SER 321 A Session

SI Session

Wednesday, September 6th 2023

6:00 - 7:00 pm MST

Agenda

{

What is SI?

About Me

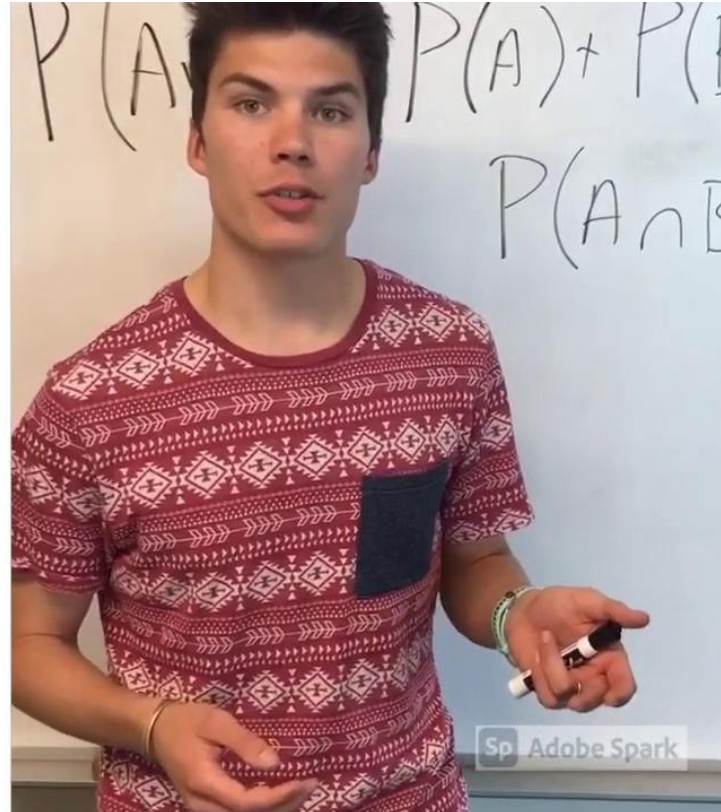Session Times & Expectations

Client & Server Review
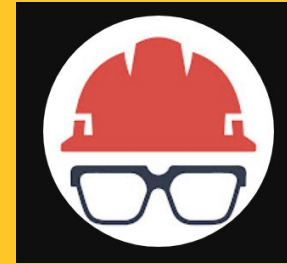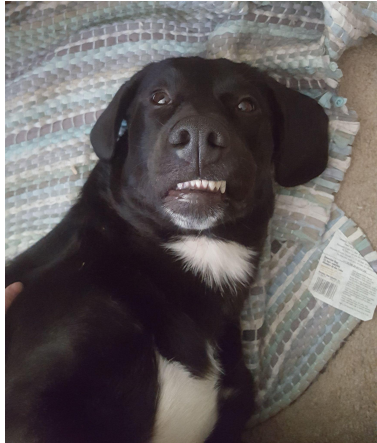
Assignment Tips

What is SI?

About Me
Katie

- Born and raised near Chicago, Illinois
  - Moved around within the Northern Suburbs
  - Currently in Grayslake
- Work at a family-owned Italian restaurant

- Received AS from my local Community College
- Transferred to ASU at the start of 2020
- Majoring in Software Engineering
- Runs in the family
  - Father is a Software Architect
  - Mother is a Quality Assurance Analyst

About Me
Katie

- Hobbies:
  - Gaming
  - Football (Go Bears!)
  - Reading
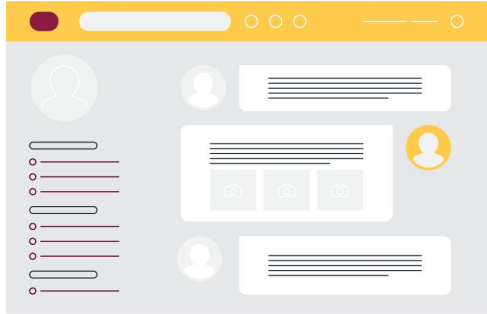 - Learning in general
  - My fur babies 😊

SI Session Expectations

Thanks for coming to the **SER 321** SI session. We have a packed agenda and we are going to try to get through as many of our planned example problems as possible. This session will be recorded and shared with others.

- If after this you want to see additional examples, please visit the drop-in tutoring center.
- We will post the link in the chat now and at the end of the session.
- Please keep in mind we are recording this session and it will be made available for you to review 24-48 hours after this session concludes.
- Finally, please be respectful to each other during the session.
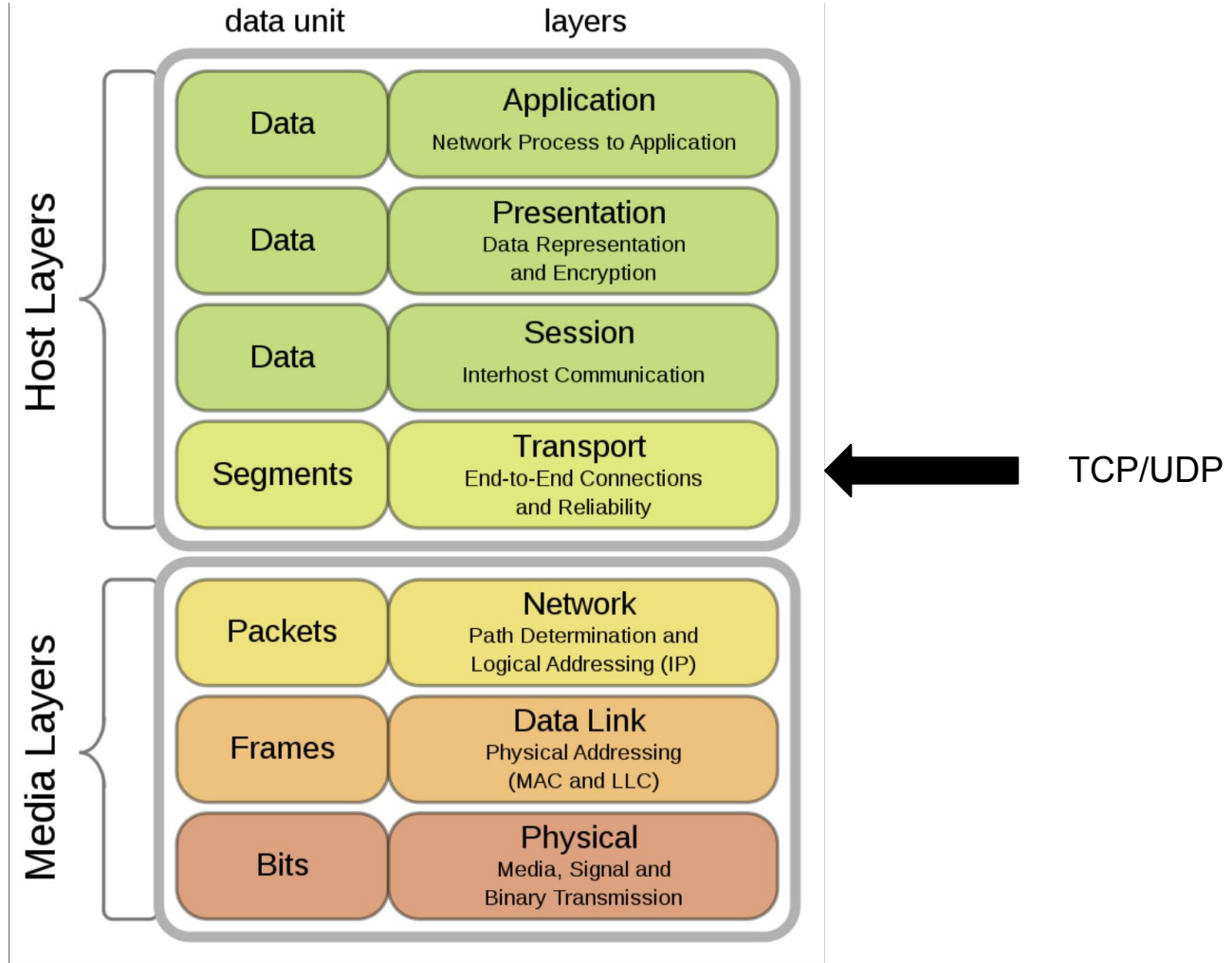
Interact with us:

Zoom Features

Zoom Chat

- Use the chat feature to interact with the presenter and respond to presenter's questions.
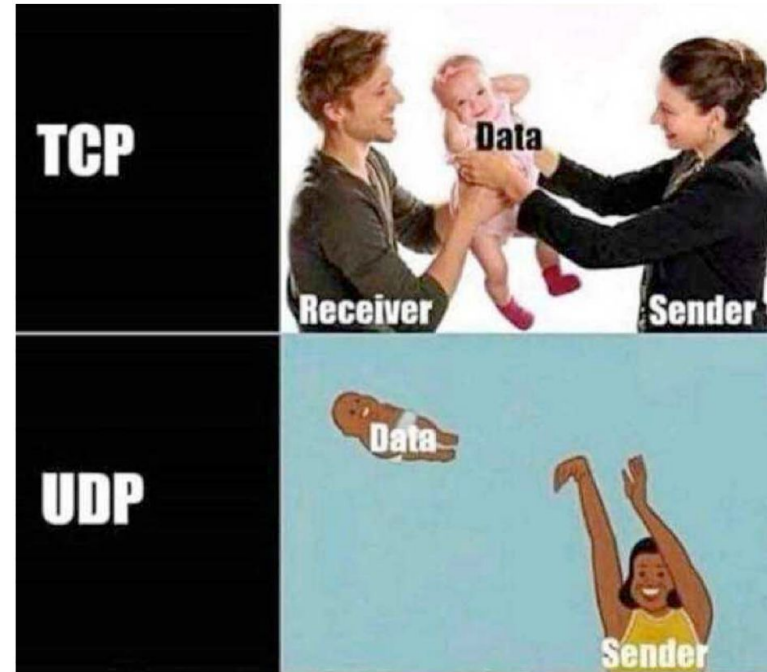- Annotations are encouraged.

SER 321:
OSI Model

data unit	layers
Host Layers	
Data	**Application** — Network Process to Application
Data	**Presentation** — Data Representation and Encryption
Data	**Session** — Interhost Communication
Segments	**Transport** — End-to-End Connections and Reliability ← TCP/UDP
Media Layers	
Packets	**Network** — Path Determination and Logical Addressing (IP)
Frames	**Data Link** — Physical Addressing (MAC and LLC)
Bits	**Physical** — Media, Signal and Binary Transmission

- TCP vs. UDP
  - Connection oriented vs. connectionless
  - Synchronous vs. Asynchronous
  - Stateful vs. Stateless
  - Data Stream vs. Datagram

- TCP is mailing a package with tracking

- UDP is tossing notes into your neighbor's backyard

SER 321:
Socket Programming

- Server uses a blocking wait to listen for clients that want to talk with it
- Client is responsible for knowing the specifics of the server
- Sockets use *streams* for input and output data
- Socket Properties
  - Location (IP address, or DNS)
  - Connection (TCP/UDP)
  - Message Format (header & payload)

- Two Main Models
  - PULL Model
    - polling/waiting model where client makes request and the server responds
    - Client *pulls* the response down from the server
    - Server's response either handles request or rejects it
  - PUSH Model
    - Server will send responses without a client request
    - Server *pushes* responses down to client

SER 321:
Socket Programming - Client vs. Server

Client or Server?

```java
sock = new Socket(host, port: 8888);
```

```java
ServerSocket serv = new ServerSocket(port: 8888);
```

SER 321:
What else goes in the Server?

Using AdvancedCustomProtocol code from the Examples Repo

1. Blocking Wait

2. Input Stream

3. Output Stream

4. Code to handle request

5. Closure of the Client specific socket

```java
sock = serv.accept();
```

```java
InputStream in = sock.getInputStream();
OutputStream out = sock.getOutputStream();
```

```java
switch (choice) {
case (1):
  returnMessage = joke();
  break;
```

```java
if (sock != null) {
  sock.close();
}
```

SER 321:
What else goes in the Client?

Using AdvancedCustomProtocol code from the Examples Repo

1. Create Socket connection

2. Input Stream

3. Output Stream

4. Code to create and send the request

5. Closure of the socket and streams

```java
sock = new Socket( host: "localhost", port: 9000);
```

```java
InputStream in = sock.getInputStream();
OutputStream out = sock.getOutputStream();
```

```java
switch (choice) {
    case (1):
        request = joke();
        break;
    case (2):
        request = quote();
        break;
```

```java
sock.close();
out.close();
in.close();
System.exit( status: 0);
```

SER 321:
Socket Programming Suggestions

- If you design the protocol, *do that first!*
- Create methods for building your requests and responses
  - Think of Base.java in SimpleCustomProtocol
  - Think of NetworkUtils.java in AdvancedCustomProtocol
- Map out a general idea of the program flow before you begin coding
- Create skeletons or placeholders in places you want to come back to later
  - Allows you to get the general structure of your code set up ASAP
  - Print something like "Not implemented yet"

SER 321:

General Assignment Tips

- ○ Start assignments early 🌟
- ○ Play around with the examples if you are stuck
- ○ Complete *one* requirement at a time
- ○ **Use print statements to check your transmissions!** 🌟
- ○ Keep a copy of the protocol handy while coding
- ○ Commit commit commit!
- ○ If you're struggling with a requirement, skip it!

Questions?

Survey:
https://bit.ly/asn_survey

More Questions?
Check out our other resources!

tutoring.asu.edu

1 –   Go to Zoom

Need help using Zoom?

2 –   View the tutoring schedule

View digital resources

1. **Click on 'Go to Zoom' to log onto our Online Tutoring Center.**
2. **Click on 'View the tutoring schedule' to see when tutors are available for specific courses.**

More Questions?
Check out our other resources!

tutoring.asu.edu/online-study-hub

Don't forget to check out
the Online Study Hub
for additional resources!