Changes:

There were no major changes to the direction of our project. From the beginning, the goal was to create a tool for users to easily access dog ownership information, specifically in the South Australia and Victoria regions of Australia. Throughout the course of the project, this goal never changed, and we ultimately created our Aussie Dogs website based on this premise as well. In this respect, we benefited from a well-defined scope.

In terms of usefulness, our goal was to make a tool that made it easy to access different types of data all at once, instead of having to traverse through multiple websites or applications. To achieve this, we created an application that anyone could use. We believe the design is simple and user friendly, appealing to people of all age ranges, nationalities, and ages. After all, pet ownership is a family affair. Perhaps more importantly, as our app demonstrates, it is also a community affair. Thus, we achieved our initial goal of usefulness.

Our overall schema did not change much, although there were definitely some notable changes that we made throughout the course of the project. Our final design included the same tables that we proposed in Stage 2 of the project, but we did change the schema of a few tables. Specifically, we changed the 'Dog' and 'Building' tables, though namely for different reasons.

First, we had to change the Dog table to better fit our design. Our original plan was to simply import the Kaggle database that we introduced in our initial proposal. However, the schema of this table did not align well with the rest of our design, so we ended up changing the columns that we used in our final design. We ended up cutting some of the unused columns and combining the South Australia and Victoria data into one large table. We did still use the original Kaggle database.

The second change was made for a different reason. Because our building data did not meet the requirements stated in Stage 3 of the project, we used a script that procedurally generated location data in accordance with our design plans. However, we were unable to reach the required amount using our original schema. As a result, we ended up using an extra, unnecessary primary key. Of course, this implementation is not ideal. In the future, with a larger dataset or more diverse generated data, we plan to correct this issue.

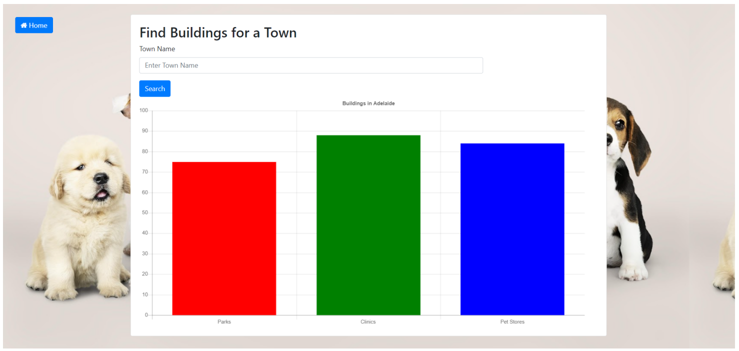
Arguably, our functionality is what changed the most from our initial project proposal. We wanted to aggregate and synthesize data (especially from credible but dense and inaccessible sources) so that we could make pet ownership decision-making more accessible and available at multiple levels of resolution. We wanted to create a platform for users and other community members to give their own ratings.

Our application does allow users to search using a variety of different parameters. In addition, we created some useful aggregation tools utilizing advanced database queries and programs, which we will later discuss in further detail. This allowed users to search for suburbs that contain certain breeds of dogs, search for neighborhoods with specific types of buildings, see a town's dog-friendliness ranking, and see the buildings of a town. These were functionalities that originally weren't part of our project proposal, but they were tools that we felt aligned with the direction of our project and were useful for users. However we are missing two key functionalities which we originally hoped to include: a map-like visual tool, and a platform for user ratings.

We will discuss the reason why we were unable to implement the visual tool in our technical challenges later on. As for the reason why we didn't implement the rating platform, the issue stems from our lack of data. We wanted to source user login data, but were unsure if online examples might be unethically sourced. We considered randomly generating usernames and using a list of most common passwords, and tried to find the equivalent of the Northwinds database (as used by Abdu in the course for examples) but for user logins. Ultimately, we scrapped the user login feature. Secondly, we played with the option of generating our own user rating data, but we felt this didn't really make sense. It was impossible for us to uniquely create ratings for every single town, neighborhood, or building, and if we had randomly assigned ratings for multiple users, the end result would look like a normal distribution of ratings. Thus, we chose not to generate user data. As a result, it was impossible for us to have a platform that evaluated and aggregated user data that simply didn't exist. We also considered creating a form for user ratings, but since none of the other functions would be there, we felt this wasn't a great use of our time.

As mentioned previously, we created aggregation tools by using advanced database queries and programs. Using advanced SQL queries, we allowed users to search for suburbs that contain certain breeds of dogs and search for neighborhoods with specific types of buildings. These two functionalities were implemented without using any stored procedures and triggers. We used a stored procedure to, upon a user request, calculate a dog-friendliness ranking for all the towns in our database, and then it creates and displays a leaderboard based on these rankings. Separately, we used a trigger on the 'Dog' table to limit valid birth years. If, during a dog registration, the user says their dog's birth year is in the future, or more than 40 years in the past (data calculated by searching for maximum medically recorded lifespan), the trigger will prevent that registration from being completed successfully. All of these functions allow for users to sift through our data in new and useful ways, so we believe that they functionally add to our overall web application design.

One final major change to our website design was a visualization tool. We were unable to implement a map-based visualization tool, which we will discuss further in our technical challenges. However, this did not stop us from creating a simpler, but different visualization. We allowed users to input a specific town, then look at all of the buildings within that town (as shown below). This was not in our original proposal, but we felt it was a fun and useful addition.



Technical Challenges:

Arman: Initially, we wanted to use Google Maps API so we could have access to real time location data. However, this would have been a multi-language process and a project unto itself. Before making a proposal, it is important to determine its feasibility. However, I believe it is good that we had a vision and challenged ourselves, even if it didn't work out perfectly.

Ebaad: Cleaning the data was difficult. We had to standardize the information for a large number of entries, this is not something that could be done automatically. Even after spending a significant amount of time sifting through the data, we still ended up having tuples with NULL values. Thus, I learned it is important to keep in mind the initial structure and quality of your databases when choosing a project, and ensure you perform a thorough cleaning.

Patrick: I ran into an issue when designing the trigger, which limited the tuples that could be entered into the 'Dog' table. The original trigger internally modified the 'Dog' table, but this ended up causing errors every time a new tuple was inserted into the table. I ended up learning that a trigger cannot modify the table that it is on. The solution to this would be to create another table which stored the modifications, and then handle those individually separate from the trigger activation. However, I ended up simply using a completely different trigger design.

Future Changes:

Firstly, it is important that we fully clean all of the data inside our database. Specifically in our 'Dog' table, we have an insignificant number of tuples that have weird or NULL values, which should be deleted in their entirety. Furthermore, in the future, we would like to implement a full platform for all user data (login information, ratings, etc.). We would also like to realize our original goal of creating an interactive and visual tool using Google Maps, as well as perhaps importing useful location information from Google Maps as well. Finally, we would like to integrate a graph-based NOSQL database using Cypher. As of right now, we think this graph-based database can be used to store our user login and rating data. We personally enjoy the structure and feel of Cypher, and we also believe the graph-based implementation accurately symbolizes the relationship between users and our current data (e.g. User -[rates]-> Town/Neighborhood/Building).

Teamwork:

Group members worked across areas of focus and met outside of class as needed to ensure the project was completed. We all communicated well with each other. We all did our best and were thoroughly pleased with the work we did (as well as our final project). The work distribution is as shown below:

Patrick:

- Created SQL CRUD functions
    - Worked with Ebaad for final implementation
- Implemented advanced database programs
- Developed most of the code for website design
    - Worked with both Ebaad and Arman in creating implementations for their ideas

Ebaad:

- Create forms (visually on our web app) for CRUD functions
    - Worked with Patrick for final implementation
- Overall UI manager
    - Each member was responsible for creating the UI for their own functionalities, Ebaad oversees how these UIs are combined together
- Bought a domain for our website (aussies.dog)!!!

Arman:

- Dog-friendliness algorithm
- Create visualization for search results
- Some ideas for advanced database programming
- Handled any otherwise necessary data manipulation
    - Data sanitization and random data generation