Outlook

[CS 488T] Sprint 12 Report, Team 11 [stewartc]

From shelbyemailrelay@gmail.com <shelbyemailrelay@gmail.com>
Date Sun 4/27/2025 9:06 AM
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 12 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	8.0	83.0	3.0	9.0	7.5	7.5	1.6	11	0 (0%)
Stewart	8.5	94.5	2.0	16.0	8.6	8.6	3.4	11	0 (0%)
Taggart	9.0	79.0	3.0	10.0	7.2	7.2	2.2	11	0 (0%)
Team Total:	25.5								

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

Team Hours:

Sprint																
1	2	3	4	5	6	7	8	9	10	11	12	Total	Min	Max	Avg	Std
8.0	27.5	24.5	28.5	25.5	22.0	18.0	0.0	29.0	25.0	23.0	25.5	256.5	0.0	29.0	21.4	8.4

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

Taggart:

- Working on implementing specific logo video search
- Working on tracker
- Implementing manual compression into videos
- Working on implementing specific logo video search
- Cleaning up code

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 114: Clean up code

Clean up all parts of the code (one sprint expected)

Stewart

Activity 110: Code clean up

Make code more readable and reduce reused code. (two sprints expected)

Activity 111: Implement FAISS Specific Search

Implement the prototype code of specific search (one sprint expected)

Taggart

Activity 112: Video Compression

Manual compression of processed videos (one sprint expected)

Activity 113: Specific Logo Search

Implementing specific logo search into application (two sprints expected)

Continuing Activities

These activities were continued from the previous sprint.

Activity 98.1: Video Processing

Opened in Sprint 9 by Taggart; expected to take three sprints.

Original description: Get YOLO model to work on videos

Progress in Sprint 10 (expected to take two more sprints): The model is now working on videos and detects logos within frames, but still needs to be actually implemented into the backend code.

Progress in Sprint 11 (expected to take three more sprints): General logo search is implemented but needs more work to send video from back end to the front end in a reasonable time amount.

Progress in Current Sprint (expected to take two more sprints): General logo search for videos now works efficiently with sending and compression, I am starting to work on specific logo search for videos.

Activity 105.1: MOSSE Tracker

Opened in Sprint 10 by Taggart; expected to take two sprints.

Original description: Implementing MOSSE tracker to interpret logo boundary boxes in between analyzed frames.

Progress in Sprint 11 (expected to take two more sprints): MOSSE tracker now works in testing code and tracks the object based off of the initial boundary box, but it still needs to be implemented into our backend.

Progress in Current Sprint (expected to take two more sprints): I ran into some issues and had to switch our tracker to be a CSRT tracker, conversion was fairly simple as they were both from OpenCV. I am working on some errors we are having with losing track of spacial area, and the tracker box expanding without any apparent reason.

Completed Activities

These activities were completed during this sprint.

Activity 93.1: Style new stats that are returned

Opened in Sprint 7 by Keck; expected to take two sprints.

Original description: Make the stats look better

Progress in Current Sprint: Stats are returned

Activity 108.1: Fine tune our website

Opened in Sprint 11 by Keck; expected to take two sprints.

Original description: Update designs, make sure everything works

Progress in Current Sprint: The website has been updated

Activity 102.1: Implement FAISS General Search

Opened in Sprint 10 by Stewart; expected to take one sprint.

Original description: After FAISS works in interactive notebook, implement it into actual backedn

Progress in Current Sprint: General video search with FAISS is working, and probably as efficient as it is going to get

---

Activity 104.1: Implement FAISS specific search

Opened in Sprint 10 by Stewart; expected to take two sprints.

Original description: After FAISS works in interactive notebook, implement it into actual backedn

Progress in Current Sprint: General video search using FAISS is properly implemented

---

Activity 107.1: Make FAISS implementation more efficient

Opened in Sprint 11 by Stewart; expected to take two sprints.

Original description: Make the FAISS implementation from backend to frontend more computationally efficient

Progress in Current Sprint: Figured out how to stream the video from the back end to the front end.

Activity 99.1: FAISS

Opened in Sprint 9 by Taggart; expected to take two sprints.

Original description: Learning and using FAISS to detect when logos in different frames are the same.

Progress in Current Sprint: FAISS works good and has been tested throughly.

Activity 109.1: Getting video to send to frontend

Opened in Sprint 11 by Taggart; expected to take one sprint.

Original description: Encoding video to send to the front end

Progress in Current Sprint: Video now sends properly as its own route and uses Flask's built in send_file function.

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 easiest aspect to understand is what we need to update within our front end codebase to better make it match our goals of displaying clear metrics to users based upon the results of the processed images or videos.
Hardest:	The hardest aspect to understand is why when we are using the CSRT tracker it is sometimes losing track of the spacial area that we initialized with it. It is difficult to understand why because it is a very complicated algorithm behind the scenes and is taking a lot of research/effort to understand what we need to do to minimize these errors.

Approach

Easiest: The easiest thing to approach has been implementing the general video search from our Jupyter Notebook into our actual application. We were able to easily approach how we were going to do it because we made sure that the code and logic in the Notebook was very solid and had a similar structure to the rest of our actual application.

Hardest: The hardest part of the current work to approach is figuring out how to minimize false positives within our processed videos. We are having trouble knowing what exactly is going on in the actual detection process, so it has made it difficult for us to devise a way to make it more accurate for the videos.

Solve

Easiest: Right now, the easiest aspect to solve has been efficiently verifying that a logo found in the video matches the reference logo that we pass in. Since we are using FAISS we have been able to easily perform a verification just by checking the FAISS vector database, and if it is existing in there we recognize it as a valid match. This helps us because we only have to perform additional similarity computations if we don't have the logo inside of the database.

Hardest: The hardest to solve part of the work currently is getting our backend to work and startup at a faster speed. We are attempting to use a library called Nuitka to recompile our Python code into C code for speed purposes, but we haven't been able to test whether it will solve our issues because it has been taking a very long amount of time to compile. Due to us not getting our compiled executable yet, we are having a difficult time seeing if this will solve the problem, or if we will need to make changes to our codebase.

Evaluate

Easiest: The easiest thing to evaluate currently is whether our processed video that we are sending to the front end is compressing into a size that we expect it to, since we are saving the outputted video and can simply view the file size.

Hardest: We are finding the hardest part to evaluate right now is what is causing our CSRT tracker to not always work as is expected, as it sometimes does not lose track of the space but believes the space that it is tracking is getting larger when it is actually not. This is causing the boundary box to be bigger than it should be, but we are having a tough time evaluating what is causing it to do so because it is from an old version of OpenCV, it is not easy to find documentation for, and there are not easy to understand logs of what it is detecting.

Completion: 80%. We feel that since we have most of the heavy logic completed and implemented we will successfully complete the project on time.

Contact: N/A

Comments: We do not have any issues or concerns for the project at this time!

Report generated on Sun Apr 27 09:06:00 PDT 2025