Capstonian: An HDSI Capstone Search Gallery

William Hu
wyhu@ucsd.edu

Mandy Xu
max002@ucsd.edu

Colin Jemmott
cjemmott@ucsd.edu

Abstract

A monumental aspect of the UC San Diego Halıcıoğlu Data Science Institute (HDSI) program is the senior capstone. Fourth year DSC students collaborate with UCSD and industry mentors through a two quarter-long project that is later showcased at the end of the winter quarter. These projects are seen as a culmination of the knowledge that students obtain from the previous years. While the annual capstone showcase is a great way to celebrate the graduating class and their hard efforts, there is no easy or accessible way to display their work past the showcase. Previous capstone projects can only be seen by manually navigating through archived links. It is difficult for current third year DSC students to get a look at past projects as they navigate their capstone domain selections. Our project aims to remedy this issue by creating an efficient, user-friendly and publicly accessible search tool for all senior capstone projects. This solution will allow current students, faculty, and industry professionals to easily explore past projects. First, we scraped the public data from the currently available archives. Next, we extracted the relevant information from each project. We then employed Elasticsearch, which ingested all the data and efficiently and accurately returns relevant projects when queried, and tuned it using NDCG, an evaluation metric. To make our project publicly accessible, we use React in JavaScript to implement our frontend website, FastAPI to manage our API endpoints, and finally a virtual machine in Azure (cloud) to deploy our full stack website.

Website: `https://mxndyxu.github.io/DSC180B-Capstone-Search/`
Code: `https://github.com/mxndyxu/DSC180B-Capstone-Search`

1 Introduction

Traditionally, search and information retrieval have only been applied to select domains, particularly those engaging in education and research. However, in more recent years, almost every aspect of our lives has benefited with the implementation of information retrieval systems from retrieving specific emails to searching for a video on YouTube. A variety of techniques have been employed to make information retrieval incredibly accurate, efficient, and adaptable including term frequency–inverse document frequency (TF-IDF), subject indexing, and semantic search (Singhal et al. 2001). While these techniques have been incredibly successful in their respective domains, there are still areas lacking comprehensive search functionalities, one being senior capstones within the HDSI department at UCSD.

Currently, capstone projects are separated by year in a list-type format that's hard to access unless you are directly given the archive source. The absence of a search tool for these projects makes it difficult for users to efficiently navigate through various projects based on their desired parameters.

We hope to provide a more visible gallery to showcase these projects through ease of navigation that also serves as an efficient way to discover new projects. To do so, we first scrape the data. The data that we are using is text-based data scraped from a standard HTML website. The type of information that we have access to include the project's name, year, domain, members, mentors, GitHub repository, report, website, and poster. Using this data, we are able to deliver an aggregated source for all of the capstone projects as well as a comprehensive and detailed overview for each project. More details are addressed in Section 2. We then implemented a search engine that takes in a string query and/or preset filters and swiftly and accurately returns projects using keyword and semantic search techniques. Finally, we package our search engine with a user-friendly interface in order to deliver a full stack website that streamlines the exploration of capstone projects. With our approach, we aim to improve the navigation of capstone projects, enhancing the overall user experience.

2 Dataset

2.1 Raw Data

We sourced our data from the official Data Science capstone website, which organizes the raw data by each capstone year. At the time of this project, there are four capstone years available spanning from 2019 to 2023. Given the relatively recent establishment of the UCSD Data Science program, there are additional components to consider. For example, the year 2020-2021 introduced a written report and the year 2022-2023 introduced the poster requirement. As a result our data scraping process had to be individually processed by year and later consolidated.

Across all years, we set out to obtain these relevant features for each project by using Python and BeautifulSoup: year, domain, project title, students involved, mentors and companies involved, GitHub repository URL, website URL, report URL, and poster URL. Moreover, using GitHub's REST API, we extracted each project group's GitHub contributor profiles, the language breakdown of the repository, and the README file. Additionally, to extract the contents of each project's report, we used the Python package PyPDF2 to extract the text and images of each report as well as gdown to do the same from Google Drive as some reports were stored there. We stored this data using Python's pickle module (”pickling”) to avoid rerunning the data extraction script. Finally, using the pickled report contents data and stored README contents, we included an additional feature by transforming the report and README contents into their respective summaries using a mix of OpenAI's GPT-3.5 and GPT-4 API and feeding it the query ”Summarize this: <text contents>”. Few projects didn't have a report. In this case, we returned ”Report summary unavailable”. As mentioned earlier, we were unable to extract all of this data for every project due to the evolving project requirements for each year.

In the following sections, we discuss the specifics and inconsistencies of our provided data as well as any deviations to our methods of data scraping as outlined above.

2.1.1 2019-2020

The 2019-2020 year introduced the senior capstone project. There were only four mentors, five domains and a total of 27 projects. Each project contained a static website and GitHub repository. One thing to note for this year was that the all GitHub links redirected to one overarching GitHub repository containing all capstone projects as folders rather than separate repositories. Because of this, we were unable to use GitHub's API to extract the contributors and language breakdown for each project.

2.1.2 2020-2021

The data from the 2020-2021 year builds upon 2019-2020. We were able to extract a project's report alongside the website and GitHub repository. Similar to 2019-2020, all projects were a folder within the overarching GitHub repository so we again couldn't extract a project's contributors and language breakdown. There were a total of 72 projects and 18 domains. Specific companies also were introduced as mentors (e.g. Intel and Viasat) but no specific mentor within those companies were displayed.

2.1.3 2021-2022

The 2021-2022 capstone dataset was identical to the 2020-2021 dataset in requirements and overall structure. There were a total of 30 domains and 46 projects. Additionally, GitHub contents were transitioned from folders to individual repositories, meaning we were able to obtain a project's members and programming language composition.

2.1.4 2022-2023

The 2022-2023 year had a total of 66 projects and offered the most data and deliverables. This was the first year where all capstone projects were required to have a report, website, poster, and consistent code structure (for our purposes, a README file) in GitHub. Moreover, unlike previous years, specific industry mentor names were now available. Therefore, we did not have to modify any of the methods described above. Similar to 2021-2022, every project had its own GitHub repository.

2.2 Data Anomalies

Within inconsistencies across capstone years, there were also some outliers within each year. We describe these data anomalies below:

- One project in the 2019-2020 year contained only the names of the team members, lacking a project title or any content. This entry was omitted from the final data due to the absence of any useful information.
- Some projects were identical but consisted of different student teams. For example, two projects in the 2021-2022 year had the exact same content–project titles, mentor, website, GitHub repository, and report–but were displayed as different groups when we were scraping our data. Another project in the 2022-2023 year was the same with the additional identical poster. These entries were kept separate as to not manipulate the original data.
- Many projects throughout the years provided no GitHub repository URLs which limited our ability to obtain their README files. Additionally, some projects provided a stale GitHub repository link as their pages were no longer available. We decided to keep the URLs within our dataset since they were provided to us but let users know that certain links may not work.
- While the GitHub API provided code language breakdowns, we are not sure how accurate they are upon further investigation.

2.3 Processed Data

During the data collection, different aspects of each capstone required different scraping/collecting procedures and we created separate dataframes for the various aspect of each project. We ended up with a total of six dataframes:

1. An overview of the project that included the project's ID, year, domain, and project title.
2. Every student to have participated in the Data Science capstone, as identified by their project ID.
3. Every mentor, industry partner, and available specific industry mentors to have participated in the Data Science capstone, as identified by their project ID.

4. Links to each group's GitHub page, report, website, and code if available, as identified by their project ID.
5. A group's GitHub data if it was available, as identified by their project ID. Specifically the repository URL, contributors' usernames and link to their profiles, the repository's programming language breakdown, and the raw and summarized contents of the README file.
6. A group's report data if it was available, as identified by their project ID. Specifically, the report URL, and the raw and summarized contents of the report.

We aggregated all of this data into a single Pandas dataframe that included all of our desired features, wherein every row represented an individual project. This helped transform the data into a format that our search engine could easily ingest and format. This dataframe is pickled and is also in our GitHub repository.

3 Methods

After processing our data into an organized and readable format, we began implementing our search tool. To do so, we employed Elasticsearch, a free and open-source distributed search and analytics engine designed for scalability and speed. At the same time, we applied React, a JavaScript library for building user interfaces. To connect the two endpoints, we used FastAPI, a web framework for building APIs with Python.

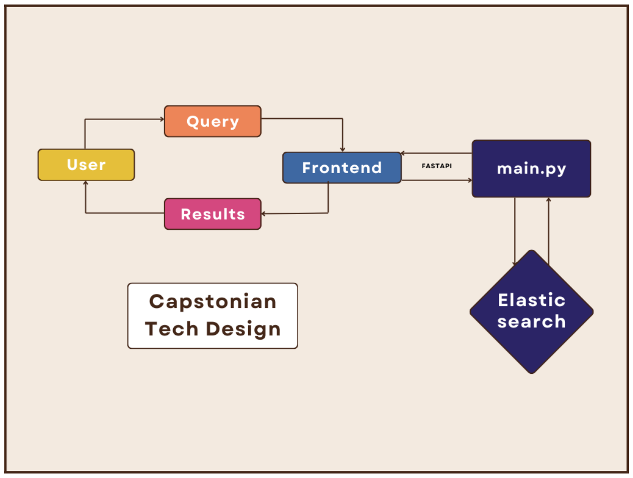


Figure 1: The technical design of Capstonian.

Below, we discuss in more detail our methodologies behind the tools that we used.

3.1 Elasticsearch

Our Elasticsearch ingested traditional keyword matching as well as semantic search elements for both queries and filters.

For the keyword matching portion, Elasticsearch has its own built-in BM25 algorithm which is a ranking function used in information retrieval to score the relevance of documents to a given search query. We applied this to both the query as a phrase and the individual words within the query. On top of that, we boosted the specific fields: year, domain, student names, and mentor names. We wanted to make sure these fields within the query were prioritized in the top results.

For the semantic search portion, we encoded the project titles, and summarizations of the READMEs and reports with SBERT (Sentence-BERT), which is a variant of the popular BERT (Bidirectional Encoder Representations from Transformers) model, a natural language processing (NLP) model developed by Google to capture bidirectional contextual information from text. The query is then encoded into the same embedding space as the document encodings mentioned above. To evaluate which results to return to the user, we use cosine similarity which measures the similarity between two vectors (in this case, the query and the encoded documents).

In addition to users being able to query for results, we also provide the ability to filter the capstone year, domain, and/or mentor. This method allows for only the desired filters to be returned. For example, if the user is interested in only the capstones from 2022-2023, they can filter for that and only get those 66 projects.

3.1.1 Evaluation

To evaluate our search engine, we employed normalized discounted cumulative gain (NDCG), an evaluation metric used in information retrieval and machine learning. It measures the quality of our search results by considering both relevance and ranking position.

We created 15 total search possibilities, taking into consideration the four search scenarios we identified:

1. **Query and filters:** the user is able to search based on their desired query and one or more filters.
2. **Query and no filters:** the user is able to search using only their desired query.
3. **No query and filters:** the user is able to search using on their desired filters.
4. **No query and no filters:** the user is unable to search without a query or filters, an error pops up prompting the user to input some form of query.

Taking into account only the first ten results, we hand labeled each result's relevance using the evaluation from the CodeSearchNet Challenge, a paper that explores semantic code search:

- 0 = result is totally irrelevant
- 1 = result is a weak match
- 2 = result is a strong match
- 3 = result is an exact match

We obtained an NDCG score of 0.85 on our baseline search engine and after fixing and finetuning filtered search and student names, we obtained a final NDCG score of 0.98.

To evaluate the search experience of our website, we created a Google form asking these questions:

1. Are you a UCSD student (DSC or non-DSC), UCSD faculty (HDSI or non-HDSI), or non-UCSD affiliate?
2. If you are a UCSD student, what year are you?
3. On a scale of 1 (confusing) to 5 (intuitive and easy to use), how was the UI/UX of the website?
4. Do you have any feedback on improving the UI/UX? If so, please fill this question out.
5. Is there anything else you'd like to see on this website?
6. On a scale 1 (confusing) to 5 (intuitive and easy to use), how was the search experience of the website?
7. Were there any hiccups/bugs you encountered during the search experience?
8. Is there anything you'd like to see implemented/improved upon on our search?

We received feedback from 12 people ranging from faculty to first to fourth years students and a range of disciplines including data science, computer science, cognitive science, and design. We also linked this feedback sheet in our About page on the actual website. We will discuss the feedback within the Section 4 section of the report.

3.2 React.js

We believed that React would provide us with the necessary tools to an intuitive and clean user interface. The most important aspects of our website is structured with a home page, search results page, and individual project pages.

The home page includes a search bar to enter in the user's queries and three drop down filters, specifically for a capstone's year, domain, and mentors.

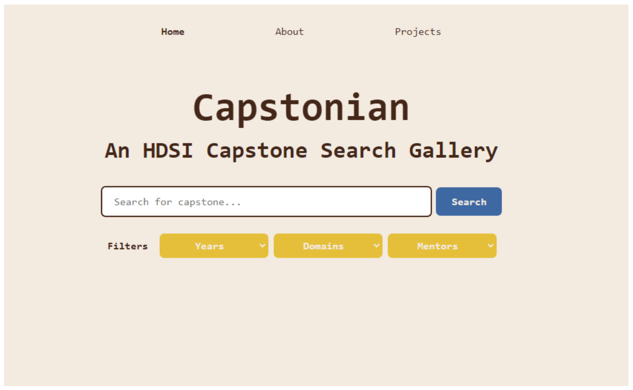
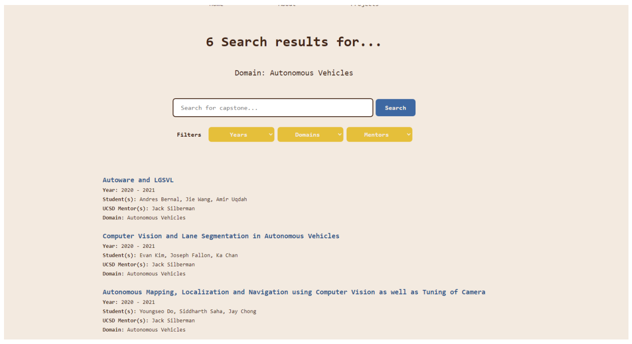


Figure 2: Home Page

For every result deemed most relevant to the user's query, we offer a concise, at-a-glance overview of a project's details: the project name, year, students involved, UCSD or industry mentors involved, and domain. Each project synthesis is redirected to the their respective pages that present more detailed information (e.g. project summary) upon a user's click.



6 Search results for...

Domain: Autonomous Vehicles

Search for capstone...

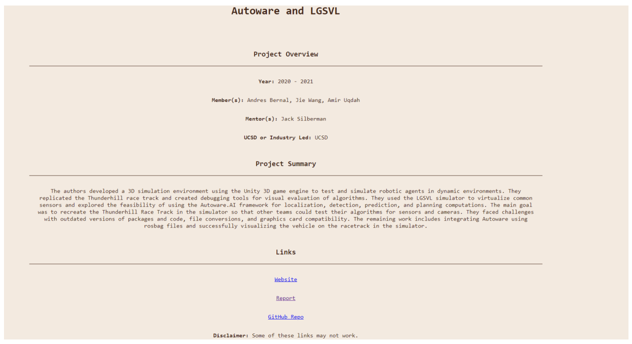
Autoware and LGSVL
Year: 2020 - 2021
Student(s): Andres Bernal, Jie Wang, Amir Uqdah
UCSD Mentor(s): Jack Silberman
Domain: Autonomous Vehicles

Computer Vision and Lane Segmentation in Autonomous Vehicles
Year: 2020 - 2021
Student(s): Evan Kim, Joseph Fallon, Ka Chan
UCSD Mentor(s): Jack Silberman
Domain: Autonomous Vehicles

Autonomous Mapping, Localization and Navigation using Computer Vision as well as Tuning of Camera
Year: 2020 - 2021
Student(s): Youngseo Do, Siddharth Saha, Jay Chong
UCSD Mentor(s): Jack Silberman
Domain: Autonomous Vehicles

Figure 3: Search Results Page

For every individual project, we display the same information as the search results page (project title, year, students, and mentors) as well as a project summary (extracted from the report using OpenAI from earlier) and URLs to each group's GitHub page, report, website, and poster (if available). In addition to this, we include the GitHub repository's README information, language breakdown of the GitHub repository and the GitHub contributors along with links to their personal profiles.



Autoware and LGSVL

Project Overview

Year: 2020 - 2021

Member(s): Andres Bernal, Jie Wang, Amir Uqdah

Mentor(s): Jack Silberman

UCSD or Industry Led: UCSD

Project Summary

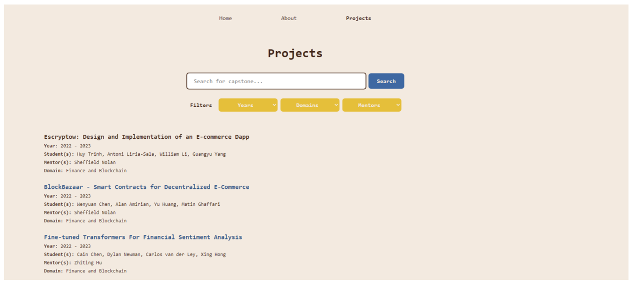
The authors developed a 3D simulation environment using the Unity 3D game engine to test and simulate robotic agents in dynamic environments. They replicated the Thunderhill race track and created debugging tools for visual evaluation of algorithms. They used the LGSVL simulator to virtualize common sensors and explored the feasibility of using the Autoware.AI framework for localization, detection, prediction, and planning computations. The main goal was to recreate the Thunderhill Race Track in the simulator so that other teams could test their algorithms for sensors and cameras. They faced challenges with outdated versions of packages and code, file conversions, and graphics card compatibility. The remaining work includes integrating Autoware using rosbag files and successfully visualizing the vehicle on the racetrack in the simulator.

Links

Disclaimer: Some of these links may not work.

Figure 4: Individual Projects Page

We also created an all projects page in case the user wanted to browse all the capstones. This page includes all projects in the same format as the search results page as well as the same search bar from the home page to be able to query and filter projects without having to re-navigate back to the home page.



Projects

Escryptow: Design and Implementation of an E-commerce Dapp
Year: 2022 - 2023
Student(s): Huy Trinh, Antoni Liria-Sala, William Li, Guangyu Yang
Mentor(s): Sheffield Nolan
Domain: Finance and Blockchain

BlockBazaar - Smart Contracts for Decentralized E-Commerce
Year: 2022 - 2023
Student(s): Wenyuan Chen, Alan Amirian, Yu Huang, Matin Ghaffari
Mentor(s): Sheffield Nolan
Domain: Finance and Blockchain

Fine-tuned Transformers For Financial Sentiment Analysis
Year: 2022 - 2023
Student(s): Cain Chen, Dylan Newman, Carlos van der Ley, Xing Hong
Mentor(s): Zhiting Hu
Domain: Finance and Blockchain

Figure 5: All Projects Page

Other small/minor features we had added to ensure a smoother user experience included:

- The enter key on a user's keyboard equating to clicking the website's search button.
- Indicating that a link has already been clicked on through a color change of the presented project name.
- Adding a popup to prompt the user to provide at least a query or filter if the search button was clicked with no search or no filters selected.
- Underlining and changing the cursor type when the user hovers over a clickable link (e.g. project title in the search results page).
- External links opening to a new tab instead of opening on the Capstonian tab.
- Displaying the number of search results returned.

3.3 FastAPI

We used FastAPI to implement our backend system for our search engine application and connect our endpoints. The backend interacts with an Elasticsearch server to handle search queries and filter results based on specified criteria. The integration of FastAPI and Elasticsearch enables seamless communication between the backend and frontend components.

Our main.py file serves as the backend for our search engine application. It consists of the following components:

- Elasticsearch Connection
  - An Elasticsearch instance is established to interact with the Elasticsearch server.
  - The Elasticsearch server's URL and the index name are provided during initialization.
- CORS Middleware
  - Cross-Origin Resource Sharing (CORS) middleware is added to facilitate communication between the backend and frontend applications running on different ports.
- API Endpoints

- The search route handles search requests and accepts optional query parameters (query, year, domain, mentor) for filtering search results.
- The search() method of the Elasticsearch instance is called with the provided parameters to retrieve search results.
- Search results are returned as a JSON response.

3.4 Azure

To host our website, we used Microsoft Azure. Specifically, we spun up a virtual machine (2cpus and 8gbs of RAM), cloned our GitHub repository, and ran the website while opening the specific ports so it is accessible online.

4 Results

4.1 Elasticsearch

For the Elasticsearch, we created a test set of example queries and graded our results with NDCG. We obtained an NDCG score of 0.98 for our final search engine. Our initial results was 0.85 and we altered the ingestion process to add key-term functionality to more fields and re-worked our filtering process to improve the results. Our Elasticsearch works really well on the key-terms and filtering. Specific mentor queries like "Colin Jemmott" and "Long Le" will yield the most relevant search results on the top, but due to the nature of our querying, if you were to try to query "Long Le and Justin Chu", the results are not relevant. Further improvement on the recognition of multiple key-terms/names can be improved.

Another issue we faced was the lack of relevant search results for semantic queries. We heavily prioritized the term matching for the queries while the semantic embeddings were limited to only three fields: report text summarizations, readme summarizations, and project titles. Oftentimes the Elasticsearch engine weighted the match on individual words of a query to be much higher, rather than the semantic embeddings. For example, querying "Super User" would be returned as the top result since there is a project with the title containing that query ("ForumRec - A Question Recommender for the Super User Community"). However, altering the query to "Power User" doesn't return the same result even though semantically, they have the same meaning.

4.2 Website Feedback

We had two main points of feedback. The first was that our website is very text based. The second was that many users didn't know what to search up. This was particularly the case for non-DSC majors which makes sense. This website is catered to DSC members but possible solutions is to add general categories to the home page that users can click on such as a certain year or an interesting domain.

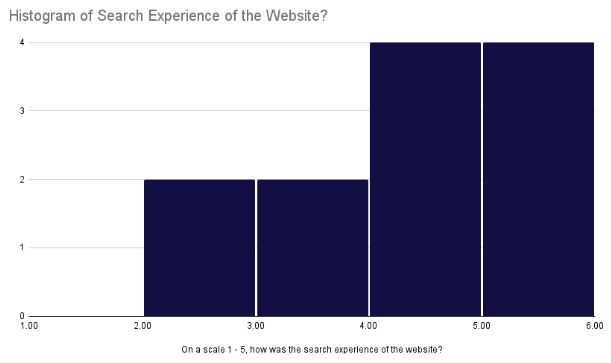


Figure 6: Feedback on search experience. The average score is 3.83.

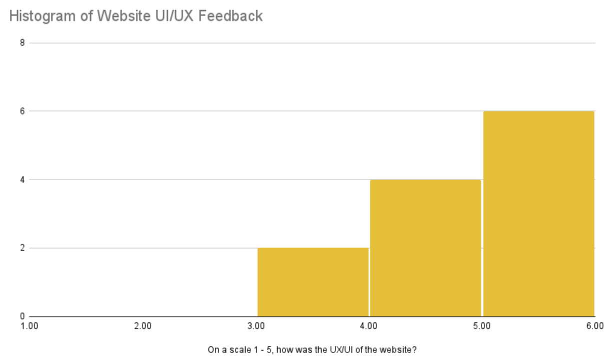


Figure 7: Feedback on user experience. The average score is 4.3.

From the figures, we can see feedback on both experiences skew positive. One thing to note is that we asked for some feedback before certain features were implemented which may account for lower scores. However, we do acknowledge that our product has room for improvement (discussed more in Section 5.1).

5 Discussion

Overall, we believe to achieved our goal for our capstone. We created a website that allows people to easily explore all of the past HDSI capstone projects. Future capstone students, faculty, and everyone else would be able to hopefully gain a better understanding of what a capstone project entails. Our search function works reasonably well and helps people obtain their relevant search results. We hope that this project can be maintained past this year to be a persistent database of all past, present, and future HDSI capstones.

5.1 Future Work

That being said, there remain several avenues for future exploration and development in this area. Some of the points below include reach goals we did not have the time for as well as user feedback.

1. Finetune Elasticsearch further to improve on student name hits. Currently, our search engine is accurate in returning queries including one student name but not accurate in returning more than one student name (even if the student names are in the same project group).
2. Query autocomplete: Being able to predict what the user is typing to help them narrow down their search and save time.
3. Suggested topics to explore on the home page. This is especially useful for non-DSC majors or those unfamiliar with the HDSI senior capstones.
4. Include images to individual project pages. Extracting report images to include on project pages and rotate through them. This helps separate the text heavy data that we obtained.
5. Multiple filters within the same category. For example, being able to filter 2021-2022 and 2022-2023 as opposed to only 2021-2022. Currently, our implementation only supports the latter.
6. Dynamic filtering: As the user chooses their desired filters, remove incompatible filters to avoid not being able to return certain results. For example, if the user chooses to filter the mentor ”Colin Jemmott”, remove the years (all years but 2020-2021) and domains (all domains but ”Recommender systems”) that ”Colin Jemmott” did not mentor.
7. Clean up the domain filter by broadening the scope of topics. A lot of domains are similar in nature due to the evolving domain names throughout the years. For example, aggregating ”graph neural networks” and ”Graphs and Deep Learning”.
8. Displaying the ”best” projects. During the capstone showcase typically held at the end of winter quarter (March), attendees are able to vote on their favorite projects. Being able to display this is an additional way to showcase projects and enhance project exploration.
9. Improve on non-relevant results scenario. We got mixed feedback between showing no results (with a message) or showing alternative results.

References

Singhal, Amit et al. 2001. “Modern information retrieval: A brief overview.” *IEEE Data Eng. Bull.* 24(4): 35–43