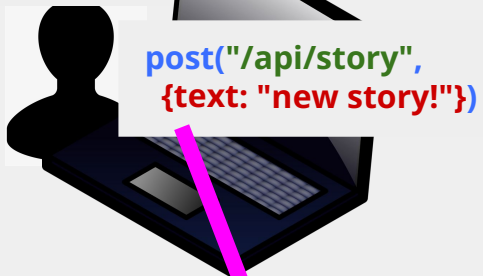


Backend (server code in api.js)

```
router.get("/comment", (req, res) => {  
  const filteredComments = data.comments.filter(  
    (comment) => comment.parent == req.query.parent);  
  res.send(filteredComments)  
});
```

Intro to Databases

Client



Server!!

Client



```
post("/api/story",  
  {text: "new story!"})
```

****adds "new story!"
as a new story in
wherever the
stories are stored****



Server!!

api.js (the server)

```
const data = {  
  stories: [  
    {  
      _id: 0,  
      creator_name: "Shannen Wu",  
      content: "I love corgis!"  
    }  
  ],  
  comments: [  
    {  
      _id: 0,  
      creator_name: "Jessica Tang",  
      parent: 0,  
      content: "Wow! Me Too!",  
    }  
  ],  
};
```

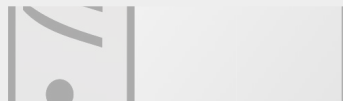
Client



```
post("/api/story",  
  {text: "new story!"})
```

Currently, Data is stored in
the server

****adds "new story!"
as a new story in
wherever the
stories are stored****



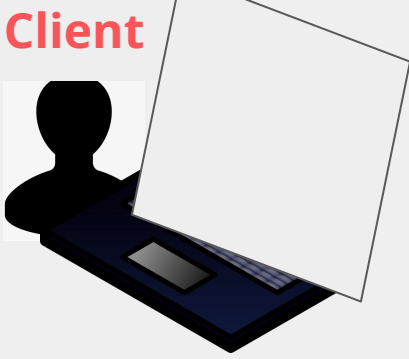
Server!!

api.js (the server)

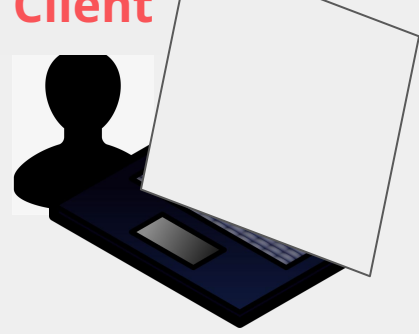
```
const data = {  
  stories: [  
    {  
      _id: 0,  
      creator_name: "Shannen Wu",  
      content: "I love corgis!"  
    }  
  ],  
  comments: [  
    {  
      _id: 0,  
      creator_name: "Jessica Tang",  
      parent: 0,  
      content: "Wow! Me Too!",  
    }  
  ],  
};
```

What's wrong with storing data in server
as a variable

Client



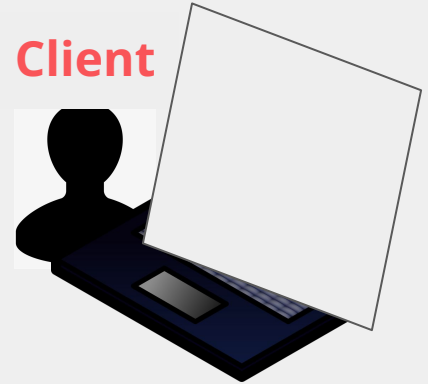
Client



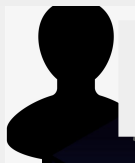
Server!!



Client



Client



```
post("/api/story",  
  {text: "hi"})
```



```
stories = ["hi"]
```

Server!!

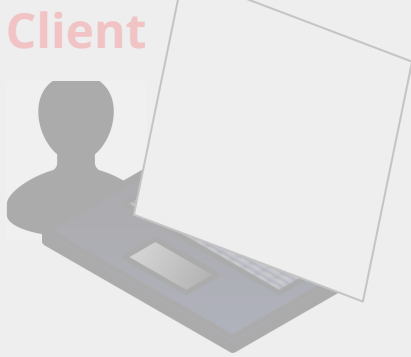
Client



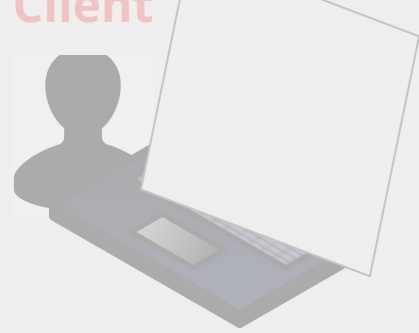
Client



Client



Client



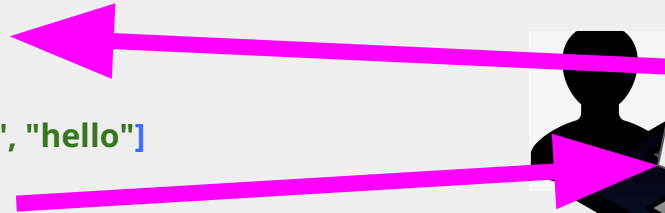
Server!!



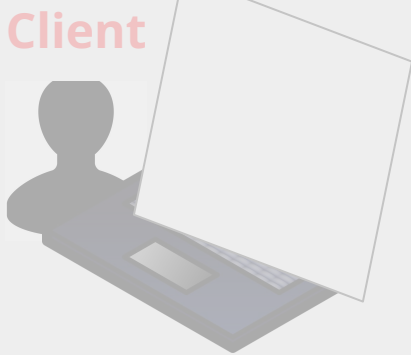
Client



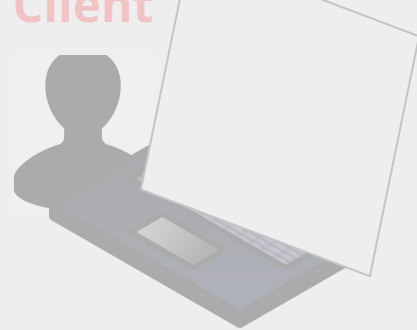
```
post("/api/story",  
      {text: "hello"})
```



Client



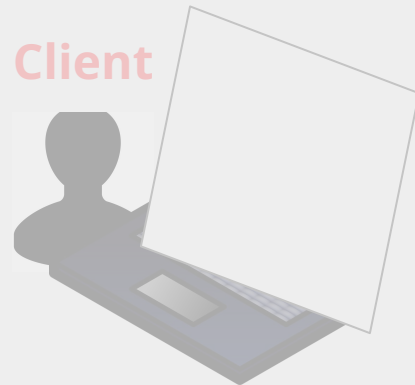
Client



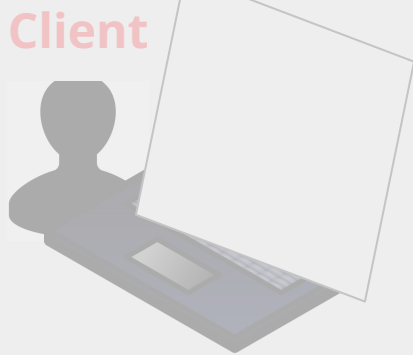
Server!!



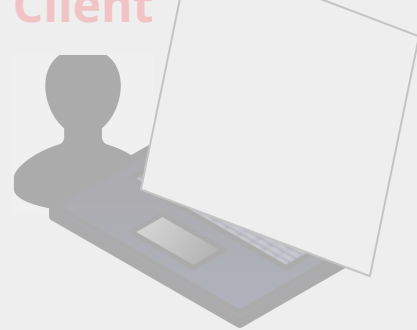
Client



Client



Client



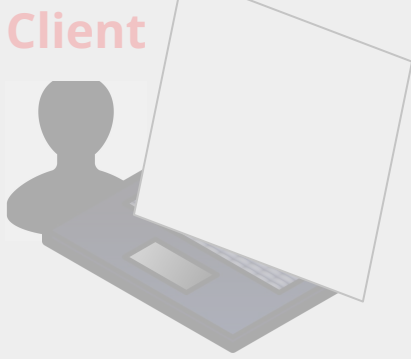
Client



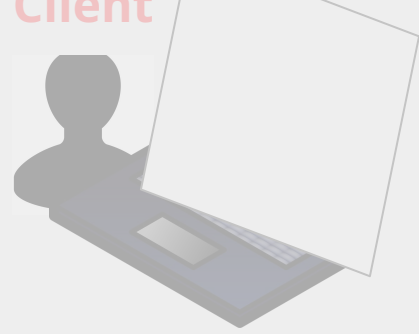
stories = ["hi", "hello"]

Server!!

Client



Client

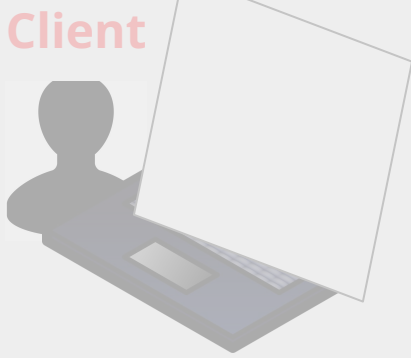


Server down

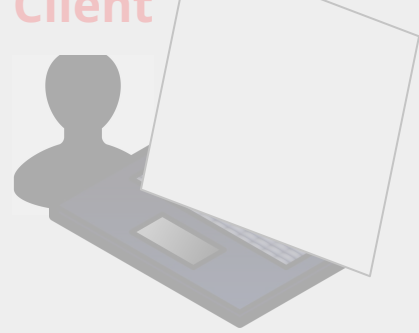
Client



Client



Client



Fully Charged

Server!!



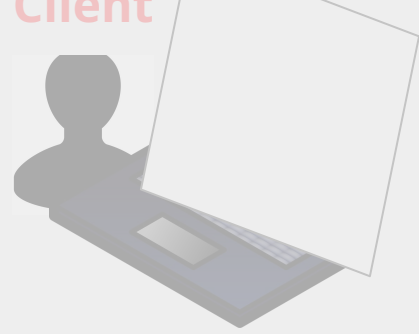
Client



Client



Client



Client



all data lost :(


What's wrong with storing data in a variable on the server

What happens if...

- You close Terminal
- You modify your server code
- Your server crashes
- Your laptop runs out of battery

All your data is gone!

Gets reset when
you restart server



```
let data = {  
  stories: [  
    {  
      _id: 0,  
      creator_name: "Shannen Wu",  
      content: "I love corgis!",  
    },  
  ],  
}
```

What's wrong with storing data in a variable on the server

What happens if...

- Catbook gets thousand of users
- You need to store gigabytes of stories and comments

Run out of memory (RAM)!

```
let data = {  
  stories: [  
    {  
      _id: 0,  
      creator_name: "Shannen Wu",  
      content: "I love corgis!",  
    },  
  ],  
}
```

Demo in Catbook of all data lost :(

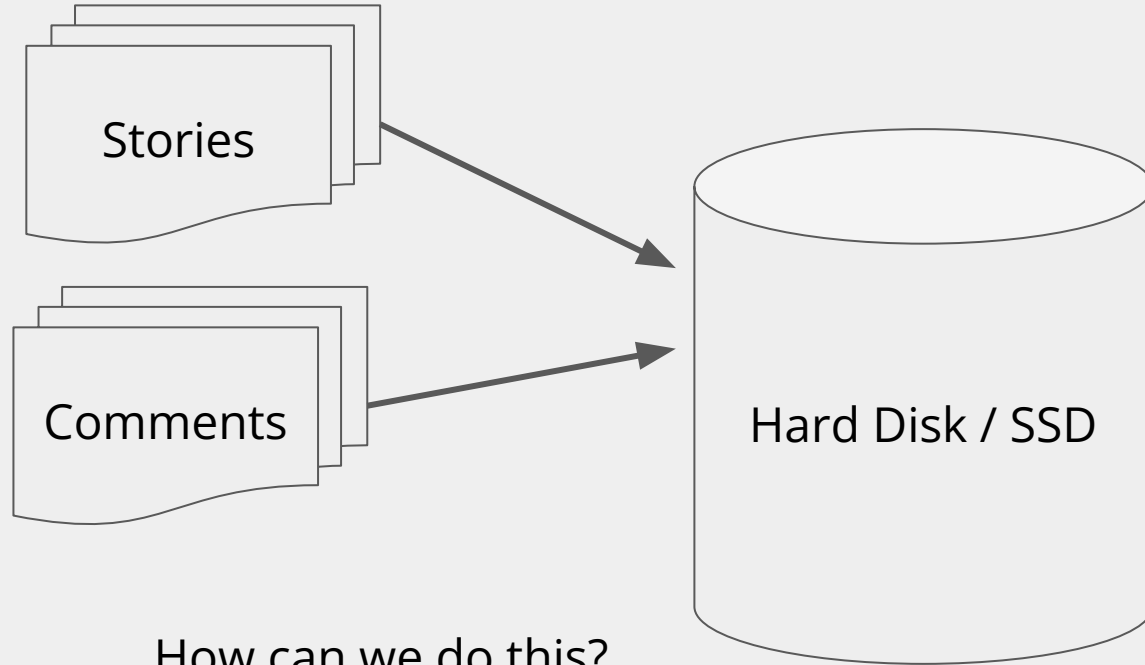
Hackerman
HI

Hackerman
HALLO

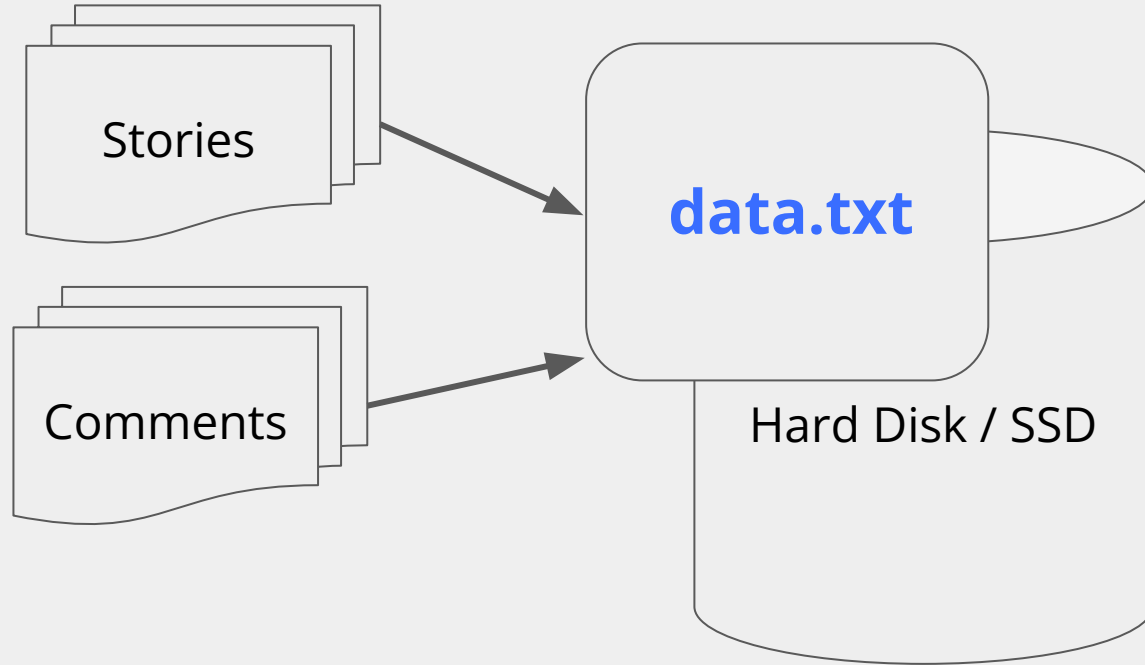
Shannen Wu
I love corgis!

Jessica Tang | Wow! Me Too!

We need to store our data permanently



Lets use a text file



Writing to File

Load data
from data.txt

```
function readDataFromFile() {  
  if (!fs.existsSync("data.txt")) return;  
  fs.readFile("data.txt", (err, fileData) => {  
    data = JSON.parse(fileData);  
  });  
}
```

Save data to
data.txt

```
function writeDataToFile() {  
  fs.writeFile("data.txt", JSON.stringify(data), (err) => {  
    if (err) console.log(err);  
  });  
}
```

Writing to File

```
router.post("/story", (req, res) => {  
  const newStory = {  
    _id: data.stories.length,  
    creator_name: MY_NAME,  
    content: req.body.content,  
  };  
  
  data.stories.push(newStory);  
  writeDataToFile();  
  
  res.send(newStory);  
});
```

Loading from file

```
// read existing data from the file when the server starts up
readDataFromFile();

router.get("/stories", (req, res) => {
  // just send back all of the stories!
  res.send(data.stories);
});
```

Demo of data.txt approach

Hooray!! We solved the problem



What's wrong with data.txt

What's wrong with data.txt

- **Write Speed:** Saves *every story/comment* whenever someone posts

What's wrong with data.txt?

- **Write Speed:** Saves *every story/comment* whenever someone posts
- **Memory Usage:** Still keeps all stories/comments in RAM

(still exists in api.js)

```
let data = {  
  stories: [],  
  comments: [],  
};
```

What's wrong with data.txt?

- **Write Speed:** Saves *every story/comment* whenever someone posts
- **Memory Usage:** Still keeps all stories/comments in RAM
- **Query Speed:** To find story with a particular `_id`, we linear search through all stories

Find me a story
with `_id = 23`

Stories

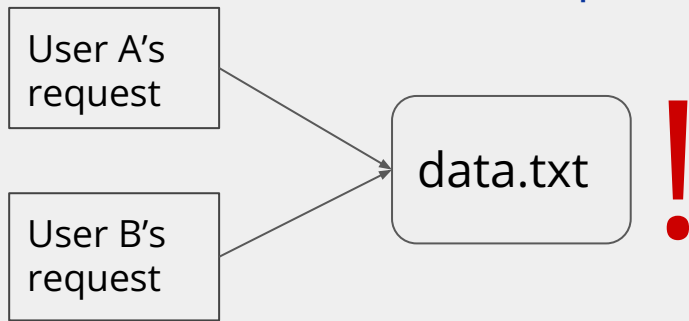
```
{_id: 11, ...  
{_id: 71, ...  
{_id: 21, ...  
{_id: 62, ...  
{_id: 73, ...  
{_id: 54, ...  
{_id: 23, ...  
{_id: 47, ...
```

What's wrong with `data.txt`?

- **Write Speed:** Saves *every story/comment* whenever someone posts
- **Memory Usage:** Still keeps all stories/comments in RAM
- **Query Speed:** To find story with a particular `_id`, we linear search through all stories
- **Single Point of Failure:** What if your laptop hard drive breaks?

What's wrong with data.txt?

- **Write Speed:** Saves *every story/comment* whenever someone posts
- **Memory Usage:** Still keeps all stories/comments in RAM
- **Query Speed:** To find story with a particular `_id`, we linear search through all stories
- **Single Point of Failure:** What if your laptop hard drive breaks?
- **Concurrency Issues:** What if two users post at the exact same time?



It is unsafe to use `fs.write()` multiple times on the same file without waiting for the callback.

What's wrong with `data.txt`?

- **Write Speed:** Saves *every story/comment* whenever someone posts
- **Memory Usage:** Still keeps all stories/comments in RAM
- **Query Speed:** To find story with a particular `_id`, we linear search through all stories
- **Single Point of Failure:** What if your laptop hard drive breaks?
- **Concurrency Issues:** What if two users post at the exact same time?
- and more...

How to fix these issues?

How to fix these issues?

All these problems have been solved for us!



Databases

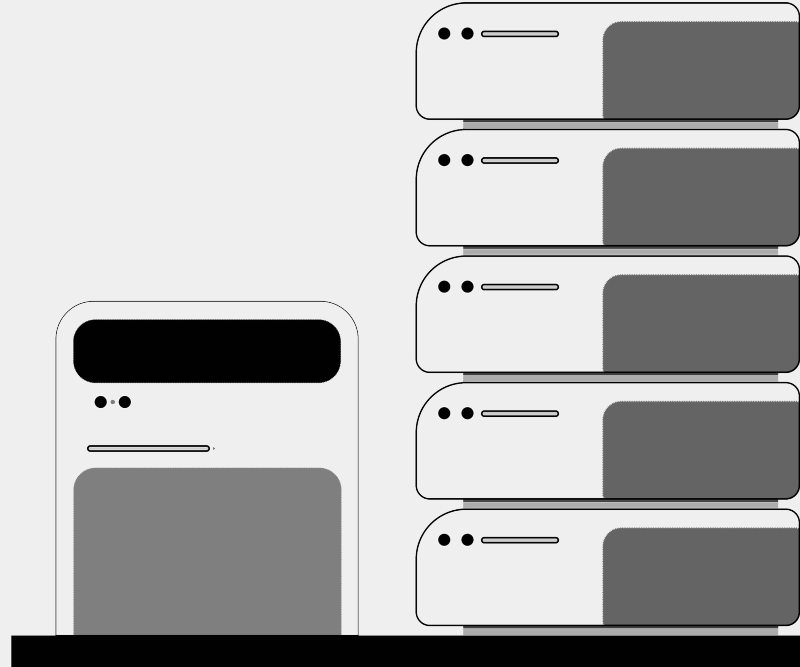


What is a database?

- **Database (DB):**

Organized collection of data

- **DBMS:** functions that let you retrieve/add/modify data



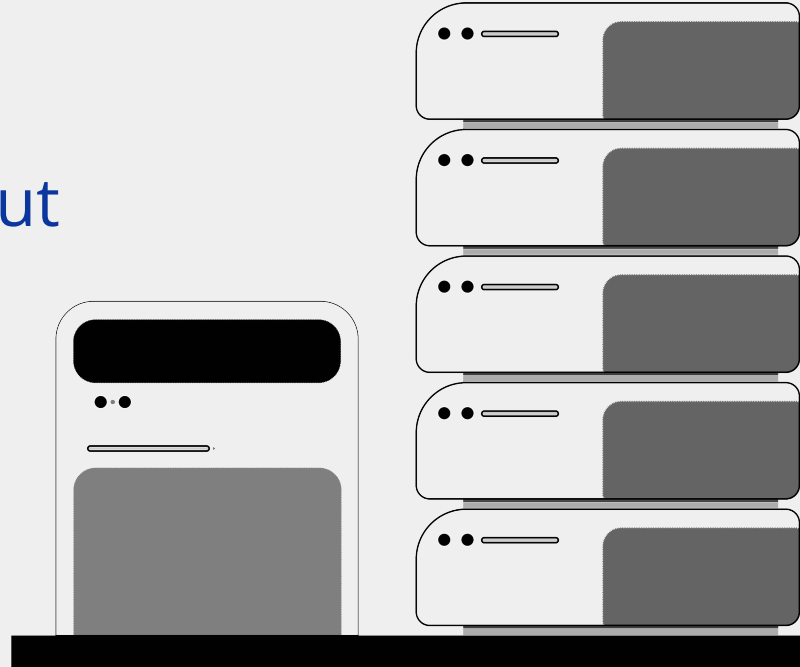
What is a database?

- Database (DB):

data.txt but better

- DBMS:

read/writeDataFromFile() but better



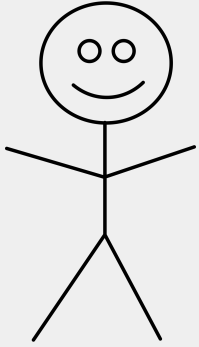
Reading from Database



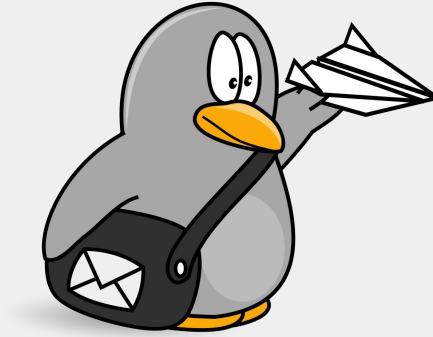
server DBMS



database



↑
server

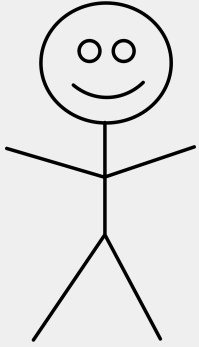


↑
DBMS

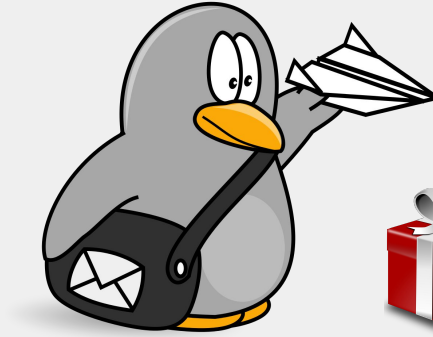


↑
database

retrieve
stories
from DB



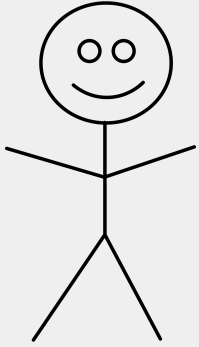
↑
server



↑
DBMS



↑
database



↑
server



↑
DBMS



↑
database



↑
server DBMS



↑
database

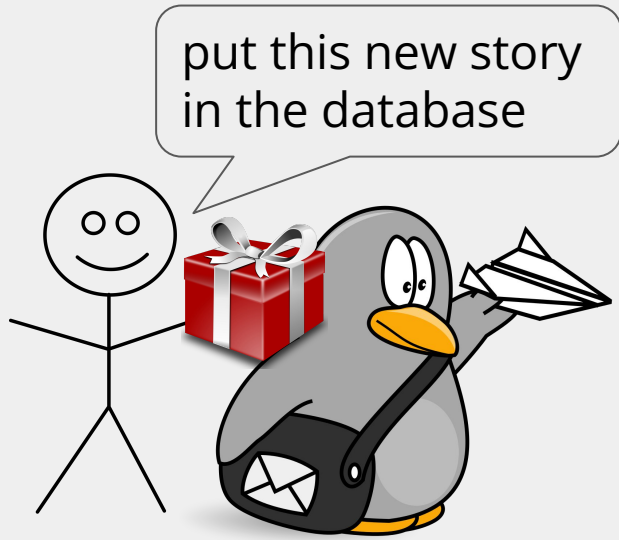


↑
server DBMS



↑
database

Writing to Database



↑
server DBMS



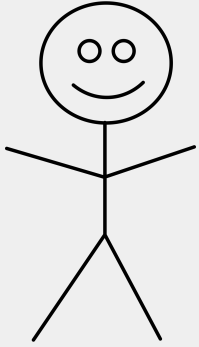
↑
database



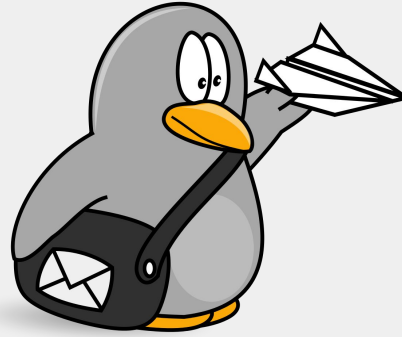
↑
server DBMS



↑
database



↑
server



↑
DBMS



↑
database

Pseudocode for a generic DBMS

```
import dbms;  
stories = dbms("give me all stories");           // read  
story4 = dbms("give me the story with _id=4");   // read with query  
dbms("put this story in the database", { ... }); // write  
dbms("delete all stories posted by Akshaj");     // delete  
dbms("change the author of story4 to Daniel");   // update
```

```
// note: this isn't real code, don't actually try to run this
```

What kind of database can I use?

Relational Database (SQL)

Stores data in a spreadsheet-like format (tables)

Stories		
_id	creator_name	content
0	cor	i like cat
1	matt	hi there
2	aaron	uwu owo
3	jynnie	henwo



What kind of database can I use?

Document Database (NoSQL)

Stores “documents”, which are basically JSON objects

Stories

_id:	1
creator_name:	matt
content:	hi there

_id:	0
creator_name:	cor
content:	i like cat



What kind of database can I use?

Document Database (NoSQL)

Stores “documents”, which are basically JSON objects

We'll be using this!

You're already familiar with JSON

Stories	
_id: 1	_id: 0
creator_name: matt	creator_name: cor
content: hi there	content: i like cat



Introducing MongoDB

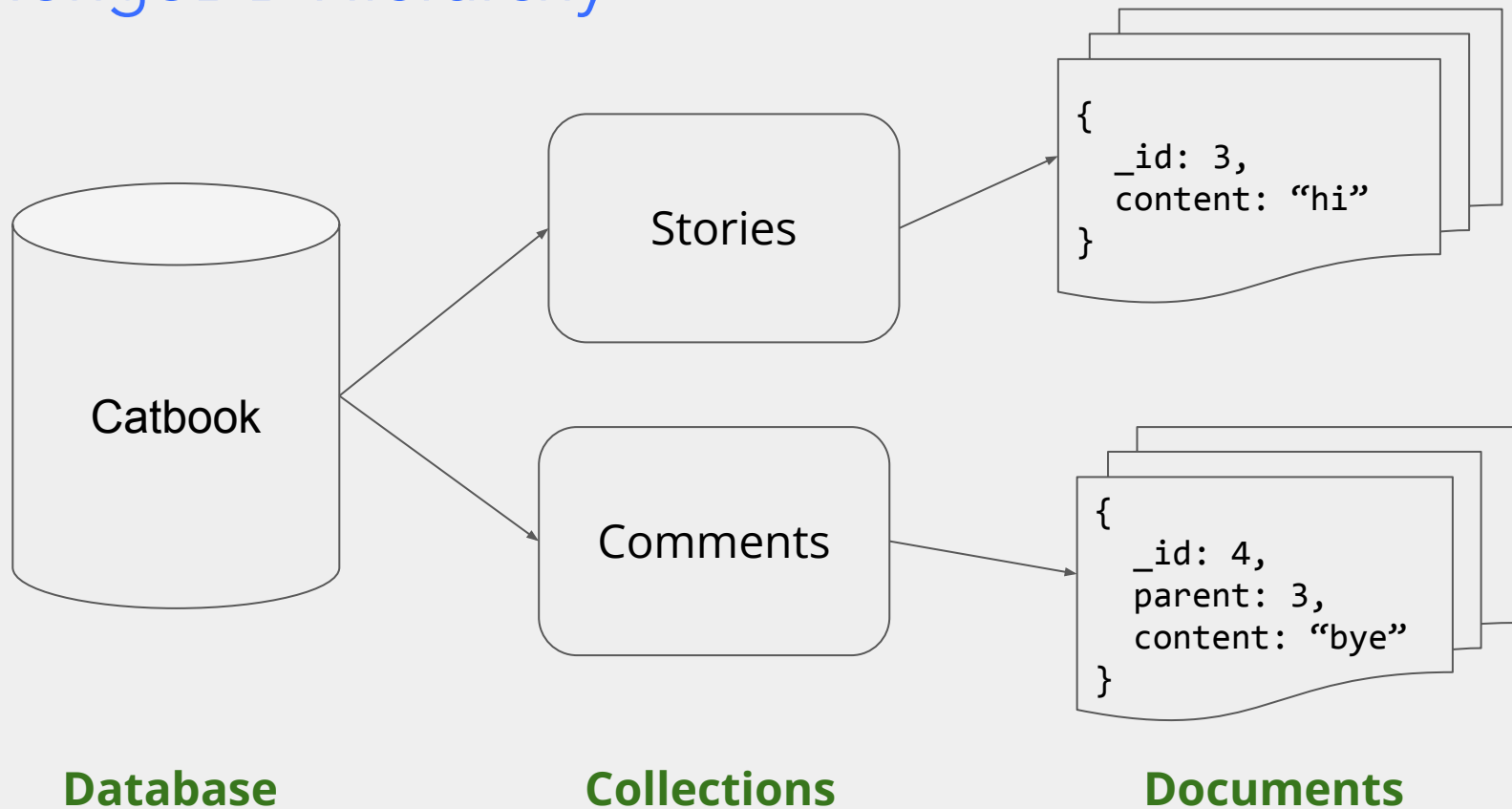


THE DOCUMENT MODEL

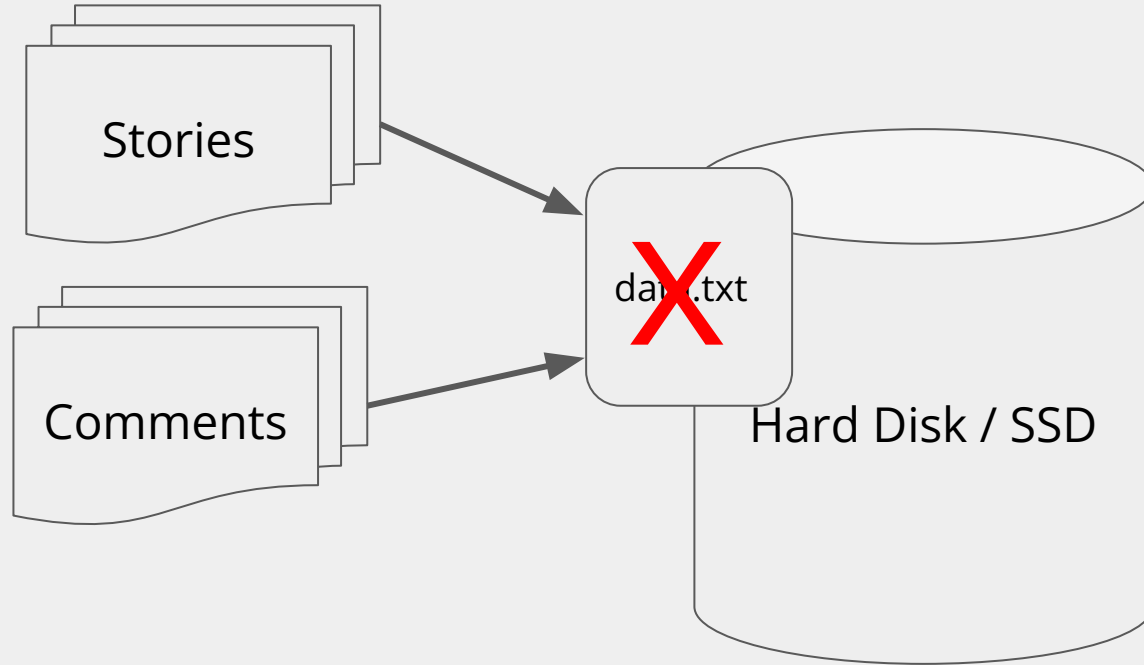
As a programmer, you think in objects. Now
your database does too.

MongoDB is a document database, which means it stores data in JSON-like documents. We believe this is the most natural way to think about data, and is much more expressive and powerful than the traditional row/column model.

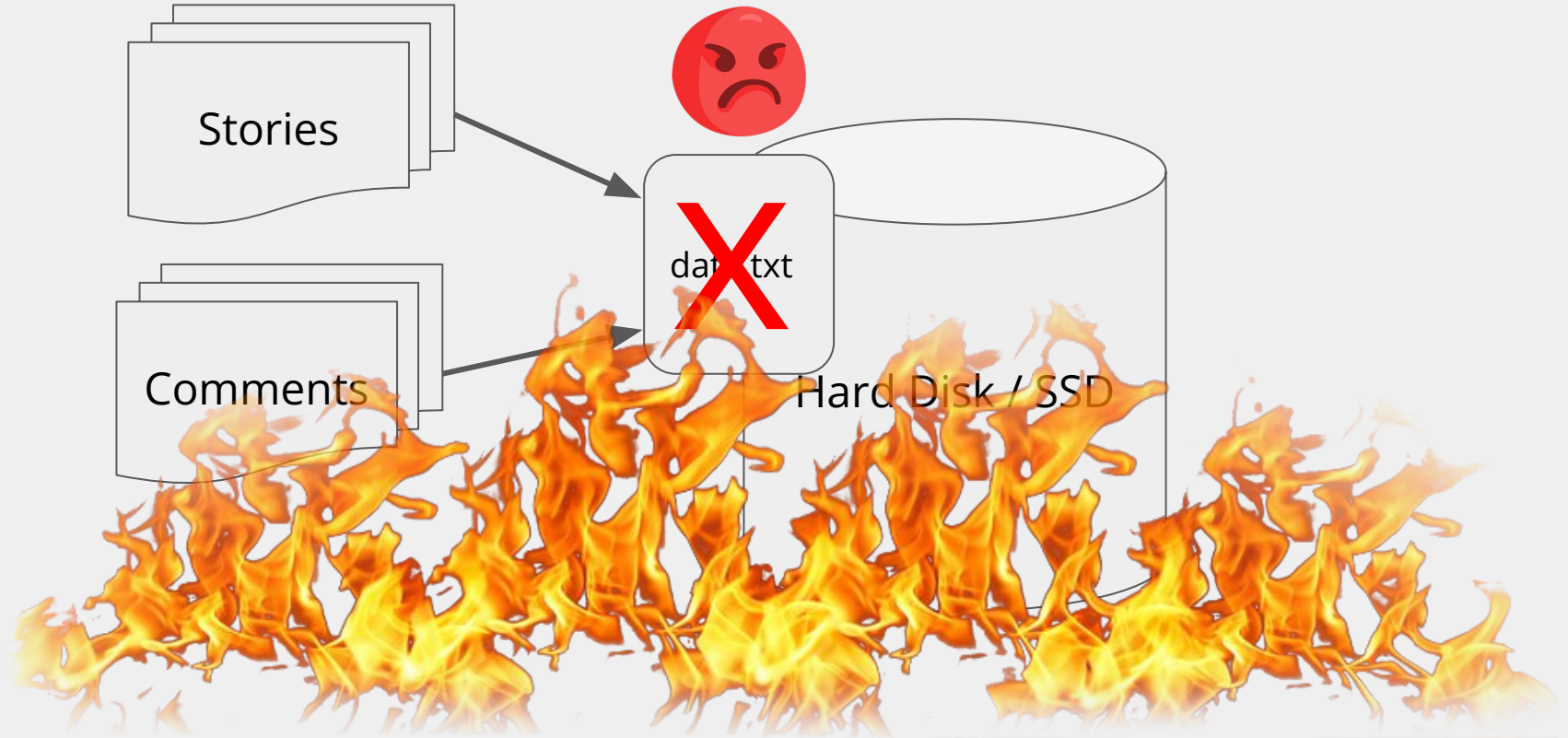
MongoDB Hierarchy



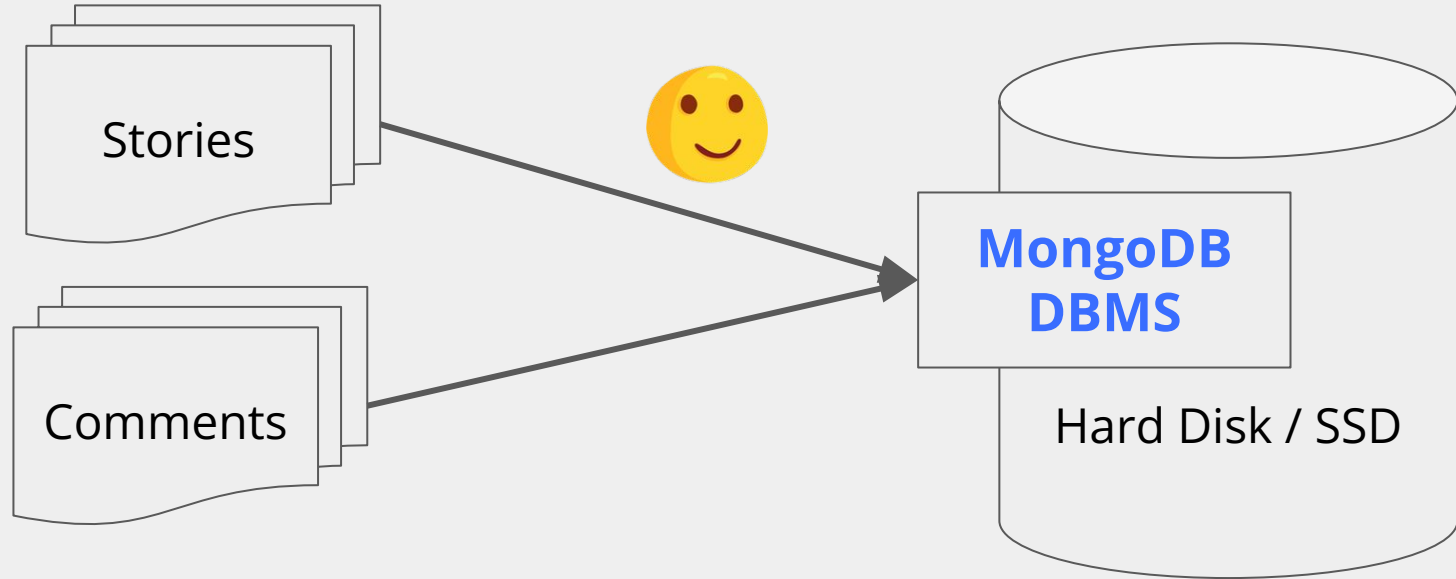
Replacing data.txt



Replacing data.txt



Replacing data.txt



Does MongoDB fix our issues?

- **Write Speed:** Optimized!
- **Memory Usage:** Optimized!
- **Query Speed:** Optimized!
- **Concurrency Issues:** Solved!

Does MongoDB fix our issues?

- **Write Speed:** Optimized!
- **Memory Usage:** Optimized!
- **Query Speed:** Optimized!
- **Concurrency Issues:** Solved!
- **Single Point of Failure:** Our hard drive could still break...

MongoDB Atlas

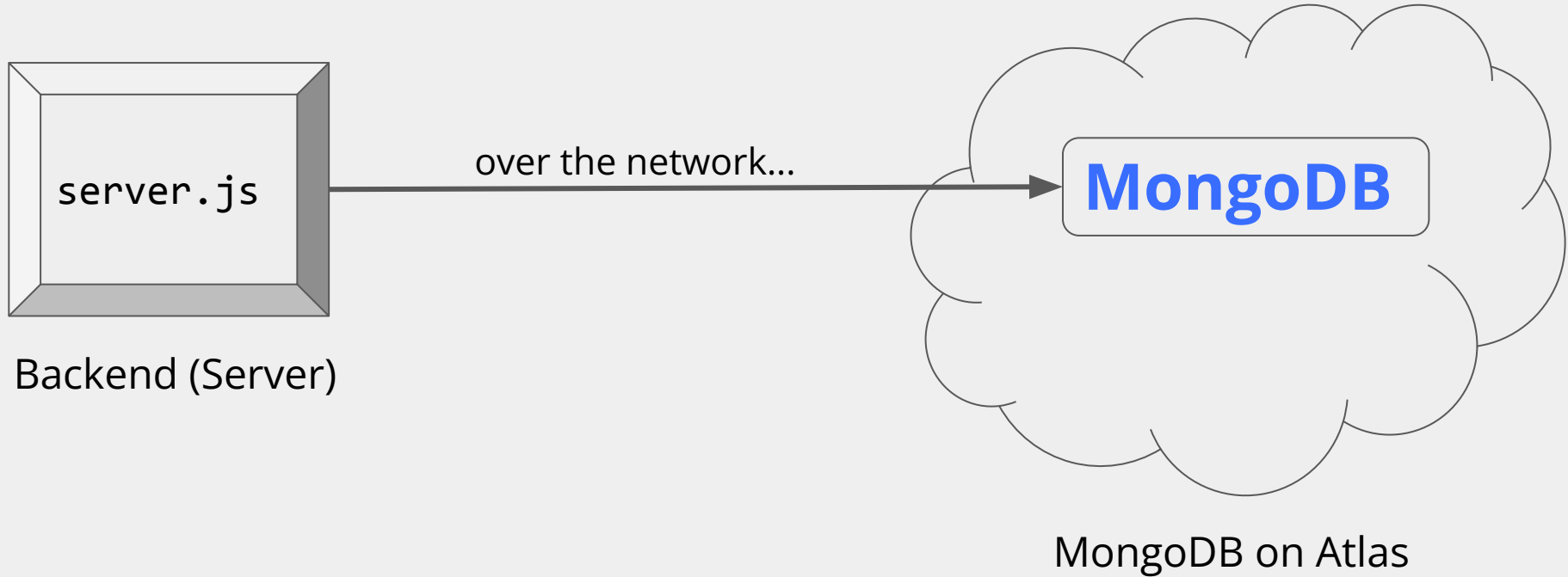


mongoDB®

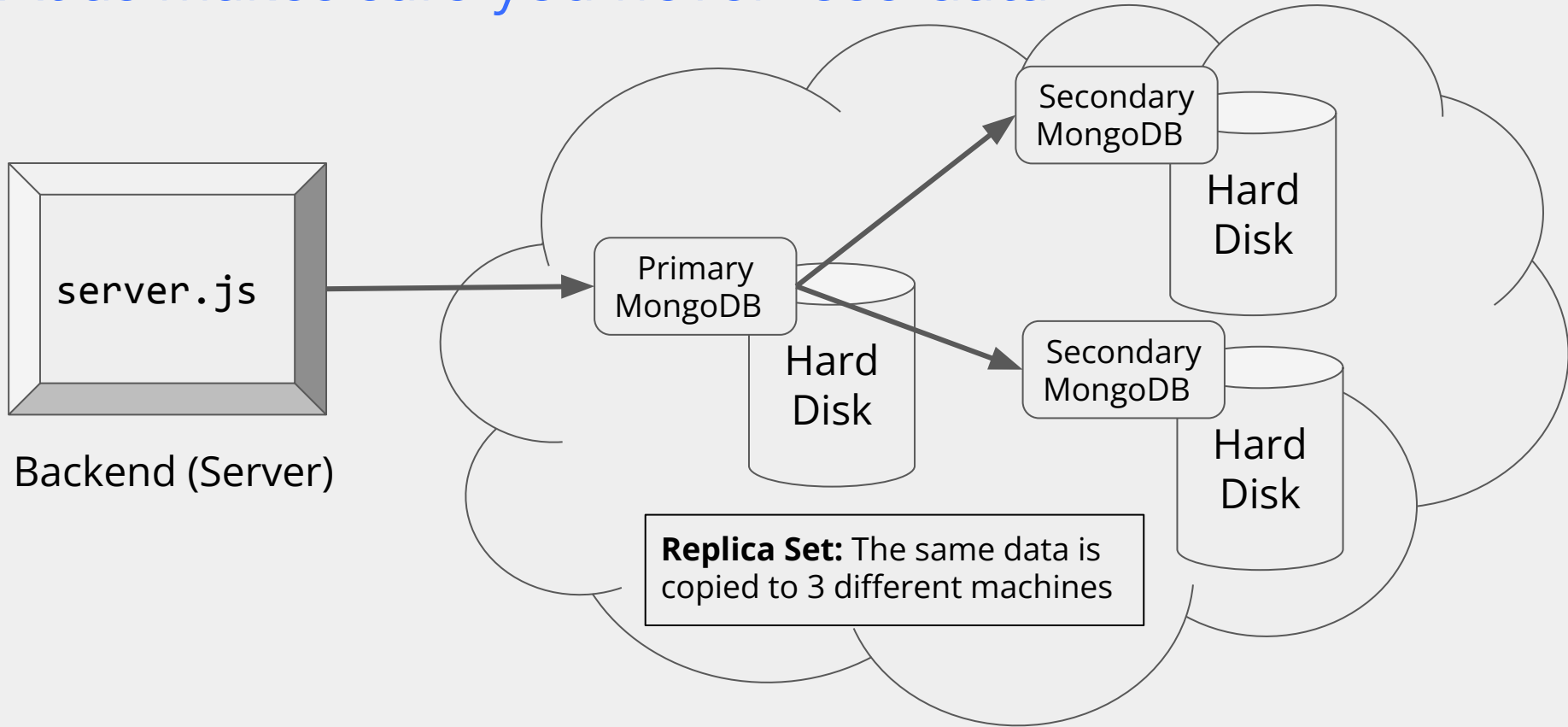
- Run MongoDB somewhere else (on the “cloud”)
- Makes your life easier
 - Don't need to run the database on laptop
 - Database is managed for you
 - **Can share data with teammates**
- More reliable than your laptop SSD
 - Replicates your data!



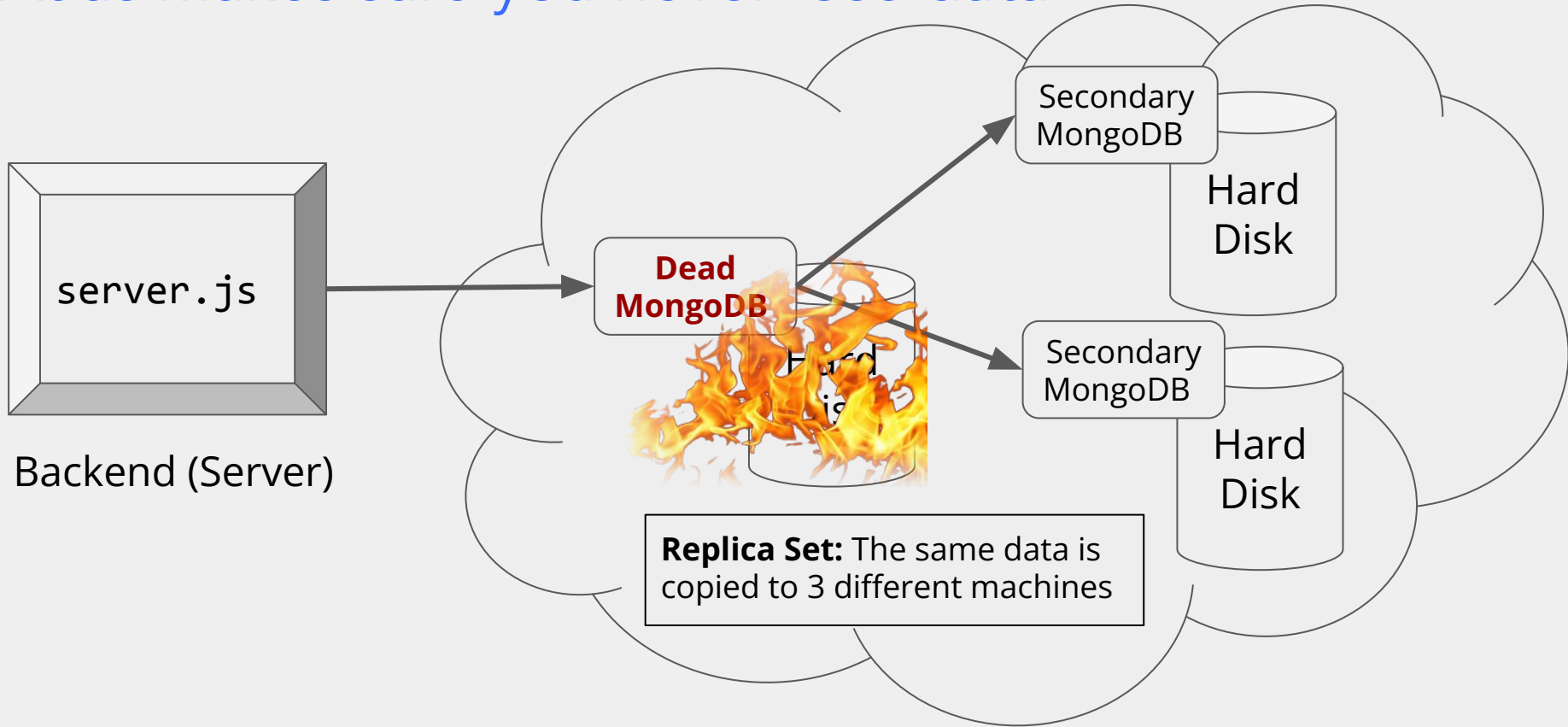
Catbook configuration with Atlas



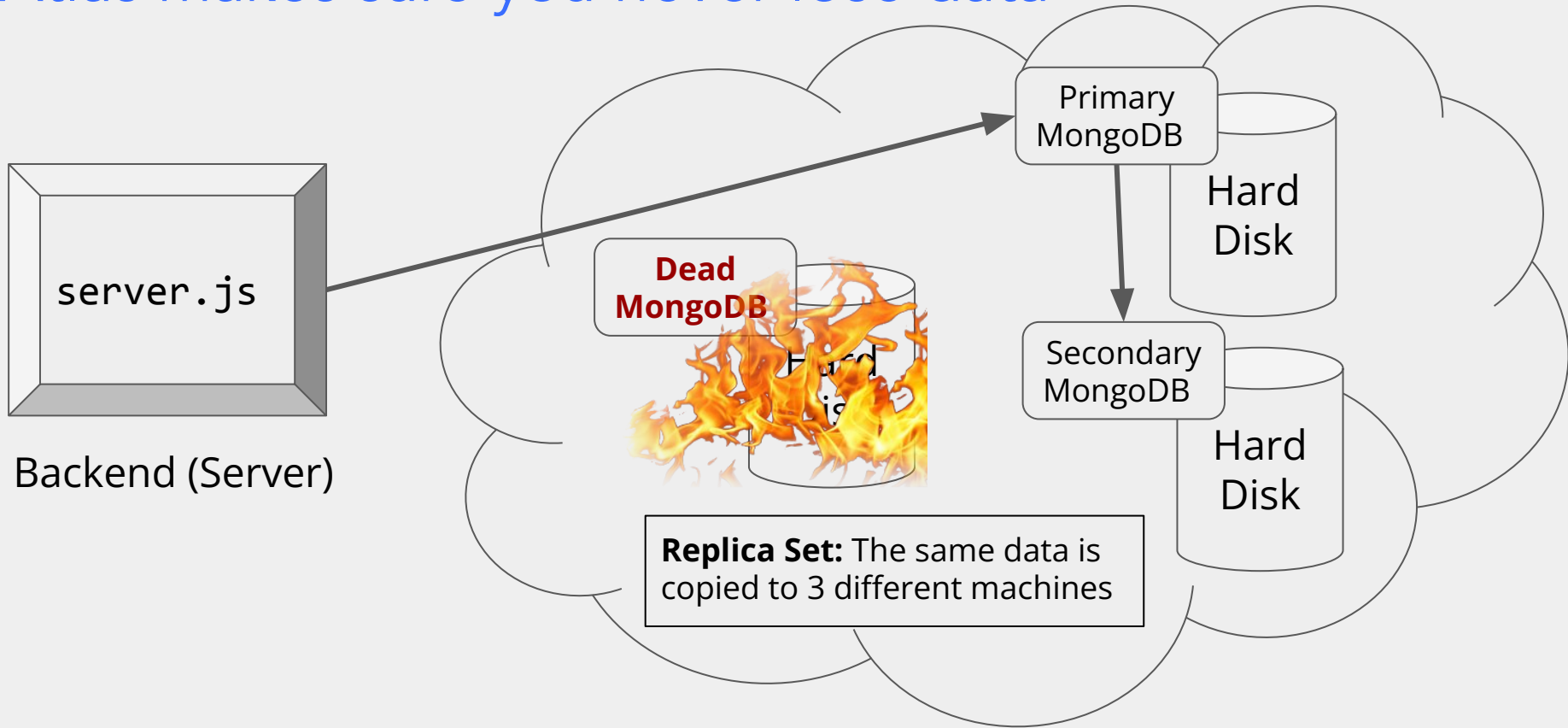
Atlas makes sure you never lose data



Atlas makes sure you never lose data



Atlas makes sure you never lose data



MongoDB Atlas Demo

catbook-workshop2.comments

COLLECTION SIZE: 6.83MB TOTAL DOCUMENTS: 171 INDEXES TOTAL SIZE: 36KB

[Find](#) [Indexes](#) [Schema Anti-Patterns](#) 0 [Aggregation](#) [Search Indexes](#) ●

[INSERT DOCUMENT](#)

[FILTER](#) { field: 'value' } [▶ OPTIONS](#) [Apply](#) [Reset](#)

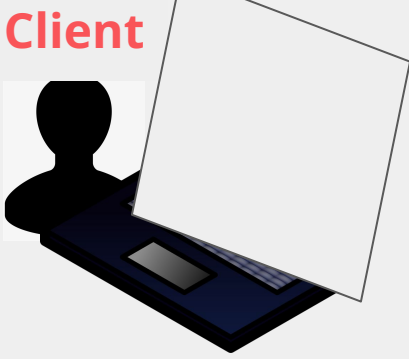
QUERY RESULTS 1-20 OF MANY

```
_id: ObjectId("61d4bf2d589c920024d001ae")
creator_name: "ANONYMOUS"
parent: "61d4b59e589c920024d001ac"
content: "hi this is a comment"
__v: 0
```

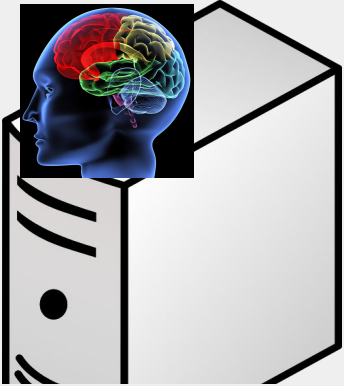
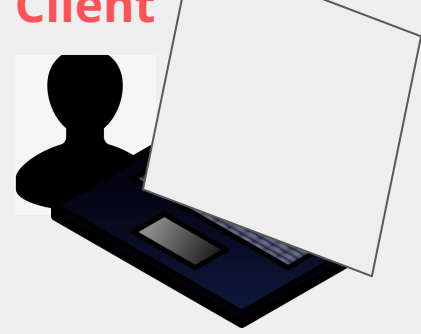
```
_id: ObjectId("61d4c321589c920024d001af")
creator_name: "ANONYMOUS"
parent: "61d4b59e589c920024d001ac"
content: "this is another comment :0"
__v: 0
```

```
_id: ObjectId("61d5ee8536abad0024097534")
creator_name: "ANONYMOUS"
parent: "61d4b59e589c920024d001ac"
content: ":0"
__v: 0
```

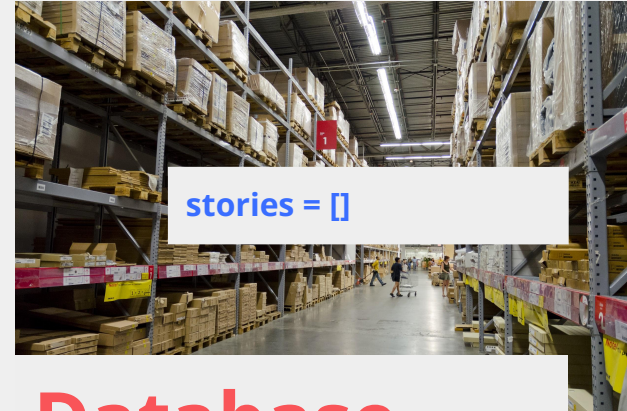
Client



Client



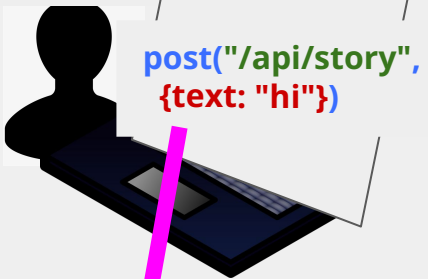
Server



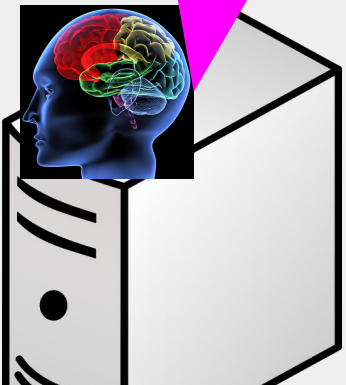
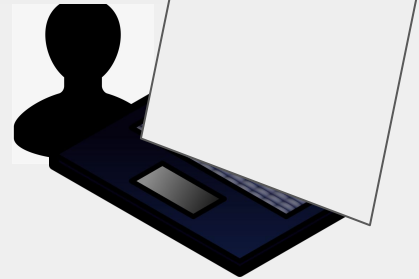
```
stories = []
```

Database

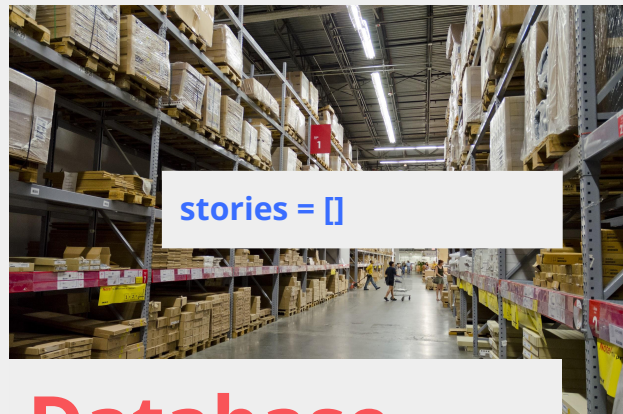
Client



Client

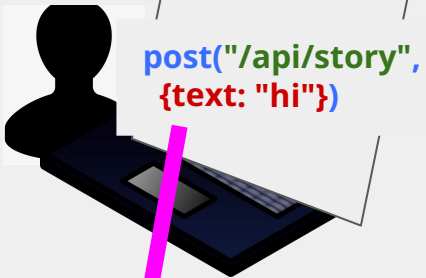


Server

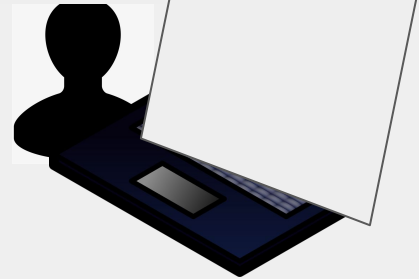


Database

Client

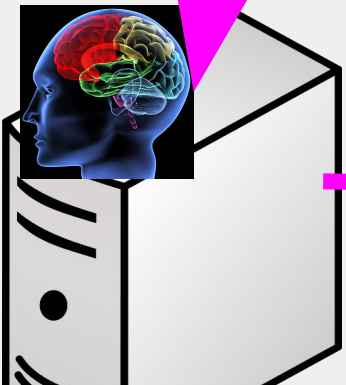


Client



add `{_id: 5,`
 `content: "hi"}`

to the stories collection in
the database

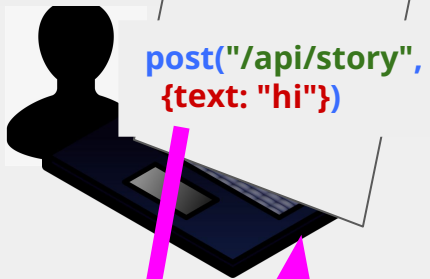


Server

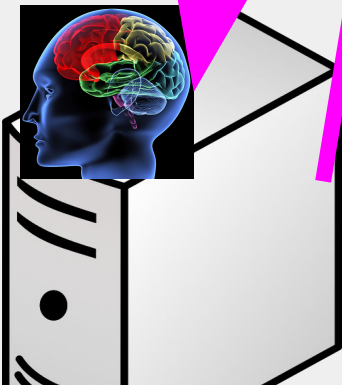


Database

Client

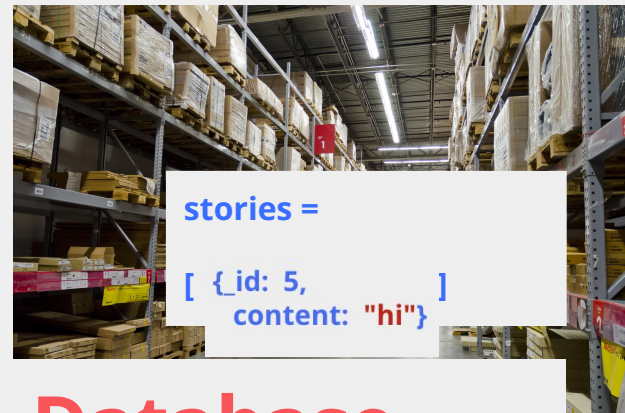
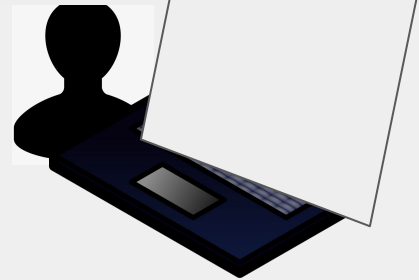


done!



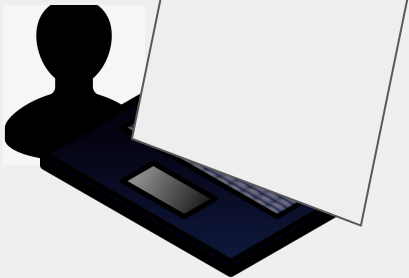
Server

Client



Database

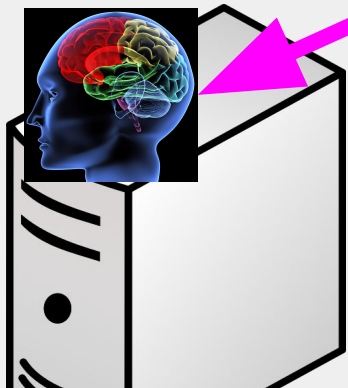
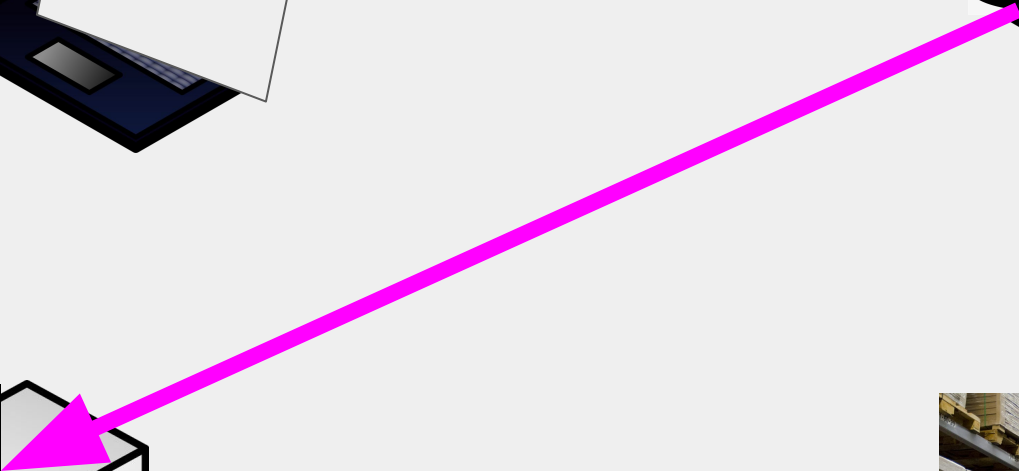
Client



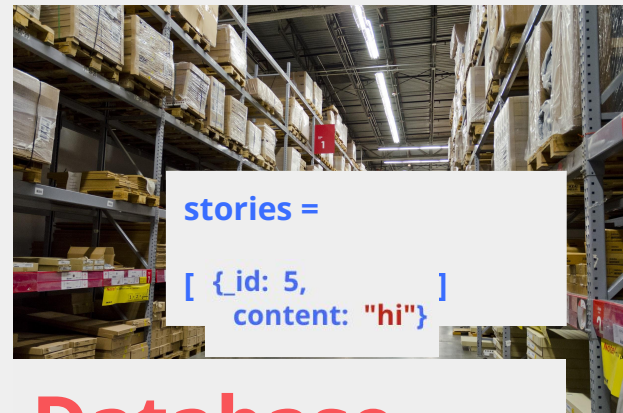
Client



`get("/api/stories")`



Server

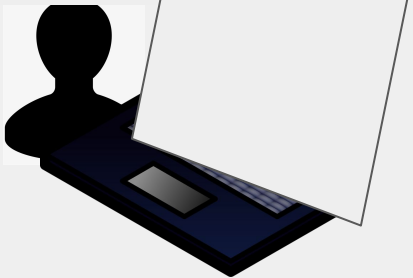


`stories =`

```
[ {id: 5,  
  content: "hi"} ]
```

Database

Client

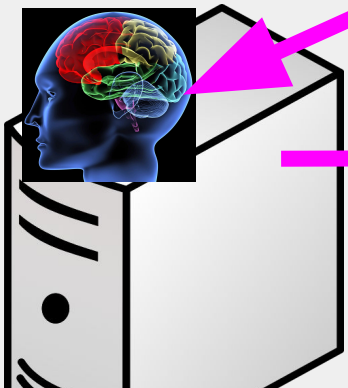


Client



find all the stories for me pls!

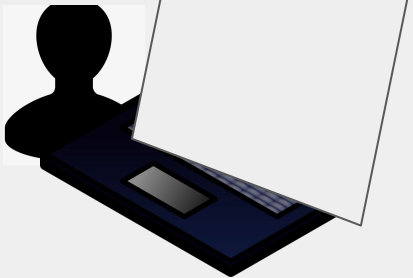
Server



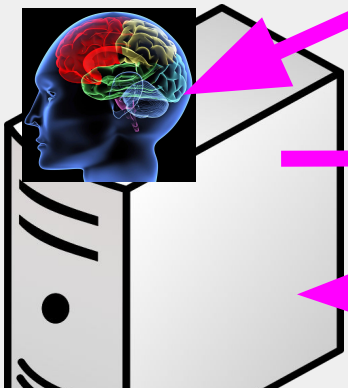
Database



Client



Client



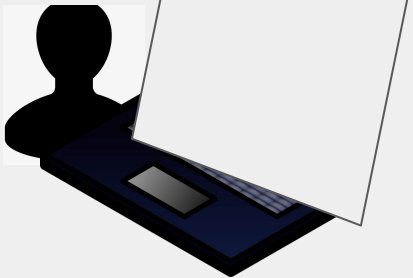
Server



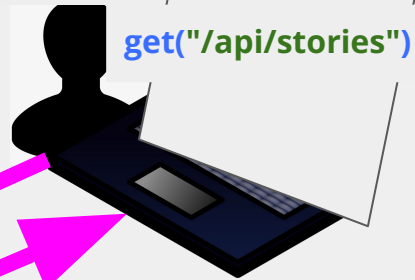
Database

[{id: 5,
content: "hi"}]

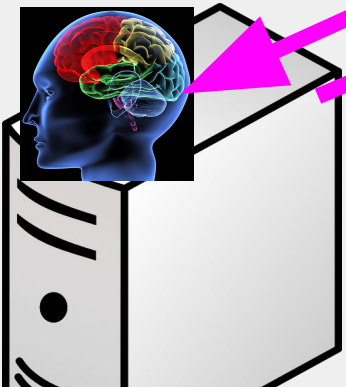
Client



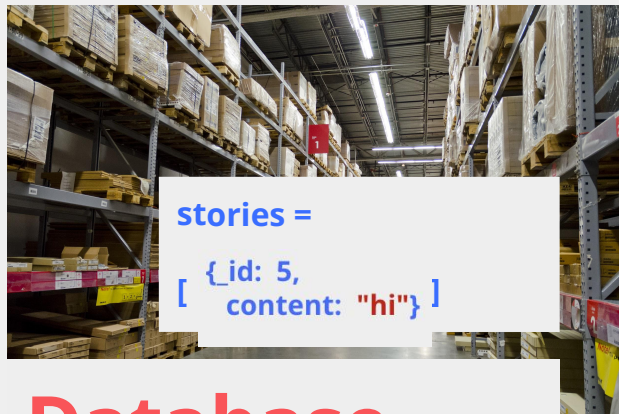
Client



[{id: 5,
content: "hi"}]



Server



Database

Summary

- Rather than store our data on the server... or in a text file...
- We can use a **database** to store our data
- We will use MongoDB as our database
- We will run MongoDB in the cloud, using Atlas

Later today: **Writing code with MongoDB**

A sneak peak...

Schemas

```
//define a story schema for the database
const StorySchema = new mongoose.Schema({
  creator_name: String,
  content: String,
});
```

```
//define a comment schema for the database
const CommentSchema = new mongoose.Schema({
  creator_name: String,
  parent: String, // links to the _id of a pa
  content: String,
});
```


A sneak peak... Retrieving Data from MongoDB

```
router.get("/stories", (req, res) => {  
  💡 // empty selector means get all documents  
  Story.find({}).then((stories) => {  
    res.send(stories)  
  });  
});
```



Gets all the stories from MongoDB

A sneak peak... Adding Data to MongoDB

```
router.post("/story", (req, res) => {  
  const newStory = new Story({  
    creator_name: MY_NAME,  
    content: req.body.content,  
  });  
    
  newStory.save();  
});
```

Saves newStory to MongoDB

Good luck in Workshop 6! Ask questions if you're lost

- Ask questions on the questions doc (weblab.to/questions)
- If stuck during class, add yourself to help queue (weblab.to/q)
- If stuck outside of class, ask on Piazza
 - or stay after 3pm today
 - extra office hours at weblab.to/q

Average Response Time:

1 min

Good luck in Workshop 6! Ask questions if you're lost

- Ask questions on the questions doc (weblab.to/questions)
- If stuck during class, add yourself to help queue (weblab.to/q)
- If stuck outside of class, ask on Piazza
 - or stay after 3pm today
 - extra office hours at weblab.to/q

Average Response Time:

1 min

- **You don't need to know what you're confused about**
 - just ask for help

Have fun at lunch

Come back at 12:55!