CIS550: Project Write-Up

Michael Zhou

Max Ying

Tang Song

Kasra Koushan

Abstract

The central motivation behind our project was to delve into and understand the determining factors in a country's performance and success in the Olympics. We wanted to investigate the relationship between aggregate data points——such as gross domestic product (GDP), population, GDP per capita, literacy rates, homicide rates, and other statistics——and medal counts in various disciplines at the Olympics. Specifically, for the scope of this project we wanted to provide users with the opportunity to explore this data themselves and view the relationships in a clean and elegant manner.

Modules and Architecture

The architecture of our application is made up of a few different components. Our application was written primarily in JavaScript, within the Node.js runtime environment. We used the Express.js framework for our server—Express is light-weight, unopinionated, and flexible server framework on top of Node. In addition, we used the Bootstrap front-end framework in order to improve the look and design of our web application, and we used the Jade template language for HTML. In addition to these frameworks, we used several specific Node.js modules: mysql and Chartjs. The mysql module was used to connect our web application to our database, and the Chartjs module was used to create dynamic, animatable charts using the data we had.

Specifically, for the database component of our application, we used two databases. We used a MySQL instance stored on Amazon EC2, and we also stored a NoSQL database instance using Firebase, which is owned by Google. Initially, our relational database was stored on Oracle and managed using Oracle's SQLDeveloper, however we had trouble connecting our Node.js application to the database so we migrated our data to our current MySQL instance.

Data Instance Use

To connect our web application to the MySQL instance, we used the mysql Node module. Specifically, we used the module's `createConnection` method to create a connection to the database with our credentials. We made queries using the `query` method on the connection that we had.

Additionally, to connect our web application to our Firebase NoSQL instance, we used a firebase-admin module to perform queries and updates to the database.

Data Cleaning and Import Mechanism

We used several data import mechanisms. One of the major sources of our data was the UN Data repository, where we were able to specify data points such as literacy and crime rates and obtain data organized by country and year. This data was downloadable in CSV format and was then uploaded to our database in the same format. We also imported some data from Wikipedia and scraped data from a 132-page UNESCO report on literacy rates in various countries and years.

For some of our data, significant data cleaning was needed. The country names from the Olympic database and the UN Data did not match – one used 3-letter abbreviations while the other had full country names. We needed to convert all the 3-letter abbreviations into full country names in order to be able to do joins across the various tables in our data. Also, the UN data was not uniform across all countries and years – some statistics were only available for certain countries and years, so we had to do some filtering to make all of the data uniform and reduce the occurrence of null values.

Algorithms and Communication Protocols

Our project used AJAX and HTTP for communication between the server and the client. While most of the application did not require use of advanced algorithms, the technical challenges included securing a connection to the database from the server, as well as managing and rendering the data that was obtained from the database queries.

Use Cases

Our application has a few use cases. The primary use case is for students or hobbyists who are interested in using the data simply to learn more about the Olympic performance of different countries around the world. They can use the generated graphs simply to learn more about the Olympics and delve into the kinds of relationships that exist between Olympic medals and aggregate country statistics. Another use case for our application is for journalists and other researchers – people who wish to use Olympic data and country data as part of their research project or some sort of news article. Finally, our web application can also be used by Olympic watchers and enthusiasts from around the world – they can inspect the data for their individual countries and see how they stack up against other countries.

Optimization Techniques Employed

One of the primary optimization techniques employed in our application was the use of scrupulous schema design. Once the original schema was designed, we used the theory learned in class on normal forms, functional dependencies, and lossless joins in order to optimize the design of our tables for the kinds of queries we expected to make. We designed keys, foreign keys, and integrity constraints to ensure the robustness and scalability of our database as more and more data was to be inserted.

Highlighted Features

One of the interesting features of our app is our randomized landing page. Every time the user loads the page, some complex query from a set of pre-made queries is chosen and sent to the database, and the results are then rendered as a graph to the user. This feature enables the user to immediately get an idea about the usage of our application and get some inspiration about the kinds of ways data can be visualized. Another important feature of the application is the custom graph page. Here, the user can specify a country and a year, and this will be converted into a query to the database to get the number of medals won by women and men by that country in the given year. This allows the user to view how different countries have historically performed in the Olympics in an interactive and easily digestible way.

Division of Work

As provided in the project description

Michael – explored many javascript libraries and Node modules in order to write a majority of the controllers, and webpages for our project. Responsible for all the overhead regarding database interaction with the frontend as well as user interaction with the frontend.

Max – came up with complex SQL queries and provided SQL knowledge to design, and clean our database throughout the course of this project.

Tang – addressing issues with AJAX and JavaScript including the scope of callback functions, routes, and passing user forms back to the Express server..

Kasra – managing the project repository, uploading data to the database, investigating and using data visualization libraries.