HoopMetrics Database Systems Project

CS 4347.502 | Professor Jalal Omer

Group 17: Joel Varghese (jjv200001) , Anish Dasti (akd200002), Manab Sthapit

(mxs200105)

4/26/2024

TABLE OF CONTENTS

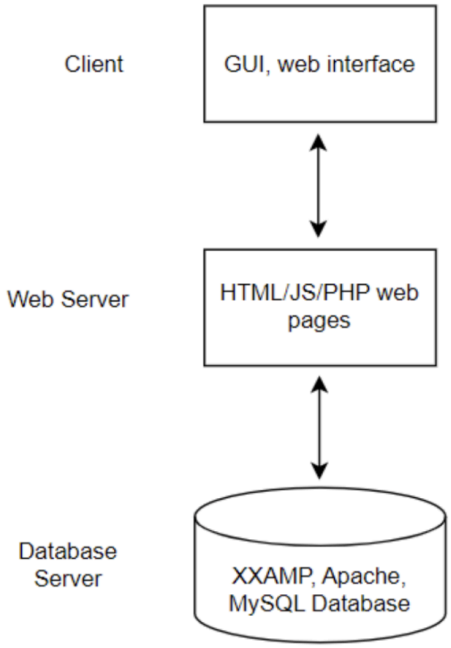
Introduction

HoopMetrics: Grassroots Basketball Analytics Platform is designed to enhance the sporting experience at the University of Texas at Dallas (UTD) through data-driven analytics specifically tailored for intramural basketball. In a setting where sports play a pivotal role in community and student life, our platform aims to revolutionize how players, coaches, and administrators engage with sports data. The project leverages modern database technologies to create a comprehensive system that not only stores but also allows for the querying of performance data. This initiative is particularly significant because it addresses the gap in robust data handling that is often found at the grassroots level of sports management, which is typically reliant on rudimentary tools like spreadsheets. By implementing a dedicated database system, we enable real-time access and analysis, facilitating informed decision-making that can enhance player performance and strategic planning.

Report Organization:
This report provides a detailed account of the HoopMetrics project, covering all technical and operational aspects:
1. System Requirements: Outlines the system architecture, interface requirements, and functional and non-functional requirements.
2. Conceptual Design of the Database: Details the Entity-Relationship (ER) model, including the data dictionary and business rules.
3. Logical Database Schema: Discusses the translation from ER model to the relational schema, showcasing the schema structure and SQL statements for database creation.
4. Functional Dependencies and Database Normalization: Analyzes the functional dependencies and demonstrates the normalization process up to the Third Normal Form (3NF).
5. The Database System: Explains the installation and usage of the system, supplemented by screen dumps for clarity.
6. User Application Interface: Describes the system's user interface and user interaction.
7. Conclusions and Future Work: Reflects on the project's outcomes and explores potential future enhancements.

System Requirements

4.1 Context Diagram (System Architecture)



4.2 Interface Requirements

For the HoopMetrics database project, our web interface is designed to be both user-friendly and functional, catering to different user roles such as players, coaches, and administrators. This interface incorporates four main HTML forms, each corresponding to a core table in the database—Players, Teams, Games, and Player Stats. These forms are integral to performing all four CRUD operations (Create, Read, Update, Delete) on each table:

1. Player Registration Form: Enables new players to register by entering their details into the Players table, with fields like FirstName, LastName, and Team, among others. The form ensures data validation before submission.

2. Team Creation Form: Facilitates the creation of new teams, populating the Teams table

3. Game Entry Form: Used for logging game details such as scores, updating the Games table and linking to team and player data.

4. Player Stats Entry Form: Allows entry of detailed performance data for players in games, updating the Player_Stats table to support comprehensive statistical tracking and analysis.

Additionally, a Query Interface supports dynamic querying for data retrieval and display, such as team standings or player statistics, enhancing the system's interactivity and usability for all users. This streamlined approach to interface design ensures comprehensive data management through straightforward, role-specific interactions.

4.3 Functional Requirements

- Data Entry: Ability to enter and update player information and stats, game results, and team information.
- Data Retrieval: Capability to query and retrieve specific data, like player performance metrics or team records.
- Data Updation: Enables authorized users to update existing records to correct or add new information as the season progresses or when errors are discovered.
- Data Deletion: Permits the removal of outdated or incorrect data, ensuring that the database maintains its accuracy and relevance.
- Password authentication: Requires a system wide password to update or retrieve sensitive personal data

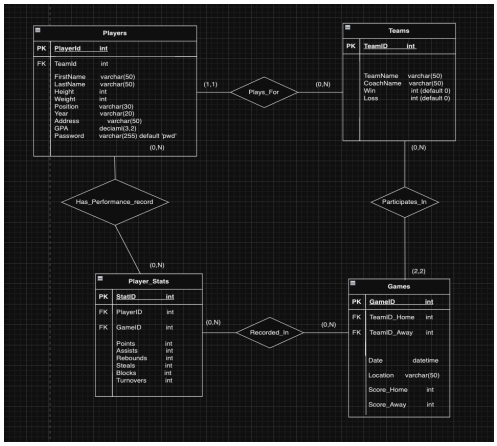
4.4 Non-Functional Requirements

- Performance: The system should perform queries and generate reports within seconds, even during peak usage.
- Reliability: HoopMetrics must be reliable, with an uptime goal of 99.9% during the sports season.
- Security: Detailed security measures to protect sensitive data, including encrypted connections and secure password protocols.
- Scalability: The system should be scalable to accommodate increasing amounts of data and concurrent users without degradation in performance.

Conceptual Design of Database

5.1 Entity-Relationship (ER) Model

The ER model visually represents the data and the relationships between the tables in the HoopMetrics database. This model facilitates understanding how data is interconnected within the system and supports the design of the relational schema.



The diagram includes entities such as Players, Teams, Games, and Player_Stats, each connected by relationships that reflect the real-world interactions within the intramural basketball league.

5.2 Data Dictionary and Business Rules

Data Dictionary:

Teams Table
- TeamID: Unique identifier for each team. Auto-incremented primary key.
- TeamName: Name of the team. VARCHAR(50).
- CoachName: Name of the coach. VARCHAR(50).
- Win: Number of games won. Default value: 0.
- Loss: Number of games lost. Default value: 0.

Players Table
- PlayerID: Unique identifier for each player. Auto-incremented primary key.
- FirstName: Player’s first name. VARCHAR(50).
- LastName: Player’s last name. VARCHAR(50).
- TeamID: Identifier of the team the player belongs to. Foreign key referencing Teams(TeamID).
- Height: Player's height in centimeters. INT.
- Weight: Player's weight in kilograms. INT.
- Position: Player's position on the team. VARCHAR(50).
- Year: Year of study or player's class. VARCHAR(50).
- Address: Player's address. VARCHAR(255).
- GPA: Player's grade point average. DECIMAL(3, 2).
- Password: Player's password for authentication. Default value: 'pwd'.

Games Table
- GameID: Unique identifier for each game. Auto-incremented primary key.
- Date: Date of the game. DATE.
- TeamID_Home: Identifier of the home team. Foreign key referencing Teams(TeamID).
- TeamID_Away: Identifier of the away team. Foreign key referencing Teams(TeamID).
- Score_Home: Number of points scored by the home team.
- Score_Away: Number of points scored by the away team.

Player_Stats Table
- StatID: Unique identifier for each player statistic. Auto-incremented primary key.
- PlayerID: Identifier of the player. Foreign key referencing Players(PlayerID).
- GameID: Identifier of the game. Foreign key referencing Games(GameID).
- Points: Number of points scored by the player in the game.
- Assists: Number of assists made by the player in the game.
- Rebounds: Number of rebounds grabbed by the player in the game.
- Steals: Number of steals made by the player in the game.
- Blocks: Number of blocks made by the player in the game.
- Turnovers: Number of turnovers committed by the player in the game.

Business Rules:
- Each player must be associated with exactly one team, but each team can have multiple players.
- Games must involve two different teams.
- Player statistics are recorded per game; each entry in the Player_Stats table corresponds to a single game played by a player.
- If a Game or Player is deleted, all Stat Entries associated with that game or player also get deleted. Team records will also be updated as a result of deleting games.
- If a Game is updated, and the score changes the outcome of the game, the team records will be updated as a result.
- A player must be assigned to a Team at all times (no free agents allowed).

Logical Database Schema

6.1 Schema Translation and Structure

In the transition from the conceptual ER model to the logical database schema, the following mappings are applied to transform entities and relationships into relational tables, ensuring that all necessary constraints, such as primary and foreign keys, are properly established to maintain data integrity and relationship rules.

- Teams Table:
  - Translated directly from the Teams entity.
  - Attributes include `TeamID` (primary key, auto-incremented), `TeamName`, `CoachName`, `Win`, and `Loss`.

- Players Table:
  - Derived from the Players entity.
  - Attributes such as `PlayerID` (primary key, auto-incremented), `FirstName`, `LastName`, and other player-specific details like `Height`, `Weight`, `Position`, `Year`, `Address`, `GPA`, and `Password`.
  - Includes a foreign key `TeamID` linking to the `Teams` table, reflecting the relationship that each player belongs to one team.

- Games Table:
  - Corresponds to the Games entity.
  - Attributes include `GameID` (primary key, auto-incremented), `Date`, `TeamID_Home`, `TeamID_Away`, `Score_Home`, and `Score_Away`.
  - Foreign keys `TeamID_Home` and `TeamID_Away` connect to the `Teams` table, representing the participating home and away teams.

- Player_Stats Table:
  - Maps to the Player_Stats entity.
  - Includes `StatID` (primary key, auto-incremented) and statistical attributes like `Points`, `Assists`, `Rebounds`, `Steals`, `Blocks`, and `Turnovers`.
  - Foreign keys `PlayerID` and `GameID` link to the `Players` and `Games` tables respectively, indicating which player the stats belong to and in which game.

This schema effectively represents the database structure required for managing the complex data relationships inherent in a basketball analytics system, ensuring clarity and efficiency in data management.

6.2 SQL Statements for Schema Construction

```
CREATE SCHEMA IF NOT EXISTS HoopMetrics;
USE HoopMetrics;

CREATE TABLE Teams (
    TeamID INT AUTO_INCREMENT PRIMARY KEY,
    TeamName VARCHAR(50),
    CoachName VARCHAR(50),
    Win INT DEFAULT 0,
```

```
    Loss INT DEFAULT 0
);

CREATE TABLE Players (
    PlayerID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    TeamID INT,
    Height INT,
    Weight INT,
    Position VARCHAR(50),
    Year VARCHAR(50),
    Address VARCHAR(255),
    GPA DECIMAL(3, 2),
    Password VARCHAR(255) DEFAULT 'pwd',
    FOREIGN KEY (TeamID) REFERENCES Teams(TeamID)
);

CREATE TABLE Games (
    GameID INT AUTO_INCREMENT PRIMARY KEY,
    Date DATE,
    TeamID_Home INT,
    TeamID_Away INT,
    Score_Home INT,
    Score_Away INT,
    FOREIGN KEY (TeamID_Home) REFERENCES Teams(TeamID),
    FOREIGN KEY (TeamID_Away) REFERENCES Teams(TeamID)
);

CREATE TABLE Player_Stats (
    StatID INT AUTO_INCREMENT PRIMARY KEY,
    PlayerID INT,
    GameID INT,
    Points INT,
    Assists INT,
    Rebounds INT,
    Steals INT,
    Blocks INT,
    Turnovers INT,
    FOREIGN KEY (PlayerID) REFERENCES Players(PlayerID),
    FOREIGN KEY (GameID) REFERENCES Games(GameID)
);
```

6.3 Database Operations and Estimated Data Volumes

The operations performed on the HoopMetrics database are crucial for maintaining its responsiveness and accuracy. Here we consider the common operations and estimate the data volumes for each table:

- Insert Operations: Frequent during the basketball season as new games are played and player statistics are updated.
  - `Players` and `Teams` tables are relatively stable post initial setup but may have occasional additions.
  - `Games` table sees regular updates each game day.
  - `Player_Stats` table has high-frequency updates during active seasons, with entries added for each player in every game.

- Update Operations: Necessary for correcting or updating existing records, particularly for player information and team records.
  - `Players` table may require updates for changes in player details, such as address or team transfers.
  - `Games` may update Team win/loss records after each game.

- Delete Operations: Less frequent, used mainly for removing outdated records or errors.
  - Data integrity must be carefully managed to avoid cascading deletions that could affect multiple tables.

- Query Operations: The most frequent operation, crucial for retrieving player performance data, team standings, and historical statistics.
  - Complex queries involving multiple joins may be necessary, especially for generating statistical summaries or league rankings.

- Estimated Data Volumes:
  - `Players`: 100-200 records per season.
  - `Teams`: 20-30 records.
  - `Games`: 100-150 records per season.
  - `Player_Stats`: 1000-1500 records per season, assuming multiple entries per player per game.

These operations and volumes suggest that the database needs to be optimized for frequent reads and moderate writes with considerations for peak times during and immediately after games.

Functional Dependencies and Database Normalization

7.1 Analysis of Functional Dependencies

- Teams Table:
  - TeamID → TeamName, CoachName, Win, Loss
    - TeamID uniquely determines every other attribute in the Teams table.

- Players Table:
  - PlayerID → FirstName, LastName, TeamID, Height, Weight, Position, Year, Address, GPA, Password
    - PlayerID is a unique identifier for each player and determines all other attributes of a player.

- Games Table:
  - GameID → Date, TeamID_Home, TeamID_Away, Score_Home, Score_Away
    - GameID uniquely determines the attributes of each game.

- Player_Stats Table:
  - StatID → PlayerID, GameID, Points, Assists, Rebounds, Steals, Blocks, Turnovers
    - StatID uniquely determines all recorded stats and the game and player they relate to.

7.2 Normalization Process

Normalization is performed to ensure that the database schema is free from undesirable characteristics like redundancy and composite dependencies that can lead to anomalies in data manipulation.

- Teams, Players, Games, Player_Stats Tables:
  - All these tables are already in 1NF because they have no repeating groups or arrays.
  - They are in 2NF as all non-primary-key attributes are fully functionally dependent on the primary key.
  - They are in 3NF as there are no transitive dependencies; all attributes are directly dependent on the primary key and not on any other non-primary key attribute.

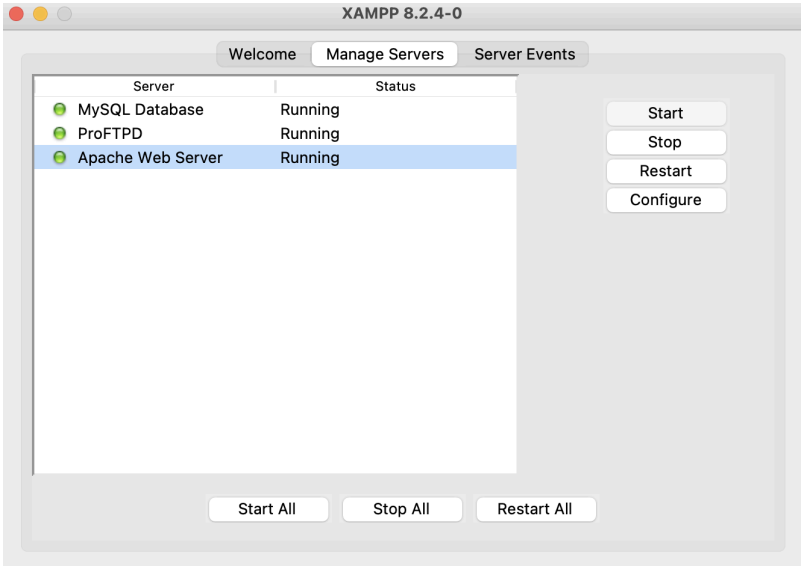
7.3 SQL Statements for Constructing Normalized Tables

The tables, as defined, are already normalized to 3NF based on the dependencies identified. Thus, no additional modifications are necessary for normalization purposes.

The Database System

In order to get the system to work, you will have to install XXAMP and Apache onto your computer. You can do so by going to this website: https://www.apachefriends.org/

You will need to run apache and MySQL.
This is what the XAMPP Panel looks like:



Once Apache and MySQL are running, create a database called HoopMetrics and create the required tables necessary using phpmyadmin at http://localhost/phpmyadmin/

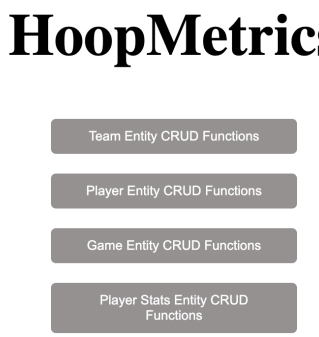
Create a folder in the htdocs folder wherever you placed the xampp directory on your computer

Download the files from this repository: https://github.com/anishdasti/HoopMetrics

Place all of the files from the project folder here on the GitHub, into the folder created in your local.

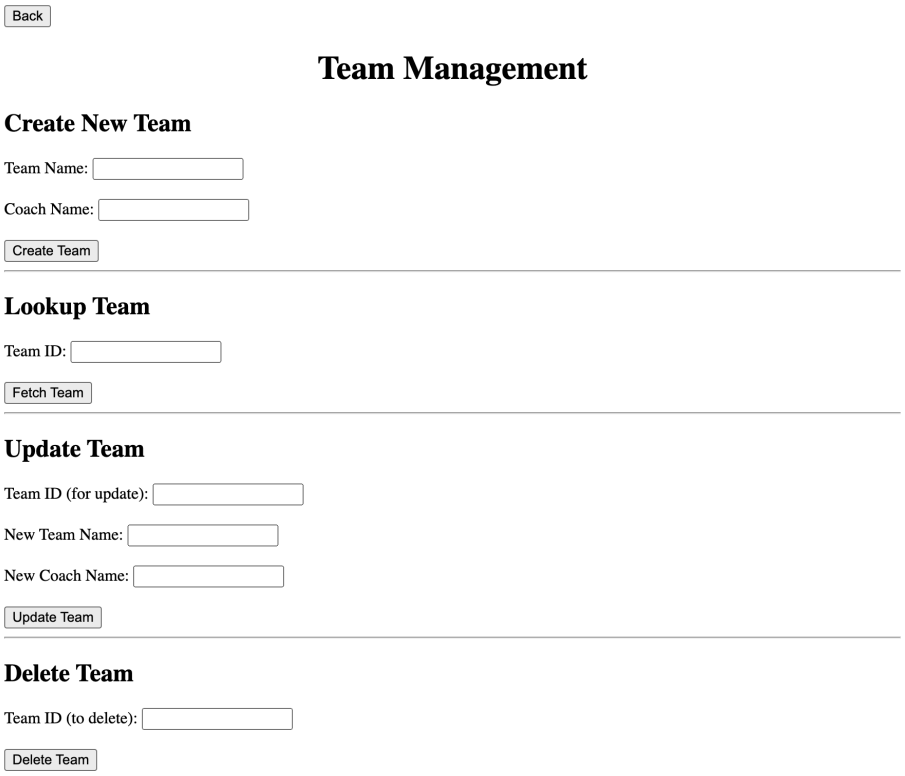
Once downloaded all the files and necessary applications, navigate to
http://localhost/demo/index.html

This is what our homepage will look like:

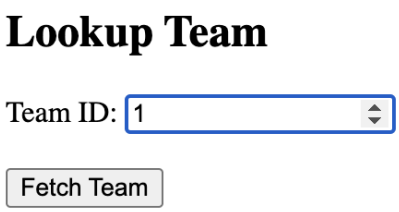


You can now access the website and execute CRUD functions on the 4 entities (Team, Player, Game, Player Stats)

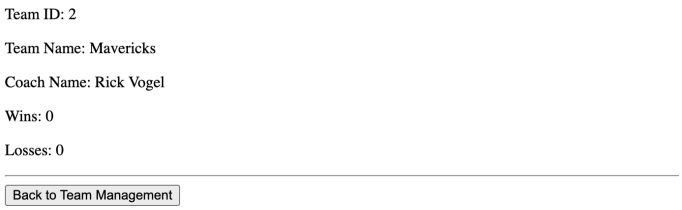
In order to execute a CRUD function on an entity, click on one of the buttons such as the Team one to see this web page:



You can press the back button to select a different entity to execute a CRUD function on, or scroll down to the function you wish to execute. This is an example to execute the read team function.
Enter a Team ID and click Fetch Team:



This is how the output will look like:



User Application Interface

The interface design of the HoopMetrics system is governed by principles of simplicity, efficiency, and responsiveness to accommodate various user needs. This approach ensures that the platform is accessible and easy to use for all stakeholders involved, including players, coaches, and administrators. The HoopMetrics system is structured around a series of HTML forms and interfaces that allow comprehensive interaction with the database, supporting CRUD operations across four main tables: Players, Teams, Games, and Player Stats. Each form is designed to handle specific data manipulations effectively.

CRUD Operations for Each Table:

- Creation Forms: These forms allow users to add new entries to each table. For example, new players can be registered, new teams can be formed, game details can be entered after matches, and player stats can be recorded post-game. Fields are tailored to capture all necessary data, such as player details, team information, game outcomes, and individual performance metrics.

- Read Interfaces: Dedicated forms enable users to fetch and view detailed information from each table based on specific criteria like player ID, team ID, or game ID. These interfaces are crucial for retrieving comprehensive data for analysis and decision-making.

- Update Forms: These interfaces facilitate modifications to existing records. Users can update player profiles, team records, game scores, and player statistics to reflect changes such as transfers, result updates, or corrections in player stats.

- Deletion Forms: To maintain data integrity and relevance, deletion forms allow users to remove outdated or incorrect entries from the database. This function is particularly critical for removing players who are no longer part of the league, disbanding teams, or correcting erroneously entered games.

- Security and Validation: Each form includes validation checks to ensure data integrity and prevent SQL injection or other forms of malicious input. Server-side validation is implemented in PHP, ensuring that only valid, sanitized data is processed and stored.

- User Feedback: After each operation—whether creating, reading, updating, or deleting—users receive immediate feedback through the interface. This feedback informs them of the success or failure of their actions and provides guidance on any errors encountered.

The user interface is designed to be accessible on various devices, including desktops, tablets, and smartphones, ensuring that users can interact with the system from anywhere. The layout is responsive, and controls are easily navigable, making it user-friendly for all users regardless of their technical expertise. The user application interface of HoopMetrics is a key component of the system, designed to facilitate easy and efficient interactions with the database. By supporting comprehensive CRUD operations across all main tables and ensuring data integrity through secure practices, the platform enhances the management and analysis of basketball league data at UTD.

Conclusions and Future Work

The HoopMetrics project successfully establishes a robust and efficient database system tailored to enhance the management and analysis of intramural basketball at the University of Texas at Dallas (UTD). By transitioning from traditional manual record-keeping to a specialized database solution, the project addresses significant gaps in data handling and analytics at the grassroots sports level. The implemented system not only facilitates real-time data updates and accessibility but also provides detailed analytics that can help in strategic decision-making and player development.

The user interface of the system is designed to be intuitive and accessible, ensuring that all stakeholders, including players, coaches, and sports administrators, can easily navigate and utilize the platform without extensive technical knowledge. The comprehensive implementation of CRUD operations across all main entities—Players, Teams, Games, and Player Stats—ensures that the system is versatile and fully functional, capable of supporting the dynamic nature of sports leagues.

While the current implementation of HoopMetrics serves the basic needs of the UTD intramural basketball community, there are several avenues for further enhancement and expansion:

- Advanced Analytics and Reporting: Future versions could include more sophisticated analytics features, such as predictive modeling for player performance and game outcomes, trend analysis over seasons, and customized reports for individual and team performance metrics.

- Integration with Other Systems: There is potential to integrate this database system with other campus systems, such as UTD's student information systems, to streamline operations and enhance user experience. Integration with fitness tracking devices and apps could also be explored to automatically update player stats based on real-time performance data.

- Mobile App Development: Developing a dedicated mobile app could enhance accessibility and user engagement. This app could provide notifications for game schedules, results, and individual performance updates, making the platform more interactive and user-friendly.

- User Customization Features: Allowing users to customize dashboards and control what data they see could greatly improve user satisfaction and engagement with the system. Personalized interfaces for players, coaches, and administrators could be developed to cater to the specific needs and preferences of each user group.

- Security Enhancements: As the system scales and handles more sensitive data, it will be crucial to continuously evaluate and enhance security measures. Implementing more advanced security protocols and regular security audits could ensure the protection of user data.

The HoopMetrics project marks a significant step forward in applying technology to enhance sports management and engagement at UTD. By leveraging modern database and interface design techniques, the project not only improves operational efficiency but also enriches the sports experience for the university community. Future enhancements and expansions promise to build on this foundation, offering even more sophisticated tools and features to support the UTD intramural basketball league.

References

"Connect to Mysql with PHP in XAMPP / Create a New Database." *YouTube*, YouTube, 11 Dec. 2013, www.youtube.com/watch?time_continue=1&v=ueWpNe0PG34&embeds_referring_euri=https%3A%2F%2Fwww.google.com%2Fsearch%3Fq%3Dhow%2Bto%2Bconnect%2Bphp%2Bto%2Bmysql%26sca_esv%3Dafdc9e5c89cffcda%26biw%3D1420%26bih%3D692%26tbm%3Dvid%26sxsrf%3DACQVn0-3vhGh&source_ve_path=Mjg2NjY&feature=emb_logo.

Appendix

Appendix is at this github link: https://github.com/anishdasti/HoopMetrics

Paths:

/doc
    Group17-Phase5-FinalReport.pdf
    Slides.pdf
/project
    Code
README