What went well:

- Scheduling/Planning/Time management
    - In our team contract and teamwork plan we unanimously decided to prioritize work on the project so we could finish each part as early as possible. Our work schedule and personal deadlines were appropriately scheduled in such a manner to give us enough days before the deadline to check over our work.
    - Because we scheduled our tasks in such a matter, it left us enough time to check over our code and add in extra features that were not initially planned in the MVP.
    - Even though our project was considered difficult, by not procrastinating on our work we were able to complete the goals we set for ourselves on time.
- Frequent team meetings with all the members
    - We scheduled frequent team meetings with all the members, and this meant that had easy access to one another and could communicate easily during that time. Therefore discussion before/during making big decisions was pretty easy.
    - Along with frequent team meetings, we also scheduled hackathons over the weekend that allowed us to communicate and ask questions about each other's code that we needed help with while working on our individual parts. These hackathons made coding and updating possible and easy, even though our app had a lot of interconnected "moving parts".
    - These meetings also helped keep everyone up to date. From our initial meetings, we delegated tasks and expectations for each model/controller/route. Because everyone knew the general idea of what they needed to do, it was easy to code up and connect the elements together.
- Initial design phase (seems bad but it was a good idea IN THE END)
    - We spent a lot of time initially debating over designs, reiterating, and changing them. While this took up a lot of time and caused an initial bottleneck with our code, ultimately it was a good decision as our design was easy to implement and connect together.
- Prewritten route names
    - Prior to starting routes, we sat down and had Maddie write an initial document with the document routes and what they handled. This made it very clear that there are no overlaps in our routes and dividing up the routes was easy; everyone just had to implement the routes they were given.
- Overall task layout/distribution
    - We decided to do our model and thoroughly testing those models before writing our controllers. By making sure that our code was thorough and robust each step of the way, we were able to guarantee that bugs that occurred later on were only dependent on that version and narrow down on its location.
- Version Control/Few Merge Conflicts
    - We used branches and version control to minimize the amount of merge conflicts that we experienced between our code.

What could be improved:

- Dealing with bottlenecks
    - We had one really big bottleneck in the beginning of the project that slowed down coding for a few days. Additionally, we had another set of bottlenecks when we started building out the UI since the only way we had thought out about building the UI was to build page by page, which was very slow.
- Coordination
    - Although we did not run into a ton of merge conflicts, the fear of having merge conflicts slowed us down quite a bit. There were several times where multiple people were working on similar things but there was a hold up because they could not work in parallel.
    - One thing that two team members did, which seemed to work well for them was work in two separate files (for the time being) and merge their work once completed.
- More thorough code review
    - Despite us planning for code review, we didn't allocate enough time for everyone to thoroughly look through each other's code and as a result, there was only one thorough code review.
    - This was an important step we kind of rushed over, as code review is important for code quality and an opportunity for everyone to learn what was going on in each other's code.
- Naming and object conventions
    - We needed to establish conventions for how we were sending error message and responses because everyone had something different and this caused a bit of issues when connecting things together.
    -

What you would do differently in the future:

- Naming convention
    - We would choose to meet earlier in the future to discuss naming conventions or have a design document that listed the conventions so that team members could have a central place of reference.
- MORE code review
    - We would definitely allocate more time for code review to make sure everyone had a look at each other's code and make sure that it was bug-free.

Feedback to Jennifer from:

- Maddie: I'm really appreciative of all the design and implementation work Jenn has taken on for the MVP. However, I sometimes felt that she left me out of the loop when making changes to the application model during the design process. Going forward, maybe she could keep a changelog of design changes or notify the group of changes via email or instant message.

- Heeyoon: Jennifer is very willing to take on a lot of work, and in general completes it very quickly and very robustly. One small thing was she had to rewrite a class several times, since it didn't work exactly. She did ask questions to us individually, as she was going through the iterations. However, perhaps she could have worked with all of us together to come with a solid implementation design first, before coding right away, though there were many unexpected problems.
- Lynda: Jennifer takes on a lot of work and is a great person who always tries her hardest to get it done on time and in the highest quality possible. However, she should try communicating to the team more about decisions that she makes and stick to them so that others do not need to constantly change their code to fit the new updates.

Feedback to Lynda from:

- Maddie: Lynda is super organized and is great at helping us manage time. Nevertheless, sometimes during group meetings she gets a bit too absorbed in her work and doesn't participate in team discussions. I appreciate her hard work but also wish she would engage with us more. Another possible area for improvement is her method signatures: sometimes the required inputs/outputs aren't so intuitive.
- Heeyoon: Lynda writes very organized and understandable code. She's also very good at communicating to the team, and bringing new ideas to the table. Sometimes, however, she does this during meeting when we're in the middle of discussing other things, which can be distracting. Also, she sometimes stops paying attention during meetings.
- Jennifer: Lynda is super enthusiastic, which is awesome because she's willing to take on lots of work and try lots of new things. On the flip side, sometimes her enthusiasm to try new things comes a little premature, and she will start working on things before we have the backing to support it.

Feedback to Heeyoon from:

- Maddie: Heeyoon is really good at writing quality code in a timely fashion. Sometimes she doesn't pay attention to our group discussions online and is out of the loop on design or implementation changes; however, I don't think this is Heeyoon's problem so much as a problem with our choice of method of communication. Maybe we can come up with a new strategy for the final product.
- Lynda: Heeyoon's code is always robust and connects well to the rest of our stuff despite the changes we may have made. One suggestion is that she should communicate better with the team and be more up to date, either by looking through our chat logs if she has time or asking another team member.
- Jennifer: Heeyoon is good at making her code modular so that it easily fits into the existing code. One thing she could work on is adhering to naming conventions a better. Additionally, sometimes it seems that she's a bit out of the loop, but she does do a really good job at figuring out the existing code.

Feedback to Maddie from:

- Lynda: Maddie is always willing to meet up and is in the loop even though she lives further away than the rest of us. She should try writing a little bit of code at a time and then debugging to make sure every one of her steps is robust rather than writing a lot of code and then having issues with debugging later.
- Heeyoon: Maddie's main strength is that she is very willing to take on work, even when she seems busy. A minor potential fix could be messaging specific members if she has a question about a particular class, rather than messaging the whole group.
- Jennifer: One of the best things Maddie is that she's super accessible, and always very easy to communicate with. Nevertheless, she could work on finding a balance for when to ask for help. Sometimes she asks a little early when the answer is not too difficult to find, but other times she spends hours debugging when someone else could have quickly helped her.