Kea Braekman, Thomas Ochsner, Brett Derham, Riley Persily

CMSI 401

11/06/19

Software Design Document

6.1 Introduction Section

This software design document provides the details of the planned design for Ballpark Bookie which provides an application to provide users with general and specific predictions about Major League Baseball games. It will have a simple design that will prominently feature two ways in which a user can interact with the application.

The application’s final output will be the predicted winner of an upcoming Major League Baseball game. The user will be able to search through a list of all the MLB teams and then select two to go head to head in our regression algorithm to determine a winner. The application will provide this information through a series of steps performed in the backend. Also, on another page, there will be a big predictions section that allows for the user to see the likelihood that a selected team will make it to the playoffs or world series.

On the left of both of these sections there will be a list of all 30 MLB teams with links to their schedules.

6.1.1 System Objectives Section

The backend will be doing all the work. The statistics will be pulled from an API that provides access to such data. The predicted winners will come from an algorithm, which is unique to Ballpark Bookie, that takes into account the enormous amount of data present in MLB statistics and weigh the probable influence of the data on the outcome of each individual game.

Our goals for this project is to compile as much data as possible into this algorithm in order to make it as accurate as possible, and if that is accomplished ahead of schedule we will be able to give it a good looking front end.

Ballpark Bookie will appeal to a range of potential users, ranging from novice betters to avid sports fans, that want an easy-to-use, accurate application that can predict the outcomes of games. People may want to know the likely outcome of a game based on their interest in the sport, investments in betting, participation in fantasy baseball, or desire to attend a game where their favorite team wins.

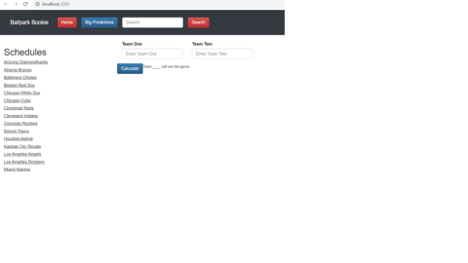
6.1.2 Hardware, Software, and Human Interfaces Section

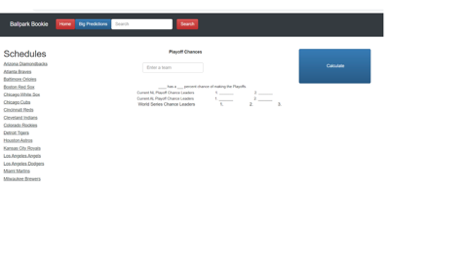
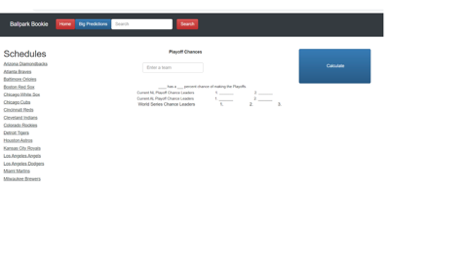
The hardware interaction comes from the user interacting with a computer, and the software interaction starts with the user interacting with a web browser of the choice (Google Chrome recommended).

The Graphical User Interface (GUI) will contain links on the left side of the page and dropdown lists and buttons on the right side (for both the home and big predictions page).

The user should only need their mouse to navigate through the application.

These are rough approximations on what the front end of our application will look like based on the progress that we are at now. The buttons, lists, and links will get sharper but the basic idea for the design will be the same.





6.2 Architectural Design Section

For the Architectural Design Section of our project, we have decided to separate four elements that would constitute Ballpark Bookie : Front End, Back End, Algorithm, and Hosting.

Front End : The front end portion of the project is pretty simple. Since the focus of our project is to predict baseball games, the front end is not the priority. We have decided to use a simple ReactJS front end framework with two prediction pages with clickable MLB links for the user to see different baseball stats and news. The main page will contain upcoming MLB games and our prediction (as a win percentage probability). The prediction scores will be populated via our algorithm that we will describe later.

Back End : The back end is pretty straightforward. We get all the data needed for scheduling as well as MLB statistics with the API “MySportsFeeds”. The data collected is in CSV format and parsed via Python using pandas. The data is then used for the development of our algorithm as well as upcoming games etc…

Algorithm : Perhaps the most complex part of the project.

Inputs : MySportsFeeds API MLB stats, team A, and team B

Output : Probability of home team winning (float)

Tools : We will use Python, pandas, numpy, and Scikit learn.

Process : We will develop a machine learning ranking algorithm. We will start by making a list of all baseball games given a past season, store a +1 value if the home team wins, and -1 otherwise in a separate list. Then, we will collect all the seasonal statistics of each team for the previous season in a matrix. Once we have collected all the data in CSV format, we will normalize the data (convert to standard normal form), and subtract both matrices. This new matrix is then going to be compared with the win/loss result array. We will then use numpy logistic regressions that will learn weight coefficient values that correlate our matrix with the win/loss array. If we repeat this enough times, the weights and coefficients could potentially be very accurate. Once we have these coefficients, we can use a numpy “predict” function that will take a matrix and determine the most likely outcome (win/loss).

Hosting : The last portion of the project. Once everything is functional and we have connected the algorithm results to the front end, we can deploy Ballpark Bookie. We still need to research what our options are, but it seems that AWS is the way to go. We will deploy using tools like Amplify and maybe purchase a domain name.

6.2.1 Major Software Components Section

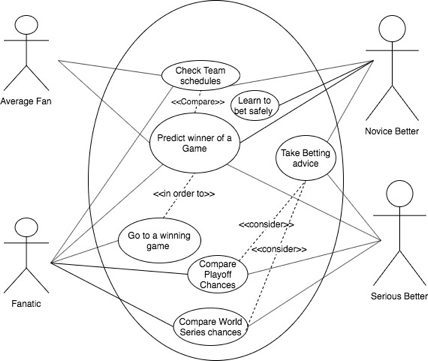
Ballpark Bookie shall not require any special computing hardware to operate. Ballpark Bookie shall be able be able to execute using any standard web browser.

6.2.2 Major Software Interactions Section

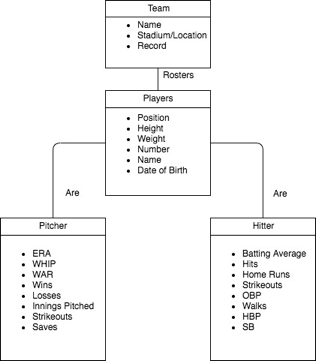
Ballpark Bookie relies heavily on the MySportsFeed API. All of the data used in our algorithm gets called each day from their database. We have written many python functions that gather all sorts of data through API calls that will later on be used by the algorithm. Ballpark Bookie was also created in and editor called Visual Studio Code. Other than these two Ballpark Bookie does not rely on other software interactions.

6.2.3 Architectural Design Diagrams Section

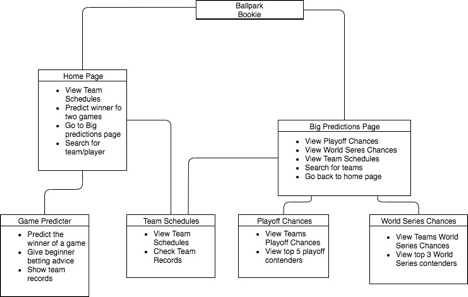
Use Case Diagram



Top-level Class Diagram



Deployment Design



6.3 CSC (Computer Software Components) and CSU (Computer Software Unit) Descriptions Section

6.3.1 Detailed Class Description Section

I will limit this section to code that has been written

basicAPIcall → This will access the desired API call.

choose\_team → Creates a numpy array for the desired team.

normalize → This normalizes all the data so that it is easier to compare.

subtractArrays → Subtracts numpy arrays of two opposing team's data.

createWinList → This goes through every game in the season and creates a numpy array based on who wins. If the home team wins there is a ‘1’ and if the away team wins then there is a ‘-1’.

Team One Form → Input for user to enter first team for comparison

Team Two Form → Input for user to enter first team for comparison

Calculate Prediction Button→ Button for user to get result of game prediction

Playoff Chances Form→ Input for user to enter a team and see their playoff chances

Calculate Playoff Chances Button→ Button for user to view results for Playoff Chances

Navbar→ Menu bar to go back and forth between pages

6.3.2 Detailed Interface Descriptions Section

The process of control transfer in the application starts with the JavaScript being run in the web app which waits for user input, then when that is detected it asks the Python Script for the data requested by the user so control is transferred to the Python script. Once the Python is in control, it contacts the API service through basic calls to sockets and requests data to be delivered to it in the form of CSV which it then takes and manipulates into calculated data containing the predictions requested by the user, this resulting data is stored in an array in the Python and is then returned back to the JavaScript call that asked it for the data, transferring control back to it, and the JavaScript takes that data and populates the sections of the web page that are meant to display the data, returning control to the user.

API Service >>> Python Script

Python script uses a function to make an API call that contacts the XXXXX server and requests data, which is then provided to the script in the form of a CSV file.

Python Script >>> Web-App

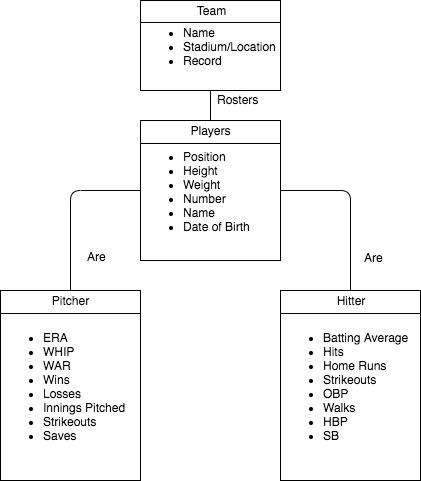
The data that has been created by the Python Script by running the collected information from the API Service will be stored in simple arrays and then pulled from those arrays by calls in JavaScript classes in the web-app that then use that information to populate the various areas on the page that are meant to display the results and relative information to users.

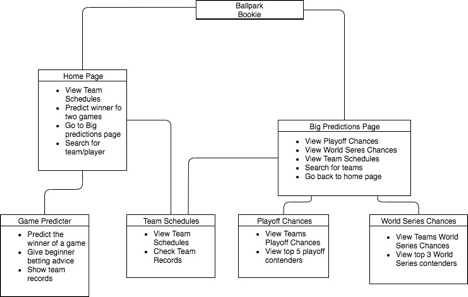
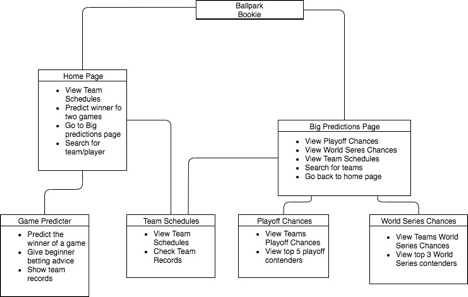
6.3.3 Detailed Data Structure Descriptions

Our information comes from an API and we access it from CSV files that are created and overwritten from our code. The selected information is then put into a numpy array so that we have our specific information and can also perform calculations with it easily.

We will normalize all the data before it goes through our regression algorithm. It will learn by guessing the outcome and comparing it to the real world results which will also be in a numpy array.

6.3.4 Detailed Design Diagrams Section





6.4 Database Design and Description Section

Not applicable because we will use csv files for information that get filled from API calls. Will not use a database.