

THE UNIVERSITY OF SYDNEY

FACULTY OF ENGINEERING

UG Capstone Projects

COMP3888\_COMP3988\_SOFT3888\_INFO3600\_COMP5615\_ISYS3888\_CSEC3888

SOFT\_3888\_TU08\_03\_P13

**Individual Contribution Report**

Student Name:

Student SID: 500468777

Tutor: Caleb The-Tjoean

Project: Visualisation tool for scene graph

Client: Yichao Hao

Unit: COMP3888

September 2023

# Statement of Work Done

## Summary of Work Done (1-2 paragraphs)

In the project thus far, I have had a variety of roles. In the first week I was the customer liaison and responsible for initiating contact with our client, Yichao and in doing so organised weekly meetings on Thursdays from 12-1. Following this, in week 2 I was allocated as manager. As manager I added a range of tasks to Notion that needed to be done and allocated them to members of the team as well as updating the wiki with an initial plan and a home page. Then in week 3 as tracker I ensured the tasks allocated that week were being completed as well as outlining tasks that needed to be completed soon to continue the development of our initial prototype. Finally, in week 4, I was the doomsayer and with a greater knowledge of the code as well as the most work in the codebase I was able to outline the areas of the project that I thought would be particularly challenging.

In week 5, I had no specific role other than programmer and worked on frontend graph generation as well as aligning the backend with the frontend requirements. Other contributions as a programmer in other weeks included implementation of a basic UI. I also assisted with system architecture, proposing a react / typescript frontend and a python backend. I implemented flask and restful designs in the API and began some initial data manipulation while a database was being set. I had some input in database architecture but was not responsible for implementing it. I also continually reviewed pull requests in bitbucket.

## 1.2 Weekly Plan

What you intended to do in that week vs what you completed

|  |  |  |
| --- | --- | --- |
| **Week** | **Work Planned (from week 2)** | **Work Completed** |
| Week 1 |  | * Organise group preferences |
| Week 2 | * Create backend python environment * Initiate contact with client * Investigate structure of VG data * Create project description presentation * Do group and individual contracts | * Initiate contact with client * Investigate VG data * Troubleshoot bitbucket |
| Week 3 | * High level design * Host backend * Create database | * Create Gantt chart for project presentation * Write brief project description * Create preliminary models to parse first images * Retrieve subject object predicate triplets by image id * Generate graphs in backend * Create flask app * Implement frontend fetch functionality * Create VG endpoint |
| Week 4 | * Deliver prototype design * Host frontend * Host database | * Refactor preliminary models to implement restful design * Switch to yarn * Switch to vite * implement d3 graph generation * Investigate alternatives to d3 * Switch d3 to react flow * Implement d3 force in react flow |
| Week 5 | * Backend development * Frontend development * Graph visualisation | * Generate graph dynamically in frontend from api data * Update frontend design for multiple graphs * Update backend VG endpoint to return multiple graphs * Render conceptnet graph in frontend |

# Extent of Work Done

**Week 1**

|  |  |  |
| --- | --- | --- |
| Task | Description | Evidence |
| Organise group preferences | I accumulated group preferences and required information for the project preferences survey | [Appendix 2.1](#organise_preferences) |

**Week 2**

|  |  |  |
| --- | --- | --- |
| Task | Description | Evidence |
| Initiate contact with the client | Email communication with Yichao, organising time for meeting with team. | [Appendix 2.2](#initial_email_to_client) |
| Investigate VG data | Initial investigation into the visual genome dataset. This included downloading data and analysis of the data structure. | [Appendix 2.3](#vg_dataset_download) |
| Troubleshoot bitbucket | We were unable to push to bitbucket after cloning and adding our SSH keys. I discovered it was because we had a final command to run. | [Appendix 2.4](#ssh_key_add) |
| Create drive folder | I created and shared a team folder on google drive for sharing of resources | [Appendix 2.5](#drive_folder) |

**Week 3**

|  |  |  |
| --- | --- | --- |
| Task | Description | Evidence |
| Create Gantt chart | For the project presentation I created the Gantt chart with Chad, showing our projected work. Unfortunately, as the requirements became clearer it rapidly became outdated | [Appendix 2.6](#gantt_chart_history) |
| Write brief project description | I wrote a brief project description for use on the wiki landing page. This also included updating the wiki with the information | [Appendix 2.7](#wiki_history) |
| Create preliminary models to parse first images | I wrote some simple python code to extract the information on the first 50 images in the database and export them to a new JSON file so that we could begin testing without using full file which severely limited performance. | [Link to bitbucket](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/82e156760fa1c765ab996111184dddf1da22979d)  [Appendix 2.8](#backend_graph_message) |
| Retrieve subject object predicate triplets by image id | Following this I implemented code that will parse the outputted JSON files in order to extract the necessary information to create the nodes and links in a graph. |
| Generate graphs in backend | Using the Networkx and matplotlib libraries I outputted the constructed graphs in the backend |
| Create flask app | Initialised flask app to facilitate communication between frontend and backend | [Link to bitbucket](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/b2f66f7c02d33f1fbe87990ef644fb0cf17d4b1a) |
| Implement fetch functionality in frontend | I created two fetch functions in the frontend that will hit an endpoint and return the response. Typed fetch will also parse the response as a certain type. | [Link to bitbucket](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/fb4414aeb6035f2e8ff84717323a6cdde055bb54) |
| Create VG endpoint | Create an endpoint in the flask app that returns the data from the VG dataset | [Link to bitbucket](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/86ad46e1cf7986e9558c3781c2046999619de930) |

**Week 4**

|  |  |  |
| --- | --- | --- |
| Task | Description | Evidence |
| Refactor preliminary models to implement restful design | After setting up flask, fetches and VG endpoint I refactored the preliminary models to implement a restful design. This included 2 files, ImageRelationshipsResource.py and ImageRelationshipsService.py. | [Link to bitbucket](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/86ad46e1cf7986e9558c3781c2046999619de930) |
| Switch to yarn and vite | I switched our frontend to use yarn and vite to speed up the development process | [Link to bitbucket](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/baab7e052e553e3e0ab9ee117eda17a8e10d11f8) |
| Implement d3 graph generation | I implemented frontend graph generation using the d3 library including a force simulation. | [Link to bitbucket](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/687de00c9e04f1f0d8c29e701f77aefd4f8ec8df) |
| Investigate alternatives to d3 | Due to the lack of compatibility with react we decided to investigate alternatives | [Appendix 2.9](#alternatives_to_d3) |
| Switch d3 to react flow | After deciding react flow suited our needs the best I reworked our graph generation logic. However, node locations were decided randomly. | [Link to bitbucket](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/54e3d92fa5e6a68086142affb6506b4f9f564f07) |
| Implement d3 force in react flow | To solve the problem of random node locations I implemented the d3 force algorithm which calculates the optimal node positioning through a simulation |

**Week 5**

|  |  |  |
| --- | --- | --- |
| Task | Description | Evidence |
| Generate frontend graphs dynamically | Previously our app would only render one image based on the id. I changed this to fetch based on an inputted word | [Link to bitbucket](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/cc1aa89674a985fc6bcd7f97d6db3c64a6cc7c7d) |
| Update frontend design for multiple graphs | Our client wanted 10 VG graphs to be shown. I added tabs to the side of the page such that only one of the 10 is rendered at a time. I also added a button that opens a dialog that shows the image for each graphs. |
| Update VG endpoint to return multiple graphs | I updated the VG endpoint in the backend to collect image data for 10 graphs rather than just 1. This mostly involved changing ImageRelationshipService.py and the response models. We still did not have the functionality to retrieve image ids by word contained so instead 10 random images were chosen |
| Render conceptnet graph in frontend | Once the conceptnet endpoint had been merged in I also rendered a graph based on the response |
| UI revamp | Now that I had the components for the 10 VG graphs and the conceptnet graph I spent time refactoring the ui so that the graphs were centered, adjacent and the same size |

# Quality of Technical Work Done and Other XP principles

|  |  |  |
| --- | --- | --- |
| Measure | Description | Evidence |
| Code review | As one of the most experienced coders on the team, I reviewed all the pull requests in bitbucket that weren’t mine. Suggestions included:   * Minor nit-picks such as changing variable names * Switching to non-deprecated packages * More elegant ways to implement logic * Recommendations for future tickets * Advice on how to align with other parts of the code * Redundant parts of the code that can be deleted   This ensured consistent quality across the entire codebase | [Appendix 3.1](#bitbucket_comments) |
| Code refactoring | I often refactored the codebase to ensure quality of code and use of common coding paradigms. Examples of this include:   * Implementation of programming paradigms such as restful programming * Modification of models to align frontend and backend * Changing of graph generation libraries | [Link to restful refactor](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/pull-requests/5)  [Link to graph library refactor](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/54e3d92fa5e6a68086142affb6506b4f9f564f07) |
| Embracing change | When another way of implementing an aspect of the codebase was identified we adapted accordingly. This is most evident with the multiple ways we have generated the graphs, moving from backend to frontend using d3 and finally react flow.  We also changed our design and implementation from feedback we received from our client. In the future we are going to implement dynamic node size and colour as Yichao suggested. |
| Restful programming | I implemented a restful design to our API. This involved creating resources with request methods and services that contained most of the logic for the endpoint. This ensured our calls are stateless and no data needs to be stored for future calls. |
| Story driven development | I added many epics to the notion as well as subtasks that needed to be completed in the stories. The stories were broad topics that needed to be completed such as backend implementation, dev workflow or graph implementation. Then, subtopics were more specific and could be assigned to members of the team. | [Appendix 3.2](#notion_stories) |
| Weekly sprints | I would often outline the work that needed to be done for a specific week and then allow team members to allocate themselves to the tasks. I did this over slack as well as verbally during meetings. Team members would then allocate themselves to the tasks on notion. | [Appendix 3.3](#sprint_goals) |
| Incremental changes | Our code was implemented incrementally. Only small portions relating to specific tasks were changed at one time. This ensured team members could work on other sections of the code at the same time without too many major changes occurring. Further, it mean that our codebase was carefully versioned on git and previous commits to master could be accessed easily if we needed to rollback. The incremental changes were driven by the careful creation of notion tasks. | [Appendix 3.4](#incremental_changes) |

# Discipline-based Contributions

My degree in computer science has provided me with the knowledge to contribute across all aspects of the project, however, my main contribution lies in the codebase. When looking at the [commit history](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/branch/master), it is clear I have had the most pull requests merged into master and have influenced all aspects of the code. My ability to solve problems comes from INFO1910 learning outcome 3 “compose a structured algorithmic design to solve the descriptive problem” while maintaining performance as shown in COMP2123 learning outcome 2 “design an algorithmic solution to a problem, analysing its time complexity and evaluating its suitability to a context”.

From my studies at university, I have gained competency in a range of languages as well as the skills to learn more independently. This has given me experience in development across the full stack and as such I had a significant role to play in defining the system architecture. Emma and I were advocated heavily for a [react based typescript frontend](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/pull-requests/2) and [a restfully implemented python backend using flask](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/pull-requests/3). Furthermore, I was instrumental in [defining database schemas](#database_schema). My involvement with system architecture was a result of COMP2123 with learning outcome L01 “demonstrate proficiency in organising, presenting and discussing professional ideas and issues in oral, written and graphic formats”.

The end goal of our client is a visualisation tool to compare the visual genome and conceptnet databases through the display of network graphs. I began learning about graphs and associated algorithms in COMP2123, data structures and algorithms, and solidified my understanding in COMP3027, algorithm design. Consequently fundamental understanding of the graph data structure has enabled me to complete this project and implement algorithms that [create the nodes and links](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/82e156760fa1c765ab996111184dddf1da22979d) as well as [displaying the graphs in the frontend](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/54e3d92fa5e6a68086142affb6506b4f9f564f07).

I have ensured the code I have written in the project is of a high quality. To do so I have employed a range of techniques I have learned throughout my studies. Using programming style conventions is a key outcome of INFO1910 and to do so my code contains [many comments](#commented_d3_force) and descriptive variable names consistent with language conventions (snake case in python and [camel case](#commented_d3_force) in typescript). I have also used a variety of debugging techniques such as breakpoints, code stubs ([see TestNetworkGraphData.ts](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/687de00c9e04f1f0d8c29e701f77aefd4f8ec8df)) and output statements to “assess code execution using debugging tools”, learning outcome 6 in COMP2017.

Furthermore, I have used many external libraries throughout the code, aligning with INFO1910 outcome LO12 “use of standard library functions” and INFO1113 outcome LO9 “demonstrate experience writing code with common interfaces and collections”. This included [React, Material UI, React Flow and d3](#frontend_libraries) in the frontend as well as [Flask, pydantic and requests](#backend_libraries) in the backend. Further, doing so required extensive reading of the documentation for these libraries which I learned in INFO1113 in accordance with the outcome LO2 “read and interpret object-oriented design documentation”.

This was particularly relevant with the d3 force algorithm and its integration into react flow which I found quite difficult. However, using COMP3027 learning outcome 2 “learn about a novel algorithm, by searching for descriptions in textbooks or online” I was able to succeed. It did however require modification to link lengths and spreading force which aligned with COMP3027 LO3 “read, understand, analyze and modify a given algorithm, as well as design efficient algorithmic solutions for given problems”.

Building off the application of code quality strategies in COMP2017 as well as my understanding of common programming principles from INFO1910 I have implemented a [restful design in the backend of the code](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/86ad46e1cf7986e9558c3781c2046999619de930). This involved the construction of resource and services as well as model classes for the parsed data. I learned this object-oriented approach from INFO1113, specifically outcomes LO1 “demonstrate an understanding of object orientation” and LO6 “create appropriate class/data structure including the data types”. Furthermore, to maximise efficiency, we have used [react query](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/commits/3d83a257f927abb129f3e7807781fd910169020f) which performs asynchronous requests, a concept I was familiar with due to COMP2017 learning outcome 10 “construct, debug, and evaluate parallel or concurrent programs”.

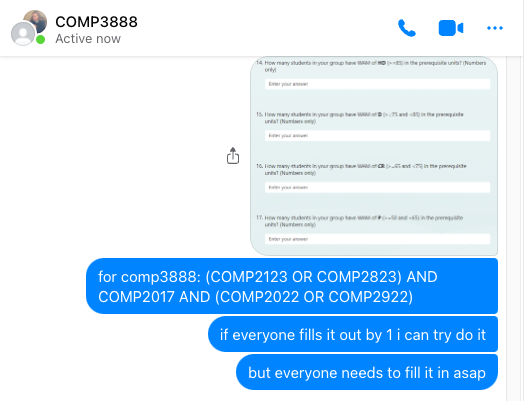
Teamwork is a key aspect of software engineering and computer science and I have learned how to collaborate effectively in subjects with group projects such as DATA1005 and COMP3308. The use of git is a key tool used by teams working together on a single codebase and it has been a component of many of my courses throughout my studies. As such, I was more experienced than many of my group members and I spent some time in the first meeting explaining the mechanics and conventions of git. This included an explanation of remote and local repositories as well as the difference between the master and other branches and how to move between them. Furthermore, I helped in the setup of SSH keys and ensured my other team members were able to access the repository locally. Evidence of this is contained in [the first meeting minutes](https://bitbucket.org/soft3888-tu08-03-1/scene_graph/wiki/Meeting%20Minutes/Week%202/Group%20Meeting%201) on our project wiki. Communication is also vital in effective teamwork and I am able to explain my solutions clearly and in a way that is easily understood due to COMP3027 learning outcome 1 "produce a clear account of an algorithm that would allow others to understand and implement it” allowing others build on what I have implemented. Additionally, I have been extremely active on the team channel during team discussions (evident in appendices [2.1](#organise_preferences), [2.4](#ssh_key_add), [2.9](#alternatives_to_d3) and [3.3](#sprint_goals)). Furthermore, I [set up a google drive](#drive_folder) for shared resources that don’t belong on git, such as datasets, contracts, and reports. All these contributions allowed our team to function effectively and complete the required work with sufficient velocity.

Outside of the coding I have also been active in project documentation. We have been doing weekly sprints where each of us are allocated tasks to complete. I have been extremely active on Notion, adding [tasks and epics that needed to be completed](#notion_stories). To do this effectively I have been [outlining tasks that need to be completed](#sprint_goals) in the following sprint as well as tracking their completion, reflecting my XP role as manager. Furthermore, as tracker, along with Chad, I [created an initial Gantt chart](#gantt_chart_history) to track our progress but this was replaced later by Notion.

Ultimately, my contribution to this project spans all required aspects. Using principles I learned throughout my studies, my focus has mostly been on the codebase where I have contributed across the full stack. This has been aided by experience working in teams in various subjects and my experience with tools such as external libraries and git. In addition, I have also contributed to group documentation and been instrumental in tracking progress as well as creating and allocating tasks to ensure we were on schedule.

Appendix

1. Evidence for extent of work done
   * + 1. Communication to organise preferences



* + - 1. Email to client

A screenshot of a white box

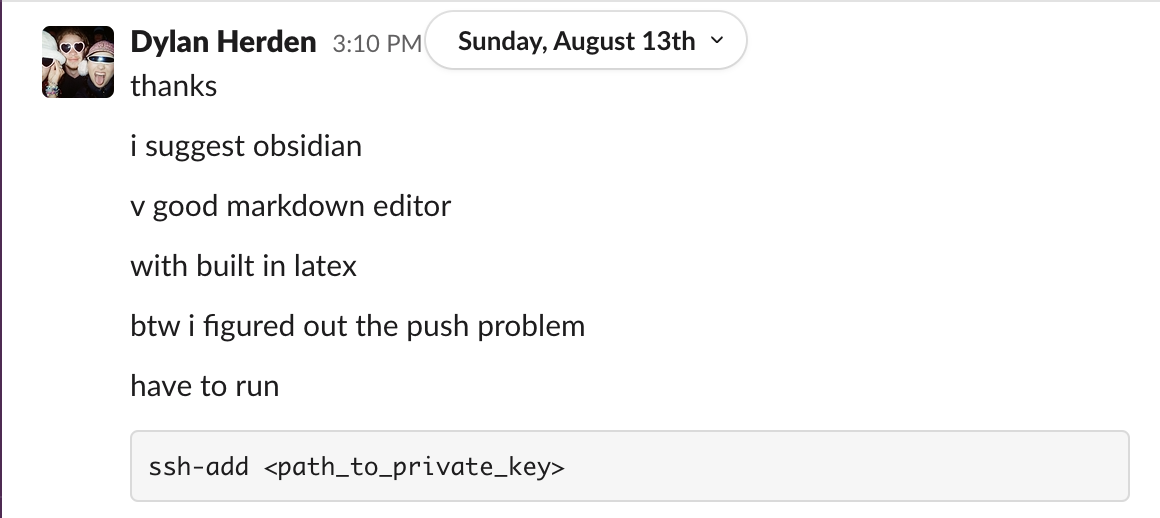
Description automatically generated

* + - 1. Screenshot of search history when VG dataset was downloaded

A close-up of a computer code

Description automatically generated

* + - 1. Messages explaining how to add SSH key



* + - 1. Screenshot of creation of shared drive folder

A screenshot of a phone

Description automatically generated

* + - 1. Screenshot of search history for creating Gantt chart

A screenshot of a computer

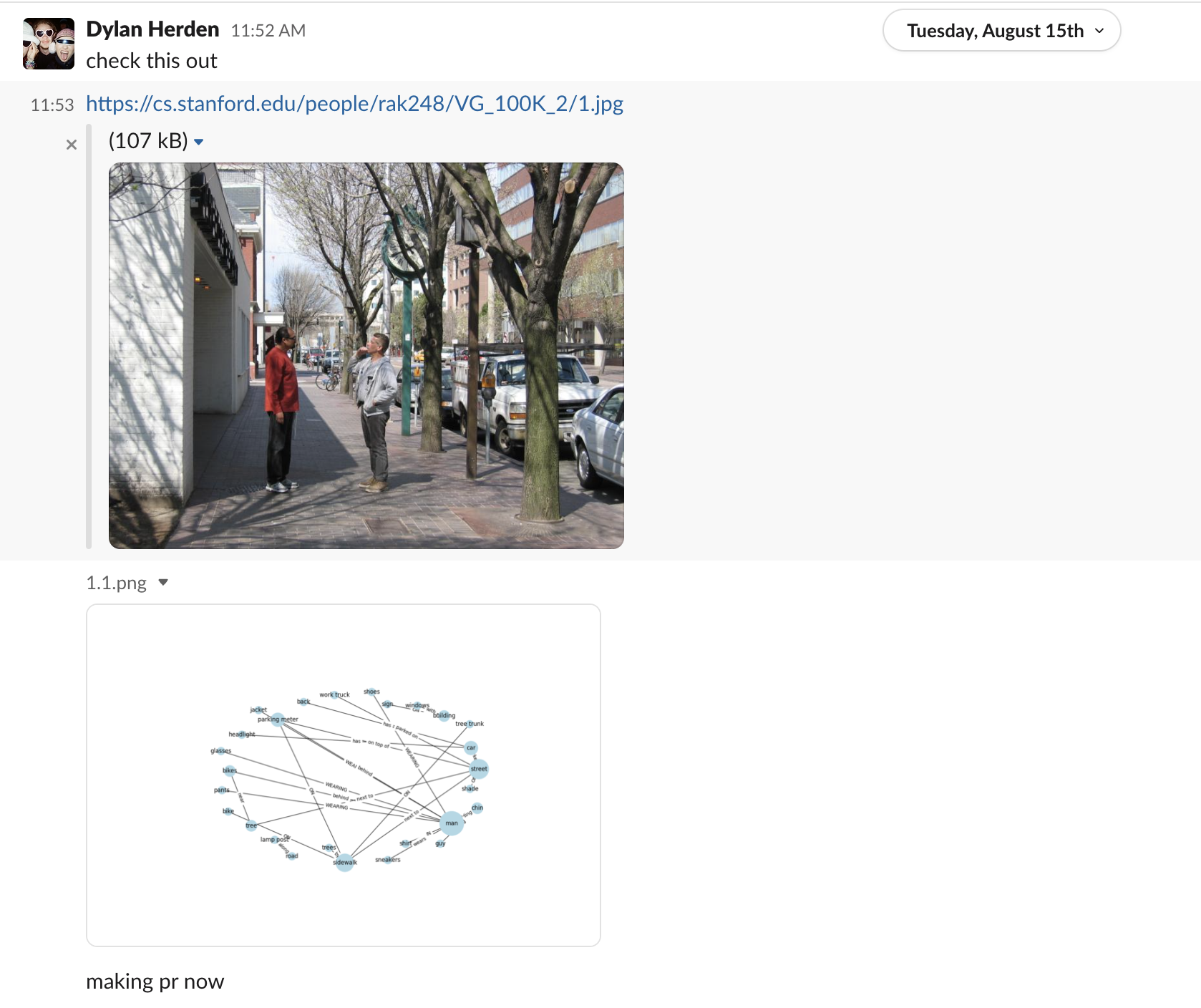
Description automatically generated

* + - 1. Wiki history

A screenshot of a computer

Description automatically generated

* + - 1. Messages showing graph constructed in backend using Networkx and matplotlib



* + - 1. Messages saying I am investigating alternatives to d3

A white background with black and white clouds

Description automatically generated

1. Evidence for extent of work done
   * + 1. Bitbucket comments

A screenshot of a chat

Description automatically generated

A screenshot of a chat

Description automatically generated

A screenshot of a chat

Description automatically generated

1. Stories on notion

A screenshot of a computer

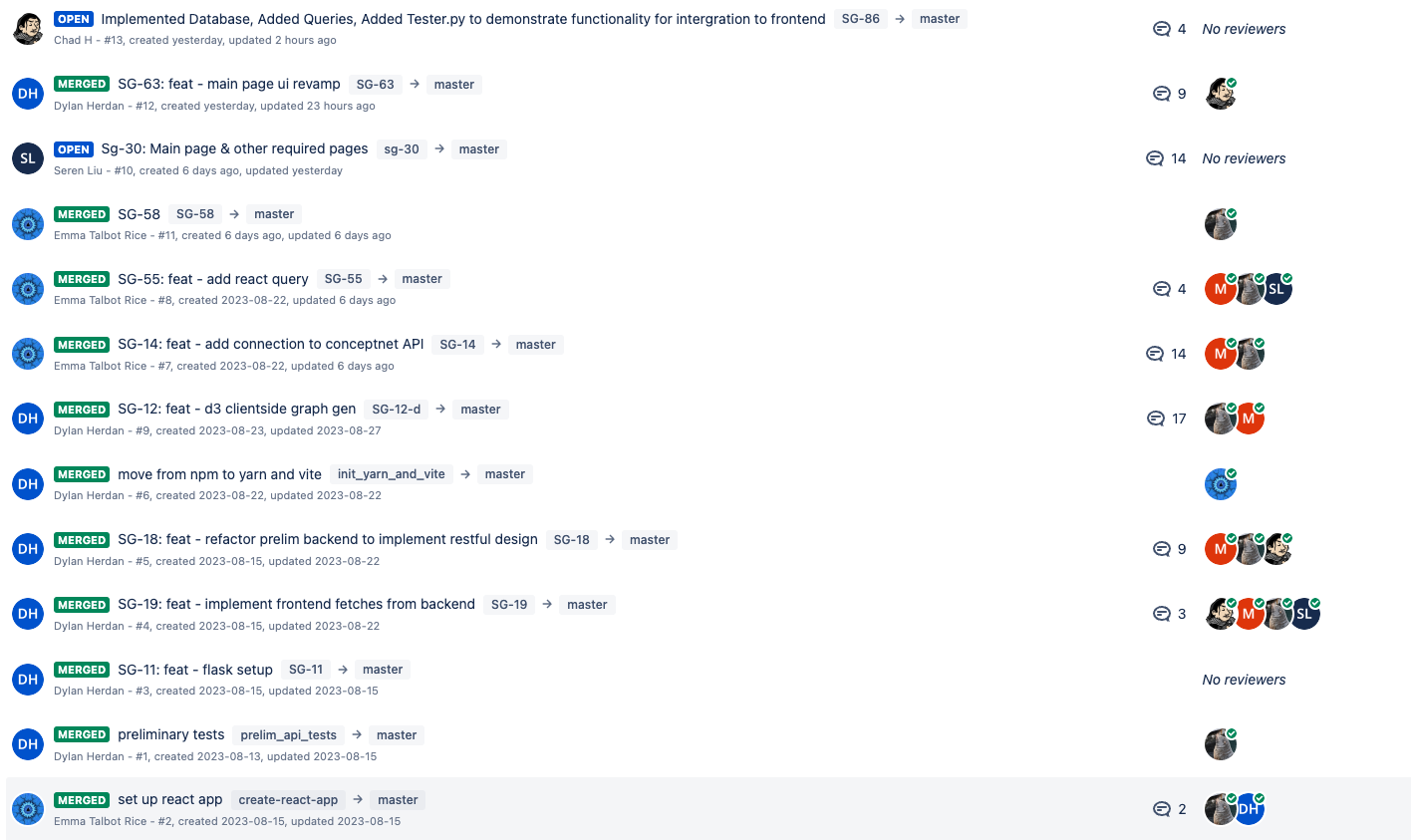
Description automatically generated

1. Sprint planning messages

A white paper with black text

Description automatically generated

1. PR history showing incremental changes

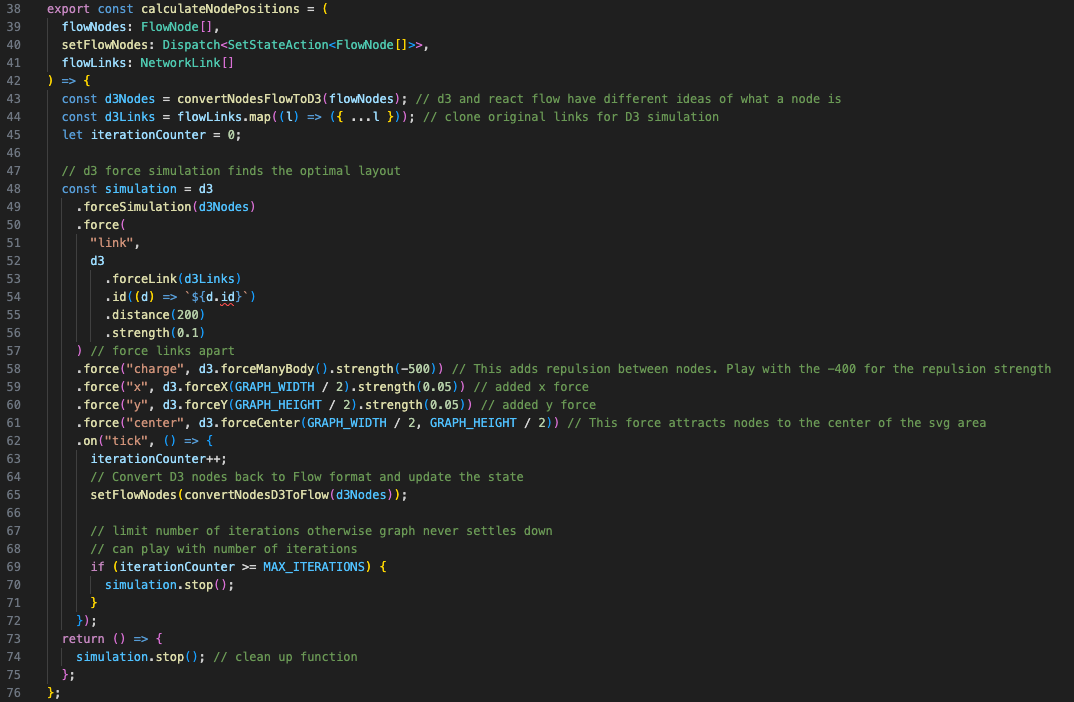


1. Evidence for discipline based contributions
   * + 1. Database schema I drew on whiteboard

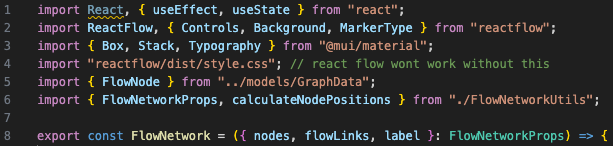
A person writing on a whiteboard

Description automatically generated

* + - 1. Commented d3 force algorithm



* + - 1. Excerpt from FlowNetwork.tsx showing use of React, ReactFlow and Material UI



* + - 1. Excerpt from ImageRelationshipsResource.py showing use of flask and pydantic

A screen shot of a computer

Description automatically generated