You will have individual tasks and Mob Tasks throughout the semester. For the Mob Tasks, you are required to follow Mob Programming principles.

Mob Programming.

- `https://youtu.be/ikilHGYk5Fs`
- `https://www.pluralsight.com/blog/software-development/mob-programming-101`
- `https://www.techtarget.com/searchsoftwarequality/definition/mob-programming`

Course Tracker. You are required to track your progress through the course using this table.

Note: Each row represents a module, a half-week's work.

Course Tracker

Half-Week	CRP	PFP	CPT	AMMT	PCTU
1	✓	✓	✓	✓	?%
2					
3					
4					
5					
6					
7					
8					

- CRP - Completed the Reading Prior to the presentation
- PFP - Participated Fully in the Presentation of the material for the week
- CPT - Completed all Personal Tasks
- AMMT - Actively Mobbed all the Mob Tasks (using mob-programming principles)
- PCTU - Percentage of Tasks Completed and Understood

Project Tracker

Week	MOB	MOC	CPT	PIU	
5	✓	✓	✓	✓	?%
6					
7					
8					
9					
10					
11					
12					
13&14					

- MOB - I participated fully in the Mob, Director, and Driver responsibilities multiple times this week.
- MOC - I met and fully mobbed with my team outside of class at least 3 times on different days this week.
- CPT - I met and fully mobbed with my team during each class period this week.
- PIU - The Percentage of materials from the modules that I Understood and can use.

0.1. **Grade Claims.** On the week indicated, bring this updated document to my office and make your claim.

Claim Week	Grade Claim	Instructor Grade	Adjusted Grade
5			
9			
13 - 14			

Here is a list of the tasks for each week of the semester. It is important that you put in sufficient time outside of class to complete these on time. Otherwise, there will not be sufficient time for the project done later in the semester.

Mob programming is used throughout the class. Watch this video on Mob programming for a brief overview.

Module 1 Setup and Erlang Tutorial

These are not listed in the order they should be done.

Personal Tasks.

- **Readings**
  - Chapters 4 & 5 in *Programming Erlang: Software for a Concurrent World, 2nd Edition*
  - Miscellaneous web pages from search
- **Videos**
  - find a few videos on Mob programming. Watch them until you are confident in your understanding of the different roles you will be fulfilling.
- **Hardware Setup Tasks**
  - Use Squarespace's service to claim a Domain (https://domains.squarespace.com/ or other service). The service you select *must* allow DNS servers to be changed to some other provider's DNS servers. We will be using Digital Ocean's DNS servers (Don't get a DO account yet).
  - Windows users install WSL and VSCode (https://learn.microsoft.com/en-us/windows/wsl/tutorials/wsl-vscode) or a VM running the latest version of Ubuntu Linux.

- Install Rebar3 on your laptop. This will also install Erlang (Linux use apt)(MacOS use Homebrew).

Mob Tasks.

- Select one mob member's GitHub repo to hold all the work from this class. You can fork this repo for use in your resume or other needs.
- Invite all mob members to be collaborators on the repo.
- All mob members must accept the invitation to collaborate.
- Get the task files from https://github.com/yenrab/ErlangTutorialTasks
- Use rebar3 to create a different app for each of the task files.
- Move each provided task file into its respective app's source directory.
- Using rebar3 and a text editor, gradually modify *eunit_test_practice.erl* until it passes all the supplied unit tests.
- Using rebar3 and a text editor, gradually modify *cone_combination_builder.erl* until it passes all the supplied unit tests.
- Each mob member runs all the unit tests on their own machine.

Module 2 - Processes, EUnit, and HTTP

Personal Tasks.

- **Readings**
  - Chapter 12 in *Programming Erlang: Software for a Concurrent World, 2nd Edition*
  - Miscellaneous web pages from search
- Copy the *hello* app from MS Teams onto your computer.
- Run the hello app.
- Use your web browser to connect to the web server running on your droplet.
- Use a file comparison tool to see the differences between the *hello* app and one of the apps your team created last week. You will need to get this from the GitHub repo created by your team.

Mob Tasks.

- Create a GitHub repo to hold this *hello* app.
- Invite all mob members to be collaborators on the repo.
- All mob members must accept the invitation to collaborate.
- Use mob programming to add end points for 3 other types of responses. These responses will be of your imaginings. Include sending HTML from your handlers.
  - `https://www.tutorialstonight.com/html-web-page-examples-with-source-code`
  - `https://designmodo.com/html5-examples/`
  - `https://www.w3schools.com/html/html_examples.asp`
  - And other things you find on the web.

- In the apps you created for the last module for the *friend_tracker.erl* and *arithmetic.erl* files,
  (1) Using rebar3 and a text editor, gradually modify *friend_tracker.erl* until it passes all the supplied unit tests.
  (2) Using rebar3 and a text editor, gradually modify *cone_combination_builder.erl* until it passes all the supplied unit tests.
  (3) Push your changes to GitHub for each repo.
  (4) Each mob member must use git clone to move your app to their machine.
  (5) Each mob member runs all the unit tests on their own machine.

Module 3 - The Generic Server Actor

- **Personal Tasks**
  - **Readings**
    - Chapter 4 and Chapter 5 through the sys Module Recap section in *Designing for Scalability with Erlang/OTP*
    - Miscellaneous web pages from search
- **Mob Tasks**
  - Create a GitHub private repo for this weeks tasks.
  - Create a rebar3 app for this week's tasks.
  - Link the app to the GitHub repo.
  - Add the provided files to your app's source directory and complete the code required to pass the supplied *gen_server* unit tests.
  - Push your changes to GitHub
  - Each mob member must use git clone to move your app to their machine.
  - Each mob member runs all the unit tests on their own machine.

Module 4 - Supervisors

- **Personal Tasks**
  - **Readings**
    - Chapter 8 in *Designing for Scalability with Erlang/OTP*
- **Mob Tasks**
  - Create a rebar3 app and a GitHub repo for the app.
  - Using the gen_server template, design, write unit tests for, and create a gen_server that does interesting computations.
  - Create a dynamic supervisor for a set of 10 gen_servers of the type you just created.
  - Create a static supervisor that supervises this new supervisor and one instance of the rr_selector. The rr_selector should select from the set of gen_servers of the type you just created.

Module 5 - State Machines

- **Personal Tasks**
  - **Readings**
    - Chapter 6 in *Designing for Scalability with Erlang/OTP*
- **Mob Tasks**
  - Design, write unit tests for and create a state machine that represents a linked smart-car. The cars need to be able to communicate with the 3 cars nearest to them that are moving. (You may want to dust off your closest-pair algorithm from the Algorithms and Complexity class). Create a gen_server that is a hub for inter-car messaging. The gen_server should use cast rather than call messaging.
  - Create an appropriate supervision tree for 10 of these smart-car state machines and the messaging hub.

Module 6 - Event Handlers

- **Personal Tasks**
  - **Readings**
    - Chapter 7 in *Designing for Scalability with Erlang/OTP*
- **Mob Tasks**
  - Design, write unit tests and implement an simulated traffic light at a four-way intersection. Each of the entrances to the intersection have a left turn lane and two traffic lanes, one each in each direction. There are no right turn lanes. Use an event handler and a state machine to simulate the red-yellow-green traffic light.
  - Create unit tests and make modifications such that 3 traffic lights are synchronized for traffic at speeds of 20 mph and a distance between the lights of 600 feet.
  - Create an appropriate supervision tree for the OTP elements for the traffic light system.

Module 7 - Monitoring and Preemptive Support

- **Personal Tasks**
  - **Readings**
    - Chapter 16 in *Designing for Scalability with Erlang/OTP*
- **Mob Tasks**
  - Select the most complicated code you've created in any language. Apply the monitoring and support principles you've learned in this reading to evaluate your code. Is it being monitored well? Is preemptive support possible? To what extent?

  - Make changes to your traffic light project. Apply what you've learned regarding monitoring and support. Use RabbitMQ to do your logging and monitoring.

Module 8 - HTTP, Mocking, Data Storage, and Data Retrieval

- **Readings**
  - Miscellaneous pages from Web Search
- **Mob Tasks**
  - Build an entire Web-available micro-service that stores, updates, and deletes data of your choice. Use Cowboy, SSL, DMZ, and Riak. Setup an appropriate supervision tree for your micro-service. Apply the monitoring and support principles learned in Module 7.

Remainder of Class

(Project) Using what you have learned, create a Web Service for Package Tracking - API Design, Software Design, Unit Testing, Implementation, System Stress Testing

0.2. **Hardware Setup.**

- Get $200 of free usage from DigitalOcean
- Create a $6 per month DO Droplet and access it using the web console.
- Setup SSH on your DO Droplet and access it using SSH from the terminal on your laptop
- Create a user on your DO droplet. NEVER run stuff as *root* on any machine.
- Secure your droplet.
- install rebar3 on your droplet using apt.

0.3. **Things to be Aware Of.**

- Make sure you keep your project app in a unique GitHub repository. If you don't you'll slow down your engineering and development.
- Put all your engineering designs in GitHub.
- Every *nix system has a limit for how many open files there can be on the system.
- There are a whole lot of things that are considered files on *nix systems.
- Don't over engineer your first design. It won't be your last. Let the bottlenecks you discover drive engineering changes that increase optimization.