# Lecture 2: September 7

ML Project Design

# Agenda

- Senior Design High Level Timeline
- Github Projects Setup & Sprint Progress Expectations
- ML Project Design
- Tech Lab Overview

# Agenda

- **Senior Design High Level Timeline**
- Github Projects Setup & Sprint Progress Expectations
- ML Project Design
- Tech Lab Overview

| Month | Expected Status | Monthly Focus | Deliverables |
|---|---|---|---|
| September | N/A | - Figure out teams<br>- Brainstorm projects | - Create teams<br>- resume |
| Mid-September | - Teams selected<br>- Handful of project ideas | - Final project selection<br>- Begin meeting w/ mentors | - Project proposal<br>- Hardware/software request<br>- Writing: Executive Summary |
| October | - Project selected & approved | - Begin technical investigations (services, apis, language, etc)<br>- Flesh out project functionality & requirements<br>- Coding should start (scaffolding, ci/cd, prototyping) | - Writing: Technical summary<br>- Presentation: Elevator pitch |
| November | - Main technologies selected<br>- project is well-defined<br>- Everyone is actively coding | - Answer all questions needed to complete TDD<br>- Lot's of coding for alpha demo | - Writing: PRD<br>- Presentation: Project Design |
| December | - Code complete for alpha demo | - more coding for beta demo<br>- Formalize design discussions into proper TDD | - Presentation: Alpha prototype<br>- Writing: TDD |
| January | - Continued focus on project development | - continued development for beta demo<br>- focus on proper testing & integration | - Website Design |
| February | - Code complete for beta demo | - Refine code from a prototype into a fleshed out project -- testing, integration, polishing<br>- continued development for prelim prototype (get as close to finished as you can here) | - Presentation: Beta prototype<br>- Presentation: Elevator pitch/promotional |
| March | - Code complete for prelim demo | - final code polishing to wrap up project<br>- complete any necessary integration work<br>- add extra features if possible | - Presentation: Pelim prototype |
| April | **- Code 99% complete for final demo** | - finishing touches for final project submission<br>- ideally you are done with coding by this point | - Presentation: Final demo<br>- Promotional video |
| May | | | - Final package due |

# Agenda

- Senior Design High Level Timeline
- **Sprint Progress Expectations**
- Github Projects Setup
- ML Project Design (industry vs academia)
- Tech Lab Overview

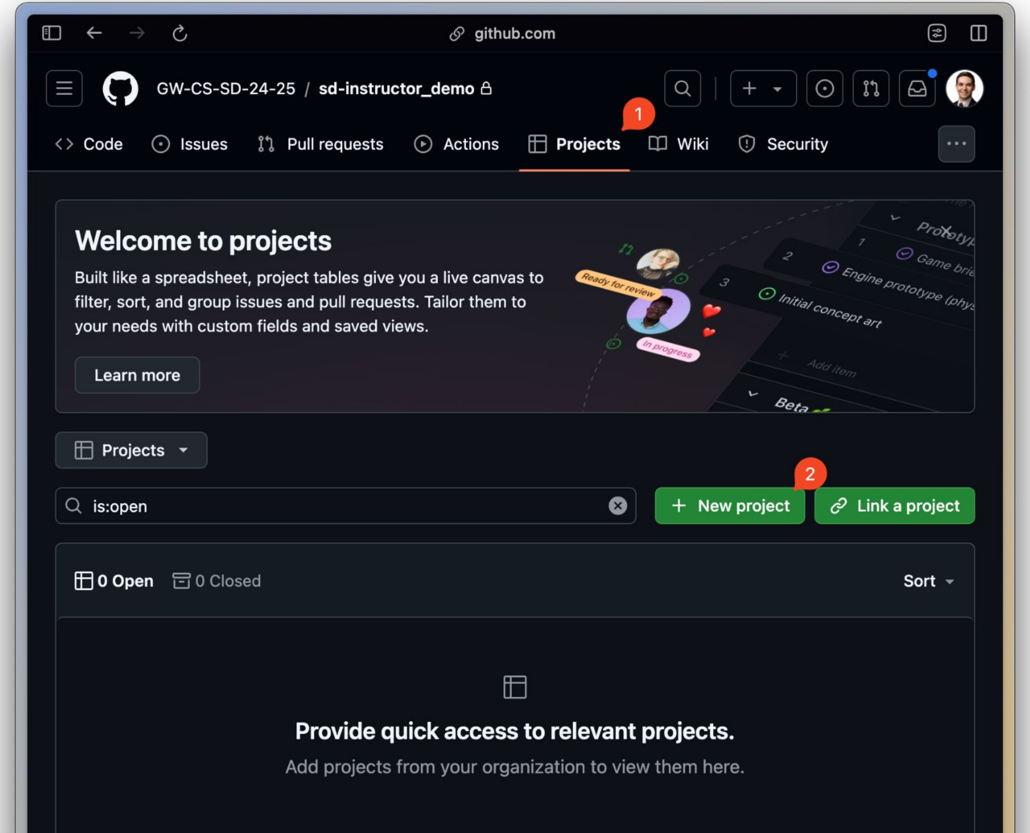
# Sprint Progress - Components

- Sprint Board
- Weekly Updates
- Slack Participation
- Code

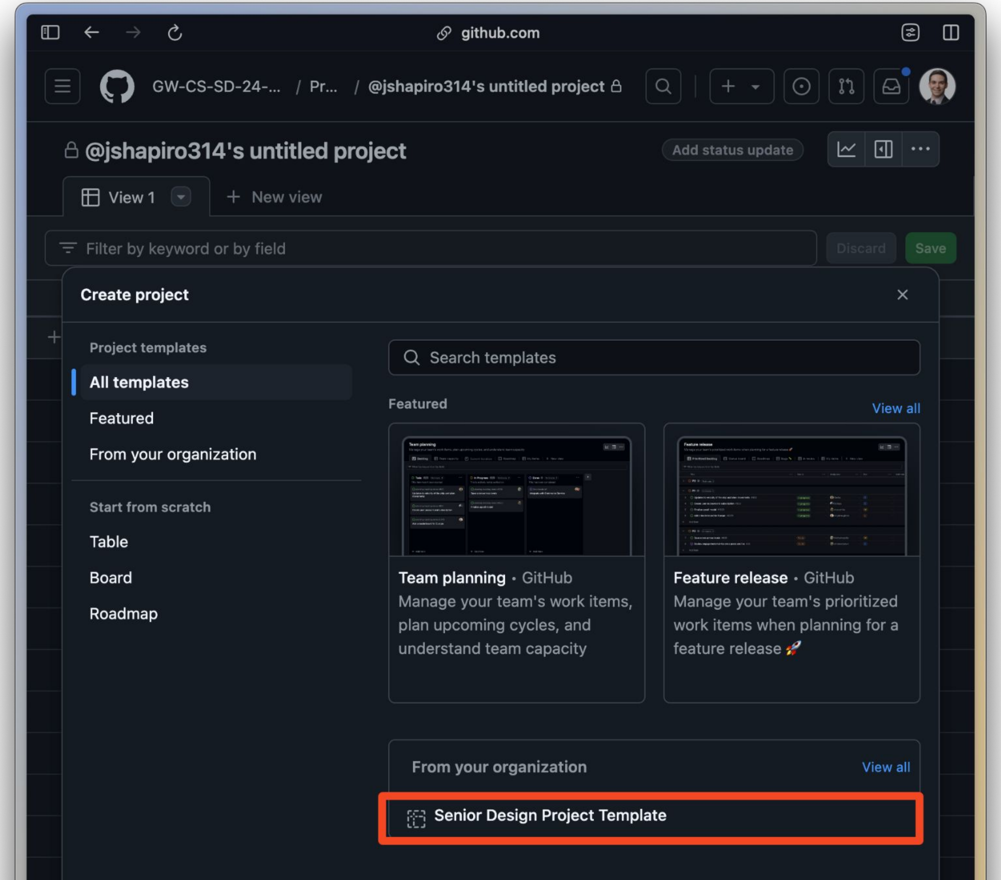
# Components - Sprint Board

- Team members should create a backlog of tickets to work on
- At the beginning of each sprint, members should pull tickets from their backlog into their sprint
- Tickets should include the following:
    - Description
    - Assignee
    - Epic
    - Due date
    - Sprint
    - Status
- By the end of a sprint, all tickets should be **done**, **won’t do**, or moved to the next sprint
- All senior design work should be accompanied by tickets (including presentations & writings)
- Tickets should be appropriately scoped to single features/prs
- Sprint boards should be created / populated during monthly sprint planning

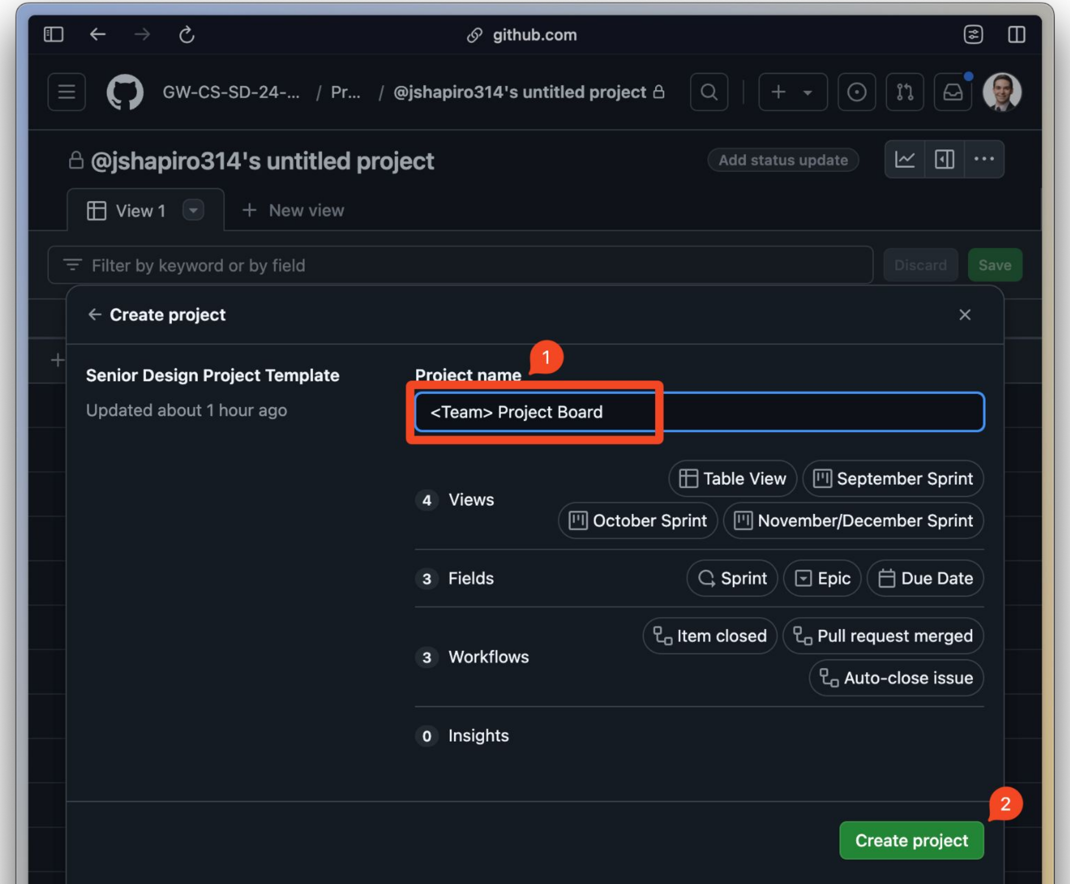
# Create Github Project Boards



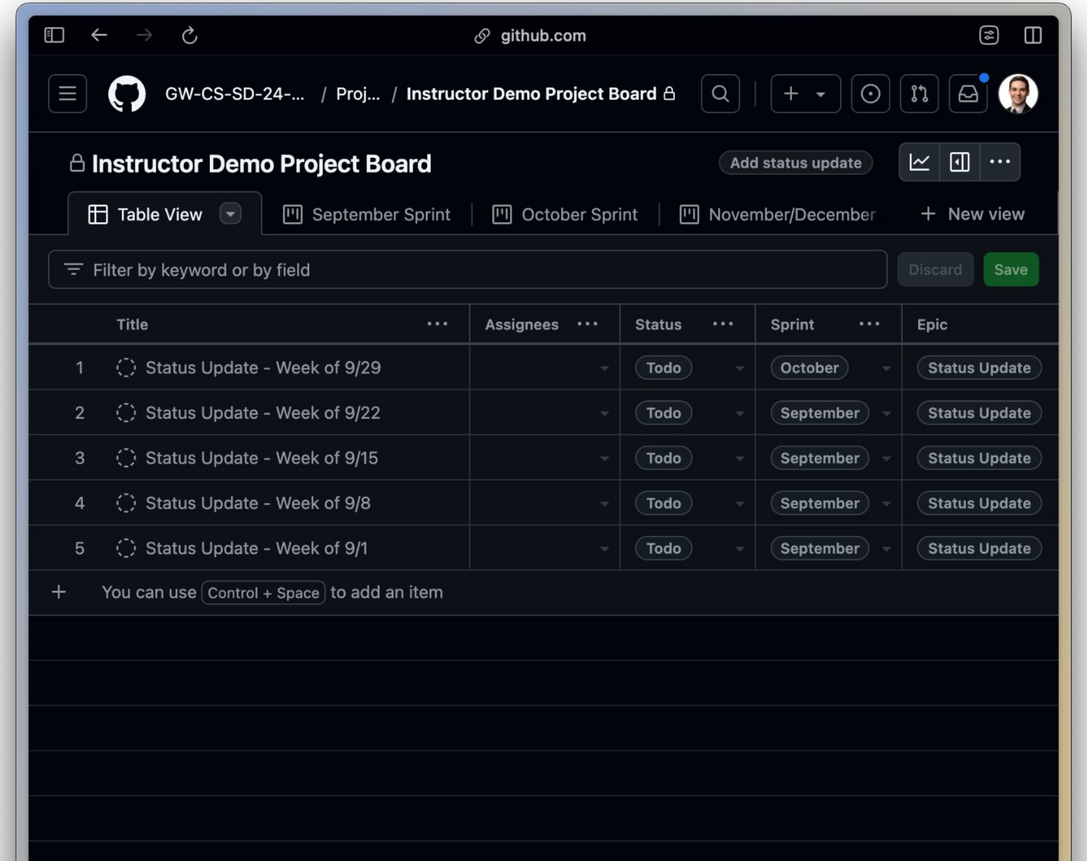
# Create Github Project Boards



# Create Github Project Boards



# Create Github Project Boards



# Create September Tickets

| Title | Epic | Due Date | Assignees | Sprint | Status |
|---|---|---|---|---|---|
| Submit Resume | Writing | 9/8 | Individual | September | TODO |
| Draft Project Proposal | Design | 9/15 | all | September | TODO |
| Refined Project Proposal | Design | 9/22 | all | September | TODO |
| HW/SW Requests | Design | 9/22 | all | September | TODO |
| Writing 1 | Writing | 10/6 | Individual | October | TODO |

# Components - Status Updates

- Students should post weekly status updates covering:
    - What they completed (can link out to tickets)
    - What they are blocked by
    - What they are currently working on
    - **Each student must leave their own comment**
    - It is ok for the update to reflect no work
- These updates should be captured on **Status Update** tickets
    - Move ticket from TODO to DONE as week progresses
    - Leave comment BEFORE DUE DATE to receive credit
- Complete these updates prior to weekly meeting with mentor (used to lead discussion)
- Create new status update tickets as sprints progress
    - Title should be **Status Update - Week of MM/YY**
    - Due date should be following Sunday

# Components - Slack / Participation

- Use slack as the main communication method between teammates and:
  - Other teammates
  - Mentors
  - Instructors

# Components - Code

- We expect all students to write code for senior design
- Only code pushed to main/master will be evaluated
- Code should be written in branches & PRed into master
- PR reviews are **highly encouraged** during the fall and required in the spring
- Code PRs should be well-scoped to single features and tied to sprint tickets

# Sprint Progress Rubric

## Fall Semester

**Full credit**

- Tickets addressed as either “done”, “won’t do”, or moved to next sprint.
- Weekly standup updates & slack participation
- Code is PRed & merged to master. Branches & PRs are well-scoped.

**Partial credit**

- Majority of tickets addressed as either “done”, “won’t do”, or moved to next sprint.
- Occasional standup updates & moderate participation
- Code is committed, PRs are sometimes present and sometimes well-scoped

**Minimal credit**

- Few tickets addressed as either “done”, “won’t do”, or moved to next sprint.
- Minimal standup updates & rare participation
- Minimal code is committed, PRs are missing or not well-scoped.

**No credit**

- No sprint board activity
- No standup updates
- No slack participation
- No code committed to master/main

# Sprint Schedule

## Fall Semester Sprints

**September Sprint**

**October Sprint**

**November / December Sprint**

## Spring Semester Sprints

**January / February Sprint**

**March Sprint**

**April Sprint (2 weeks!)**

# Agenda

- Senior Design High Level Timeline
- Github Projects Setup
- Sprint Progress Expectations
- **ML Project Design (industry vs academia)**
- Tech Lab Overview

# Context

1. We anticipate many teams will explore some form of machine learning as part of their senior design project

2. Implementing ML in a product is not easy. It's also not a skill set typically taught in class.

3. These are the problems I focus on day to day, and I’ve found success in this approach.

# Goals

- Know when to apply ML to a problem
- Know how to build out an ML solution
- Understand what different ML roles in industry entail

# Non-Goals

- Explain how to train models
- Explain how to productionize models

# Agenda

- How to tell if a problem is well-suited for an ML solution
- How to approach an ML solution
- Different roles in the ML field

# What makes an idea good for ML?

1. Can the problem be uniquely solved by ML?
   a. Can a human solve this task manually?
   b. Does a rules-based approach work?
   c. What are the existing bottlenecks to solving this problem?

2. Do you have data / can you get data?

3. The **IVO** test: can the user **immediately validate** the **output**?
   a. Change the user
   b. Make validation easier
   c. Change the output format

https://cloudonair.withgoogle.com/events/startup-school-ai/watch?talk=emea-amer-talk2

# Agenda

- How to tell if a problem is well-suited for an ML solution
- **How to approach an ML solution (an ML Technical Design Doc)**
  - Defining the input/output
  - What is your data
  - What are the metrics
  - Establishing baselines & benchmarks
  - Model training/exploration
  - Approaching ML in Senior Design
- Different roles in the ML field

# Approaching an ML Solution: **Inputs & Outputs**

1. Identify the interface of your product user experience
2. Identify the interface of your ML model(s)
   a. What is the input?
   b. What is the output?

**Why?**

- Adds structure to ambiguity – can't just lean on ML for scope creep
- Engineering is easier with interfaces. ML is hard, isolating from the rest of a system is important.
- Your interfaces will dictate the data you need and the training approach you're using (regression, classification, clustering, generation)

# Approaching an ML Solution: **Data**

1. Do you have data that matches your input/output interface?
2. How costly is it to collect labelled data? Are there other ways of getting "labelled" data?
3. Do you have/need unlabeled data?
4. What is your training/validation/test set?
5. What are the characteristics of your data? (amount, biases, etc)

**Why?**

- If you don't have data, you're going to have a bad time.
- Figure out early if ML is not the right approach
- Data needs can change during experimentation

# Approaching an ML Solution: **Metrics**

1. What offline “correctness” metrics do you care about?
2. Are there separate online metrics that are important?
3. Are there performance metrics that impact your solution?
4. What is the one metric that matters most?

**Why?**

- Need a way to objectively measure different approaches
- Need a way to evaluate a system once in production
- Forces you to focus attention on a small number of things to optimize

# Agenda

- How to tell if a problem is well-suited for an ML solution
- **How to approach an ML solution (an ML TDD)**
  - Defining the input/output
  - What is your data
  - What are the metrics
  - Establishing baselines & benchmarks
  - Model training/exploration
  - Approaching ML in Senior Design
- Different roles in the ML field

# Approaching an ML Solution: **Human Performance**

- Using the data & metrics defined previously, how does a human measure on the task?
- What is needed to collect this data?

**Why?**

- Ensures you can evaluate your system
- Sets a bar for performance to aim for (higher precision, higher recall, faster)

# Approaching an ML Solution: **Quick Baseline**

- What is the simplest approach we can take to solve this problem? (Almost always logistic regression, xgboost, non deep learning or ml techniques)
- How does the simple approach measure up?

**Why?**

- Helps build out pipeline for evaluation without focusing on experimentation
- Can be used as a placeholder while building out the engineering system
- Sets a minimum bar for performance
- Identifies the gap between humans & ml

# Approaching an ML Solution: **Upper Bound Baseline**

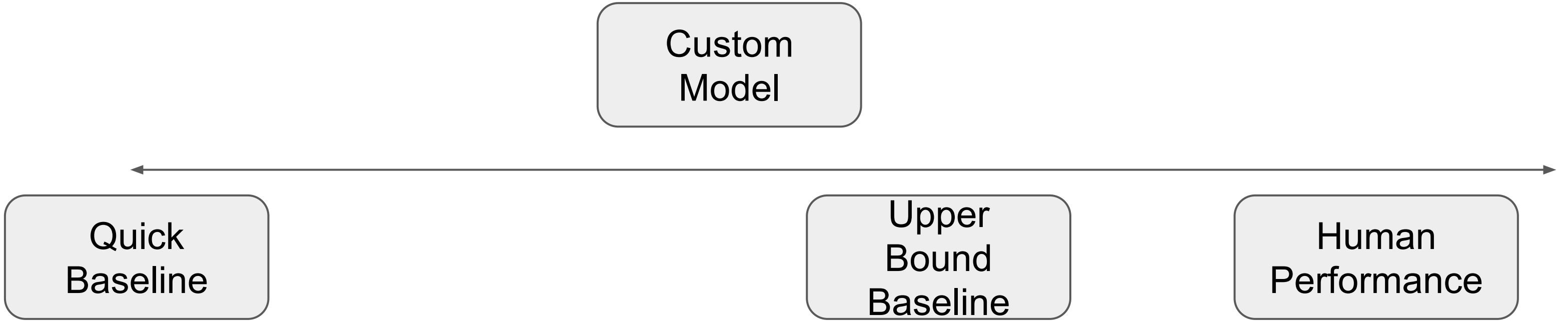
- If compute/money was no object, how would we do? (Throw an LLM at the problem)
- How does zero shot vs few shot affect results?

Why?

- Sets a pseudo-upper bound to expected ML performance
- Helps you understand tradeoffs between “accuracy” metrics & performance metrics

# Approaching an ML Solution: **Experiment!**

- You’ve done your homework, now train your own model



# ML in Senior Design

- Identify if ML is the right solution to your problem

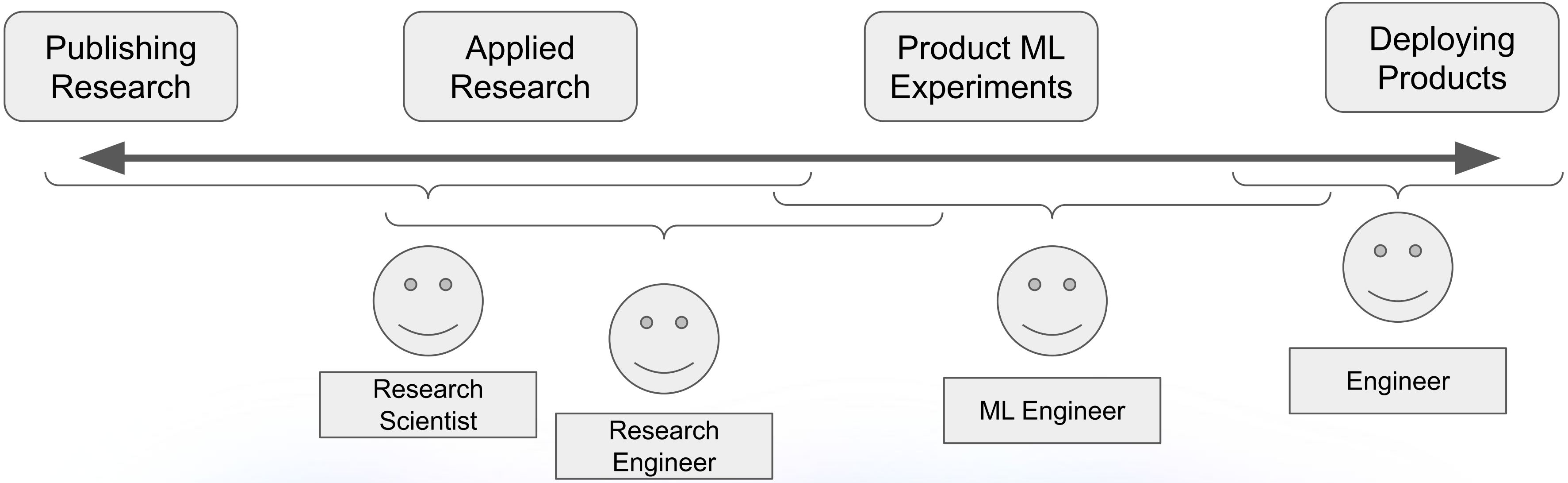
- It is not enough to integrate ML into your solution – you must be able to explain why it is necessary / how much it helps

- Creating a full eval pipeline can be time consuming. Up to your team whether or not this is something worth prioritizing.

# Agenda

- How to tell if a problem is well-suited for an ML solution
- How to approach an ML solution (an ML TDD)
    - Defining the input/output
    - What is your data
    - What are the metrics
    - Establishing baselines & benchmarks
    - Model training/exploration
    - Approaching ML in Senior Design
- **Different roles in the ML field**

# Roles in the ML field



# Research Scientist

Expectations:

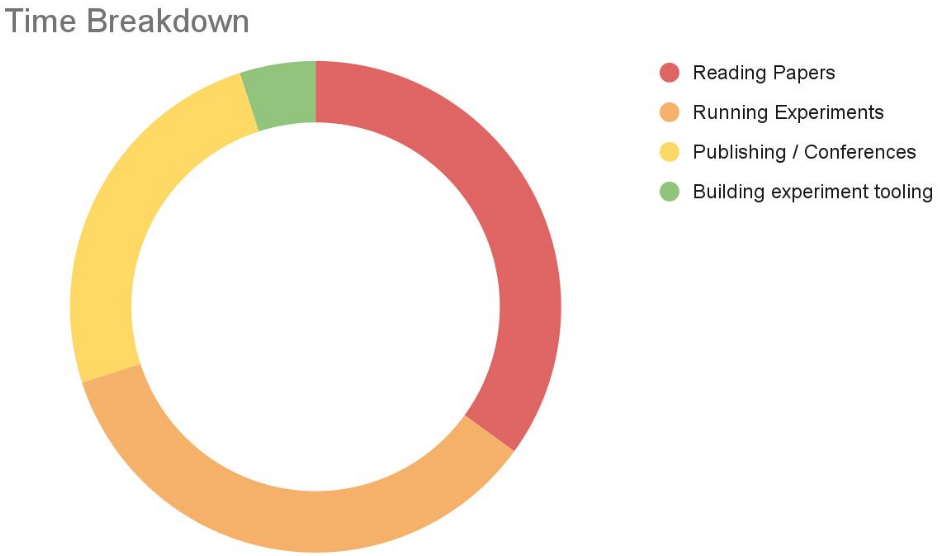
- Publish papers
- Create patents
- Novel ideas 1-2 years out

Challenges:

- Running lots of experiments & analyzing results
- Getting eng / infra help for experimentation
- Compute
- Working with teams to get data

Teams:

- Research engineers
- ML engineers
- Data science



# Research Engineer

Expectations:

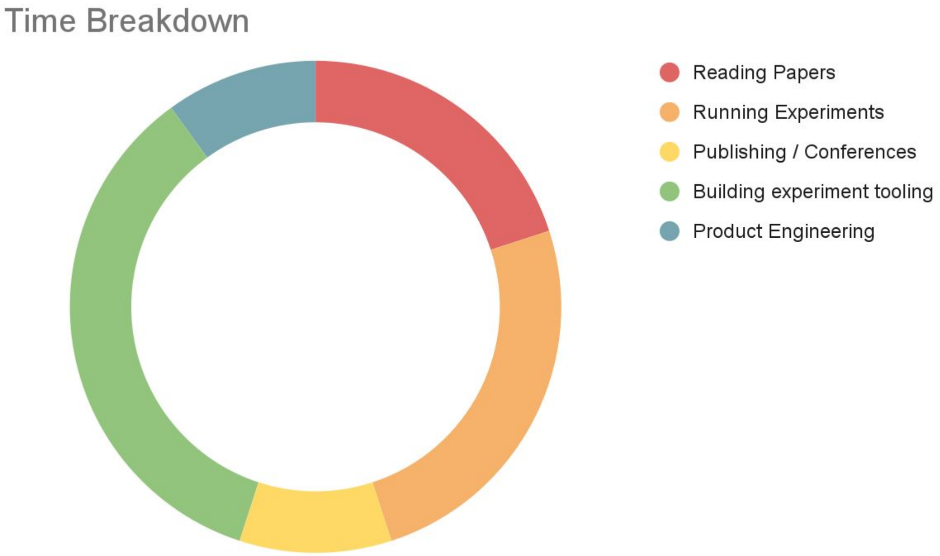
- Make experimentation easier
- Novel ideas 6-12 months out
- Publish papers/patents

Challenges:

- Build infra for research scientists
- Act as liaison between ml & research

Teams:

- Research scientists
- ML engineers
- Data science
- Product



# ML Engineer

Expectations:

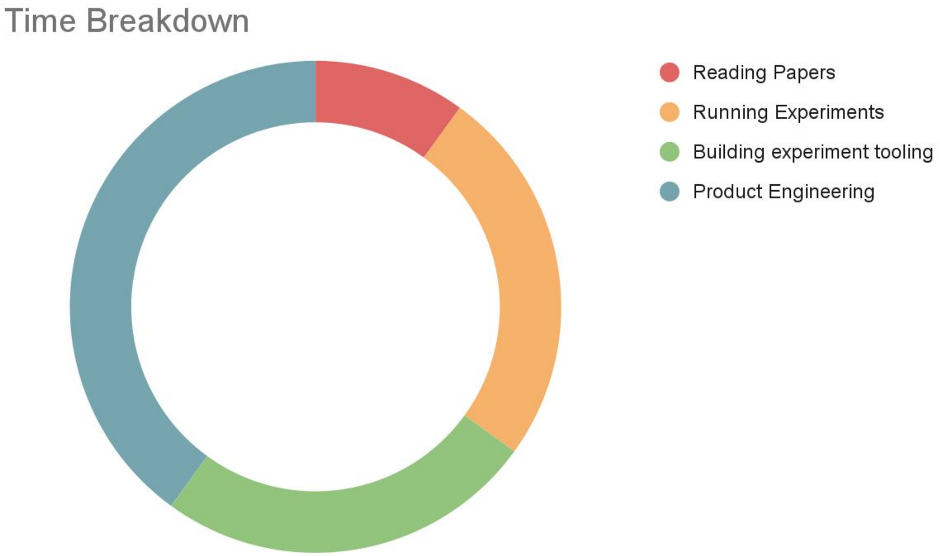
- Productionize applied research
- Build ml services
- Short-term experiments (1-2 months out)
- Monitor ml services

Challenges:

- Convert product ideas to ml problems
- Identify how to safely deploy ml models

Teams:

- Research engineers
- Data science
- Product engineers
- Product



# Tools & Technologies used

**Programming Languages**: Python, C++, Cuda

**ML Frameworks**: PyTorch, Jax, sklearn

**Common Libraries**: Hugging Face, Pytorch Lightning, Pandas, Numpy

**Experiment Tracking**: Weights & Biases, MLFlow, Tensorboard

**Other Technologies**: Docker, Kubernetes, SQL, Airflow/Prefect

# Agenda

- Senior Design High Level Timeline
- Github Projects Setup & Sprint Progress Expectations
- ML Project Design
- **Tech Lab Overview**

# Tech Labs

- You’ll likely be using technologies you aren’t familiar with to complete your SD project
- It can be difficult knowing where to start and finding time to go through introductory tutorials
- Individuals usually run into similar problems with setup, but hit them at different times during the project.
- Setting up your development environment can take time, this forces you to do so early on in the semester.

**You’ll spend next week’s lab choosing a high-level topic to focus on, and spend a few hours completing a tutorial.**

# Tech Labs - Requirements

1. You can work on these labs together, but each student must submit their own code
2. Each team must complete at least 2 different tutorials (not everyone can work on the same thing)
3. You can choose one of the suggested topics, or choose your own

# Tech Labs - Topics

1. Backends:
   a. Python backend web app (django, flask, fastapi)
   b. Node.js / Express.js
2. Frontends:
   a. React
   b. iOS
   c. Android
3. ML
   a. Google Colab
   b. Pytorch
   c. sklearn
4. IoT, Raspberry Pi, Arduino

# Tech Labs - Python Web Apps

Common python frameworks for creating backends

1. Django
   - Full-featured all-in-one web framework. Includes ORM, authentication, admin UI, etc
   - Suitable for complex web applications, but comes with a steep learning curve
2. Flask
   - Lightweight library good for rapid development
   - Lacks a ton of built-in features, relies on additional extension libraries
3. FastAPI
   - Modern, asynchronous python framework good for rapid prototyping
   - Relies on type annotations for I/O interface, self-documenting
   - Relatively new, might lack mature solutions

# Tech Labs - Python Web Apps

*Choose a framework and complete at least the first tutorial*

1. Django
   - https://docs.djangoproject.com/en/5.0/intro/tutorial01/ (parts 1-4)
   - https://code.visualstudio.com/docs/python/tutorial-django
2. Flask
   - https://flask.palletsprojects.com/en/3.0.x/tutorial/
   - https://code.visualstudio.com/docs/python/tutorial-flask
3. FastAPI
   - https://fastapi.tiangolo.com/tutorial/ (basic & advanced tutorial)
   - https://www.tutorialspoint.com/fastapi/index.htm
   - https://code.visualstudio.com/docs/python/tutorial-fastapi

# Tech Labs - Node.js / Express.js

If you're familiar with javascript, you can write your backend in javascript as well

**Node.js**: javascript runtime allowing developers to run javascript server-side

**Express.js**: a minimal, flexible web app framework for Node.js

*Choose one of the following (do both if you have time)*

- https://codexam.vercel.app/docs/project/xt/xt1
- https://codexam.vercel.app/docs/project/mernchat (fullstack + db + react)

# Tech Labs - Front Ends

- **React**: common front end for web-apps, written in javascript

- **iOS**: mobile operating system in the Apple ecosystem. Defines a framework for developing mobile apps, written in Swift. Used for frontend, can also be used for backend.

- **Android**: mobile operating system from Google. Defines a framework for developing mobile apps. Used for frontend, can also be used for backend.

# Tech Labs - Front Ends

- **React**:  (choose one, do both if you have time)
    - https://react.dev/learn/tutorial-tic-tac-toe
    - https://www.freecodecamp.org/news/react-tutorial-build-a-project/
    - https://codexam.vercel.app/docs/project/mernchat (fullstack + db + react)
- **iOS**: (complete the first, get as far as you can in the second)
    - https://www.swift.org/getting-started/swiftui/ (focused on swift ui)
    - https://developer.apple.com/tutorials/app-dev-training (thorough but very long, won't finish)
- **Android**:
    - https://developer.android.com/get-started/overview

# Tech Labs - ML

*Complete the intro to Google Colab tutorial. Then choose at least one of the pytorch tutorials OR the sklearn tutorials.*

- **Google Colab**: web-based jupyter notebook that provides free access to gpu compute
  - https://colab.research.google.com/# (intro to colab)
- **Sklearn**: library providing non-deep learning ml algorithms + training utilities
  - https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.02-Introducing-Scikit-Learn.ipynb
- **PyTorch**: library for deep learning commonly used in industry
  - https://pytorch.org/tutorials/beginner/basics/intro.html
  - https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html
  - https://colab.research.google.com/github/phlippe/uvadlc_notebooks/blob/master/docs/tutorial_notebooks/tutorial2/Introduction_to_PyTorch.ipynb
- **Datascience handbook:** useful resource on ml & datascience as a whole
  - https://github.com/jakevdp/PythonDataScienceHandbook/tree/master
  -

# Tech Labs - IoT / Raspberry Pi / Arduino / etc

- Any tutorials with a hardware component. Bring your own hardware and we're happy to help!
    - Arduino: https://docs.arduino.cc/built-in-examples/
    - Raspberry Pi: https://tutorials-raspberrypi.com/
- ROS: robotic operating system – used as part of the RTX projects
    - https://www.youtube.com/watch?v=979IZWOXC_0&list=PL8MgID9MCju0GMQDTWzYmfiU3wY_Zdjl5
    -

# For Next Week

- Complete weekly status update (get into the habit, it's ok if you don’t have much to report)
- Create September sprint tasks that would be useful for your project
- Submit Resume (blackboard)
- Continue refining project ideas
- Decide which tech lab(s) you’ll focus on next week