Bawlin' App Documentation

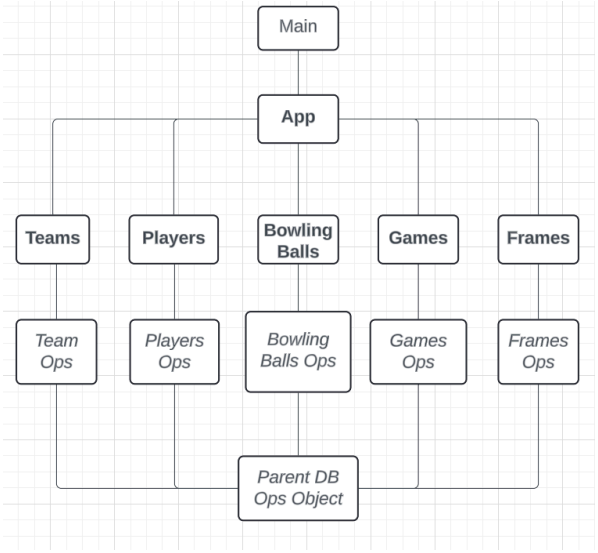
Business Problem: Performance statistics are vital for serious players, but can also act as guidance for more casual players looking to improve. Both bowling players and teams (regardless of skill level) would greatly benefit from having a simple and convenient method of keeping track of their bowling stats.

Executive Summary: The Bawlin' application provides bowling players and teams an easy and straightforward way of keeping track of their bowling stats. Teams and players also have the ability to view and modify their stats or personal information.

Technical Rundown:

→ Source Files

There are a total of 13 python source files and 7 SQL files. 5 of the SQL files solely contain procedures that are called by their respective python operations file. Their file names indicate which table the procedures are primarily focused. One other SQL file contains all database creation queries, and the final one is the database dump. Of the 13 python files, only 12 of them are fundamental to running the application as one of them is simply the main file. The image below shows all the connections between the python source files with one another. One connection is not shown, that being a connection between the "app.py" file and the "database_conn.py" file (Parent DB Object). A "DbOps" object is created within the "app.py" file in order to ensure that the database connection is closed upon exiting the application.



The "app.py" file creates the initial main menu GUI and is connected to 5 different other source files (main excluded). Each of these connected source files (teams.py, players.py, etc.) create a new GUI window that allows a user to interact with the database through various GUI widgets. The general structure of each of these files is:

initialization function, UI creation, and command functions for their respective widgets. Keep in mind that the UI creation portion is the only messy part of each of these files. These non-operation files connect to their respective operation files in order to be able to interact with the database by calling the stored procedures (located in the SQL files). The operations files inherit from the parent database operations object as they all need to connect to the database as well as get the results from the procedure they called. So in short, the structure is: “app.py” is the main menu, which opens a new window (teams.py, players.py, etc.), each of those windows uses their respective operations file to interact with the database, and each operation file inherits from the parent database object for 2 universal operations; connecting to the DB and getting procedure results.

→ Database Implementation/Design

There are 7 tables in total, 5 of which are the main tables and 2 are simply join tables; each table is normalized up to the third normal form. One of the join tables, players to games, contains an attribute besides the respective IDs which is the final score of a game; multiple players can participate in the same game. The teams table is connected one to many to the players table. Many players are connected to many bowling balls, as it is assumed that players can share a ball with each other. Many players are connected to many games as multiple players can participate in the same game. Finally, each game is connected to many frames (a round in a bowling game). Initially there was a login table, but that was removed as a more offline approach of the app was favored, and it was simply redundant. However, if one were to need an online version of the database, then a login table could easily be incorporated into the database’s design. Also, the final score was originally part of the games table only, but this prevented multiple players from being part of the same game (which was the original intention). This was promptly moved to the player and game join table to fix this relationship. Out of all the numerical primary key IDs, only game ID is supposed to be shown to the user. This was primarily done for the sake of elegance, and game ID was left viewable as it made the most logical sense to leave it displayed. If one wishes to see every exact attribute associated with each table, refer to the entity-relationship or functional dependency diagram.

→ Using the App

The general usage structure of the app is top to down when starting from the main menu. Start at teams, then players, then bowling balls, then games, and finally frames. However, it is more appropriate to finish on games rather than frames as one gets their final score after playing through all the frames in a bowling game. So the actual order would be: Teams, Players, Bowling Balls, Frames, and Games. Bowling balls could be ignored as an option since it doesn’t affect much throughout the rest of

the database. Regardless, simply follow the aforementioned order, and everything else should be intuitive to use/go through.

→ Issue(s)

From my testing, there are no major issues with the application. However, there is a lack of error checking throughout the application, so use common sense to avoid causing problems. Generally speaking, using the app incorrectly or making errors won't cause critical problems or a crash (at least from my testing). It will simply make the database being interacted with rather messy, and that's all.

→ Continuing Development

If one were to continue developing this application, the first thing to implement would be proper user error checking to ensure a clean database. Also, the only truly important files are the SQL files containing the stored procedures and database creation. All other python source files are mainly for directly interacting with the database and creating a GUI. The GUI could be made in any other language or package. As long as one can connect to the database and add new SQL procedures, the frontend and database interaction can be anything one prefers or needs.

Contributions: All work was done by me. Any and all references used will be in the README file.