Revision Summary:

We made many changes to the design of our application during the development process. Pretty much all of our user stories went through at least minor changes, but the *itinerary*, *create-trip*, and *currency* user-stories were completely revamped. A common theme throughout our user stories was having too many features, which made the story over-complicated and too diffiult to implmeent given our limited time and experience. In addition, we made changes to the tech stack that we were using as well as the libraries that we were using. Finally, we also changed the name of our application in order to reflect the changes made in the stories.

1. User Story: login

This user story was "optimized" through several minor changes. For instance, instead of a registration redirecting the user to login, we thought it would make more sense to just automatically log them in instead (this is what most apps do anyways). Furthermore, after logout the user will no longer be redirected to a login page, but rather will have to navigate there themselves. Finally, the description of the layout of the page was changed. These changes were made to simplify what we thought was an over-complicated login process.

2. User Stories: itinerary and create-trip

We initially wanted a "trip" to be primarily a flight, which would be selected based on the budget. A trip would also have an itinerary, which was made up of hotels and restaurants. It turns out that the flight API which we thought was functioning didn't actually work. As a result, we had to turn a "trip" into a restaurant and a hotel. This meant we had to merge the *itinerary* story with the *create-trip* story, since the original *trip* feature was primarily based around flight (in

fact, we didn't end up having enough time to get the restaurant api working). Finally, some bloat/extraneous features were removed.

3. User Story: view-trips

Only minor changes were made to this user story, mostly for the purpose of simplification because original story was over complicated (this statement is really true for all of our user stories). More specifically, we removed the search bar feature for finding trips as well as the ability to "copy" the trip into text in order to share it easily.

4. User Story: currency

Initially, we wanted the currency being used to be tied to a specific trip. In this way, users can see the cost of each trip in their preferred currency. However, the currency API took us a long time to get working and we ran out of time to implement this more complicated feature. As such, we decided to make an independent currency converter page. This way, some utility is still provided to users as they can still see how much their money is worth in other countries. At the same time our work does not totally go to waste.

5. Renaming the Applicaiton:

As mentioned previously, we remove the function of flight searching. As a result, the name TripAdvisor is no longer a good representation of what our web application does, so we decided to change the name to TravelHelper.

6. Tech Stack and Libraries

First of all, we switched from MongoDB to MySQL. The reason we did this was that a member of the group had experienced working with SQL, so this was easiest for us. Finally, since our application is small and data we are storing is relatively simple, the performance benefits of using NoSQL database systems such as MongoDB would not be relevant.

We made several changes to how we used our chosen tech stack. Although the core components of Flask and React remain unchanged, the libraries used within Flask were completely changed. Initially, we had decided to use *flask-login* and had constructed the whole backend around it. However, we found out very late that *flask-login* does not work with React (flask-login decorators require HTML redirects). We then reconstructed the login/registration scheme around the library *flask-jwt-extended*. Furthermore, we also had to add *flask-cors*. In the frontend, we ended up using *axios*, which we didn't plan on initially. Given that we essentially had to redesign the entire backend, we ran out of time to properly integrate our backend with the APIs and the frontend.