Outlook

[CS 488T] Sprint 4 Report, Team 11 [stewartc]

From shelbyemailrelay@gmail.com <shelbyemailrelay@gmail.com>
Date Sun 3/2/2025 6:18 PM
To Stewart, Caleb <cstewart15@ewu.edu>

Caleb,

This report describes the activities of your EWU Senior Project team over the previous self-evaluation period (usually Saturday through Friday). It contains only public information. Private information and comments, etc. are available only to the instructor. If you notice any discrepancies or have questions, please contact Dan Tappan at dtappan@ewu.edu.

Sprint 4 Team Report

Team 11: Trademark ID & Analysis Engine

· Lane Keck
· Caleb Stewart
· Logan Taggart

Logged Hours

The team is generally free to work whenever they want during the sprint. The expectation for a team of three members is 45 hours total (15 per member) on average. However, this number will vary throughout the course.

Individual Hours:

		All Sprints							
Member	Hours	Total	Min	Max	Avg[1]	Avg[2]	Std[2]	Count[1]	Missed
Keck	9.0	29.0	3.0	9.0	7.2	7.2	2.5	4	0 (0%)
Stewart	10.5	36.5	2.0	16.0	9.1	9.1	5.0	4	0 (0%)
Taggart	9.0	23.0	3.0	9.0	5.8	5.8	2.6	4	0 (0%)
Team Total:	28.5								

[1]including and [2]excluding missed submissions for required sprints

Team Hours:

Sprint								
1	2	3	4	Total	Min	Max	Avg	Std
8.0	27.5	24.5	28.5	88.5	8.0	28.5	22.1	8.3

The following is optional descriptions of daily work that is not captured as activities below:

Taggart:

- Starting to implement cosine similarity
- Making slight changes to front end
- Deploying front and back end
- Finalizing backend API design
- Restructuring backend file strcture

Activities

Activities are member-defined units of work that are formally tracked from sprint to sprint (unlike the optional descriptions above). Every activity must be accounted for from its creation until it is completed or abandoned.

New Activities

These activities were created by during this sprint.

Keck

Activity 80: Create newly designed frontend

Frontend has recieved a new design, must implement it (two sprints expected)

Activity 81: Connect frontend with backend

After design is complete, make it work with the backend (two sprints expected)

Stewart

Activity 88: Research embedding algorithms

Research the best embedding algorithms to use with cosine similarity (one sprint expected)

Activity 89: Get cosine similarity working

Be able to detect similarities between to images by embedding the image into the vector, and then applying cosine similarity (one sprint expected)

Taggart

Activity 78: Restructuring Python backend

Getting back end file structure to be more organized and easy to understand. (one sprint expected)

Activity 79: Implementing cosine similarity

Getting cosine similarity to be able to recognize a specific logo in an image instead of just all logos. (two sprints expected)

Continuing Activities

These activities were continued from the previous sprint.

Activity 76.1: Work on matching upload image

Opened in Sprint 3 by Keck; expected to take two sprints.

Original description: Match an uploaded logo with logo found by model

Progress in Current Sprint (expected to take one more sprint): We have a way of matching a reference logo with logos in an image. We use cosine similarity and for the most part it works. Now we must test it on a variety of images

---

Activity 72.1: Setting up back end server

Opened in Sprint 2 by Taggart; expected to take two sprints.

Original description: Get Python and Flask back end to function

Progress in Sprint 3 (expected to take one more sprint): I have got the Python backend integrated and working with our trained model, and have began to setup our API routes with the Flask framework.

Progress in Current Sprint (expected to take two more sprints): All of the API routes have been setup, and I am currently working on getting the cosine similarity feature to work on it.

Completed Activities

These activities were completed during this sprint.

Activity 70.1: How to utilize YOLO model

Opened in Sprint 2 by Stewart; expected to take one sprint.

Original description: Figure out how to use all features of the YOLO model

Progress in Current Sprint: We have a solid idea of what our model can do, and its limitations

---

Activity 74.1: Applying YOLO Model

Opened in Sprint 3 by Stewart; expected to take one sprint.

Original description: Figuring out how to use and apply the YOLO model we trained

Progress in Current Sprint: We can appropriately use the YOLO model to detect logos within an image

---

Activity 75.1: Research how to apply similarities

Opened in Sprint 3 by Stewart; expected to take two sprints.

Original description: Research and apply how to apply similarities between logos found

Progress in Current Sprint: We landed on using Cosine Similarity

---

Activity 71.1: Setting up front end website

Opened in Sprint 2 by Taggart; expected to take two sprints.

Original description: Get React front end to be functional

Progress in Current Sprint: I have set up the components to have the React front end to be functional and it is now able to handle everything that our backend is currently capable of doing.

Activity 73.1: Integrate front end with back end

Opened in Sprint 2 by Taggart; expected to take two sprints.

Original description: Get React app to work with Python backend

Progress in Current Sprint: React front end can now make the required request(s) to our Python backend API routes and receive the corresponding response.

Activity 77.1: Implementing YOLO model in Python

Opened in Sprint 3 by Taggart; expected to take one sprint.

Original description: Getting boundary box to be drawn around logos in a still image.

Progress in Current Sprint: I now have it where rectangular, white boundary boxes are correctly being drawn around the logos that our model detects in a still image.

Team Reflection

This section refers to the team's collective perception of and reflection on the project over this sprint.

The instructions are: Consider the following four pairs of questions hierarchically. They are not the same question. If you think they are, then you are likely not using an appropriate breadth and depth of software-engineering thought. This course is a practical application of the aspects of product, process, and people. We are trying to account for everything: not just to create a good product, but also to learn from the process to improve the people. Reflect on the experience of the entire team collectively over this sprint. You do not need to account for all work, just two examples that are most representative of easiest and hardest.

For reference, *understand* relates to the comprehension of what needs to be done; *approach* to how you think it should be solved; *solve* to implementing the actual solution; and *evaluate* to demonstrating to yourself and your team (if applicable) that the performance of your solution is consistent with everything else in the project. Remember The Cartoon from CS 350.

Understand

Easiest:	The overall goal of the project and its requirements remain straightforward and easy to understand. Tasks such as setting up the frontend and integrating the YOLO model into the backend are clear because we are following our documented process and iterating through our work.
Hardest:	The specifics of applying cosine similarity and the math behind logo matching are the hardest elements to grasp. Luckily, we don't really need to know specifics on how exactly how certain embedding algorithms work, we just need to utilize their output for cosine similarity.

Approach

Easiest:	Setting up the backend with Flask and the front end with React has been the easiest to approach. Both technologies are well-documented and widely used, allowing for quick development.
Hardest:	Integrating a similarity detection into our existing YOLO-based detection model is hard to approach because it involves combining machine learning techniques that isn't fully prebuilt for our use case. We need to connect all our tools and algorithms together to make it simple for the user to use.

Solve

Easiest: Implementing bounding boxes around detected logos in still images and sending that data to the frontend has been relatively straightforward. using openCV, we can easily draw bounding boxes based on detected coordinates in the backend.

Hardest: Fine-tuning the similarity model and trying to find the appropriate thresholds for each embedding algorithm that make two logos 'similar'. This involves testing multiple embedding algorithms and trying to compare their performance

Evaluate

Easiest: The success of logo detection using YOLO. Either it works, or it doesn't. If a bounding box is correctly drawn around a logo, the detection is successful.

Hardest: The effectiveness of the cosine similarity approach is harder to evaluate because it requires defining and measuring similarity thresholds, handling false positives, and comparing different embedding methods.

Completion: 25%

Contact: N/A

Comments: None at this point.

Report generated on Sun Mar 02 18:18:24 PST 2025