Joshua Ayaviri
Lizzy Hwang
Professor Kathleen Durant
CS 3200 Database Design
20 April 2020

FINAL PROJECT REPORT

I. README - Files, Software, and Libraries

This project .zip file contains:

- This report - AyaviriHwang_final_report.pdf
- MYSQL self-contained dump - soccer_project_dump.sql
- Java database application - DBDFinal.java

The link to our presentation can be found **here**.

Our database was created using **MYSQL Workbench Version 8.0.19** and **Eclipse IDE for Java Developers Version 2020-12 (4.18.0)** in Java SE-11, using the **JDBC connector library**.

Downloading and Installing MYSQL WorkBench

To install MYSQL WorkBench Version 8.0.19, click **here**, select *8.0.19* from the *Product Version* drop-down menu, and choose your *Operating System* from the drop-down menu directly below. Upon selection, a file(s) will appear below, along the naming convention of "macOS (x86, 64-bit), DMG Archive", depending on your operating system. Hit download, and go through the installation process for your machine.

Opening Our Project in MYSQL WorkBench

Once you have MYSQL downloaded, open the application and open a new local connection. Open the provided SQL dump of our project (File > Open SQL Script > your/directories/... soccer_project_dump), put your cursor at the beginning of the first line, and hit the lightning bolt to run the entire file. Click the refresh button in the upper right of the schemas panel. You should now have the schema set up on your connection.

Downloading JDBC Library

To download the JDBC mysql-connector-java-8.0.23 library, click **here**, select *8.0.23* from the *Product Version* drop-down menu, and choose your *Operating System* from the drop-down menu directly below. If you do not see your OS, choose *Platform Independent*. Upon selection, a file(s) will appear below. Download the file, save it somewhere, and then unzip or open it to access the .jar file.

Downloading and Installing Eclipse IDE

To install Eclipse IDE for Java Developers, click **here**, and select your operating system beneath the heading, *Download Links*. You will be taken to another page. Click the big orange "download" button, and go through the installation process for your machine.

Opening Our Project and Installing the JDBC Library in Eclipse IDE

Once you have Eclipse downloaded, create a new project (File > New > Java Project) and name it as desired before hitting *Finish*. On the left hand side in the *Package Explorer* window, you should now see your project. Right click on the project and go to *Properties*, which opens a new window. On the left panel, hit *Java Build Path* and make sure you're on the *Libraries* tab. Click on *Classpath*. On the right, select *Add External Jars*, and then select mysql-connector-java-8.0.23.jar file from whichever directory you saved it in. Hit *Open*, then *Apply and Close*. Lastly, click the *src* folder of your java project, then go to File > Open File, and open the provided .java file for our project.

Connecting to the Database from Eclipse IDE

Once you have our .jar file opened, go to the top of the JavaMySql class and change the variables to your own username, password, servername, and portNumber. You are now ready to view and access both the SQL and Java ends of our database.

II. Project Description

Overview

Our database keeps track of entities that make up a soccer club. Users interact with the club's database in order to perform tasks such as registering players, getting contact information, and viewing game schedules. There are four types of users: admin, coaches, managers, and parents. Players have one parent. A collection of players form a team, and a collection of teams form an age group. Teams play in matches.

User interface

The user interface is currently housed in the Eclipse IDE Console. Upon running the program, the Eclipse Console window opens and the user can interact with it. Each type of user has access to a certain set of procedures they are allowed to call.

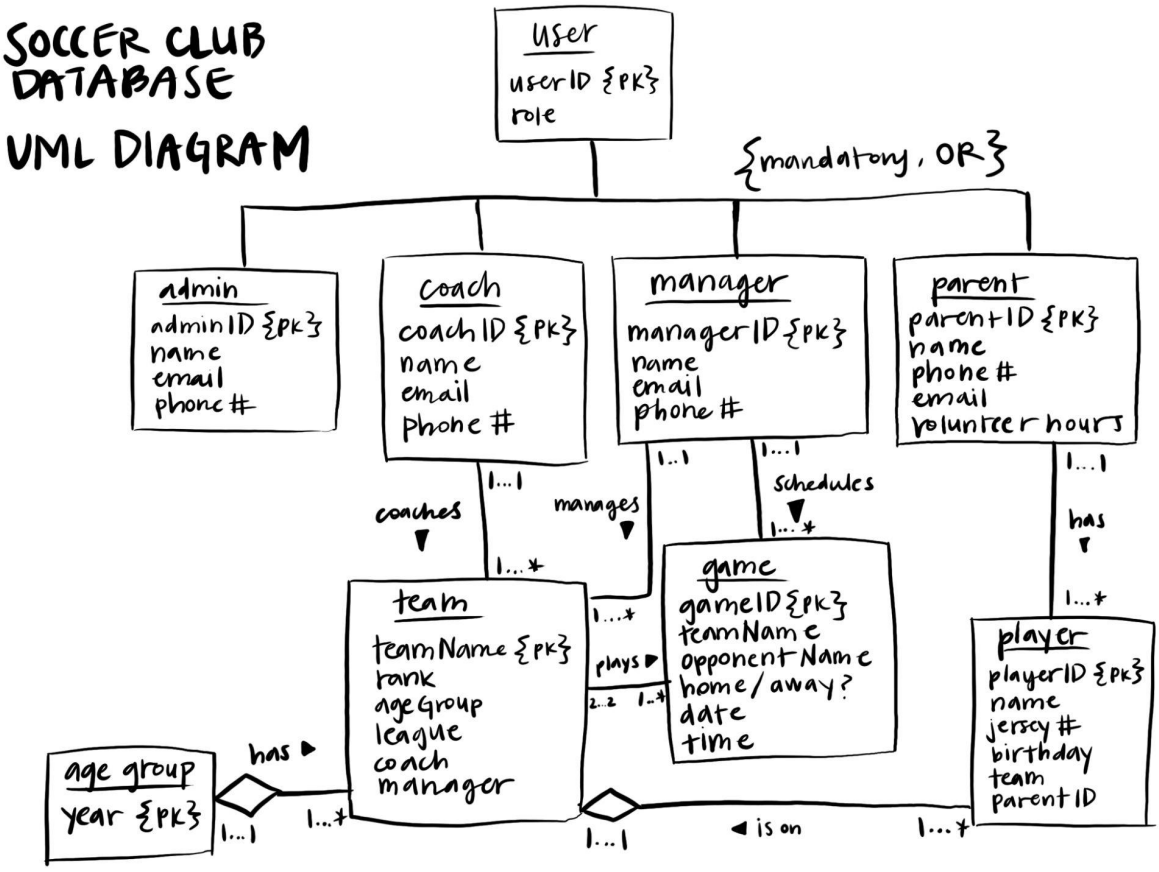
The user logs in with their userId and is prompted with an enumeration of tasks they are able to perform, as well as an option to log out of the application. The user can then type in the number corresponding with each task in order to select which procedure they would like to call. For example, a "parent" user might type "1" in order to "[1] Register a Player". If the selected procedure requires input values, the database prompts the user for each variable. The user types in their response and moves on by pressing Enter. For example, if registering a player, the program will prompt, "Please provide the player's name". The parent can type in "Johnny Appleseed" and hit Return, which will then prompt them for the player's jersey number, birthday, etc, in a similar fashion.

Once finished, the program confirms that the task was completed successfully before returning the user back to their original list of tasks. The user can decide whether to perform another task or log out of the application.

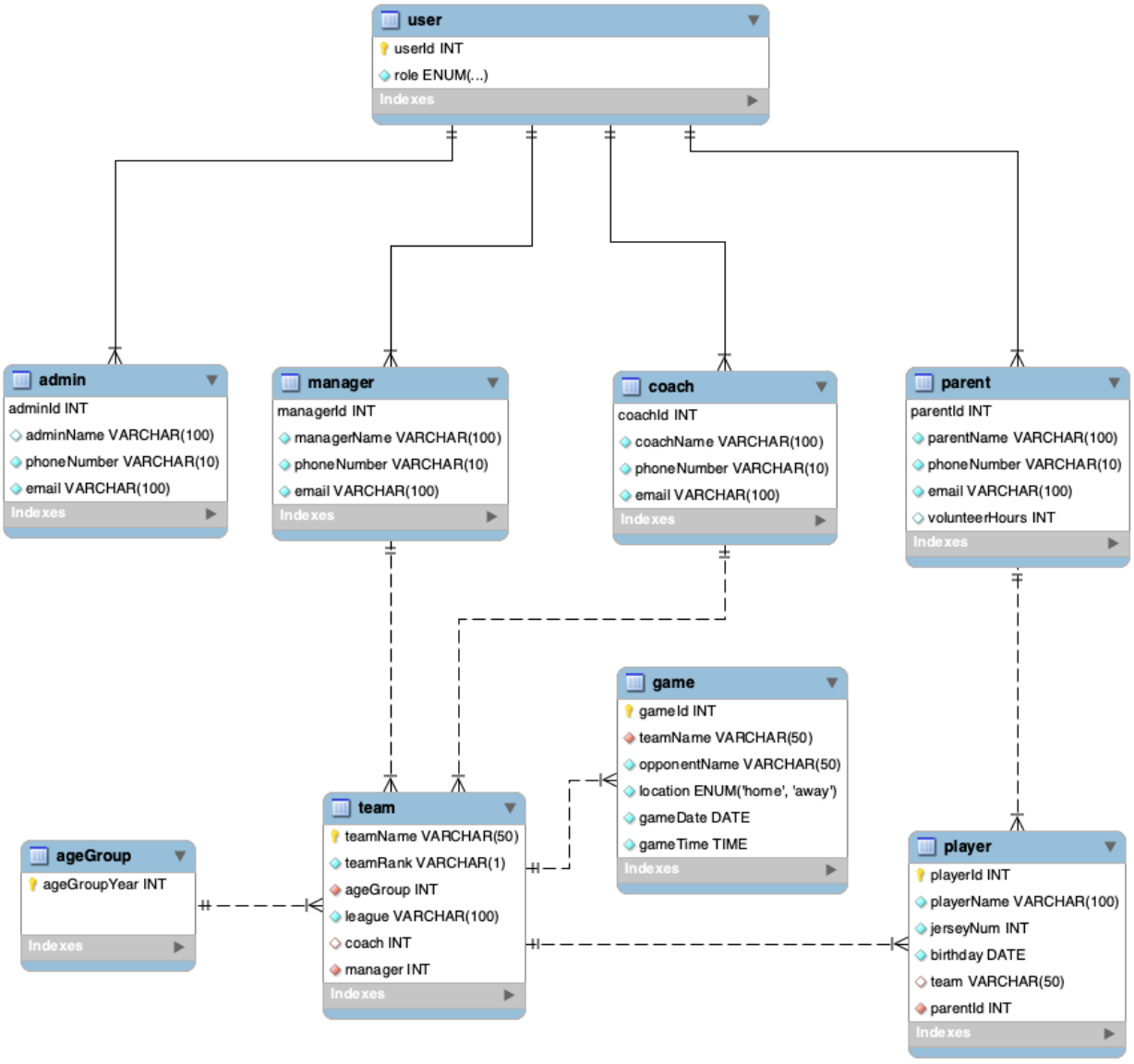
Error handling

There are several examples of error handling in the Java program that accesses the database. Every method surrounding its accompanying procedure in MySQL has a try / catch statement in order to ensure that the user is alerted when a procedure does not go through. In addition, there are several instances of invalid input checking. If a team manager is trying to create a new match, they may not input a team that they do not directly manage. If they try to do so, the program will prompt the user again for a team name. Since a while loop controls this, the user will not be able to continue until a valid input is given. This invalid input checking extends to various other operations, such as managers updating a match's start time or removing a match, and parents removing a player. Removing a player has two layers of invalid input checking: the program ensures that the player is registered in the database and that the parent is actually the parent of the player being removed.

III. Current Conceptual Design



IV. Logical Design (MYSQL workbench diagram)



V. Procedures for User Interaction

A. **CREATE**:

1. register_player(jerseyNumber, playerName, birthday, team, parentId)
    a) A Parent can register a new player (their child).
2. create_game(teamName, opponent, location, gameDate, gameTime)
    a) A manager can create a new match (for their team).

B. **READ**:

1. get_coach_contact(coachName)

a) Parents can retrieve contact information for any coach.

2. get_team_players(teamName)
    a) Managers and Coaches can get the roster for any team in the club.
3. get_parent_contact(parentName)
    a) Managers can get any parent's contact information
4. get_teams()
    a) Admin can get a list of team names from the entire club.
5. season_schedule(teamName)
    a) Managers can get a list of all the games scheduled for a specific team, in order by date then time.

C. **UPDATE**:

1. update_parent_phone(parentId, phoneNumber)
    a) Parents can update their contact information
2. update_game_time(gameId, gameDate, gameTime)
    a) Managers can update a match's time
3. update_player_team(playerId, teamName)
    a) Admin can change which team a player is on
4. update_coach_phone(coachId, phoneNumber, email)
    a) Coaches can update their contact information

D. **DELETE**:

1. delete_player(playerId)
    a) Parents can unregister their child.
2. delete_game(gameId)
    a) Managers can delete a game (cancel it).
3. delete_coach(coachId)
    a) Admin can remove a coach from the database (in the case of firing them)

E. **OTHER**:

1. biggest_age_group()
    a) Admin can see which age group that has the most players
2. month_with_most_games()
    a) Admin can see the month with the most games scheduled, and how many games are scheduled for it.
3. parent_child_team_roster(teamName)
    a) Managers can get a list of players and their parents + contact information for a given team.

VI. Lessons learned

Technical expertise gained

This project required expertise in both MYSQL and Java. Our partnership is ideal because Lizzy is best in MYSQL, and Joshua is most comfortable in Java. While we conceptualized together, Lizzy handled structuring the tables and creating procedures in MYSQL while Joshua wrote the

application in Java and designed the user experience. Our fluency from opposing ends let us help the other in designing the best code for the two programs to merge seamlessly.

Although we were strong in each of our softwares to begin with, the process of creating our database further strengthened our understanding of MYSQL and Java.

Lizzy became efficient in creating .csv files of example data and using the import table wizard to file data into tables she created. She also gained expertise in creating data dumps and using the MYSQL Workbench to reverse engineer schemas into a logical database design. She also became much more comfortable with which order to create tables in (due to foreign key references), and how to represent one to many relationships in tables. Lizzy also learned more commands to manipulate the database's example data, such as resetting auto-increments, pulling the dump code for creating tables to find auto-named foreign key constraints, and altering tables to remove or add foreign keys.

Joshua grew more comfortable with the use of the PreparedStatement and ResultSet objects. He learned how to read any given cell from a ResultSet, and he also learned how to pass in arguments to stored procedures created in MySQL. In addition to this, Joshua also learned more about try / catch statements, which are very useful in error handling. Overall, he obtained more practice with all of the objects and methods related to the Java Database Connectivity (JDBC) API.

Insights

We went into this project a bit daunted and nervous about creating our own database. However, after discussion and conceptual planning, we started feeling more confident about our design approach and the work that needed to be done to complete it. Although the tasks no longer seemed as difficult as we initially thought, we were both surprised at how much time it actually took to write the code on both ends, and the learning curve that came with creating an application from scratch, and in hindsight, we should have spent more time early on in order to reduce how much work we had to do as the deadline drew nearer. As we progressed through the project, we learned that it's important to develop good work flow, such as re-using code and planning out our time with detailed to-do lists, so that we can work as quickly and efficiently as possible. We also became much more comfortable asking for help and learning new skills instead of giving up and writing "easier" code or procedures. We utilized our resources, like office hours, online library documentation, and community forums, in order to learn skills we didn't know.

Realized alternative design approaches to the project

Although fully functional, there are many design improvements to be made on the Java program that connects to the MySQL database. Every procedure written in MySQL has an equivalent method in Java within the JavaMySQL class, and many perform the same tasks or obtain the similar result sets. Therefore, abstraction could have been done to shorten the code greatly. A

more elaborate class design could also be implemented to distribute the work among several classes, decluttering the code and making it much easier to revisit. However, this is a console-based program; the code can only be shortened so much. On the MySQL side, there are few (if any) improvements to be made to the design. The current organization of tables is sleek, and as a result, the procedures written for these tables are not required to do heavy lifting to work around poor design choices.

VII. Future Work

Our program can be used for any soccer club that needs a system for different types of users to interact with stored data or add data to their database. Its current greatest asset is handling permissions and tasks based on identifying which type of user is logged in. Although the application is not fully fleshed out, it has the potential to keep a soccer club organized and driven by user type.

Plans for Additional Functionality:

- Have the option to either “Sign In” or “Join”. Currently, the user can only “Sign In”.
- Allow the user to create a login password.
- Allow a child to be linked to more than one parent. Currently, a parent may have more than one child, but a child can only have one parent.
- Allow parents to also be managers.
- Allow players to be rostered on more than one team.
- Allow teams to have an optional assistant coach.
- Add more error handling on user input, such as string-length
- Let parents update their volunteer hours, but only have it actually update once an admin approves the update.
- Once a parent has 100 approved volunteer hours, automatically refund their $100 Volunteer Fee that was paid at their child’s registration.
- Be able to connect a bank account or other financial services so users can make payments or receive refunds. This would need to be encrypted for security and privacy of the user.
- Move the user interface from the Eclipse console to a website.