# CS 411 Stage 4 Demo 2

Implementation Challenges:

- We had a ton of challenges setting up the connectivity to GCP and understanding how GCP worked
- We couldn't successfully set-up SSH for VSCode so we had to use GitHub to pass all our local changes to the Compute Engine on GCP which slowed down our implementation of the backend and frontend code
- Implementing the stored procedures on the SQL Cloudshell and connecting it to our backend in Node JS took a few tries due to the specific syntax when calling the stored procedure
- We had trouble implementing the trigger through SQL because in order for an action to activate the trigger and the trigger to actually work we had to figure out a get around the problem of not being able to restart the server to accommodate the trigger
- One of the greatest challenges we faced was implementing the results page by correctly calling the SQL queries. While engineering advanced queries was simple, actually implementing the backend and frontend and having them properly call on each other to populate on our frontend became more cumbersome than initially thought.

Design Deviations:

- Our design was initially to have an immersive map to plot all these points for restaurants and crime data, however, we needed to pivot since we couldn't successfully connect Google Maps API to our application
- This led us to modify our search results display, but we believe that it's more clean and understandable this way
    - The user can use the location and name to further learn more about the destination they are looking for around their Airbnb location
    - A map would not have been very effective because a lot of the locations are within very close distances of each other so a map would have looked very cluttered
- We were going to add the ability for a user to add their AirBnB to the website, but we realized in terms of functionality this doesn't make sense and instead the AirBnB should first exist on the actual AirBnB website and database before it shows up on our page so that users of our website will not be shown misinformation in the form of fake AirBnBs
- We also decided to stray away from CitiBikes and use Subway Station data to have a more accessible application for anyone to use, as subway stations are cheaper and more frequently used than citi bikes across demographics.
- Another design deviation was that we initially included a button that would allow users to view all restaurants within a certain radius and modify/delete those restaurants, but we decided to deviate from that design for 2 reasons:
    1. Modifying and deleting restaurants would require user authentication (including a role attribute in the user table) and checking whether that user was a restaurant owner or not – if they were, only then would they be able to modify or delete. However, this user authentication was outside

the scope of our project and we decided to tackle this with our creative component:

2. We worked around this obstacle by implementing the creative component, where the user would be able to select the number of nearby restaurants and subway stations they wanted to see. This actually ended up being a better implementation – rather than simply seeing all the restaurants in new york, the user could see the top x amount of restaurants **nearby** their specific airbnb. Then, we were able to implement modify and delete functions on the user login, which was more practical for this project.

NoSQL Integration:

- There are many options for a NoSQL integration. One option we could have used instead is MongoDB. This would have also made it easier because it would have made it easier to host our datatables and connect it to a cloud service. This time we had to manually connect it and this was a huge implementation challenge.