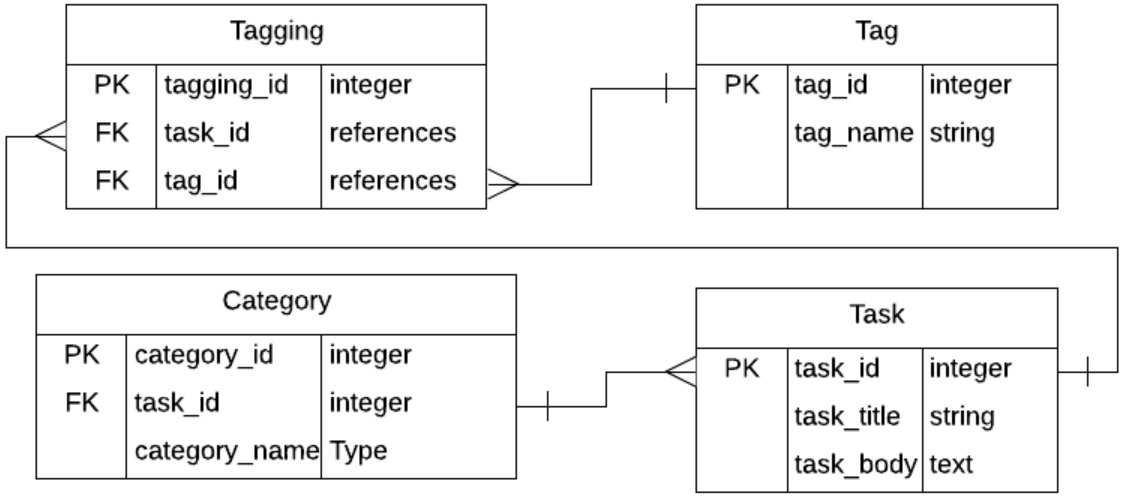
As of writing, I have not given *React* more than a cursory read, choosing to focus on following *The Odin Project (TOP)* because everything is new, and **Source** was my first language. I'll be ignoring React for this submission since I haven't learnt it yet. I've documented my entire learning journey so far (via notes on tutorials/projects completed) and have attached relevant links at the end of the document. Since I've not begun on the actual app, I'll incorporate the Execution Plan within the Use Cases and *attempt* to write out tests for these first before building. I hope to get to Iteration 5 at least by final submission.

Use Cases and Execution Plan

- ITERATION 1: Basic Models and their CRUD actions:
  - Basic CRUD actions on these models: **Task, Tag, Tagging** (to make the *many-to-many* relationship easier), **Category** (differs from tags to help chart time planned for each category e.g. Academics, Family, Hobby...) and **Date**.
  - **Tags** allow user to set whatever search term he/she wants (like in telegram) and **Categories** are to provide the user some *data visualisation* and *target-setting*. Here's my attempt at an Entity-Diagram Model for self-implemented models:



- ITERATION 2: Searching through Tags:
  - Utilizing gems such as fuzzy_match that handles a form input for searching through tags and routing the result to an index view for that tag. There are some gems available that abstract the form-creation and searching but their feasibility can only be known after Iteration 1.
- ITERATION 3: Accounts, Authentication and Sessions handling
  - User account creation minimum/basic goal: use authlogic (complicated) or sorcery (easier, have done it in a tutorial). I haven't learnt how to handle sessions yet, other than conceptual understand of a Session model and effectively using cookies.
  - Google account integration would be useful for (Iteration 6), depends on time

- ITERATION 4: Basic Front End (should be clearer once I cover React)
    - Likely just going to end up bootstrapping something decent and not resource intensive in the meantime until ITERATION 7.

- ITERATION 5: Visualising Data and Time-Targetting
    - A Category has many tasks, and so, time planned per category can be aggregated, allowing the User to visualise, through simple charts, his/her use of time (across various time-periods)
    - There are no incentivising features to this webapp, so a Target setting feature could be implemented. E.g. I target to spend 4 hours with my family in a week doing...
- ITERATION 6: Creating Google Calendar Events for Notifications
    - Can capitalise on google calendars for notifications, yet to explore Google API uses but there's some guidance on it here.
- ITERATION 7: Front end magic
    - Implement a Zen-mode (a la VsCode): to declutter the page from the available settings and controls and just display the todo.
- ITERATION 8: Convenience Features
    - Progressive Web App. The link claims it's not difficult.
    - Try using the webapp as a new-tab extension of sorts

Problems I'm facing:

- Still have very limited knowledge on React, it is taught in-depth via the Odin Project, so I shall complete that, but time is really limited and there's a learning-applying tradeoff that's hard to balance. Currently my plans are based on rendering views via erb, unsure how much more complicated using React will be.
- Heroku is a real mystery. I have spent two days trying to figure out why I can't deploy my attempt of Blogger 2 tutorial (see link below), but I can't fix it. Seems like **one** of the problems is in writing out the database.yml file to account for the fact that Heroku only accepts Postgresql while the original app was developed using Sqlite3.
- The API implementations look really difficult and time-taking. That's why I don't think I'll be done by submission 2, but I really want to use this, so I'll finish it regardless.

What I have done so far:

This is my first attempt at self-learning something of this magnitude and the journey has been rewarding. I was lucky that the learning outcomes of TOP almost entirely overlap with this assignment's learning objectives. The biggest problem of such self-learning has been falling into the many rabbit-holes there are. Almost everything I'm learning is new and majority of it is interesting. I tried to timetable out my expected progress for first submission, hoping to show a barebones version of my app but couldn't stick to it. Nevertheless, the Blogger2 Tutorial is pretty much identical in specification to the assignment. Following this tutorial, I've learnt how to implement a blog-site (works locally, couldn't deploy to Heroku though) with these features: Creation of Blog Posts and Tagging them, setting up the database for the many-to-many relationship between posts and tags, adding a commenting feature, utilizing partials and layouts to create views via erb(in accordance to the DRY fundamentals), learning how to use gems to incorporate media attachments, adding user accounts and authentication and doing these via Rail's magical generators as well as having some understanding of what goes on under the hood.

I chose to postpone learning React because as part of TOP's WebDev101 course, basic front end (HTML, CSS, JavaScript and DOM manipulation) was covered and React is part of the Courses eventually. A mini project involved implementing this Etch-A-Sketch and it was a morale boost. I spent a while fully learning pure JavaScript to help me understand React better, in the future. I think the decision to learn step-by-step was worth it because I think I understand MVC and Databases enough to debug basic issues.

Relevant Links:

- Odin Project Repository (whatever I have learnt so far)
  - Notes:
    - Rails Notes, stuff on MVC and other Rails magic
    - More Detailed Ruby Notes, stuff on Serialization and others..
    - Basic Ruby Notes
    - JavaScript notes and takeaways from JS exercises
    - Basic Git and Front End Takeaways
  - Morale-Boosting Implementations:
    - Etch-a-Sketch: Manipulating DOM elements (select dimension value of 16 if on mobile also I didn't have time to waste on making it work with touch gestures/multitouch. Refresh if it doesn't show a dialogue window first). Here is the repo for it.
    - Blogger2: blog with tagging, posts, commenting, user authentication and attachments (very similar in features to the assignment). Couldn't deploy on Heroku, works locally though.
- My CVWO Assignment Repository (pretty empty, just has these submission files, installation screenshots and such)