Welcome

Welcome to CKCS145!

There are 13 modules in this course. Each week, you'll have a new module with new readings, content and learning activities. You will have the opportunity to participate in weekly Zoom sessions to interact with your instructor and peers.

The work will be iterative, with more general foundational work at the outset of the course. The first six modules are extremely important and their completion will allow you to start working on your final project.

Before moving into the first week's content, this course overview will help you to navigate through the course and modules. The pages that follow will cover:

1. Getting around the course
2. Working your way through the modules
3. Setting norms and expectations

Getting Around the Course

If you haven't already done so, review the "Resources for Learning Online" link under the "Help" area in the main course menu for a general introduction to the tools used in online courses at The Chang School.

This is the first of what we call weekly modules or lessons. Within each module, you'll notice that you'll have very similar menu links on the left side of the individual modules. Make sure you look through all of them. Each week, you'll have some brief introductory material, learning objectives, readings, module content and learning activities. (You may also have additional resources in some modules.)

The introductory materials will provide you with a brief introduction to the week's content, and may also remind you of links to previous weeks' concepts that you might wish to refer to or review prior to starting on the new module.

The learning objectives are provided as a tool to ensure you have adequately worked though the content of the week and have a good understanding of the concepts and material. Use the objectives as your own indicator of whether you've sufficiently met the learning goals for that week.

The main module content could consist of text, visuals, video demos or links or other kinds of media that are provided to help you learn more the topics covered that week. Our hope is that you find the materials here as conversational and understandable as in-class lectures. It is expected that you review this content before attending weekly Zoom sessions (more on that under "A Note on Weekly Activities" below).

The module readings are another important part of your learning each week. Some weeks have more reading than others, so make sure you set aside time each week to go through the readings and try to understand the concepts and discussions that the authors are presenting. Most of the module content will help you “unpack” the concepts and material you’ll read, so doing the readings alongside the module content is always a good idea.

The activities section is to test your own knowledge – this is optional but helpful.

The assignments section will describe which assignments you are required to do each week to complete the module (with clear deadlines). The assignments in this course are weekly labs. For your final week/day of the course, your assignment is a lightning-talk presentation. The assignments are also listed in the Course Outline, which is located in the “Course Materials” area of the course. It is very helpful to review the “Course Schedule” page in the Course Outline prior to reviewing the content of each module.

The resources in the references section of your modules are not required readings or resources that you are required to access. They are simply references, readings and web resources for you to refer to in the event that you want to learn more about a topic, in addition to the module content and readings.

Working Your Way Through the Modules

It is recommended that, each week, you read the introduction and learning objectives and make sure you are familiar with any assignments required of you. Then, do the readings that are assigned. After you have gone through the readings once, read through the content for the week; you should start to have a better understanding of the material as a result and can then engage in the activities during the weekly Zoom session and complete the weekly labs. As a final step, go back to the learning objectives to make sure that, for the most part, you can meet each of the objectives. If you can’t, you’ll need to refer back to the content and readings, along with the activities and assignments, to try to ensure that you can meet the objectives.

Remember, while the modules are week-by-week, you can always refer back to a previous module to review concepts or revisit difficult topics. The content, discussions, activities and assignments will all be visible for the entire course, so that in addition to your more linear week-by-week movement through the course, you can also move around as you wish to ensure you’re learning what you need, when you need it.

One important part of working through an online course is keeping up. While, yes, the modules are all visible throughout the life of the course, you are required to engage in the module content and weekly Zoom sessions during the specific week it is running.

A Note on Weekly Activities

Each module represents a week and the active work will take place during that week. That means that you are required to complete the learning activities during the module week, whether it is week one or week six.

Unlike typical online courses at The Chang School, this course includes weekly Zoom sessions. The Zoom sessions will consist of two hours of activity, followed by one hour for optional office hours with your instructor. You should come to the Zoom session having reviewed any module content and videos. During the first 20–30 minutes, your instructor will introduce or review key concepts for the week. This will be followed by an opportunity to ask your instructor any questions about the week’s module or to go over the previous week’s labs (except for the first Zoom session, of course). After the Zoom session students may work on that module’s lab. The lab must be completed within 1 week. It is strongly recommended that you attend these Zoom sessions. Part of your marks for the course will be your participation in these sessions.

A Discussion board (under the “Communication” area of the course) will be available throughout the course for any questions. Although it is expected that you will bring your questions to the weekly Zoom sessions, questions may arise during the week. Your instructor may choose to respond in the Discussion board or during the weekly Zoom session.

Setting Norms and Expectations

As part of any learning experience, whether in the classroom or online, setting norms is an important step to creating a learning environment that supports open discussion and critical thinking. While you’ll be encouraged to share your views and opinions, make sure you do this in a civil and constructive way. It’s okay to disagree – in fact, it’s great! – but let’s make sure that we do so in a way that is respectful of our own and other people’s differences. At the end of the day, often, we’ll find we have to agree to disagree. The important part is not whether we agree; rather, it’s the quality of the discussion we have.

The same kind of “netiquette” rules that apply to any discussion, apply here, as well. Please see the Course Outline for more details.

Introduction

Welcome to Module 1! This week, we will learn about 3-Tier and N-Tier architectures. Python and Flask will be used to create middleware, a component of the 3-Tier architecture. Middleware will be coded using Python and Flask. Once coded, a web browser may be used to access functionality through web requests.

Topics and Learning Objectives

Topics

- Choosing a code editor
- Linux shell basics
    - man pages and – help
- Using Python and virtual environments
- Creating a “Hello World” web application
- Running a local web server for application development/testing

Learning Objectives

By the end of the module, you should be able to:

1. Create and run a Python/Flask web application
2. Run a middleware locally on a Linux machine

Readings & Resources

Reading

Required

- Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python* (2nd ed.). O'Reilly Media, Inc.
    - Chapter 1: Installation
    - Chapter 18: Additional Resources
- Reitz, K., & Real Python. (2022). The Hitchhiker's Guide to Python: Zen of Python.
- Thoughtbot, Inc. Mastering the Shell.

Client-Server Architecture

The simplest architectural infrastructure is monolithic architecture. A **monolithic architecture** is where all functionality or business logic resides in a single location or module. Monolithic architectural models introduce the concept of centralized computing. This means that all clients connect to a single module or a single server. A refinement of the monolithic architectural model is the client-server model. It allows a single server to fulfill requests from many clients (see Figure 1.1 below). In a client-server architecture, the applications and data are centralized. They are available on a single server.

The advantages of this single server are:

1. Data can be easily secured since it exists in a single location.
2. The server can be maintained since it is a well known entity.
3. Data is always available and up to date since it exists on a single server.

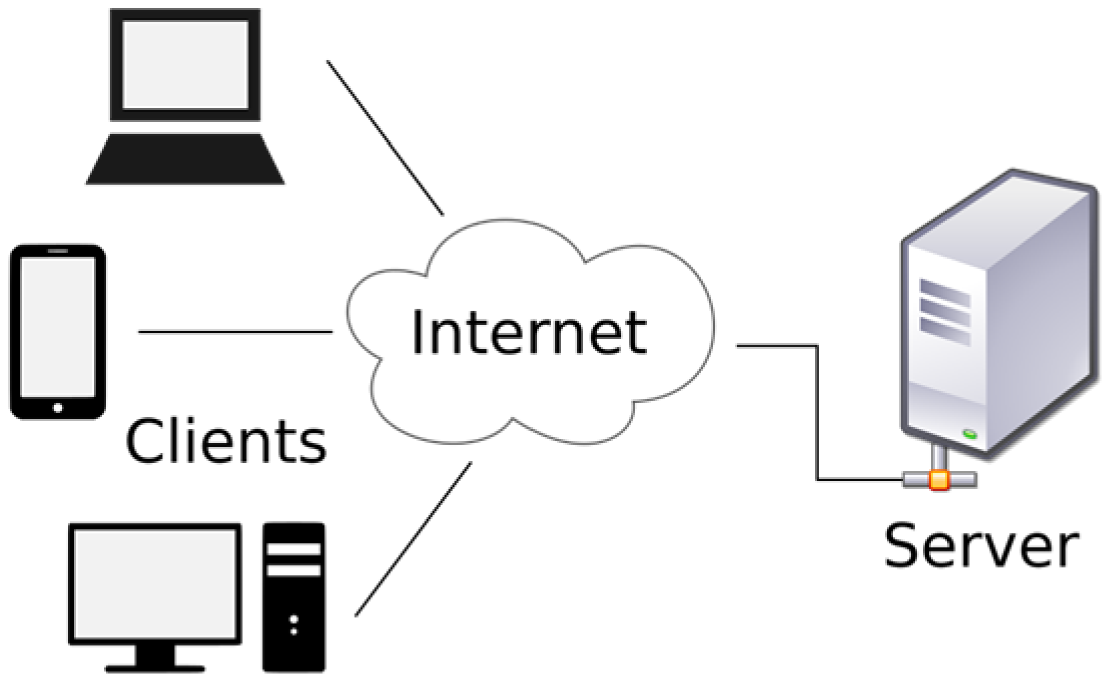

Figure 1.1. A diagram of the client server model.

[Source: Wikimedia Commons]

3-Tier Architecture

The 3-Tier architecture is an enhancement on the client-server architecture. This architecture introduces flexibility that is lacking in the client-server architecture. The client-server architecture provides an “all in one” functionality. In the 3-Tier architecture, each layer has a dedicated purpose (see Figure 1.2 below). In other words, each layer fulfills a specific purpose: presentation, business logic and data.

The 3-tier architecture includes the following layers (from top to bottom in the image):

- **Presentation tier:** The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.
- **Logic tier:** This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.
- **Data tier:** Here, information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

Benefits of 3-Tier Architecture

A 3-tier architecture has 4 benefits:

1. **Security:** Each tier can be secured independently of other tiers with security strategies appropriate for that tier.
2. **Ease of management:** Each tier can be managed independently of the other tiers. For example, the database may be updated independently of the application. The front end may be updated independently of the middleware.
3. **Scalability:** The resources required for each tier may differ from other tiers. Hence, servers that meet these requirements may be used.
4. **Flexibility:** Each tier may be configured according to their unique requirements. What is required for one tier may be totally different from what is required for another tier.

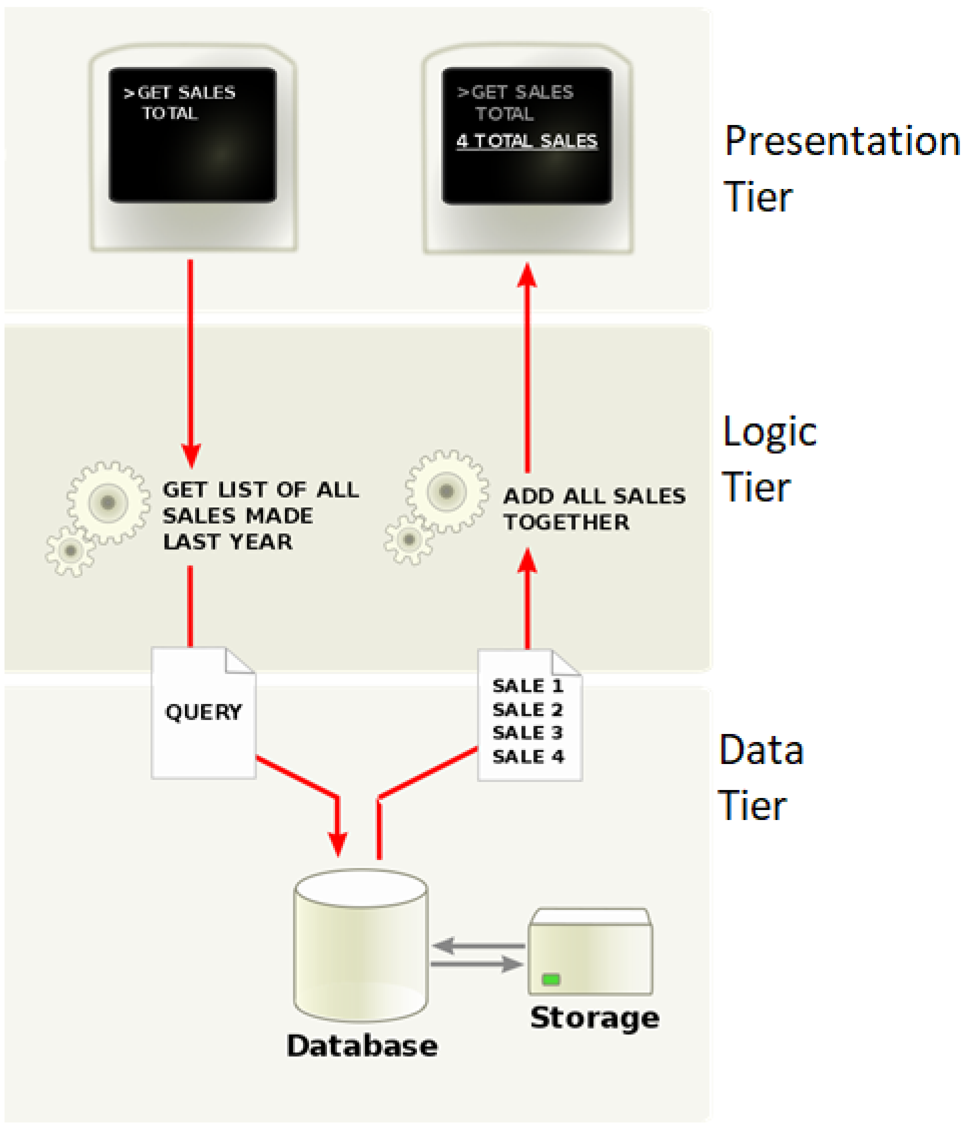

Figure 1.2. A diagram of the 3 tier architecture.

[Source: Wikimedia Commons]

3-Tier Architecture – Web Enabled Systems

- 3-tier architecture is used by many web enabled systems (see Figure 1.3 below), e.g., client (web browser), server (application server), data store (database), all connected to each other.
- 3-tier architecture is technology independent.

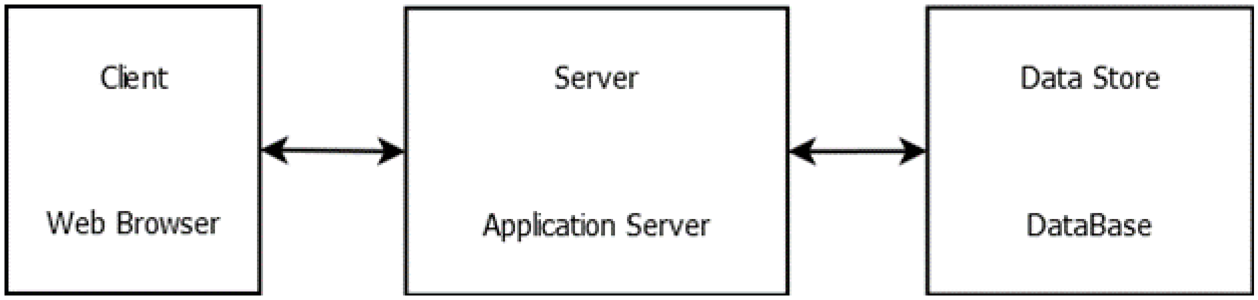

Figure 1.3. How the 3 tier architecture works with web enabled systems.

Source: Mihal Miu

N-Tier Architecture

N-Tier architecture is an extension of the 3-Tier architecture. In the N-Tier architecture, we separate the business logic from the 3-Tier architecture. N-Tier architecture allows the adaptation of new technologies and services from different providers. N-Tier architecture is component-oriented, hence maintenance and development is flexible. Each component can be updated independently of other components (see Figure 1.4 below). Modules may be reused by different applications.

In N-Tier architecture, the web front-end and application API does not change. What changes is the business logic. To be correct, the business logic does not change but evolves by using (or consuming) third party services, such payment gateways, search and indexing services and data store services. For example, a simple payment gateway may not be sufficient. You may require a sophisticated payment gateway that accepts credit cards from different banking institutions all over the world. Also, the acceptance of crypto currencies may require a separate payment gateway.

All of the layers in this architecture are sequentially connected to each other:

The web frontend (where the user is) is connected to the application API, which is connected to data access API and third party API facade. The data access API in turn connects to databases. The third party API facade connects to payment gateways.

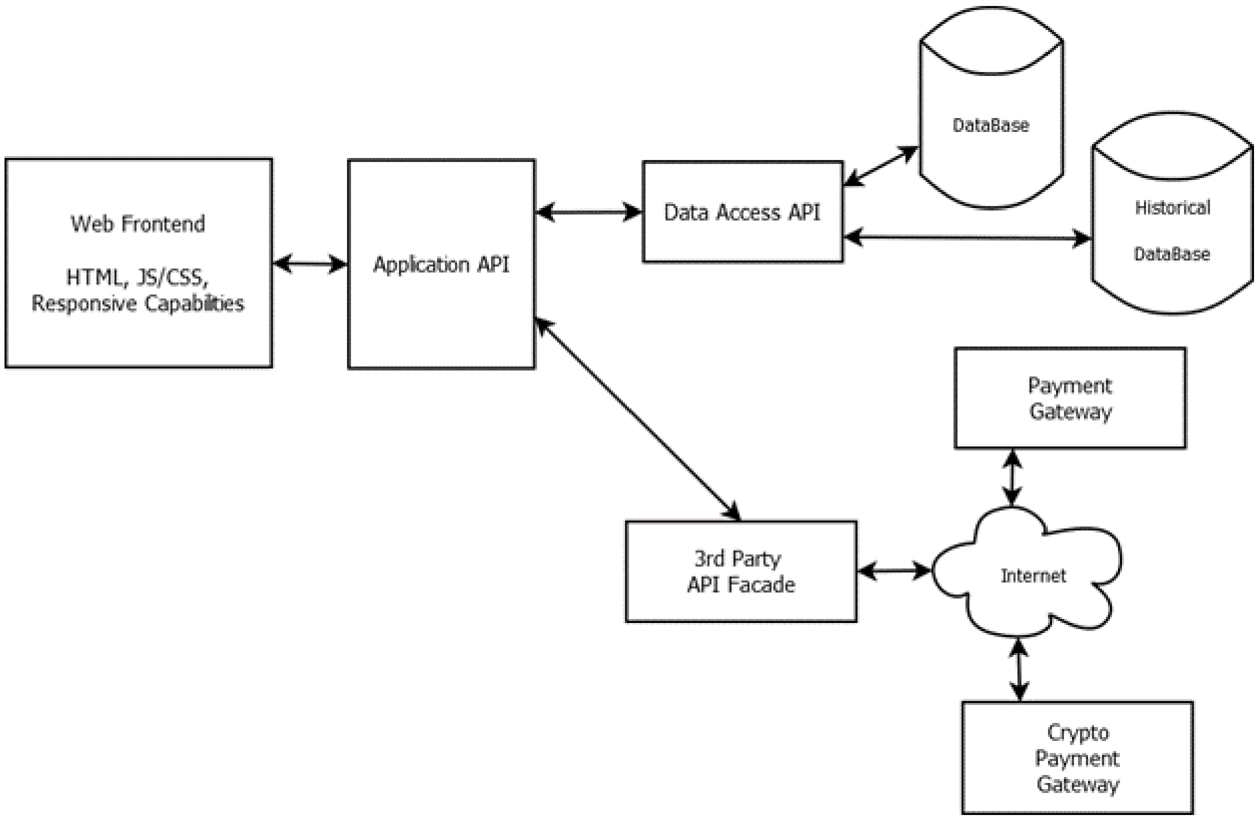

Figure 1.4. The various components of n-tier architecture.

Source: Mihal Miiu

Benefits of N-Tier Architecture

The same benefits provided by 3-Tier architectures apply to an N-Tier architecture. The only difference is that the dimensions dramatically increase:

1. **Secure**
   - Each tier/module can be secured separately using different methods
   - This architecture can provide defence in deep from any hacker
2. **Easy to manage**
   - Each tier/module can be managed, updated and modified separately
3. **Scalable**
   - Resources can be added as needed
   - Resources are added per tier/module without affecting other tiers
4. **Flexible**
   - Each tier/module can be expanded with respect to changing requirement

Python

Python is a high level, general purpose programming language. Furthermore, Python is easy to learn and easy to use. It supports multiple programming paradigms: structure/procedural, object-oriented and/or functional programming. It is recommended that, when developing applications in Python, only one paradigm be used per program. Python is designed to be highly extensible via modules. Two common modules that we use are Flask and SQL Alchemy. **Flask** provides support for web services/applications and **SQLAlchemy** provides support for data access from relational databases. Packages may be installed using the Python package manager named **PIP** (Preferred Installer Program). PIP is a command line application.

Python has been around for at least thirty years. The first release of Python was version 0.9 in 1991. Version 2.0 was released in 2000. Python version 3.0 was released in 2008. Many advances have been made since the turn of the century. It was apparent that, in order to implement these advancements, Python required a rewrite. Version 3.0 of Python is considered a rewrite of Python version 2.X, hence version 3.0 is not backwards compatible to any previous versions. The aim of new development is to use Python version 3.7 or higher. The exact version may depend on which libraries are supported by Python.

Flask

Flask is a micro web framework written in Python. It started out as a microframework, since it does not require particular libraries. It does not provide a database abstraction layer, form validation or any other components where third party libraries provide sufficient capabilities.

Flask is the de facto support library for RESTful web services. Flask is highly dependent on the Jinja template engine and the Werkzeug WSGI Toolkit. Flask has gone through various iterations and it is currently at version 2.2.

Flask enables the creation of middleware that support 3-Tier or N-Tier architectures. Flask does not support the entire 3-Tier or N-Tier architectures but a smaller component. For example, Flask is used to provide access to the application API. This can be seen in Figure 1.4 earlier in this module.

API

Application Programming Interface (API) is a connection between computers or between computer programs. It is a software interface, offering a service to clients or other software. The main purpose of APIs is to hide the internal details of how a system operates, exposing only those parts a programmer will find useful while keeping complicated internal details hidden. Flask allows us to implement Web APIs; more specifically, it allows us to implement RESTful APIs.

REST

Representational State Transfer (REST) is a software architectural style that was created to guide the design and development of the architecture for the World Wide Web. Web resources are defined as functions (business logic) identified by URLs. The goal of REST is to increase performance, scalability, simplicity, modifiability, visibility, portability and reliability. REST or RESTful based applications may be implemented by 3-Tier or N-Tier architectures.

Video

Please watch the following video on Simple Middleware. This is our first attempt in implementing a 3-tier architecture using Python/Flask and a web browser. (Note: this implementation does not have the data layer. The data layer will be introduced in Modules 5 and 6.)

Simple Middleware
TMU Video

Virtualization

Virtualization is the act of creating a virtual version of something at the same abstraction level, including virtual computer hardware platforms, storage devices and computer network resources. Virtualization may occur for hardware, operating systems, and applications. **Hardware virtualization** is the creation of a virtual machine that acts like a real computer with an operating system. Software executed on these virtual machines is separated from the underlying hardware resources. Host Machines are represented by the operating system (OS) that runs directly on hardware. Guest Machines are represented by the OS that is virtualized or the virtual machine (VM).

Desktop Virtualization

Desktop virtualization is the concept of separating the logical desktop from the physical machine or physical hardware. VirtualBox is a x86 and AMD64/Intel64 virtualization product for enterprise and home use.

A hypervisor is computer software or hardware that allows partitioning the resource of a CPU among multiple OSes or independent programs. VirtualBox is an open-source and freely available hypervisor. VirtualBox runs on top of a Host Machine and allows for the execution of multiple Guest Machines. This assumes that the Host Machine has available resources such as CPU threads, memory and storage space.

Video

Please watch the following video on Virtual Machine Usage.

Virtual Machine Usage
TMU Video

Summary

In this module we coded middleware or middle tier from a 3-Tier architecture. Furthermore, we deployed middleware in a virtual machine.

Assessments

Each week, you'll find here reminders of assignments or labs that you should be working on. For Week 1:

- Lab 1: Creating a web application with Python/Flask (5% of the final grade) is due in 7 days, i.e., the end of day Monday of Week 2.

Reminder

You have a weekly Zoom session. The date/timing for your Zoom session can be found in the Course Schedule in the Course Outline. The link to the Zoom session will be provided separately to you by your instructor. The Zoom session will consist of 20–30 minutes of lecture, followed by questions and answers (Q&A), followed by a lab period. You may ask questions about the previous or current week's content or labs.

Please bring any questions to the weekly Zoom session. However, if you have any questions during the week outside of the Zoom session, you may post them in the Discussion Board. Your instructor may respond in the Discussion board or during the weekly Zoom session.

Click-n-reveal: **Where is the Discussion board?**

Click on the "Communication" area at the top main course menu, and select Discussions from the dropdown menu.

References

None for this week.