iD-Tech

By Nicholas Lacroix, Akuang Saechao, Tipper Truong

Table of Contents

Acknowledgements

This project was created with the help of deep learning and machine learning frameworks thanks to Emmanuel Benazera, the creator of Deepdetect, Joseph Redmon and Ali Farhadi, the creators of Darknet Yolo, and the creator of Darkflow.

We also thank Dr. Jahan Ghofraniha for his continuous feedback throughout the semester on this project and the constructive criticism given. Additionally, we also give thanks for the opportunity and platform to work and learn frameworks. With this newly gained knowledge, we have gained the the skills to solve various problems surrounding deep/machine learning that can contribute to not only our career but also to society.

Executive Summary

The purpose of this project is to create an object detection and recognition web-app that can be iterated and altered for other users' specific needs. The goal in mind is to create a baseline web-app that can be slightly altered so that future users do not have to recreate the wheel, and instead make minor (or larger) changes to evolve this project.

Additionally, this project serves as a means of learning the major frameworks used across the industry. With this knowledge and practice, we hope to further the research and technology that uses deep learning, machine learning, and data science. The boom in the previously mentioned topics serve as a motivator to learn and reach a level of proficiency through which we can talk about the frameworks in depth with colleagues with the proper vocabulary, and clearly define problem current or future problem statements.

As mentioned, the goal was to create a baseline web-app using these frameworks. Through this, we took into account real world scenarios through which our project can be applied to. These real world scenarios include license plate verification, face matching for criminals and real time data analytics through video or file mediums.

Each framework uses multiple frameworks within to create the overarching tool to solve a specific problem. Deepdetect, for example, is created from a custom version of Caffe, along with Tensorflow and XGBoost. Darknet Yolo and Darkflow also use similar frameworks to create what defines Darknet Yolo and Darkflow. In the case of Darkflow, it is an iterated version of Darknet Yolo with opencv packaged in it to optimize video processing. More in depth information is included in the Background/Introduction section of this report.

Background/Introduction

This project uses four frameworks to solve object detection and recognition problems: Annoy, Deepdetect, DarkFlow, and Darknet Yolo V1.

Annoy: From the Annoy Github:

“There are some other libraries to do nearest neighbor search. Annoy is almost as fast as the fastest libraries, (see below), but there is actually another feature that really sets Annoy apart: it has the ability to use static files as indexes. In particular, this means you can share index across processes. Annoy also decouples creating indexes from loading them, so you can pass around indexes as files and map them into memory quickly. Another nice thing of Annoy is that it tries to minimize memory footprint so the indexes are quite small.

Why is this useful? If you want to find nearest neighbors and you have many CPU's, you only need the RAM to fit the index once. You can also pass around and distribute static files to use in production environment, in Hadoop jobs, etc. Any process will be able to load (mmap) the index into memory and will be able to do lookups immediately.

We use it at Spotify for music recommendations. After running matrix factorization algorithms, every user/item can be represented as a vector in f-dimensional space. This library helps us search for similar users/items. We have many millions of tracks in a high-dimensional space, so memory usage is a prime concern.

Annoy was built by Erik Bernhardsson in a couple of afternoons during Hack Week.” (source: https://github.com/spotify/annoy see the background section.)

Deepdetect: From the Deepdetect website:

“DeepDetect (http://www.deepdetect.com/) is a machine learning API and server written in C++11 developed by Jolibrain. It makes state of the art machine learning (such as deep learning) easy to work with and integrate into existing applications. Its goal is to simplify and secure both the development and production phases by using possibly different servers and passing models from one to the other.

It originates from the need for industries, businesses and researchers to quickly fit a machine learning pipeline into existing applications, starting with well-known models, and moving toward more targeted ones while measuring accuracy.

DeepDetect allows this by coupling a generic API and a server with high performance machine learning libraries. At the moment it has support for the deep learning libraries Caffe, Tensorflow and distributed gradient boosting XGBoost.

DeepDetect implements support for supervised and unsupervised deep learning with applications to images, text and other data, and focus on simplicity and ease of use, test and connection to existing applications." (Source: https://www.deepdetect.com/meta/about/)

Darkflow: Darkflow works very similarly to Darknet Yolo, with the main difference being OpenCV is packaged into the framework. This allows easier video processing, and real time detection and recognition. For more information on how it works, see the Darknet Yolo V1 section below.

Darknet Yolo V1: From the Darknet Yolo website:

"Prior detection systems repurpose classifiers or localizers to perform detection. They apply the model to an image at multiple locations and scales. High scoring regions of the image are considered detections.

We use a totally different approach. We apply a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

Our model has several advantages over classifier-based systems. It looks at the whole image at test time so its predictions are informed by global context in the image. It also makes predictions with a single network evaluation unlike systems like R-CNN which require thousands for a single image. This makes it extremely fast, more than 1000x faster than R-CNN and 100x faster than Fast R-CNN. See our paper for more details on the full system." (Source: https://pjreddie.com/darknet/yolo/)

Problem Statement

Real-time object detection comes in play in many areas in technology. Some of these examples include self-driving vehicles, license plate identification, real-time data analytics, and so forth.

Implementing real-time object detection is important in artificial intelligence software since humans rely on software to manage their daily task or workload. One of the daily task that majority of humans have to deal with is commute. Some scenarios that could happen to commuters are being impatient while stuck in traffic or having a few drinks at a work party and unable to drive afterwards. While the idea of implementing real-time object detection sounds cool to a developer's point-of-view, there's not a lot of resources that helps them understand the idea of how real-time object detection works through machine learning.

With all the industry-leading deep-learning frameworks out there, it's hard for developers to grasp or understand how these frameworks are implemented into software since instructions or the research behind it tend to be vague.

Purpose/Motivation

Topics such as machine learning, data science, artificial intelligence, and deep learning have become popular over the years. As undergraduate students, we were curious and wanted to understand the foundations of machine learning so we could not only improve our knowledge, but also helping ourselves and other developers understand how machine learning frameworks work under industry standards.

Some practical industry frameworks used are Darknet Yolo, Darkflow, and Deepdetect. With these frameworks, we are hoping to gain exposure and have proficient understanding of how they operate, create a baseline API for web-based machine learning for other developers so they could implement their own real-time object detection application, and why certain frameworks have a better algorithmic implementation than others.

Differentiator/Contribution

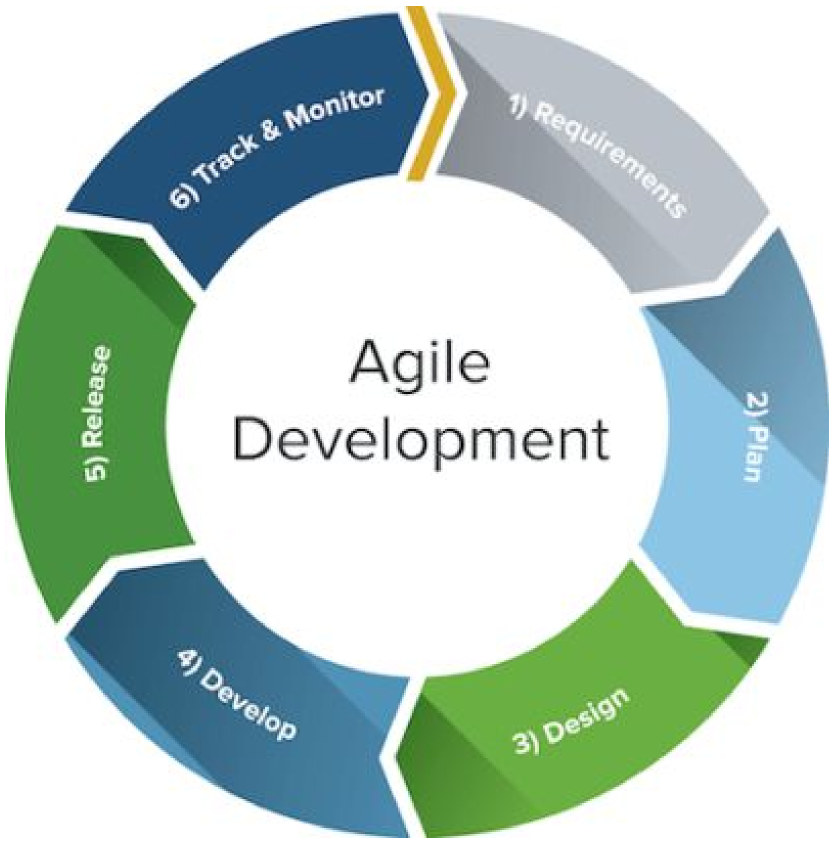
What makes our project different from what is currently released is that it uses open source frameworks, is open source itself, and has the goal of providing developers with a kit and environment where they can pick and choose which framework they want to work with while having it interfaced to Python and a completed website. Additionally, the backend can be easily edited to replace the framework, or allow the developer(s) to use their own.

When creating this project, we were looking for ways to make it accessible for people across all platforms. We came to the conclusion that a website would best fit our needs, as bug fixing is much easier through a browser than it is through an executable. If a bug were to be found on the website, we or the developer(s) who use our project can fix the bug and have it deployed to everybody instantly. Conversely, if we were to make this project purely a software application, the users would have to redownload everything for small bug fixes.

This pitfall was evident when the GUI was created. Unlike the web-based detection and recognition service, the GUI has specific dependencies and quite a bit of overhead. Although it was all packaged into docker, an application that puts all dependencies in a singular container for ease in portability across systems, we still encountered dependency issues on various operating systems. Either some computers did not have the correct Python version installed, did not or could not install specific libraries needed to display the GUI, or some operating systems could not natively run docker, making it extremely difficult to distribute the GUI.

Through experiencing this pitfall, we decided that sticking to a web application would be best to avoid these system and library dependent problems. iD-Tech is distributable, scalable, and can be transferred easily to web application frameworks other than Django and Flask.

Methodology



The methodology we followed is Agile, basically developing software in iterations or sprints. By following Agile development, we created user stories to draw a clear picture of how we want our application to be and prioritized requirements we needed to demonstrate to our target audience. Afterwards, we've compiled a plan on how to approach the development of iD-Tech and drawing out designs to make a clean and seamless UI/UX for our users. For every sprint, Nicholas was our Scrum Master and he made sure we've completed our task by the end of the sprint and helped us work around any difficulties we were encountering throughout the development process. Prior to our release, we went through rigorous testing, making sure our application is working as expected and once we've finished testing, we were ready to demonstrate iD-Tech's features to our audience.

To track our sprints, we used Taiga, a project management platform for agile developers & designers and project managers who want a beautiful tool that makes work truly enjoyable. With a clean and simple-to-use UI/UX, it helped us tracked our progression in our project easily and we highly recommend it to other teams who are following agile methodology to use Taiga as their project management platform.

Implementation & Results

Originally we had planned to purely use Darknet Yolo in combination with Django to create three specific web applications: a single image upload, batch image upload, and video upload. For the single image upload, the user would upload one image and the website would refresh and display five images, along with percentages, to describe how close of a match the five images are to the user uploaded image.

For the batch upload, the user would upload a zip file or select a folder to be used as the data comparison set. For this, a custom service had to be created through Darknet Yolo. For this service and the video service, hardware is extremely important. The better the hardware, the faster the learning process. Generally, an Nvidia GTX 980i would be sufficient to create this service and learn from the user uploaded image set. After the user uploads the zip or folder, the uploaded data would be processed by annoy and produce statistical details for each image. These statistics would produce a k-means cluster and save the nearest neighbor of each image. After that, the service is ready to be used and the uploaded zip or folder would be the dataset used by the application to detect and recognize similarities between a user uploaded image and the data set they uploaded.

The service for Videos is similar to the batch upload service. However, Darknet Yolo is written in C++ and does not interface to python. Thus, OpenCV had to be integrated into the learning service. When the Video service ran, OpenCV would separate the video frame by frame, send the frame to Darknet Yolo for image detection and recognition, then save the time stamp, frame, and box coordinates x1 y1, x2 y2 in a JSON file. This proved to be extremely inefficient and horribly slow, and was a result of processing frame by frame. A function was created to convert the coordinates to bounding boxes, and in real time display each box at the correct time stamp and frame while the video played. Additionally, users would be able to type an objects name in a search bar, press enter, and replay the video with only the bounding boxes around the searched object would be displayed.

Darkflow solves Darknet Yolo's video processing issue by processing the video through the integrated and optimized OpenCV, and saves a new video with bounding boxes that can be sent directly to the front end without having to convert coordinates to bounding boxes and display them at the correct time. Essentially, the video is processed much faster than Darknet Yolo, and we were able to save a video with the bounding boxes, name of the recognized object, and percentages and quickly redisplay the new video to the front end. This was a much more arduous task using Darknet Yolo. A demo of the Darkflow implementation, along with the source code saved on github, can be found on the last page in the Appendix section. Additionally, Darkflow allows users to upload multiple pictures to be processed. Darknet Yolo V1 does not have this functionality, thus allowing us to combine single upload and batch upload into one webpage service instead of multiple services.

Nicholas Lacroix

Nicholas, our Project Manager, helped us understand the machine learning frameworks we were implementing in our web application and assigned tasks accordingly. Since he has the most experience in implementing Darknet Yolo, Darkflow, and Deepdetect, we decided to place him as our Project Manager so he could manage our progress, help us grasp an understanding of convolutional neural networks, and advise us on how we could improve our application

Akuang Saechao

Akuang, our Front-End Engineer, helped us developed and designed our web application. He helped provide recommendations on how we could improve the UI/UX of iD-Tech and made sure users were having an enjoyable experience going through it. In addition, he also made the application mobile-friendly using a front-end framework called Bootstrap. If we ever got a chance to deploy our application, users are able to see how object detection works through their mobile devices

Tipper Truong

Tipper, our Back-End Engineer, helped us set up a micro web framework called Python Flask and integrating Darknet Yolo, Darkflow, and Deepdetect into our web application to detect objects in images and videos. Prior to integrating the frameworks to our web application, she was responsible for installing these frameworks and making sure it was running as expected.

While we all had specific roles assigned to each other, we all contributed equally and shift roles when necessary or based on how we are able to resolve a certain issue we’ve encountered.

Conclusion

In its current state, the project has achieved its purpose of being a template through which developers can alter the code and learning services to their need. The completed project uses Flask, taking advantage of small scale apps in exchange for scalability. If scalability is a requirement, Django or Node.js would be a couple of fair suggestions.

Through this project, we have gained proficient understanding of some of the major frameworks used in the industry to solve problems involving tech and ambiguous data. We have gained the vocabulary to clearly articulate how the frameworks work, how to use them to solve problems involving detection, and how to make our own learning services via Darknet Yolo. We discovered that for video processing, Darkflow is faster and more efficient that Darknet Yolo due to opencv being packaged into Darkflow, and the fact that Darkflow is interfaced to Python, the main language OpenCV uses.

On this note, we are also able to discuss the pitfalls of each framework, and which frameworks best for a specific problem. Deepdetect can process images extremely fast, and comes packaged with all the dependencies needed for get started on a project. The developer can then package their project into docker and transport it easily to other computers without having to worry about over head and dependencies.

In terms of how these frameworks on an algorithmic level, we have gained sufficient understanding and vocabulary to both read and understand their respective formulas for the learning, detection, and recognition phases of each framework. Understanding the frameworks on a mathematical level is extremely important in that it helps discussion involving which framework would be best to solve what problem in a certain timeline. This is especially specific to Darknet Yolo, where creating a service takes quite a bit of time and requires expensive hardware, whereas Darkflow provides optimized learning and detection, but sacrifices the speed in which a custom service can be created.

In the future, we hope that our project can serve as a template for others interested in learning the frameworks mentioned in this report, and to create something they can take an iterate on and create their own detection and recognition application.

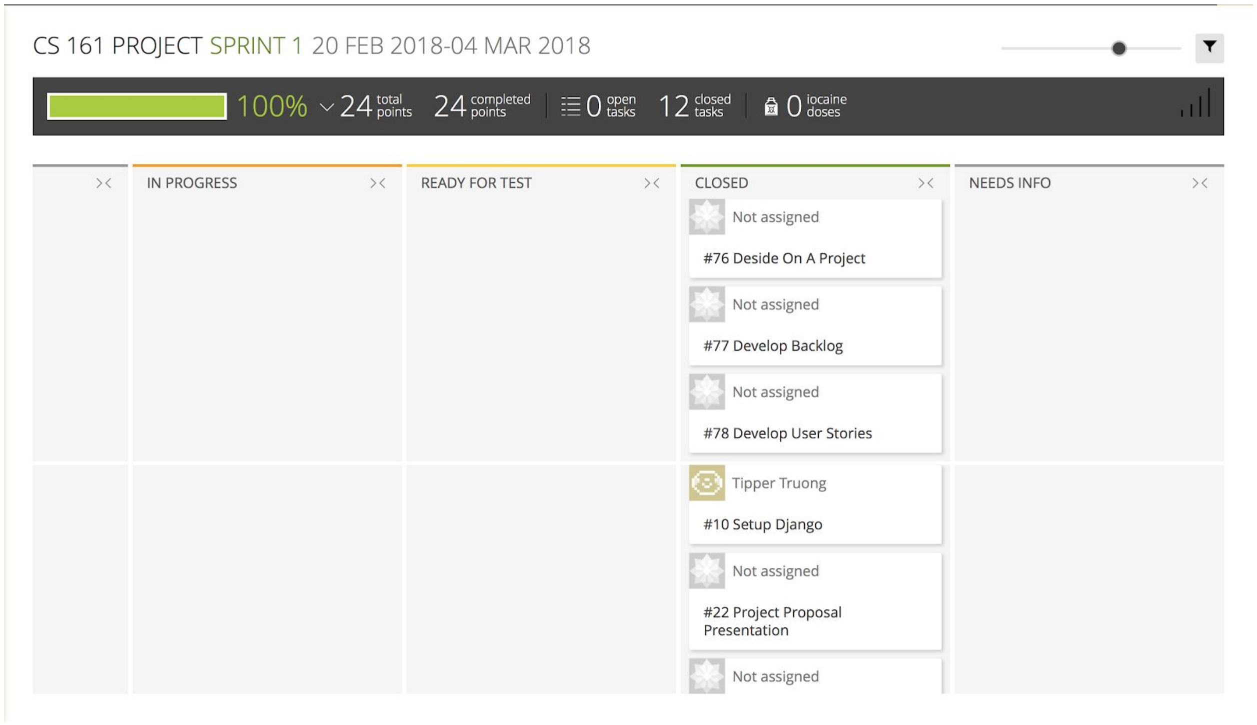
Appendix

Link to iD-Tech:
iD-Tech: https://github.com/tqtruong95/iD-Tech
Video Demo: https://www.youtube.com/watch?v=gYAZjj1jOYQ&feature=youtu.be

References:
Caffe: http://caffe.berkeleyvision.org/
Darkflow: https://github.com/thtrieu/darkflow
Darknet Yolo V1: https://github.com/pjreddie/darknet
Deepdetect: https://github.com/beniz/deepdetect
Django: https://www.djangoproject.com/
Flask: http://flask.pocoo.org/
OpenCV: https://opencv.org/
TensorFlow: https://www.tensorflow.org/
Agile Development: https://number8.com/kanban-versus-scrum/
Taiga: https://taiga.io

Sprint Reports:

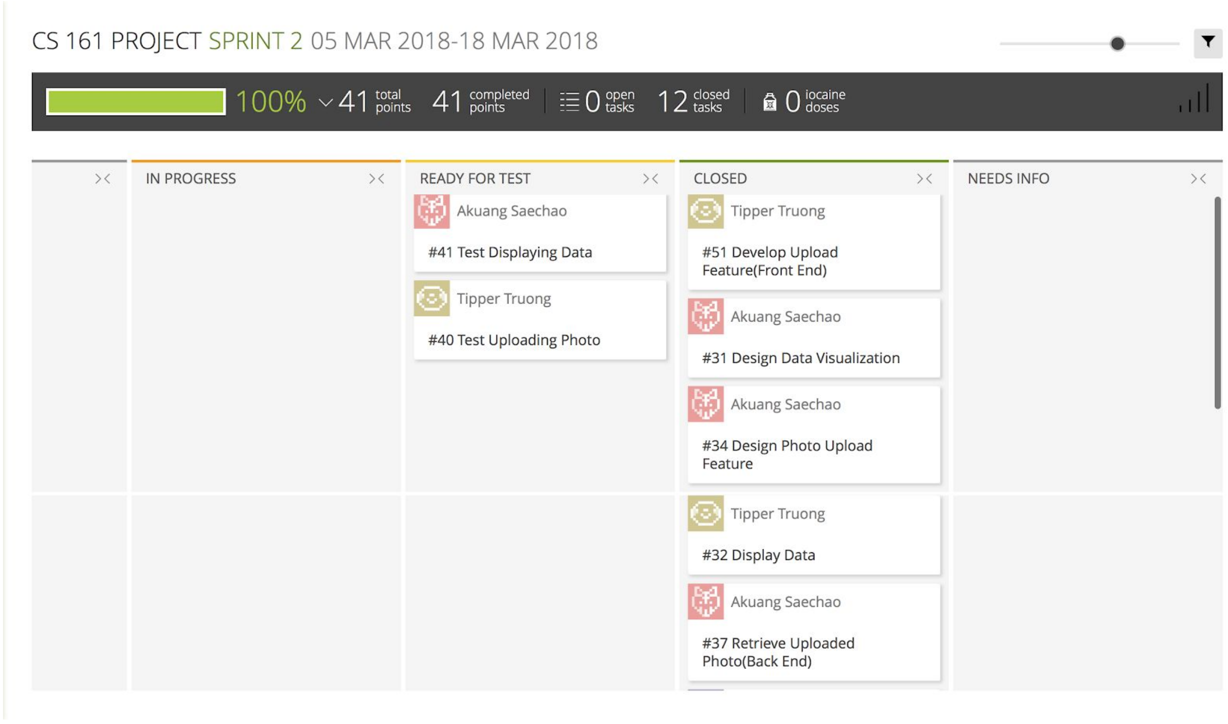
Sprint 1 report:


CS 161 PROJECT SPRINT 1 20 FEB 2018-04 MAR 2018

100% 24 total points 24 completed points 0 open tasks 12 closed tasks 0 iocaine doses

	IN PROGRESS	READY FOR TEST	CLOSED	NEEDS INFO
			Not assigned #76 Deside On A Project	
			Not assigned #77 Develop Backlog	
			Not assigned #78 Develop User Stories	
			Tipper Truong #10 Setup Django	
			Not assigned #22 Project Proposal Presentation	
			Not assigned	

Sprint 2 report:



CS 161 PROJECT SPRINT 2 05 MAR 2018-18 MAR 2018

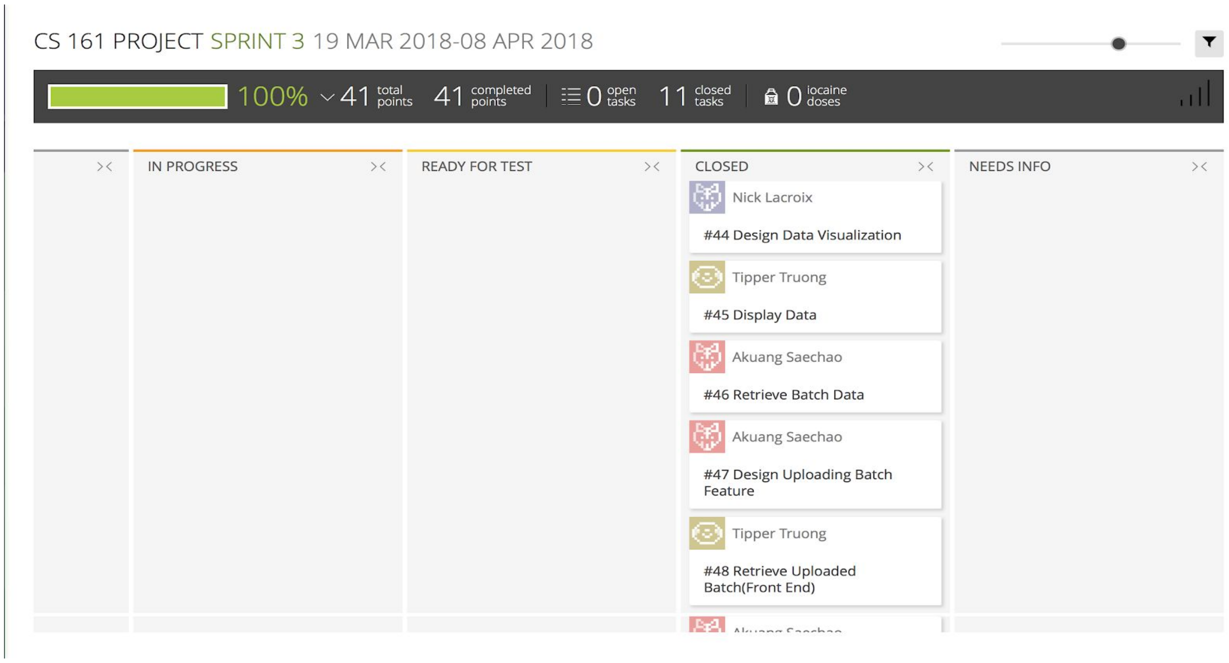
100% 41 total points 41 completed points 0 open tasks 12 closed tasks 0 Iocaine doses

IN PROGRESS | READY FOR TEST | CLOSED | NEEDS INFO

READY FOR TEST:
Akuang Saechao
#41 Test Displaying Data
Tipper Truong
#40 Test Uploading Photo

CLOSED:
Tipper Truong
#51 Develop Upload Feature(Front End)
Akuang Saechao
#31 Design Data Visualization
Akuang Saechao
#34 Design Photo Upload Feature
Tipper Truong
#32 Display Data
Akuang Saechao
#37 Retrieve Uploaded Photo(Back End)

Sprint 3 report:



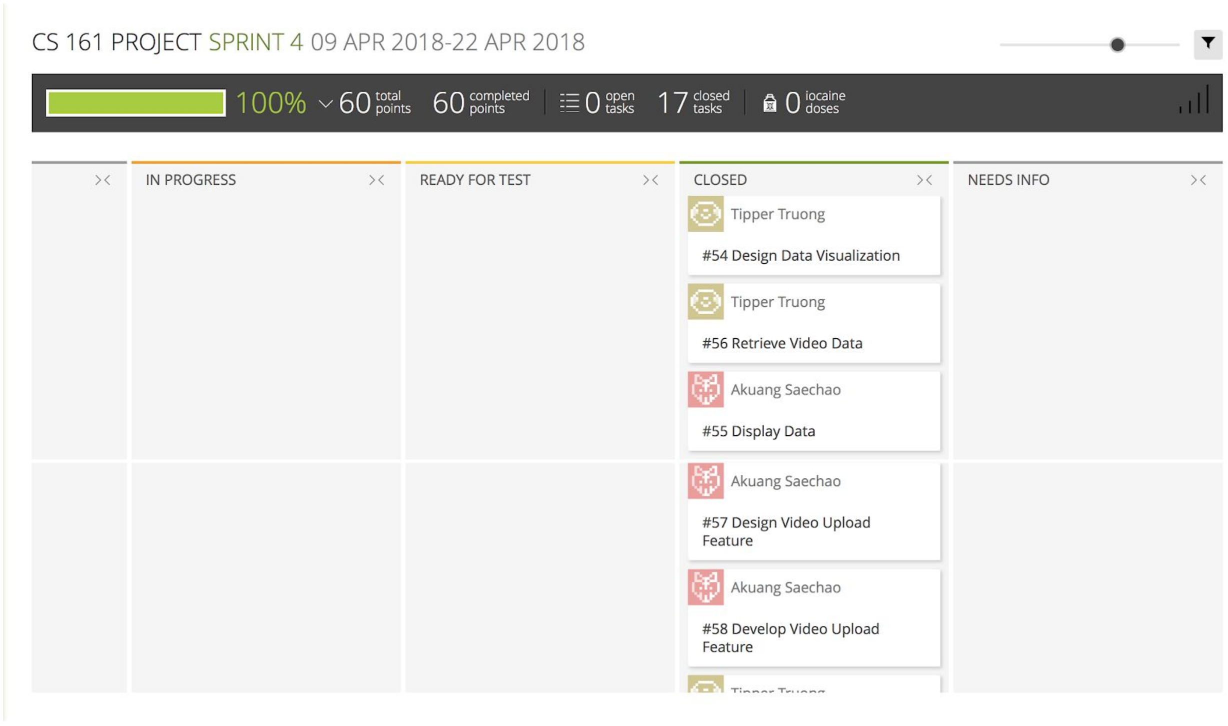
CS 161 PROJECT SPRINT 3 19 MAR 2018-08 APR 2018

100% 41 total points 41 completed points 0 open tasks 11 closed tasks 0 Iocaine doses

IN PROGRESS | READY FOR TEST | CLOSED | NEEDS INFO

CLOSED:
Nick Lacroix
#44 Design Data Visualization
Tipper Truong
#45 Display Data
Akuang Saechao
#46 Retrieve Batch Data
Akuang Saechao
#47 Design Uploading Batch Feature
Tipper Truong
#48 Retrieve Uploaded Batch(Front End)
Akuang Saechao

Sprint 4 report:

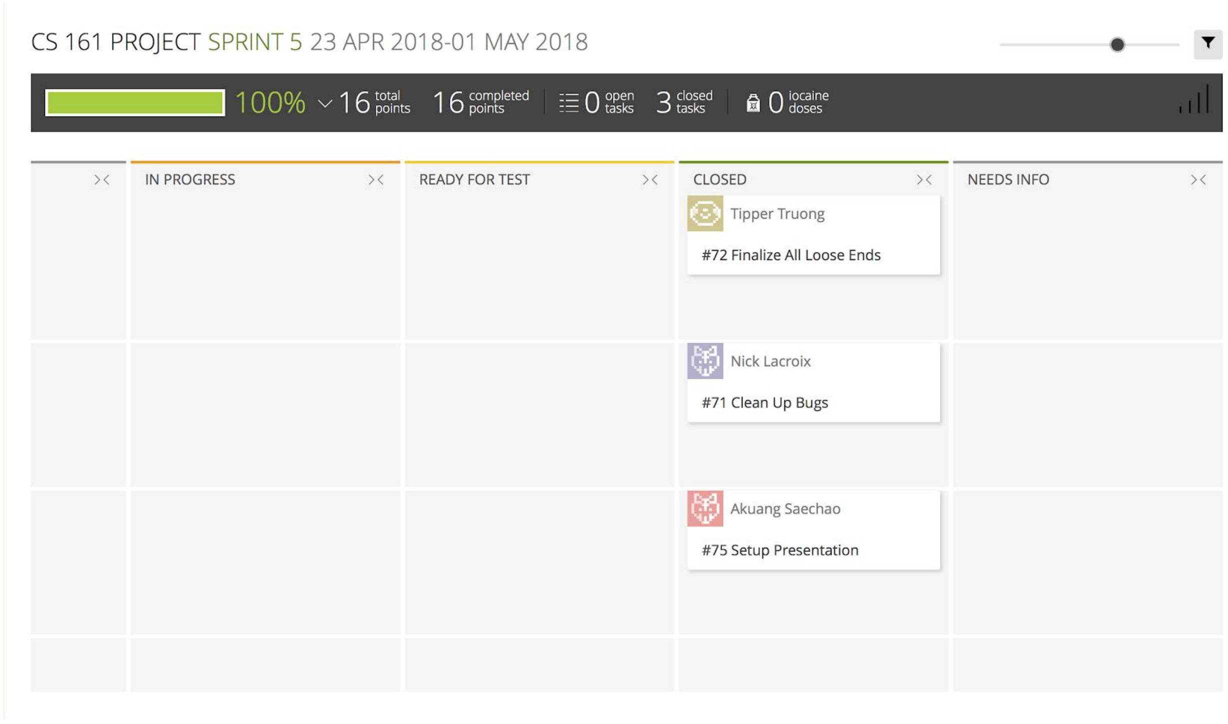


CS 161 PROJECT SPRINT 4 09 APR 2018-22 APR 2018

100% 60 total points 60 completed points 0 open tasks 17 closed tasks 0 iocaine doses

IN PROGRESS	READY FOR TEST	CLOSED	NEEDS INFO
		Tipper Truong #54 Design Data Visualization	
		Tipper Truong #56 Retrieve Video Data	
		Akuang Saechao #55 Display Data	
		Akuang Saechao #57 Design Video Upload Feature	
		Akuang Saechao #58 Develop Video Upload Feature	

Sprint 5 report:



CS 161 PROJECT SPRINT 5 23 APR 2018-01 MAY 2018

100% 16 total points 16 completed points 0 open tasks 3 closed tasks 0 iocaine doses

IN PROGRESS	READY FOR TEST	CLOSED	NEEDS INFO
		Tipper Truong #72 Finalize All Loose Ends	
		Nick Lacroix #71 Clean Up Bugs	
		Akuang Saechao #75 Setup Presentation	