StatSpot - Reflection Report

1. Changes in the directions of your project

There were no major changes to the initial proposal of the project. One UI change was that the Login page and the Registration page were combined into a single page which is the base page the application lands on.

2. What you think your application achieved/failed to achieve regarding its usefulness

Achievements

1. StatSpot was supposed to be an application which provided a one stop destination for all information needed by fantasy game players. We achieved this by sourcing data from multiple sources, creating a unified view for leagues, clubs, and players, and providing visualizations using graphs.

2. The subscription page provides the users an opportunity to get data in an excel format directly sent to their email address that helps users get data they can use as a source for their prediction tools.

Failures

1. Our application can only handle static visualizations currently. The History page shows 2 graphs based on specific queries and the application needs code changes to add new graphs to this screen. A more generic solution to add graphs on the go without any code changes will help us be more flexible.

3. Discuss if you change the schema or source of the data for your application

1. The initial application was supposed to be based on the “2021 Toyko Olympics" dataset. However, since we were going for a one stop solution for all sports, we added an additional data set called the “Football Transfer Market" dataset. Our database contains a combination of the 2 kaggle datasets merged into one set of tables after the extraction and data cleaning stage.

2. Added auto-increment IDs. We didn’t account for how we will be generating the ID field for the various entities present in our tables, and therefore had to make all IDs auto incremental later on.

3. Added a trigger to move Athlete information to an archive table - on delete of an Athlete, we move the data to an archive table called AthleteHistory. This adds a soft delete functionality.

4. Changes to ER diagram and/or your table implementations.

1. Addition of AthleteHistory table to add a soft delete function so that the data about an athlete is not lost on delete from the UI. This can in the future add a restore feature in case of mistakes.

2. Addition of 2 Report tables called FinalReportTable and FinalReportTable2 to hold information about the athletes to send as an email. These tables were created as a part of the stored procedure run to generate reports about an athlete.

5. Functionalities added or removed

1. Implemented the interactive graph component as an additional feature where the graph is able to take a user input (either young player age cut off or season) and update the graph accordingly.

2. The popups are multipurpose - we decided to use the same popup for edit, delete, and data view. This popup displays default data that can be edited or the entire player can be deleted.

6. How do advanced database programs complement your application?

1. Our application has 2 triggers - one is to move deleted athletes to the history table and the other updates the club's total market value if the player's value is updated. The first trigger is useful as it provides a soft delete function and can be used in the future to provide functionality to restore deleted data. The second trigger is useful as it maintains data consistency - a club's value depends on the players of the club.

2. The stored procedure is used to generate reports that are later sent as an email to the user. This idea of not moving large amounts of data to the backend over the network and performing all operations in the database helped our application have less load on the backend server in terms of memory, network bandwidth, and processing power.

7. Technical challenges faced by each team member

1. **Akshath SK (sk117)** - Handling the email module was a difficult task as the amount of data was very large and it could not be processed in memory. So, a stored procedure was used to solve this problem.
   Creating a stored procedure that can accept parameters proved to be a challenging task that required significant time and effort. Debugging the procedure was particularly difficult, as errors were occurring at unknown points during execution, despite the syntax appearing to be correct. Ultimately, I was able to resolve the issue by breaking down the stored procedure and executing it one step at a time.

2. **Jaelyn (jaelyne2)** - Data consistency was difficult to achieve as the dataset has a lot of interconnected data items. The club information had to be updated with changes to the players. To handle this problem a trigger was implemented on update of an athlete. This allowed the club's total market value to be updated as and when the player's information was updated. The development of the frontend presented its own set of challenges. The complexity of the nested data retrieved from the backend made it difficult to integrate and display the information effectively in the frontend. It was crucial to establish precise communication with the database to ensure seamless application performance. Through an incremental development approach and attention to detail, I was ultimately able to achieve the desired end result.

3. **Jessica (js81)** - Identifying the appropriate columns/attributes to which indexes should be applied. While the EXPLAIN ANALYZE command proved helpful in determining whether an index was improving query performance, some indexes had the opposite effect. As a result, it was necessary to employ a trial-and-error approach to determine the ideal attribute combinations to apply triggers on. Another challenge I faced was streamlining the backend code that executes whenever a particular table was inserted. To address this, I developed a trigger that eliminated the need for sequential programmatic queries by moving redundant DML to the database layer.

4. **Anthony (arihani2)** - Loading data from 2 datasets into a single database was a complicated task as it involved extracting data, matching similar data, cleaning data and adding data to one single database with a set of tables. The data had to be manipulated to fit the defined schema. This involved writing a custom jar that sourced data from the 2 datasets and populated data maintaining the constraints. The jar also considered the foreign key constraint.

8. Other things that changed comparing the final application with the original proposal

1. The subscribe page now accepts an email id field which allows the user to send emails to any of his email ids and doesn't restrict the user to get mails only on the registered email account. This gives the users more flexibility while using the subscription service.

9. Future work that the application can improve on

1. Implementation of Materialized Query Tables - Since the application is generated from a fixed set of tables and queries, we can implement MQTs to hold this data and refresh on every update.

2. The backend can be moved to a microservice architecture to handle clubs, leagues and athletes with their own microservice.

3. Dynamic Graph Generation - using NoSQL DB to provide inputs to graph generation. We can move to a state where adding a new json to the mongo tables creates a new graph on the UI.

10. Describe the final division of labor and how well you managed teamwork.

Task	Description	Assignee
Login & SignUp	1. UI, UX and Backend design of the Login and SignUp page 3. Store session related information in the MySQL@GCP database 5. APIs design for sign-up and sign-in with backend validation	Anthony (arihani2)
Data sourcing and cleanup - 2021 Olympics in Tokyo	1. Write a backend script to read data from the kaggle dataset "2021 Olympics in Tokyo" 2. Design and create tables to hold the sourced data 3. Data cleanup will be done wherever necessary 4. Trigger should be written to update the statistical data like win/loss count on addition of new player data (should be transactional)	Anthony (arihani2)
Data sourcing and cleanup - Football Data from Transfermarkt	1. Write a backend script to read data from the kaggle dataset "Football Data from Transfermarkt " 2. Design and create tables to hold the sourced data 3. Data cleanup will be done wherever necessary 4. Trigger should be written to update the statistical data like win/loss count on addition of new player data	Akshath (sk117)
Statistics - Landing Page View	1. UI, UX and backend design for the Statistics landing page 2. Data sourced from the "Football Data from Transfermarkt " dataset will be displayed in the landing page 3. The tile view will show a summary of the player, club or league 4. Clubs and leagues will have their logo displayed on the screen (default logo if not available 5. Provide backend API to GET data sourced from the dataset and stored in the MySQL GCP tables 6. Users will have a search bar to search across clubs, players and leagues	Jaelyn (jaelyne2)

Statistics - Add, Edit & Delete Operations	1. UI, UX and backend design to support add, edit and delete of player 2. 3 backend APIs to POST, PUT and DELETE data from the tables 4. Successful data modifications should be immediately visible on the UI	Anthony (arihani2)
History Page - Visualizations	1. Data sourced from the "2021 Olympics in Tokyo" dataset will be displayed on the History Page 2. UI should be able to generate Bar Graphs for any competition 3. Backend API to provide an API to provide data to render the pie chart and bar graph	Jaelyn (jaelyne2)
History Page - More Details	1. Data related to athletes and coaches sourced from the "2021 Olympics in Tokyo" dataset will be displayed on the History Details Page 2. UI displays the More Details page on click of the bar graph or the pie chart 3. Details about the Players and Coaches are displayed as a table on this screen 4. API to provide Player and Coach details for the selected competition	Jessica (js81)
Subscribe Page - New Subscription	1. UI, UX and Backend design for the subscription page 2. This page will have a date time picker which works in the user's timezone 3. The search functionality will search through players, clubs, leagues together 4. POST API at the backend will store the subscription details for each user	Akshath (sk117)
Subscribe Page - Scheduler	1. Mails are expected to be sent 2. The subscription details including players, clubs and leagues are taken from the DB including the user details	Jessica (js81)
Subscribe Page - Excel	1. For each subscribed user, the players of interest, clubs and leagues are retrieved from the DB 2. Use a Java Excel library to create an excel in the backend	Jessica (js81)

Subscribe Page - Email	1. Setup an SMTP server to send emails 2. The subscribed data will be sent to the user 3. The excel will be an attachment to the email	Jaelyn (jaelyne2)
Deployment on GCP	UI, Backend, DB should be deployed on Google Cloud	Akshath (sk117)

Team management – Weekly Scrum meetings were conducted to gain insights into individual tasks and address any obstacles encountered. It was ensured that each person's skills were put to good use and aligned with their assigned work. Everyone took responsibility for their designated task and completed it within the estimated time.