SER 321 C Session

SI Session

Monday, July 1st 2024

6:00 pm - 7:00 pm MST

Agenda

Middleware

What is it?

Why do we care?

Assignment 6 Structure

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.
  - tutoring.asu.edu
- 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
Middleware

We have been:

- Serializing Messages
- Sending Messages
- Parsing Messages
- Handle Messages

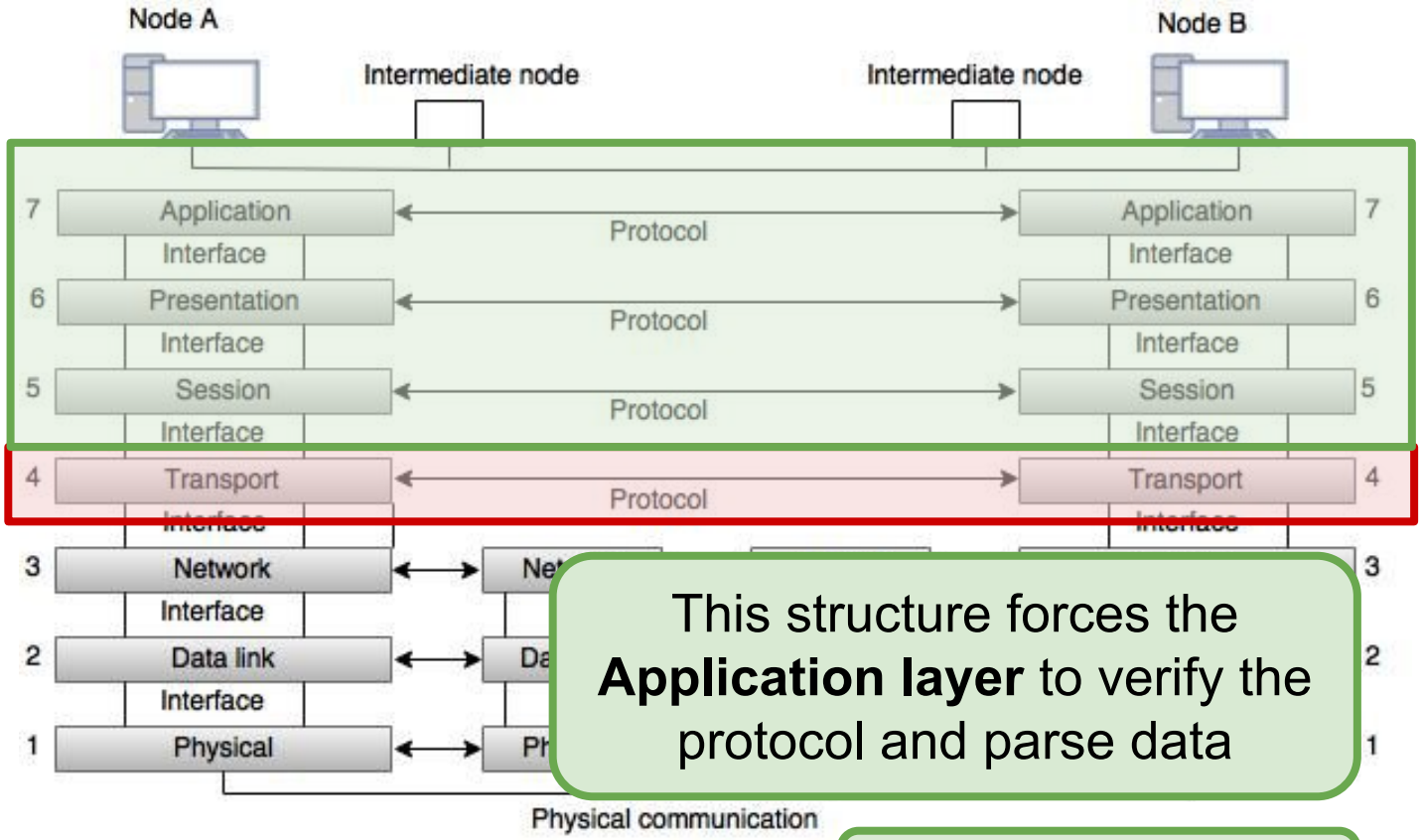


Fig: OSI Model

This structure forces the **Application layer** to verify the protocol and parse data

Not really its job…

SER 321
Middleware

With Middleware:

- Serializing Messages
- Sending Messages
- Parsing Messages
- Handle Messages

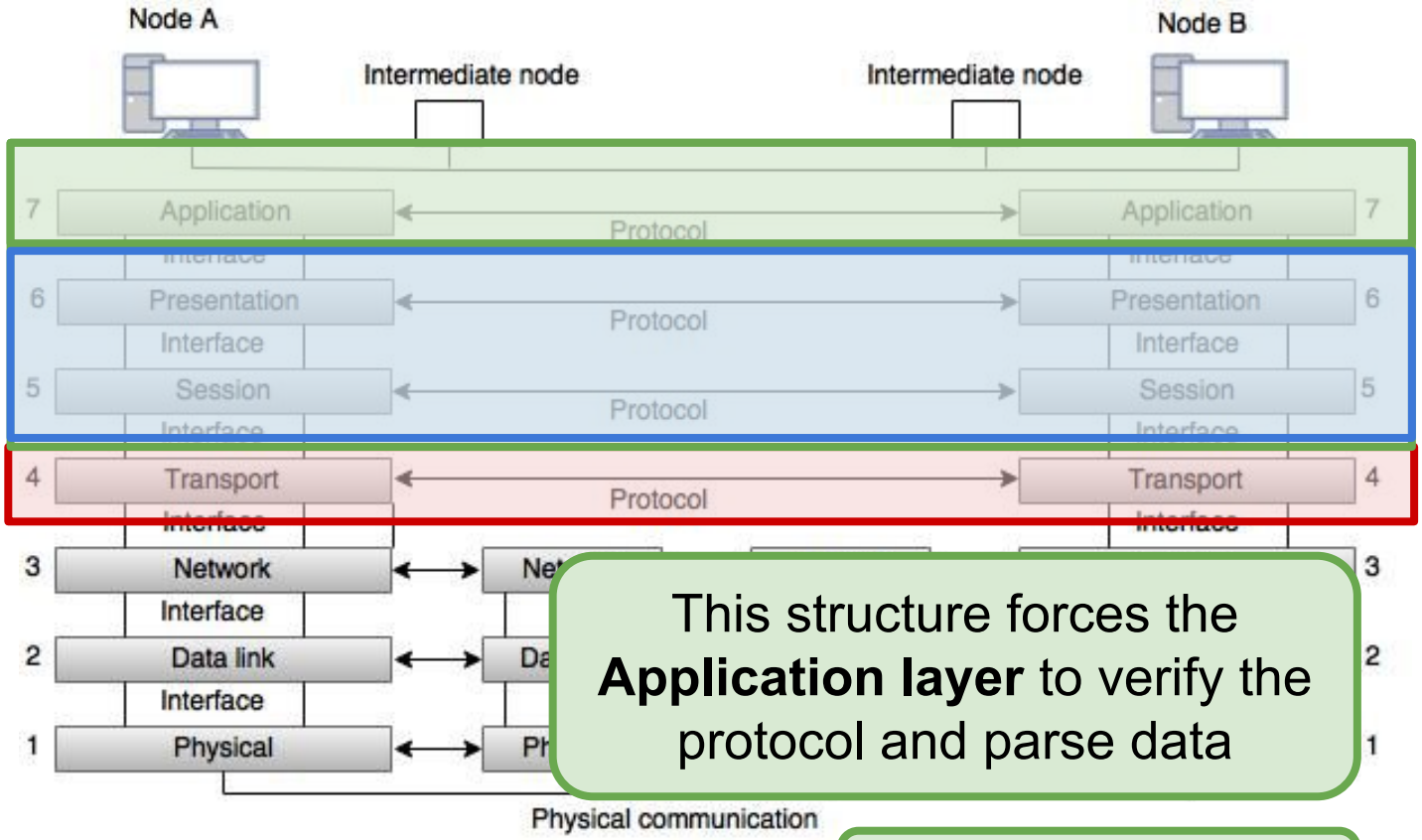


Fig: OSI Model

This structure forces the **Application layer** to verify the protocol and parse data

Not really its job…

SER 321
Middleware

With Middleware:

Serializing Messages

Sending Messages

Parsing Messages

Handle Messages

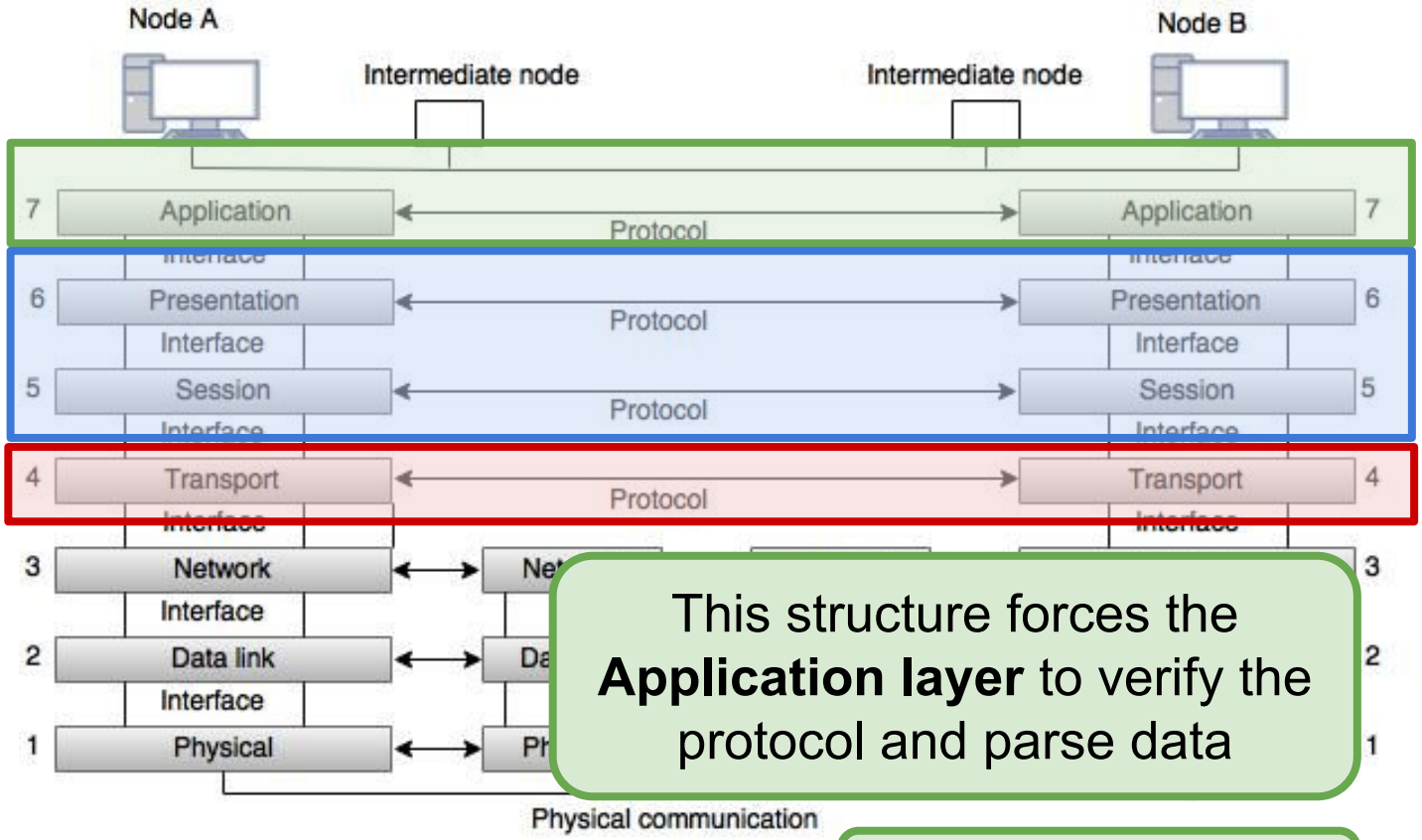


Fig: OSI Model

This structure forces the **Application layer** to verify the protocol and parse data

Not really its job…

SER 321
Middleware

Middleware:

Session Layer Responsibilities:

Authentication

Authorization

Session Management

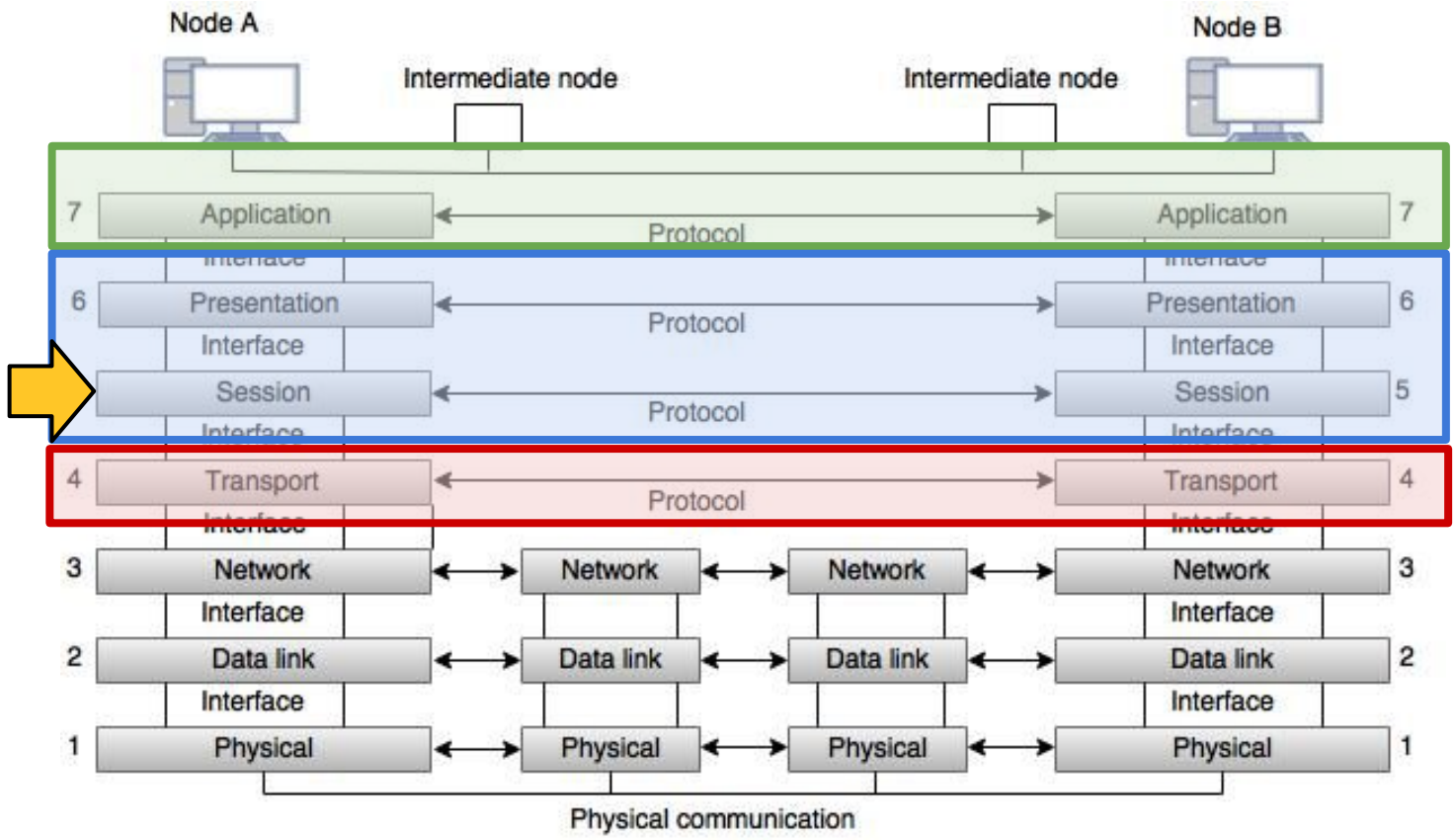


Fig: OSI Model

SER 321
Middleware

Middleware:

Presentation Layer Responsibilities:

Translation

Compression

Encryption

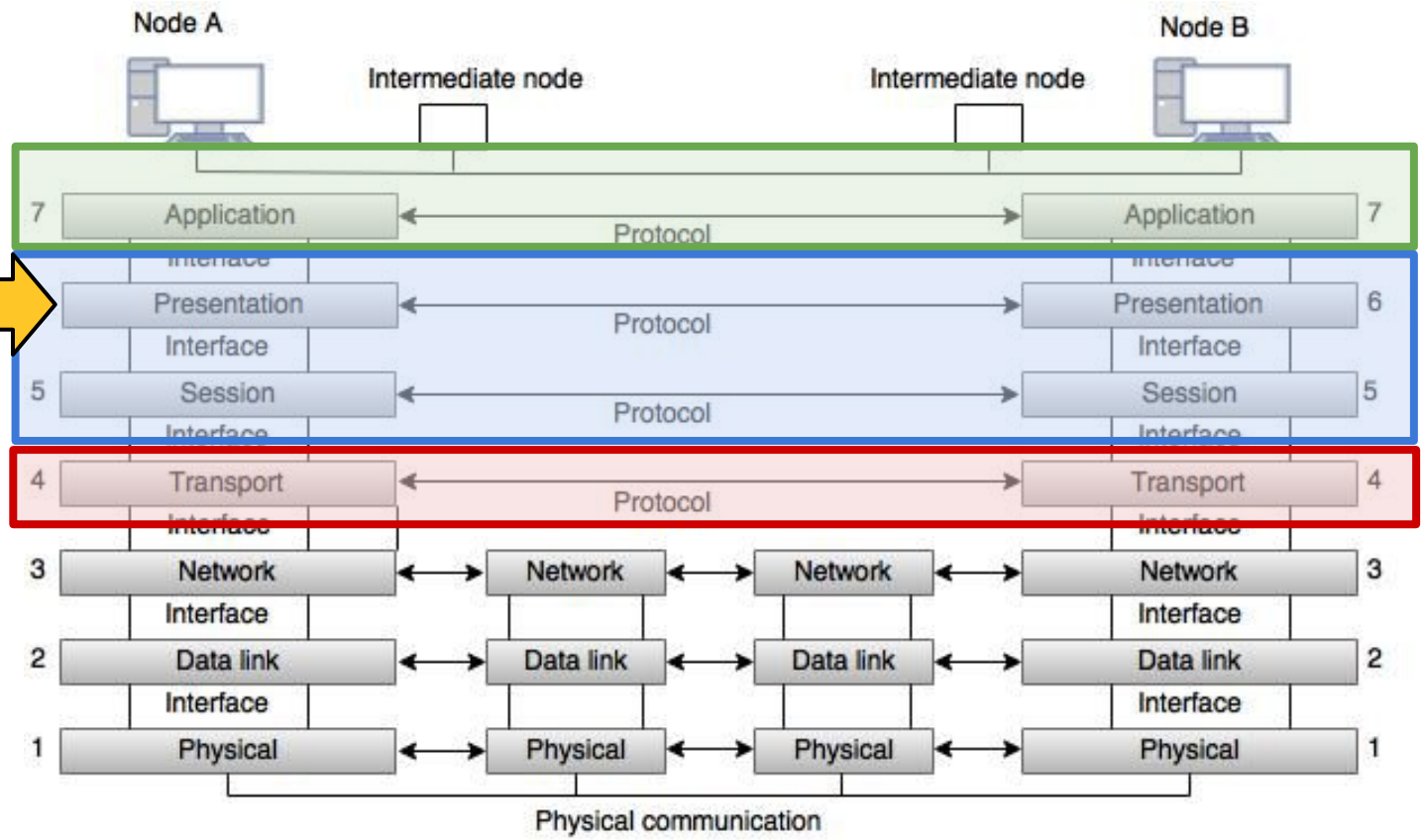


Fig: OSI Model

SER 321
Middleware

Examples?

Message Oriented Middleware (MOM)

Web Frameworks

Remote Procedure Calls (RPC)

App. Programming Interface (API)

SER 321
Middleware

Why do we care?

- Agility
- Reusability
- Efficiency
- Cost Effectiveness
- Portability

SER 321
Middleware

Why do we care?

Sort of like publishing a contract

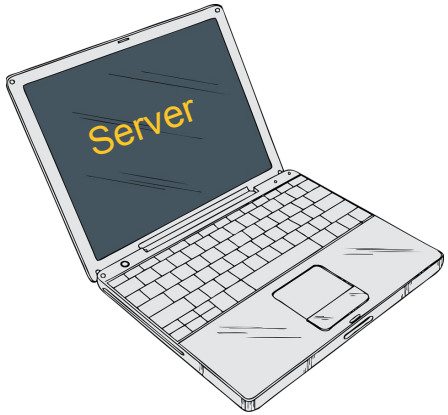
“If you follow these rules, I will handle your request.”

SER 321
Middleware

With Middleware:

Add User:
Name = Katie
Password = password

API



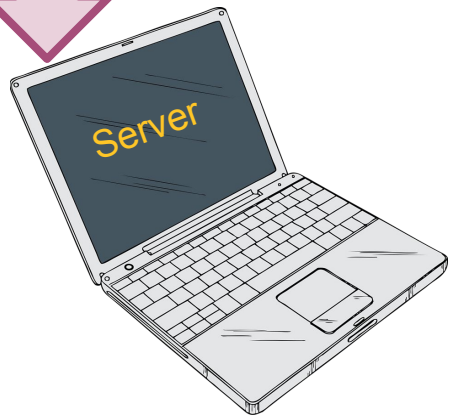
SER 321
Middleware

With Middleware:

Add User:
Name = Katie
Password = password

API


```
{
        “type” : “addUser”,
        “name” : “katie”,
        “password” : “password”
}
```



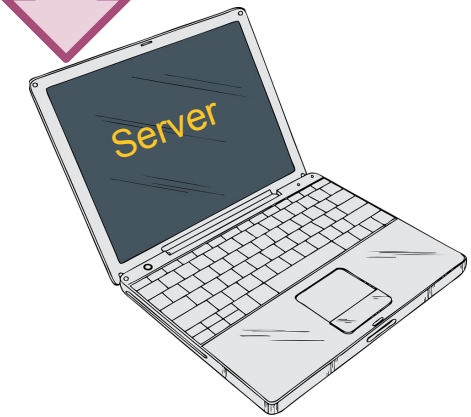
SER 321
Middleware

With Middleware:

Get repositories for a specific user

API

GitHub REST API



Code samples for "List repositories for a user"

Request example

GET /users/{username}/repos

cURL | JavaScript | GitHub CLI

```
curl -L \
  -H "Accept: application/vnd.github+json" \
  -H "Authorization: Bearer <YOUR-TOKEN>" \
  -H "X-GitHub-Api-Version: 2022-11-28" \
  https://api.github.com/users/USERNAME/repos
```

SER 321
Middleware

With Middleware:

Get repositories for a specific user

API

GitHub REST API

https://api.github.com/users/kgrinne3/repos

Code samples for "List reposito

Request example

GET /users/{username}/repos

cURL JavaScript GitHub CL

```
curl -L \
  -H "Accept: application/vnd.gi
  -H "Authorization: Bearer <YOU
  -Api-Version: 2022
  github.com/users/U
```


```
[
  {
    "id": 550568457,
    "node_id": "R_kgDOINECCQ",
    "name": "assign1git",
    "full_name": "kgrinne3/assign1git",
    "private": false,
    "owner": {
      "login": "kgrinne3",
      "id": 115493885,
      "node_id": "U_kgDOBuJL_Q",
      "avatar_url": "https://avatars.githubusercontent.com/u/115493885?v=4",
      "gravatar_id": "",
      "url": "https://api.github.com/users/kgrinne3",
      "html_url": "https://github.com/kgrinne3",
      "followers_url": "https://api.github.com/users/kgrinne3/followers",
      "following_url": "https://api.github.com/users/kgrinne3/following{/other_user}",
      "gists_url": "https://api.github.com/users/kgrinne3/gists{/gist_id}",
      "starred_url": "https://api.github.com/users/kgrinne3/starred{/owner}{/repo}",
      "subscriptions_url": "https://api.github.com/users/kgrinne3/subscriptions",
      "organizations_url": "https://api.github.com/users/kgrinne3/orgs",
      "repos_url": "https://api.github.com/users/kgrinne3/repos",
      "events_url": "https://api.github.com/users/kgrinne3/events{/privacy}",
      "received_events_url": "https://api.github.com/users/kgrinne3/received_events",
      "type": "User",
      "site_admin": false
    },
    "html_url": "https://github.com/kgrinne3/assign1git",
    "description": "Katie Grinnell",
    "fork": false,
    "url": "https://api.github.com/repos/kgrinne3/assign1git",
    "forks_url": "https://api.github.com/repos/kgrinne3/assign1git/forks",
    "keys_url": "https://api.github.com/repos/kgrinne3/assign1git/keys{/key_id}",
    "collaborators_url": "https://api.github.com/repos/kgrinne3/assign1git/collaborators{/collaborator}",
    "teams_url": "https://api.github.com/repos/kgrinne3/assign1git/teams",
    "hooks_url": "https://api.github.com/repos/kgrinne3/assign1git/hooks",
    "issue_events_url": "https://api.github.com/repos/kgrinne3/assign1git/issues/events{/number}",
    "events_url": "https://api.github.com/repos/kgrinne3/assign1git/events",
    "assignees_url": "https://api.github.com/repos/kgrinne3/assign1git/assignees{/user}",
    "branches_url": "https://api.github.com/repos/kgrinne3/assign1git/branches{/branch}",
    "tags_url": "https://api.github.com/repos/kgrinne3/assign1git/tags",
    "blobs_url": "https://api.github.com/repos/kgrinne3/assign1git/git/blobs{/sha}",
    "git_tags_url": "https://api.github.com/repos/kgrinne3/assign1git/git/tags{/sha}",
```

SER 321
Assign 6

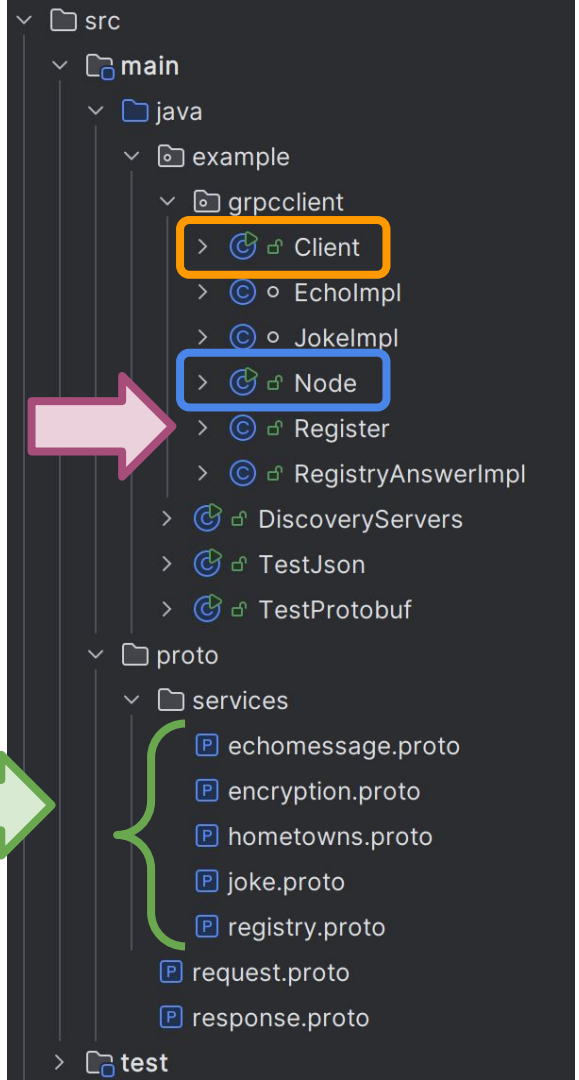
Protocol Buffers!

Client

Node

Registry

Service

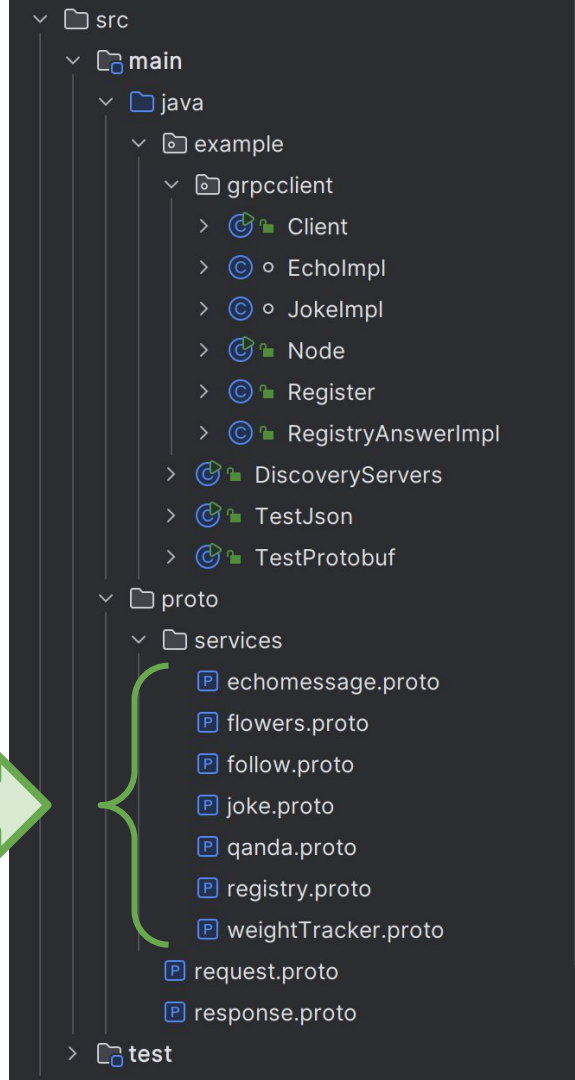


SER 321
Protobuf Review

All nodes and clients have agreed to these contracts

So ***DON'T CHANGE THEM!***

Think of these as a *contract*



SER 321
Protobuf Review

joke.proto

```
syntax = "proto3";

option java_multiple_files = true;
option java_package = "service";
option java_outer_classname = "JokeProto";

package services;

service Joke {
  rpc getJoke (JokeReq) returns (JokeRes) {}
  rpc setJoke (JokeSetReq) returns (JokeSetRes) {}
}

// The request message
message JokeReq {
  int32 number = 1;
```

```
// We are reading how many jokes the clients wants and put them in a list to send back to client
@Override  1 usage
public void getJoke(JokeReq req, StreamObserver<JokeRes> responseObserver) {
    System.out.println("Received from client: " + req.getNumber());
    JokeRes.Builder response = JokeRes.newBuilder();
    for (int i=0; i < req.getNumber(); i++){
        if(!jokes.empty()) {
            // yes, I take the joke out when it was used already,
            // should probably be done differently since this way
            // a joke cannot be told twice even to different clients
            response.addJoke(jokes.pop());
        }
        else {
            // this is more of a hack, better would be to either
            // check the number at the beginnig and say right away
            // if you do not have enough. Or send an error code or
            // similar as well.
            response.addJoke( value: "I am out of jokes...");
            break;
        }
    }
    JokeRes resp = response.build();
    responseObserver.onNext(resp);
    responseObserver.onCompleted();
}
```

```
// We take the joke the user wants to set and put it in our set of jokes
@Override  1 usage
public void setJoke(JokeSetReq req, StreamObserver<JokeSetRes> responseObserver) {

    System.out.println("Received from client: " + req.getJoke());
    JokeSetRes.Builder response = JokeSetRes.newBuilder();
    if (req.getJoke().isEmpty()) { // we do not want to add empty jokes
        response.setOk(false);
    } else {
        jokes.add(req.getJoke());
        response.setOk(true);
    }

    JokeSetRes resp = response.build();
    responseObserver.onNext(resp);
    responseObserver.onCompleted();
}
```

SER 321
Protobuf Review

joke.proto

```
syntax = "proto3";

option java_multiple_files = true;
option java_package = "service";
option java_outer_classname = "JokeProto";

package services;

service Joke {
  rpc getJoke (JokeReq) returns (JokeRes) {}
  rpc setJoke (JokeSetReq) returns (JokeSetRes) {}
}

// The request message
message JokeReq {
  int32 number = 1;
```

How do we use Protobufs again?

Use a ***Builder*** to construct the proto object

Fill with *setters*

Build when done!

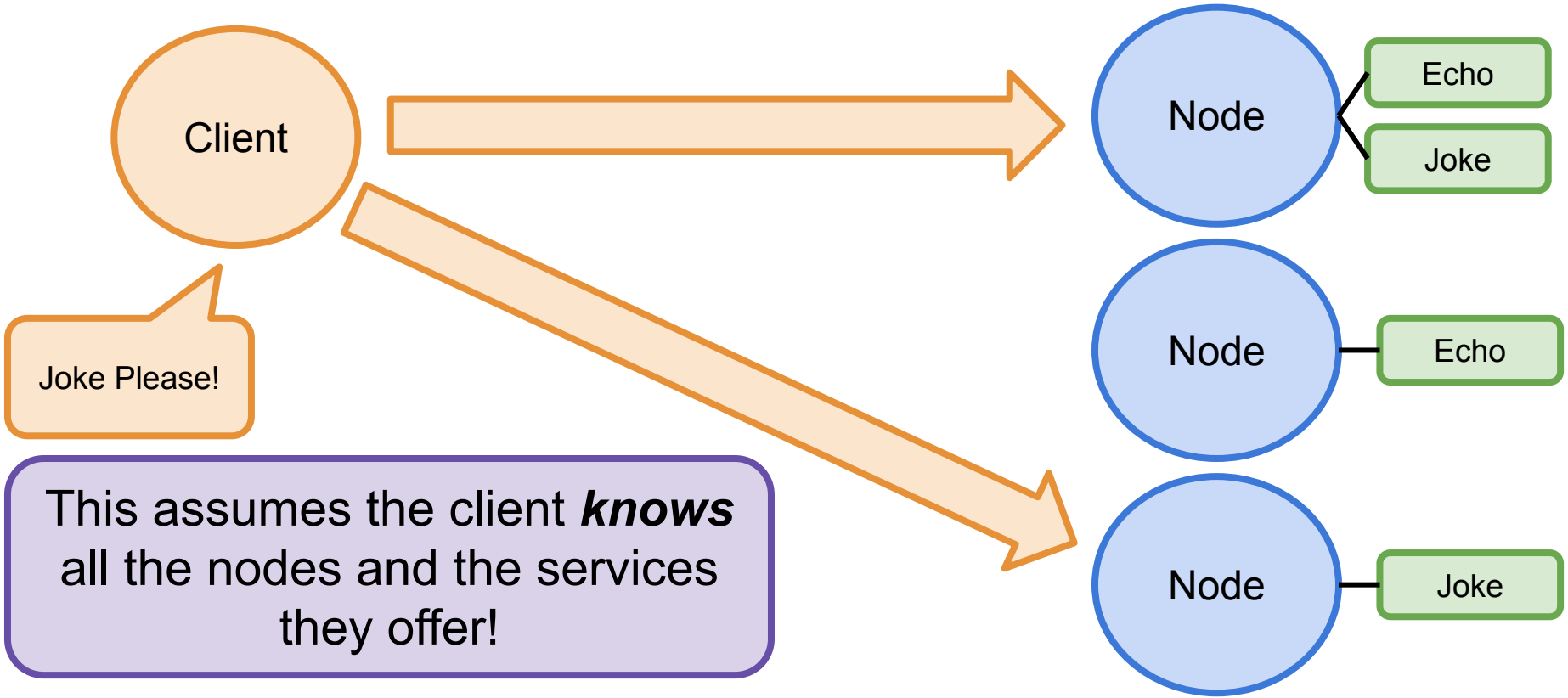
```
// We take the joke the user wants to set and put it in our set of jokes
@Override  1 usage
public void setJoke(JokeSetReq req, StreamObserver<JokeSetRes> responseObserver) {

    System.out.println("Received from client: " + req.getJoke());
    JokeSetRes.Builder response = JokeSetRes.newBuilder();
    if (req.getJoke().isEmpty()) { // we do not want to add empty jokes
        response.setOk(false);
    } else {
        jokes.add(req.getJoke());
        response.setOk(true);
    }

    JokeSetRes resp = response.build();
    responseObserver.onNext(resp);
    responseObserver.onCompleted();
}
```

SER 321
Assign 6

Previously…



This assumes the client ***knows*** all the nodes and the services they offer!

SER 321
Assign 6

With GRPC…

Client

Joke

Registry

Joke Please!

Node — Echo
Node — Joke

Node — Echo

Node — Joke

SER 321
Assign 6

With GRPC…

Client

Joke

Registry

Node

Echo

Joke

Node

Echo

Node

Joke

Joke Please!

Registry directs us to a node that can handle our request!

SER 321
Assign 6

Okay so how do we actually *use* this setup?

```java
public void setJoke(String joke) {
  JokeSetReq request = JokeSetReq.newBuilder()
      .setJoke(joke).build();
  JokeSetRes response;

  try {
    response = blockingStub2.setJoke(request);
    System.out.println(response.getOk());
  } catch (Exception e) {
    System.err.println("RPC failed: " + e);
    return;
  }
}
```

Client.java

```java
Client client = new Client(channel, regChannel);

// call the parrot service on the server
client.askServerToParrot(message);

// ask the user for input how many jokes the user wants
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

// Reading data using readLine
System.out.println("How many jokes would you like?"); // NO ERROR handling of wrong input here.
String num = reader.readLine();

// calling the joked service from the server with num from user input
client.askForJokes(Integer.valueOf(num));

// adding a joke to the server
client.setJoke("I made a pencil with two erasers. It was pointless.");

// showing 6 joked
client.askForJokes(Integer.valueOf(6));
```

Client.java (Main)

```java
// W
@Ove
public void setJoke(JokeSetReq req, StreamObserver<JokeSetRes> responseObserver) {

    System.out.println("Received from client: " + req.getJoke());
    JokeSetRes.Builder response = JokeSetRes.newBuilder();
    if (req.getJoke().isEmpty()) { // we do not want to add empty jokes
        response.setOk(false);
    } else {
        jokes.add(req.getJoke());
        response.setOk(true);
    }

    JokeSetRes resp = response.build();
    responseObserver.onNext(resp);
    responseObserver.onCompleted();
}
```

JokeImpl.java

Looking at **SetJoke**

SER 321
Assign 6

Okay so how do we actually *use* this setup?

```
public void setJoke(String joke) {
  JokeSetReq request = JokeSetReq.newBuilder()
      .setJoke(joke).build();
  JokeSetRes response;

  try {
    response = blockingStub2.setJoke(request);
    System.out.println(response.getOk());
  } catch (Exception e) {
    System.err.println("RPC failed: " + e);
    return;
  }
}
```

Client.java

Client provides the info

Client creates request

```
Client client = new Client(channel, regChannel);
```

```
// Implement the joke service. It has two sevices getJokes and setJoke
class JokeImpl extends JokeGrpc.JokeImplBase {  1 usage

    // having a global set of jokes
    Stack<String> jokes = new Stack<~>();  7 usages

    public JokeImpl(){  1 usage
        super();
        // copying some dad jokes
        jokes.add("How do you get a squirrel to like you? Act like a nut.");
        jokes.add("I don't trust stairs. They're always up to something.");
        jokes.add("What do you call someone with no body and no nose? Nobody knows.");
        jokes.add("Did you hear the rumor about butter? Well, I'm not going to spread it!");

    }
```

JokeImpl.java

```
client.askForJokes(Integer.valueOf(6));
```

```
@Ove
public void setJoke(JokeSetReq req, StreamObserver<JokeSetRes> responseObserver) {

    System.out.println("Received from client: " + req.getJoke());
    JokeSetRes.Builder response = JokeSetRes.newBuilder();
    if (req.getJoke().isEmpty()) { // we do not want to add empty jokes
        response.setOk(false);
    } else {
        jokes.add(req.getJoke());
        response.setOk(true);
    }

    JokeSetRes resp = response.build();
    responseObserver.onNext(resp);
    responseObserver.onCompleted();
}
```

JokeImpl.java

Everything else we have had to do is handled in the Implementation Class!

SER 321
Scratch Space

What does that imply for the system?

```java
public void setJoke(String joke) {
  JokeSetReq request = JokeSetReq.newBuilder()
      .setJoke(joke).build();
  JokeSetRes response;

  try {
    response = blockingStub2.setJoke(request);
    Sys
  } cat
    Sys
    ret
  }
}
```

Client.java

Client Class acts like a middleman!

```java
Client client = new Client(channel, regChannel);

// call the parrot service on the server
client.askServerToParrot(message);

// ask the user for input how many jokes the user wants
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

// Reading data using readLine
System.out.println("How many jokes would you like?"); // NO ERROR handling of wrong input here.
String num = reader.readLine();

// calling the joked service from the server with num from user input
client.askForJokes(Integer.valueOf(num));

// adding a joke to the server
client.setJoke("I made a pencil with two erasers. It was pointless.");

// showing 6 joked
client.askForJokes(Integer.valueOf(6));
```

Client.java (Main)

```java
// W
@Ove
public void setJoke(JokeSetReq req, StreamObserver<JokeSetRes> responseObserver) {

    System.out.println("Received from client: " + req.getJoke());
    JokeSetRes.Builder response = JokeSetRes.newBuilder();
    if (req.getJoke().isEmpty()) { // we do not want to add empty jokes
        response.setOk(false);
    } else {
        jokes.add(req.getJoke());
        response.setOk(true);
    }

    JokeS
    respo
    respo
}
```

JokeImpl.java

Implementations need to be robust and thorough!

Everything else we have had to do is handled in the Implementation Class!

SER 321

Scratch Space

Questions?

Survey:

http://bit.ly/ASN2324

28

Upcoming Events

SI Sessions:

- Wednesday, July 3rd at 6:00 pm MST - ***2hr Review Session***
- Sunday, July 7th at 6:00 pm MST - ***Final Session***

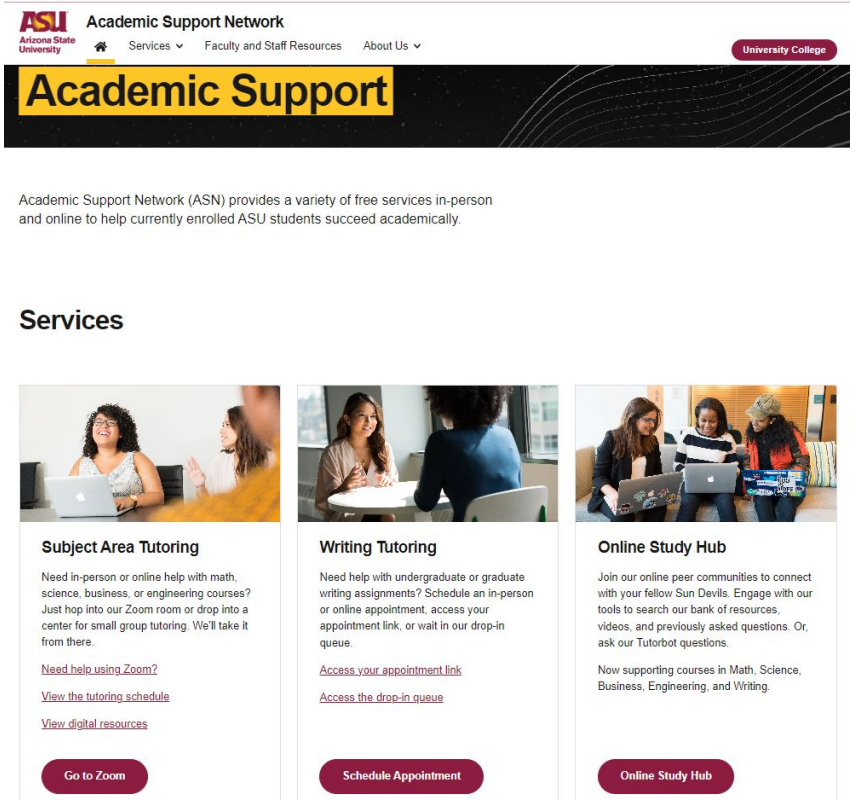
Review Sessions:

- Review Session - **Wednesday**, July 3rd at 6:00 pm MST (2 hr Session)
- Q&A Session - Sunday, July 7th at 6:00 pm MST (Final Session)

More Questions?

Check out our other resources!

tutoring.asu.edu

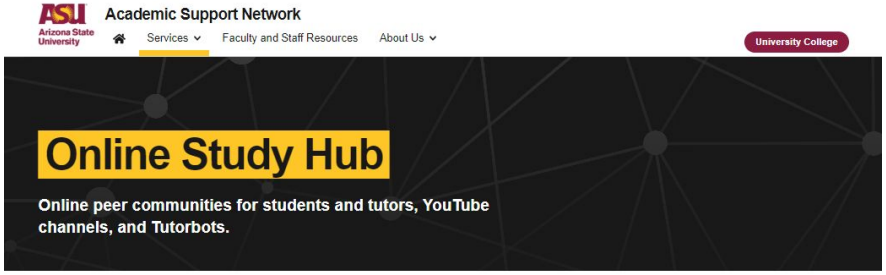


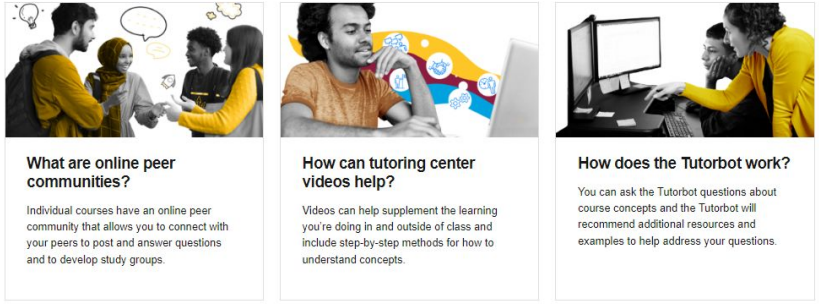
1. **Click on ‘Go to Zoom’ to log onto our Online Tutoring Center.**
2. **Click on ‘View the tutoring schedule’ to see when tutors are available for specific courses.**

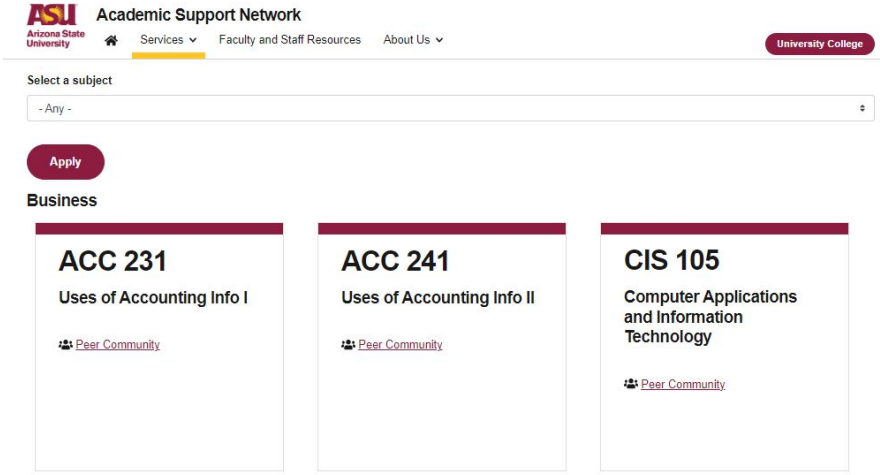
More Questions?

Check out our other resources!

tutoring.asu.edu/online-study-hub



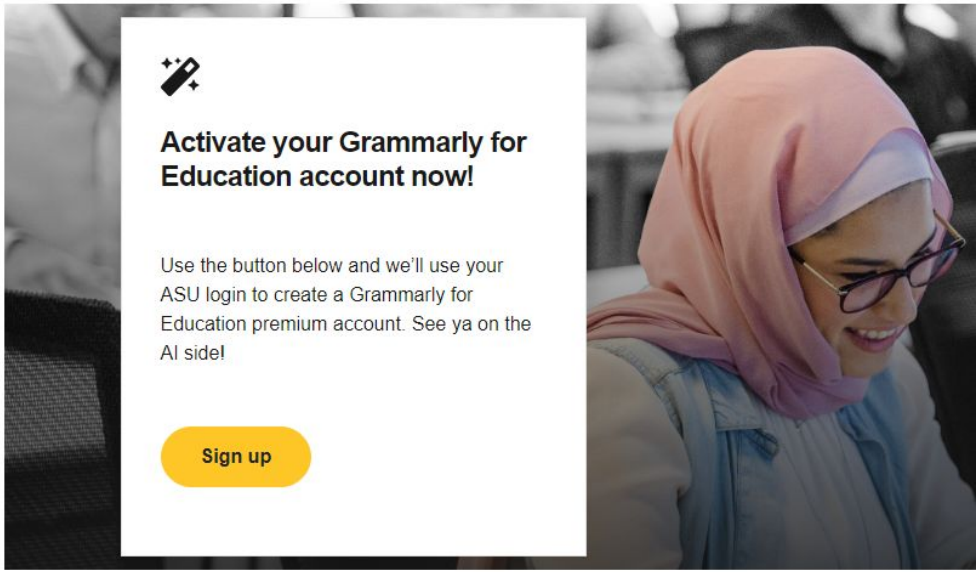




Don't forget to check out the Online Study Hub for additional resources!

Expanded Writing Support Available

Including Grammarly for Education, at no cost!



tutoring.asu.edu/expanded-writing-support

*Available slots for this pilot are limited

Additional Resources

- Course Repo
- Gradle Documentation
- GitHub SSH Help
- Linux Man Pages
- OSI Interactive
- MDN HTTP Docs
  - Requests
  - Responses
- JSON Guide
- org.json Docs
- javax.swing package API
- Swing Tutorials
- Dining Philosophers Interactive
- Austin G Walters Traffic Comparison
- RAFT