CS 337 Passwords

Benjamin Dicken

Announcements

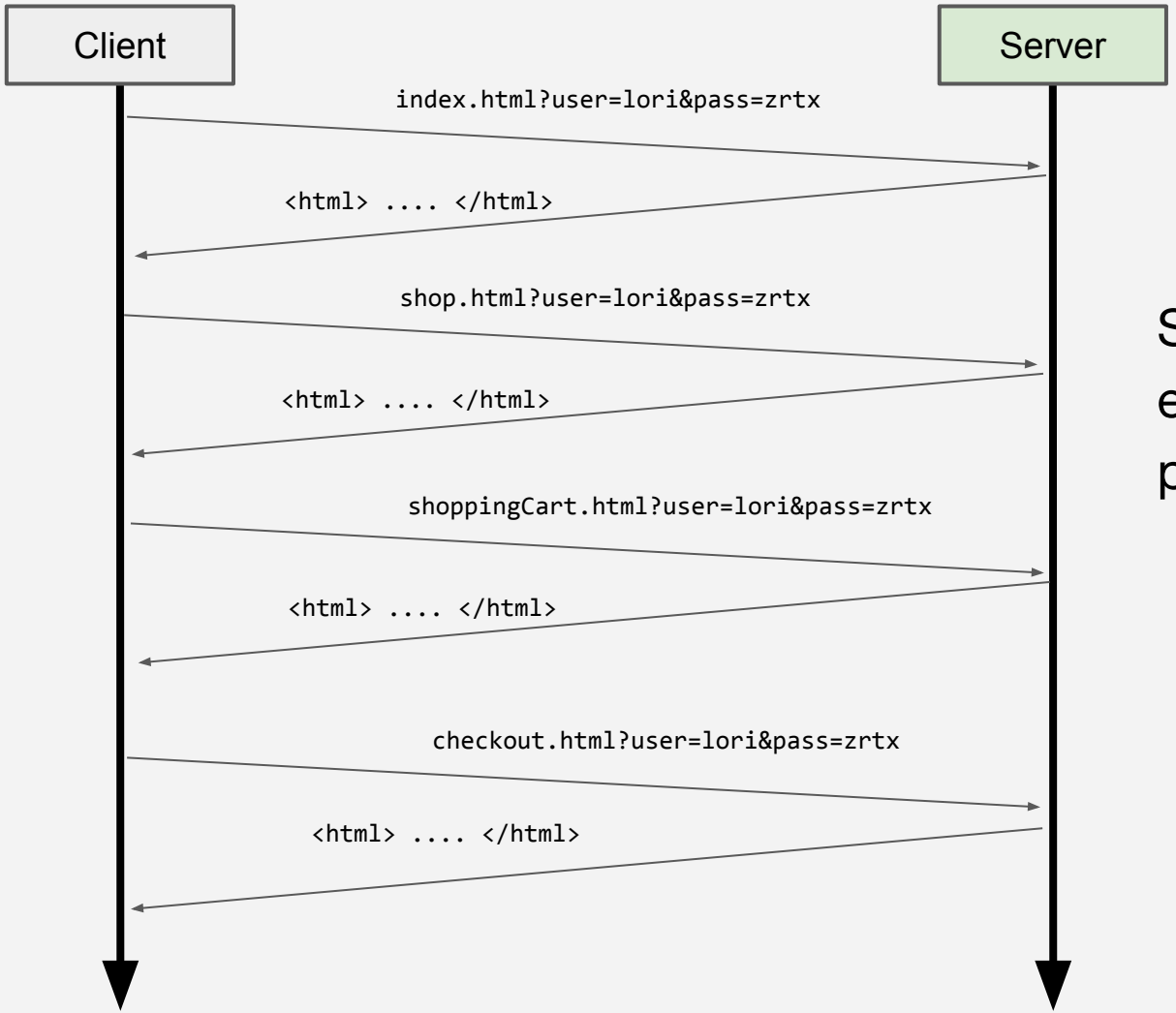
- Group Submit
- Exam 2 next week
  - Cumulative
  - Includes all subjects from before exam 1 as well as more recent topics such as Node, Express, AJAX, DBMSs, MongoDB, Mongoose, etc.
  - 20-30 short answer questions, 3-5 longer questions.
    - The longer questions could have multiple parts
  - Topic cutoff end of this week

User Schema for Database

```
var UserSchema = new Schema({
  username: String,
  password: String,
  email: String,
  phone: String,
  . . . .
});
var User = mongoose.model('User', UserSchema );
```

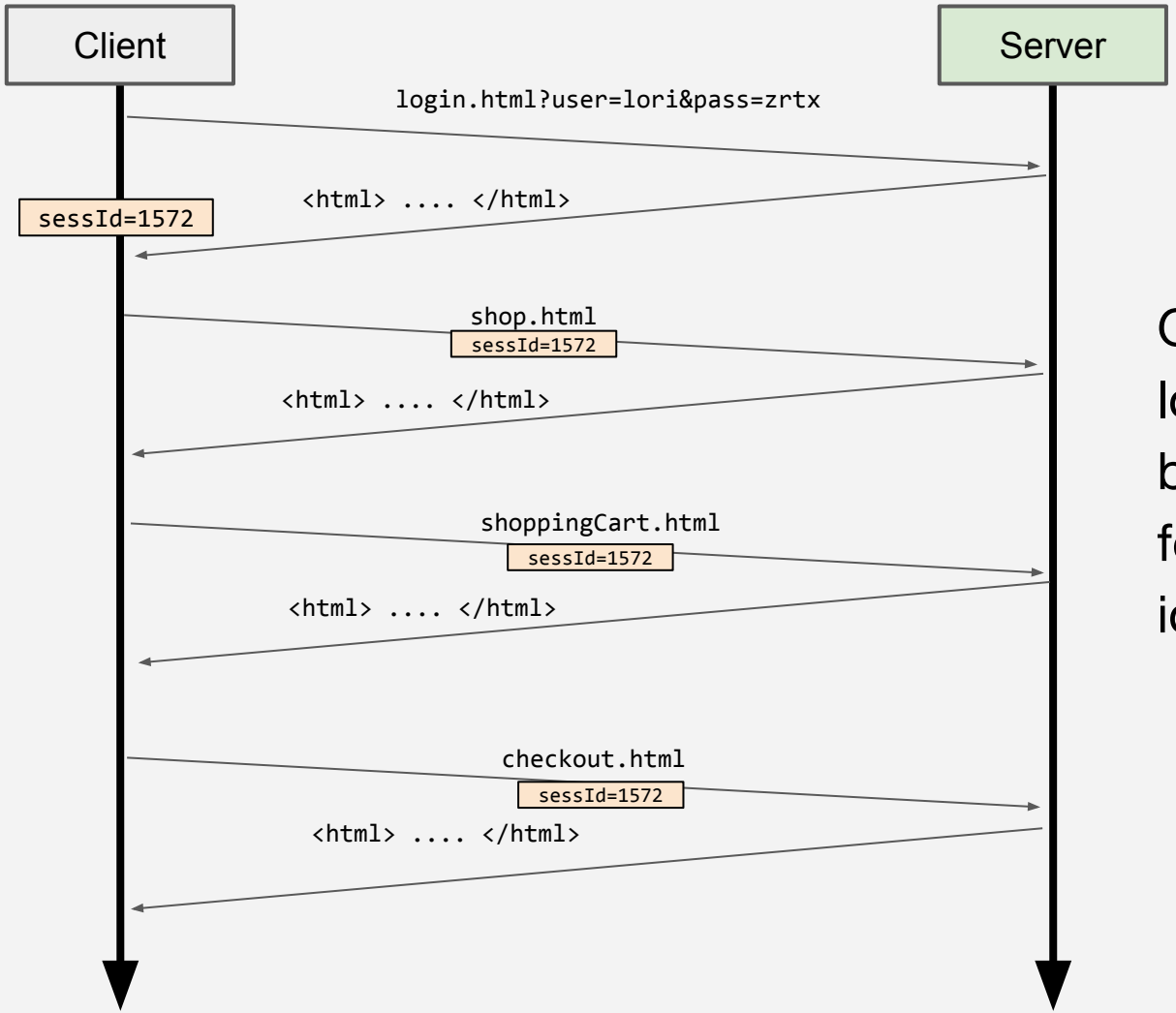
What else is needed?

1. Create new accounts
2. login functionality (check if username and password matches one in our database)
3. remember that a user has already logged in (cookies)
4. Security (salting, hashing, etc)



Without Cookies

Send along credentials each time new restricted page is loaded



With Cookies

Given sessionID upon login, continue to send back to server on follow-up requests to identify

Storing passwords

- Is storing passwords as text in a database secure?
- What if the web app was meant to handle sensitive information, such as medical records or financial info?
- If it were up to you, how would you change the structure of the server / database to store passwords and log users in more securely?

Avoid storing plaintext password

- Rule of thumb: never store (save to hard drive) a user’s password in plain text on your server
- Use a hashing function to store a hash instead

Hash function

- A function that can be used to map data of arbitrary size to a value of fixed size

`"password"`	→ hash function	→ `"lp31"`
`"Abc123z"`	→ hash function	→ `"z1ey"`
`"dfh83hqkjbsdoi234a"`	→ hash function	→ `"xrt7"`

Cryptographic Hash function

- A hash function that has some additional properties, such as:
  - Is fast
  - Is a one-way operation
  - Similar inputs should not give similar outputs

`"password"` → hash function → `"id6qwfi37fdiuyf"`
`"passw0rd"` → hash function → `"zq02odmnccdyg01"`
`"passwords"` → hash function → `"mncb8werh763rfs"`

Salting and Hashing

- Add extra, random data to a password
- Avoids having two people with the same password produce the same hash

`"password9384"` → hash function → `"cs763req65esdtr"`
`"password1723"` → hash function → `"128ydv7qt38q728"`
`"password2301"` → hash function → `"q2sqwa32eaasd2q"`

Salting and Hashing

- When a user goes to create an account:

1. Username and password get sent to server
2. Generate a salt
3. Concatenate the password + salt
4. Hash the password + salt
5. Save the salt and hash in database

Salting and Hashing

- When a user goes to login:

1. Username and password get sent to server
2. Find user with matching username
3. Concatenate the password + salt
4. Hash the password + salt
5. See if the hash matches the user hash

How would you do this?

- Can you come up with an algorithm to make a 1-way hash?
- How would you go about it?