Joshua Li

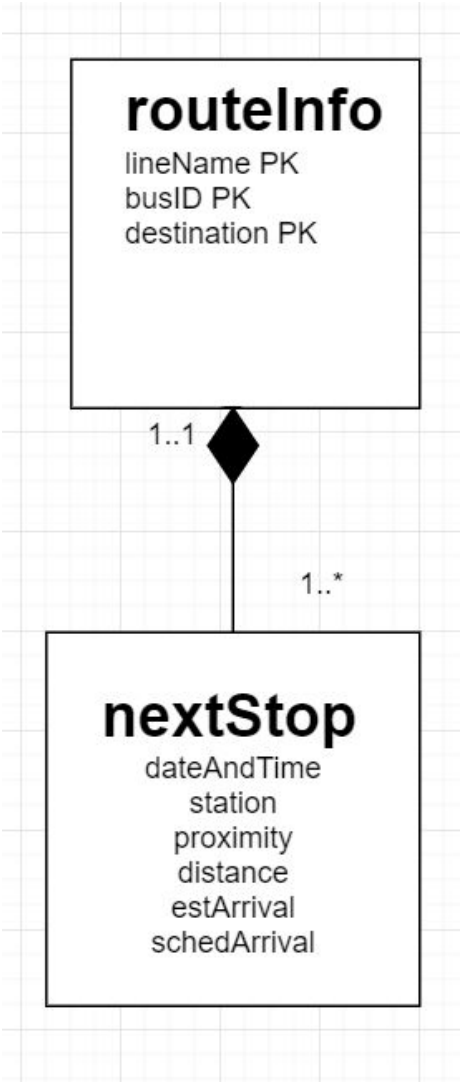
Introduction

My application is a bus tracker, using June 2017 data, that can be used to track every bus in NYC from June 1st to 30th, 2017. The dataset has a size of 1.38 GB. I wanted to do this project. First and foremost, I'm a native New Yorker who has depended on the bus and subway system all my life. As such, this dataset (literally) brings me close to home. I also want to promote bus usage in general all across America since buses are a valuable and environment-saving resource that too few Americans have respect for, especially in the South. For example, the entire Nashville bus system has a yearly ridership of 9-10 million, which is the same as the Q58, a SINGLE bus route in NYC (albeit the 8th most popular one). Through this project, I wanted to learn how I can make something useful and pleasing for Nashville residents to use the buses more often.

You might also be wondering why I chose to embark on this journey alone. Part of it was honestly due to my natural independence, but also due to my relative inexperience in programming, this being my third semester taking CS classes. I didn't want to weigh down other people in the class, but I also wanted to push myself to learn new things on my own. Lastly, I feel like no one in Vanderbilt is nearly as passionate about public transit as I am, so I thought only I could do it justice. In hindsight, I probably should've worked with someone, haha. Guess it's too late now.

Final Implementation

I used MySQL as my back-end, as instructed and used Java as my front-end. As I said in class, my brain is still stuck in 1101, so I decided to stick with what I know. However, I used Eclipse IDE instead of IntelliJ because I found it more programmer-friendly. The whole program is very simple: you just pick a route, terminal number, date in June, and time, and all the buses within a 10-minute range of your inputted time pop up with the stations that they are expected to arrive at and their arrival times. My normalization of the dataset was very sparse because I only used a few columns (especially excluding the longitude and latitude columns). So my UML diagram looks like this:



The routeInfo relation basically contains all the possible route configurations. Most buses have 2 terminals, although due to short-turns, special school trips, alternate terminals, and Limited bus routing (a quasi-express variant of some local buses), some may have up to 10 terminals. Each terminal has its own busID, which is used to make it easier for the user to choose which terminal they want. All three attributes are included as primary keys since making one attribute the primary key may eliminate data from the other attributes (although in hindsight, I probably should've excluded destination).

The nextStop relation essentially expands on the routeInfo relation and contains all the data pertaining to bus locations and arrival times. I intended on using proximity, distance, and schedArrival in my final product, but stuck with estArrival and station for cleaner outputs.

Unfortunately, the only advanced feature I used was a single Stored Procedure (load_buses) that basically plugged in all the User Inputs and made an easy-to-print

package. I'm aware that I'll likely get a lot of points deducted, but I didn't allow myself enough time to implement more features.

Testing

I used a much smaller subset of the dataset to test and present to the class. I extracted Q17 (my home bus route) data from June 1 to June 4 and built and presented the program around that. Eventually, I implemented all 300+ bus routes and all 30 days of June (which can be observed in my YouTube demo).

Walkthrough

The application is fairly self-explanatory, I assume. One needs to:

1. Open the MTA_buses directory via a Java IDE (I used Eclipse).
2. Open the Main.java class in the src folder
3. Click Run.
4. Pick a route (Q/M/Bx/B/S/X for Queens/Manhattan/Bronx/Brooklyn/Staten Island/Express + a number, generally between 0 and 116). Some number and borough combinations do not exist. It is generally safer to choose a lower number.
5. Given the terminal numbers and their respective terminals, pick a terminal.
6. Pick a day between 1 and 30, as June has 30 days.
7. Pick a time in military format (00-24:00-60).
8. Enjoy! Repeat if you want.

Summary

Of course, I wanted to go further with this project, maybe using a flashier UI or even a mobile app. However, as a solo team who is also a chronic procrastinator, I haven't been able to fulfill all my goals (or even all the goals outlined in the project spec). Nevertheless, I am proud of myself for kind-of completing my first full-fledged CS project! I feel like I learned a lot along the way and obviously made many mistakes that I will strive not to make again. Do I regret working alone? No. Will I work alone the next time I have a project like this? Definitely not.

The hardest and most time-consuming part of this project was figuring out how to load a huge CSV file into MySQL. I was under the impression that I had to make a secondary Python or PHP app to do the job and I was even tempted to by a service from online. After perusing through hundreds of StackOverflow threads, I finally stumbled on the solution: using LOAD DATA INFILE and using *double slashes!!!* After discovering that, the project was relatively smooth sailing. However, if I had discovered that earlier, who knows how much

more I could've accomplished. Connecting MySQL to Java was also challenging, but a YouTube tutorial led me the right way.

The Future of this Application

Over the summer, I hope to use this project as a shell for new learning endeavors, such as app development, other database languages like MongoDB, and other front-end languages like JS. I also want to learn how to turn it into something practical, using real-time data instead of a historical dataset.