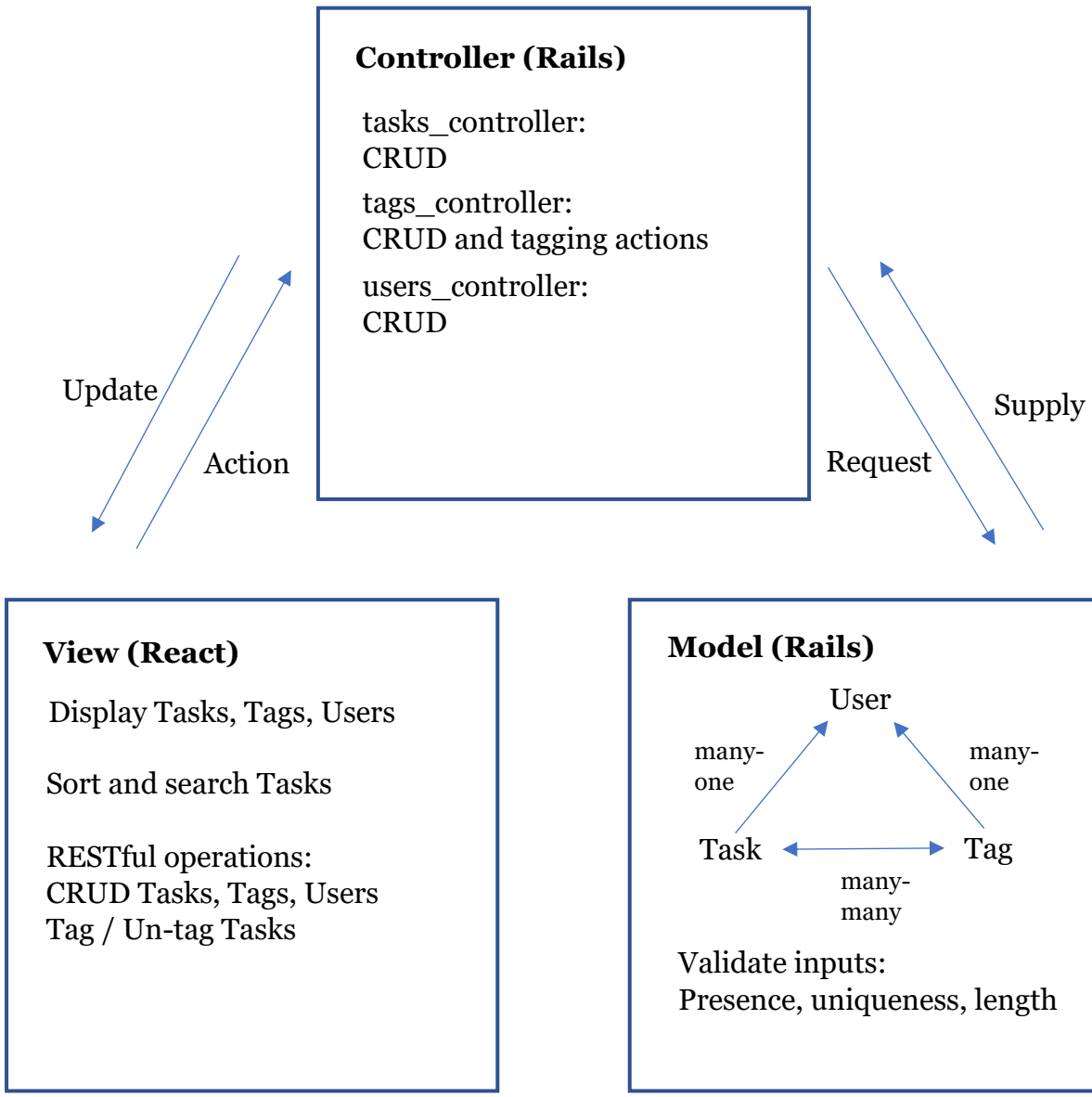
Moo Jun Wei

Riding on Rails: Final Submission

1. Overview



2. My Progress

2.1. *Starting web development:* At the beginning of this project, I had very little knowledge about web development. I found "The Odin Project" immensely helpful in teaching me the fundamentals of the web, in addition to building static websites with HTML and CSS.

2.2. *Starting Rails:* Learning Rails was very intimidating, as many online tutorials are targeted at people who already have experience with other frameworks and Ruby. I had trouble understanding how the different parts in the Rails documentation fit together. Here, I found YouTube tutorials to be very helpful in getting a broad conceptual understanding of how Rails works. I also took a detour to learn Ruby programming, after which I found that I can understand the Rails guides better.

2.3. *Starting React:* I did not have experience with JavaScript prior to this, so I learnt React by experimenting with it and searching around for solutions to problems that I faced. Reading online discussions and guides taught me several better ways of doing things, which I applied to my code.

2.4. *First attempt at the to-do app:* This was probably the most difficult part of this project as I had little idea of how I would integrate React with Rails. I encountered many errors that were due to an improperly set up environment, which I managed to resolve after searching and looking at guides on the internet. After getting a 'hello world', I experimented with the new app until I could get a page that could handle CRUD. As I added more features, the code for this app got messy and unmaintainable, so I decided to scrap the code here.

2.5. *Beginning anew:* Setting everything up to the point where I could get React to communicate with the API-only Rails server was much smoother this time. I had a better idea of the code I needed to write. With a better understanding of how React and Rails work, I could also define features that I am confident in writing code for.

2.6. *Iterative development:* I made the Task model and controller first, then wrote React and CSS code to display them, followed by a series of improvements to the user interface. Here I reached the second difficult point of this project, which was to add a Tag model that has a many-to-many relationship with Tasks. I read through the Rails documentation on associations, then branched my existing code to experiment until I understood what I needed to do to create the Tag model.

2.7. *Adding Users:* The last model I added to this project was the User. I had a good idea of what I needed to do at this point, so I could write the new controllers to fetch Tasks and Tags associated with Users. After this, there were multiple rounds of bug fixes and UI improvements.

3. What I Learnt

3.1. *How to learn a framework:* After going through the process of learning React and Rails, I am now much more confident in picking up another new framework. Specifically, I found that I learn best with a project-based approach, in which I build progressively more complex apps and refer to guides and documentation as needed. I am also able to find solutions from official documentation and online discussions.

3.2. *How to develop software:* While doing this assignment, I experimented with both backend-led and frontend-led development. Now, I have better intuition on selecting the right approach to different situations. Additionally, I learnt how to use test-driven development to get a clearer idea of the code I need to write.

3.3. *How to write better code:* I learnt how to manage the complexity of an app by breaking it up into smaller components, so that the code I write can be more maintainable and robust. I also learnt to use mental models such as MVC to achieve separation of concerns.

3.4. *Technical aspects:* I managed to pick up the Ruby language, and use Rails to handle routing, controllers, and models. I am also able to use HTML and CSS alongside React to create a dynamic webapp. This is also the first time I used git to handle versioning, and I have found it to be an indispensable tool in managing programming projects.

4. What Can Be Improved

4.1. *Better modularity:* The react code for this app is not as modular as I would like. Some components are heavily co-dependent on others, which might make it difficult to maintain or add features in the future. In future projects, I will develop with modularity in mind to make it easier for myself and others to work on my code.

4.2. *Better compatibility and accessibility:* As the React frontend has been tested only on a desktop browser, the page might not display correctly on a mobile browser or high-dpi displays. It also does not use accessibility guidelines to make the page easier to use for people with disabilities. In future projects, I will test frontend code in more situations and develop with accessibility in mind.

4.3. *Better documentation and readability:* As I learnt Rails and React alongside the development of this app, documentation for my code is sparse. In future projects, I will write documentation and track changes throughout the course of development to make my code easier to understand.