FootyBettor

Sports Betting Platform (SBP)

Software Design Document

Prepared by:
Zahra Cheeseman
Esa Chen
Junhao Qu
Shamdeed Kabir

Release Date:
Monday 6th March, 2023

Contents

Executive Summary

Sports Betting Platform (SBP) is a unique form of entertainment that combines the excitement of sports with the thrill of gambling. SBP is intended to exist in similar markets as other sports betting websites, but provides more data-backed informed odd calculations, and uncluttered, understandable, and easy-to-use operation pages. SBP softwares will be distributed as standalone applications for use on end-user hardware without network connectivity.

This document provides technical information regarding the software design of SBP.

Document Versioning

Date	Owner	Comment
03/03/2023	Zahra Cheeseman, Shamdeed Kabir, Esa Chen, Junhao Qu	Executive Summary and Project Description
03/06/2023	Zahra Cheeseman, Shamdeed Kabir, Esa Chen, Junhao Qu	Feature Matrix and Feature Discussion
03/06/2023	Shamdeed Kabir	High-level System Architecture
03/06/2023	Zahra Cheeseman and Esa Chen	Major Components
03/06/2023	Zahra Cheeseman and Esa Chen	Detail Design
03/06/2023	Shamdeed Kabir and Junhao Qu	UML Diagrams
03/06/2023	Zahra Cheeseman and Esa Chen	Data Format
03/12/2023	Zahra Cheeseman, Shamdeed Kabir, Esa Chen, Junhao Qu	Update Detail Design, UML Design and Data Format

Project Description

FootyBettor is looking to enter the sports gambling market. This market is saturated with platforms with clunky and complex operation pages, limited data availability, and implicit probability distribution. Sports Betting Platform (SBP) is FootBettor's creation which provides a solution to the problems which traditional betting platforms present which create difficulty in users being able to make informed decisions, place winning bets and succeed in the highly competitive sports betting industry. SBP allows for impressive data visualization of previous season team statistics and a smooth user experience for easily betting on the result of matches between sports teams. Users can create an account in which they navigate through the program, and view a balance which they can add to and which gets updated in relation to the return on their bets. SBP generates precise odds which are informed by a multitude of statistics used in calculations which optimize accuracy.

SBP is highly extendable and thus would have the potential to be able to support different applications such as a variation in the sport selected and type of bet available.

Additionally, SBP can support a graphical user interface, however, it does not necessarily require one to interact with the program. The program may be run completely via command line interface support. Currently, there does not exist a consummate sports gambling program which is for low-power and low-capability embedded platforms. SBP provides a resolution which supports such devices plus ones which have sufficient requirements to support a GUI, allowing for an increased potential pool of devices.

Features

The feature matrix enumerates the technical features required to support each of the business requirements. The discussion section provides details regarding the constraints and functionality of the feature. The ids are used for traceability. Features that can be removed should strike-through the feature id and have a comment added to identify why this feature can be removed without impacting the BRD requested functionality.

Priority Codes:

H - High, a must-have feature for the product to be viable and must be present for launch
M - Medium, a strongly desirable feature but product could launch without
L - Low, a feature that could be dropped if needed

Feature Matrix

ID	Pri	Feature Name	Comment	BRD ID
P.1	H	Language	Implementation of JAVA	s.1 , s.3
I.1	H	CLI		ux.1 , e.2
C.1	H	System Loop		e.1 , e.3
C.2	H	Save Data	Store user profile	e.4 , e.5 , e.10
C.3	H	Actions		e.5 , e.6 , e.7 , e.8
C.4	M	Represent Data		e.6
T.1	H	Error Handling		ux.3 , e.1

Feature Discussion

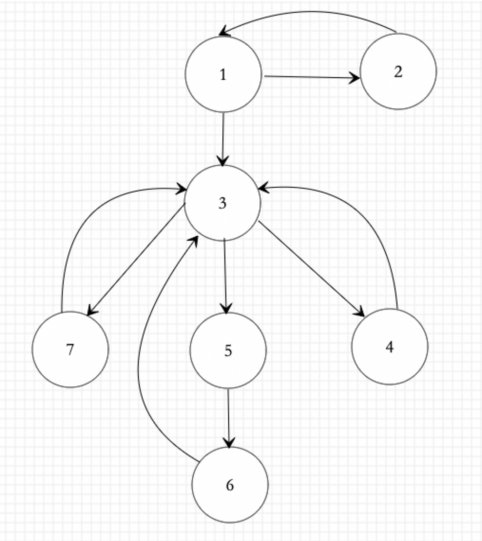
P.1 - Language

There are many possibilities for the choice of programming language that SBP will utilise. Java provides the best fit for the capabilities that SBP must support. Java has cross-platform support, includes capabilities for making executable bundles, has consistent GUI behaviour across platforms, and also supports CLI interaction.

I.1 - CLI

SBP provides a command line interface for interacting with the software in a command line terminal, thus it must have a fairly simple user interface pattern that is well suited for CLI operation.

C.1 - System Loop



All SBP will use the same basic loop system.

1. User log-in
2. User sign up and return to sign in
3. Get the user's choice
4. Show previous session stats and return to 3
5. Bet on a game
6. Show the betting result for the user and return to 3
7. Show balance and return to 3

C.2 - Save Data

Save Data means to store the user's profile including username, password, and balance. Each user profile is stored in a text file named by the username. A folder is built to store all the text files of users' accounts. SBP provides options for logging in and signing up for an account. To log in to SBP, the user can enter the username and password, then the system would check whether there is a text file with a corresponding filename to the username. If the file exists, the system would read that text file to check with the password and record the user's account balance. For the option of signing up for an account, the platform asks for the user's input for username and password. The text file for the new account would be created and stored in the folder, which allows the user to log in.

C.3 - Actions

Actions means all of the actions the platform will perform when users make a choice. The platform provides a menu that contains all of the actions for users to choose from. These actions can store user data, represent data, and track balance.

C.4 - Represent Data

An integral part to SBP is that previous team statistics can be viewed by the user, in order for them to make an informed decision on their bet. They can view both complete previous season statistics and data specific to the teams that they select.

T.1 - Error Handling

If the system reaches an unsafe state or the provided AL/EL has system stopping defects during parsing or execution, error information should be provided to inform the end user that an error has occurred. The user should then be presented with an option to have SBP produce a more detailed error report that could be sent back to the game developer as part of a game bug report.

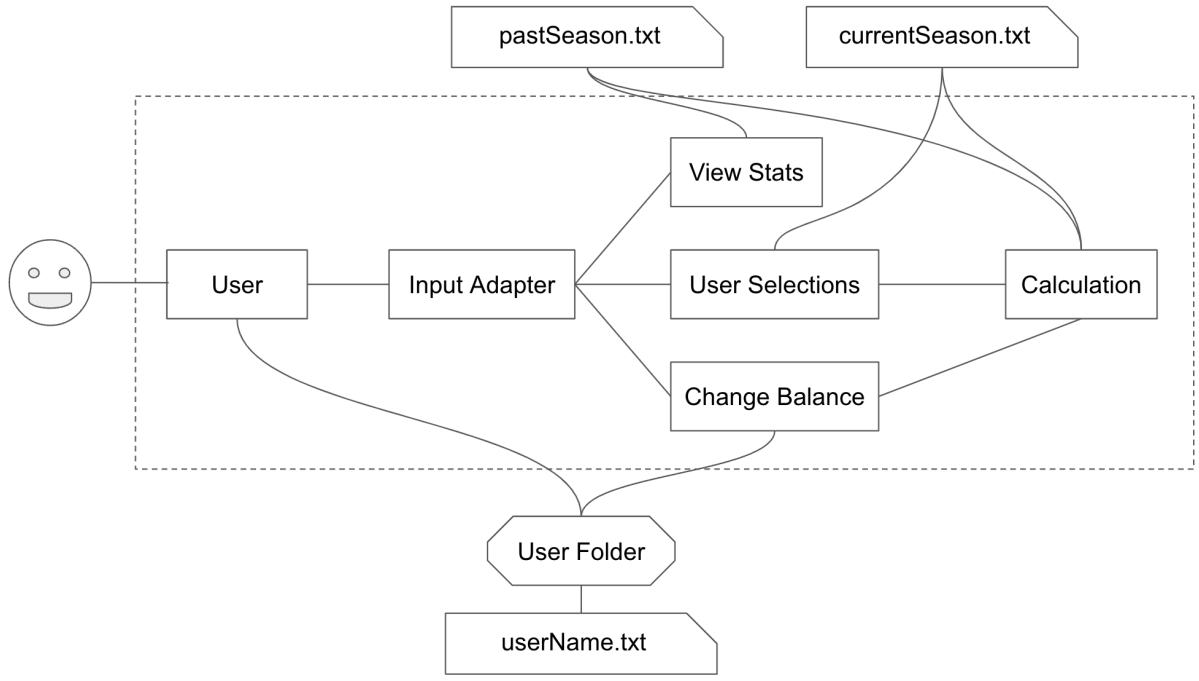
System Design

This section describes the system design in detail. An overview of the major system components and their roles will be provided, followed by more detailed discussion of components. Component discussion will reference the technical features the component helps satisfy.

Note: Design co-evolves with implementation. It is important to start with some sense of components and their interactions, but design docs should be updated during implementation to capture the actual system as-built.

Architecture Overview

High-level System Architecture



High-level Architecture Diagram

Major Components

InputAdapter

I.1, C.1

InputAdapter is responsible for managing the input choice of the user and then determining which actions to run in response to that.

User

C.2
This component is responsible for storing user information when users sign up and checking the user's password when they log in.

User selections

C.3
This component controls all user selections- including the teams which they select, their predicted result, and their wager.

Change/Update balance

C.2, C.3
The Change/Update balance component is responsible for the functionality of displaying the current balance and adding money to the current balance. After the change is made by the user, the system should update the balance stored in the user's account text file in the user folder.

Calculation

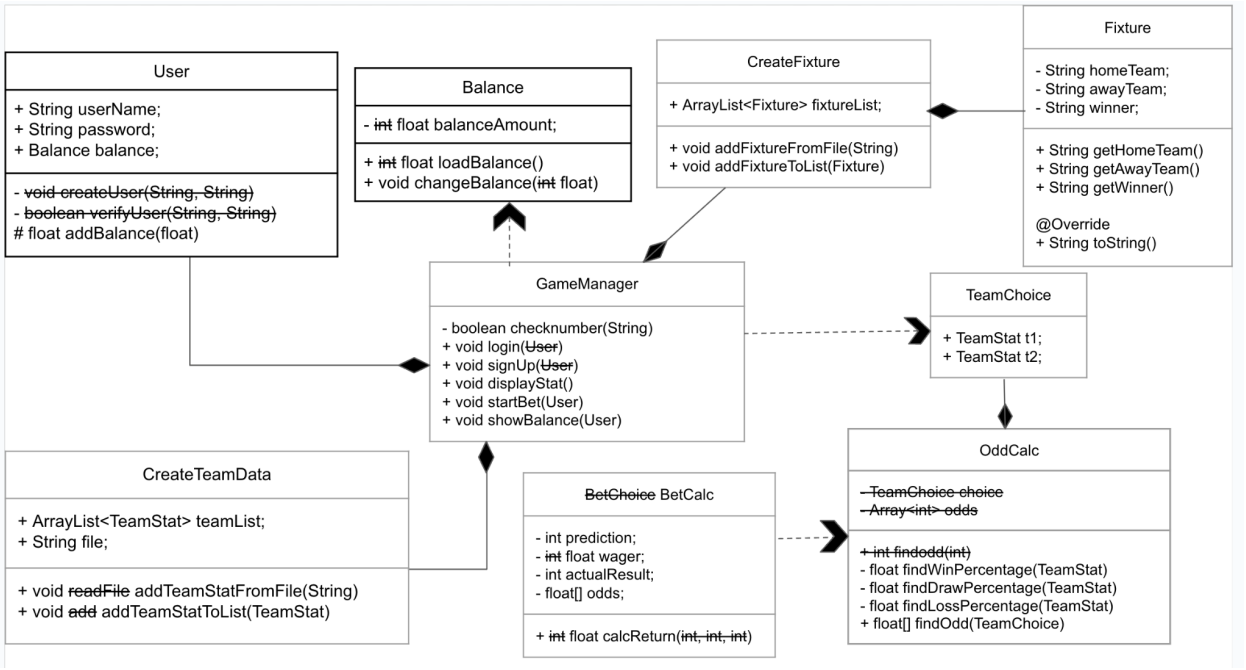
C.3
The calculation component manages all calculations- including taking in data in order to generate odds and returns.

View Stats

C.4
Controls the display of the statistics from the previous season's team data.

Detail Design

InputAdapter Component



Class Diagram

GameManager

Class which has only methods which are used in Main to control the system state/ user pathway. It includes the login(), signUp(), displayStat(), startBet(User), and showBalance(User) methods. It interacts with all of the other classes. The method checknumber(String) supports the function of checking whether the user input is a positive number (including decimals) or not.

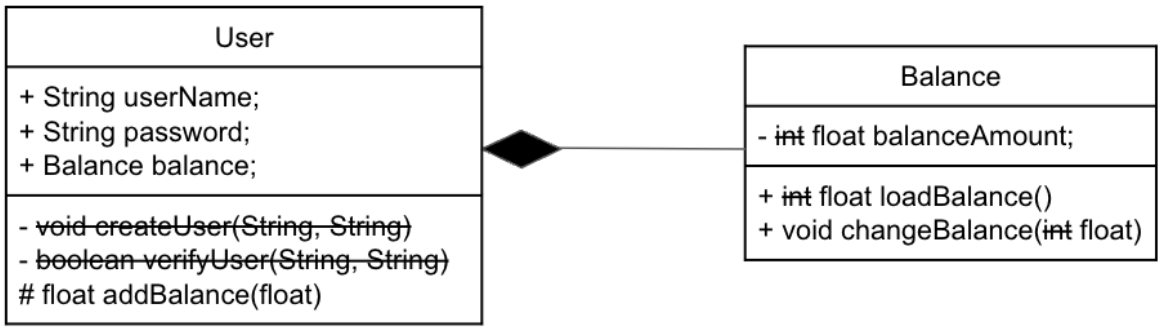
CreateFixture

The CreateFixture class reads the text file (pySoccer.txt) containing current season data of all the matched games line by line by the method addFixtureFromFile(String) which requires an input of filename. As each line in the file represents a game, the class creates an object Fixture for each line and stores all the Fixture objects in an ArrayList<Fixture> fixtureList by using the addFixtureToList(Fixture) method which adds the input Fixture object to the ArrayList fixtureList.

Fixture

The Fixture class stores data of the user-selected game. The home team name, away team name, and the winner of the matched game are stored as private string variables in this class. The methods getHomeTeam(), getAwayTeam(), and getWinner() are public getters for those three variables. The method toString() supports the functionality of printing out all the information of the selected game in one single string.

User Component



Class Diagram

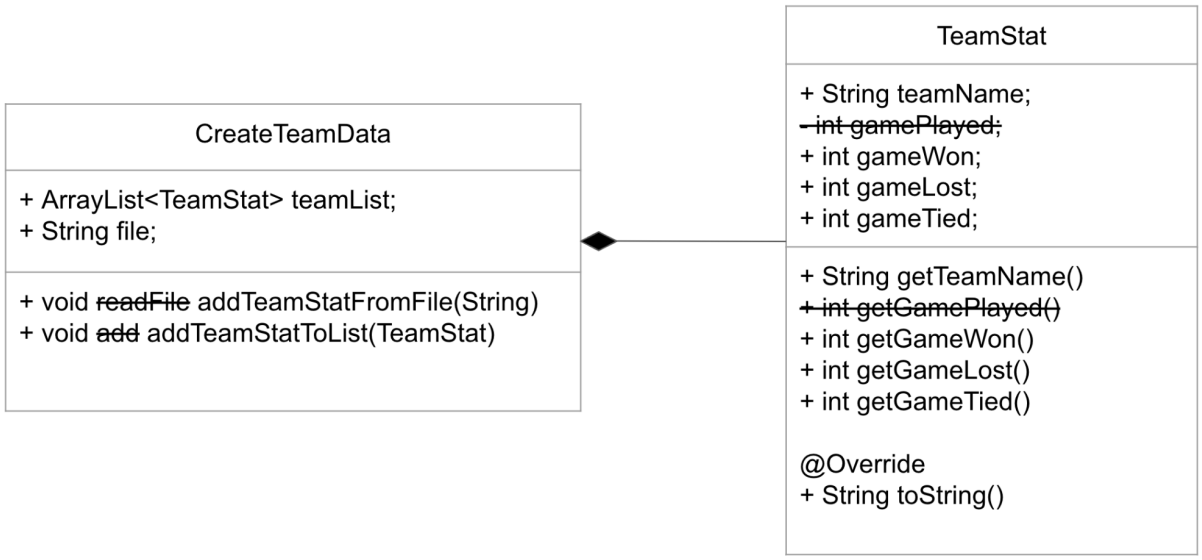
User

Creates a user object which defines a user by their username, password, and their balance. The userName and password are stored as string variables, and the balance is an instance of the Balance class. The addBalance(float) method is used to add money to the balanceAmount of the balance object stored in the User class by calling the changeBalance(float) method in the Balance class, and then this method rewrites the user's account text file in the user folder to update the amount of balance.

Balance

The Balance class is responsible for the user's balance management. It stores the current balance in the user account as a float variable. The loadBalance() method is the public getter method for obtaining the value of balance. The changeBalance(float) method supports the function of changing the balance by adding the input float number.

View Stats Component



Class Diagram

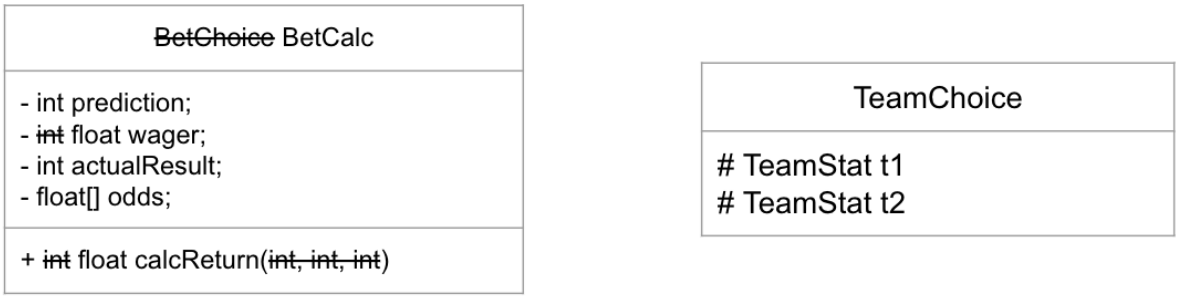
CreateTeamData

The CreateTeamData class reads the text file (pastSeason.txt) containing data of all the teams line by line by the method addTeamStatFromFile(String) which requires an input of filename. As each line in the file represents a team, the class creates an object TeamStat for each line and stores all the TeamStat objects in an ArrayList<TeamStat> teamList by using the addTeamStatToList(TeamStat) method which adds the input TeamStat object to the ArrayList teamList.

TeamStat

The TeamStat class manages the team data by storing the information of a team such as teamName, gameWon, gameLost and gameTied. The getTeamName(), getGameTied(), getGameWon(), and getGameLost() methods are public getter methods that support the function of returning the private variables stored in the TeamStat class- including String teamName, int gamePlayed, int gameWon, and int gameLost. The toString method is also overridden, allowing the display of all of these variables in an easy to read format.

User selections Component



Class Diagram

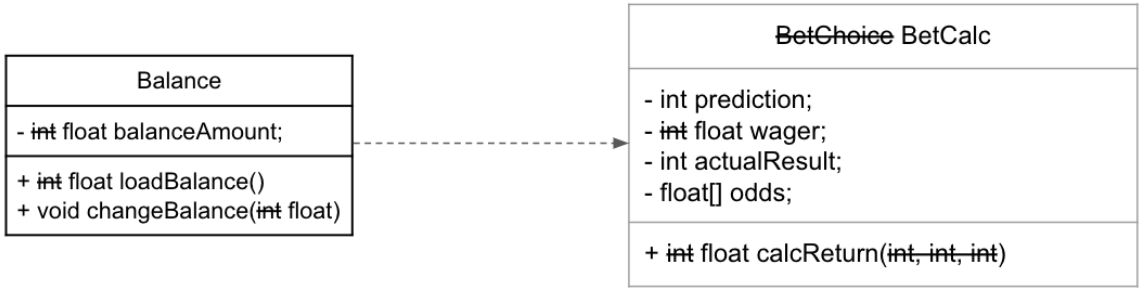
BetCalc

The BetCalc class takes in the user's prediction, their wager, the actualResult and the odds of each outcome to calculate the return for the user in the method calcReturn().

TeamChoice

The TeamChoice class stores the user's two selected teams by creating two TeamStat objects, t1 and t2. These two objects can be accessed directly without getter methods since they are public.

Change/Update balance Component



Class Diagram

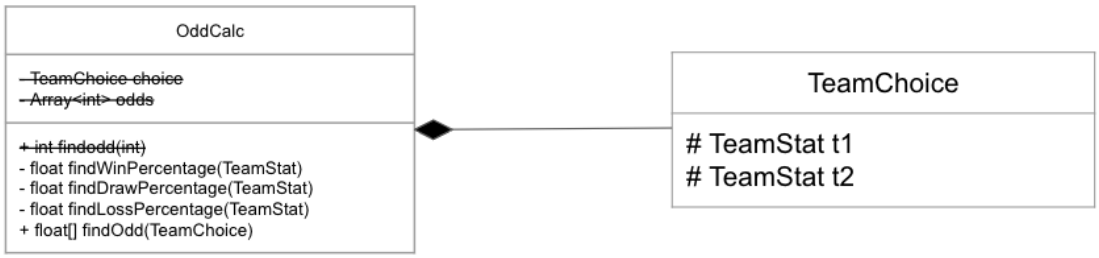
User

Creates a user object which defines a user by their username, password, and their balance. The userName and password are stored as string variables, and the balance is an instance of the Balance class. The addBalance(float) method is used to add money to the balanceAmount of the balance object stored in the User class by calling the changeBalance(float) method in the Balance class, and then this method rewrites the user's account text file in the user folder to update the amount of balance.

Balance

The Balance class is responsible for the user's balance management. It stores the current balance in the user account as a float variable. The loadBalance() method is the public getter method for obtaining the value of balance. The changeBalance(float) method supports the function of changing the balance by adding the input float number.

Calculation Component



Class Diagram

oddCalc

This class contains the algorithm which calculates the odds for the three situations of game result between the two teams that the user has selected: one team wins, the other team wins or there is a draw. These odds are stored by and can be accessed from an Array and has the method findOdd(TeamChoice) which takes in their choice of teams and produces the odds. The findOdd(TeamChoice) method calls on the private methods: findDrawPercentage(TeamStat), findWinPercentage(TeamStat) and findLossPercentage(TeamStat), in order to calculate the odds.

TeamChoice

The TeamChoice class stores the user's two selected teams by creating two TeamStat objects, t1 and t2. These two objects can be accessed directly without getter methods since they are public.

TeamStat

The TeamStat class manages the team data by storing the information of a team such as teamName, gameWon, gameLost and gameTied. The getTeamName(), getGameTied(), getGameWon(), and getGameLost() methods are public getter methods that support the function of returning the private variables stored in the TeamStat class- including String teamName, int gamePlayed, int gameWon, and int gameLost. The toString method is also overridden, allowing the display of all of these variables in an easy to read format.

Data Format

For the CLI, SBP must store 3 categories of data in text files: users, past season data, and current season data.

User

There will be a folder, userFolder, which will store a .txt file for every user.
The .txt file will contain the user's username, password, and balance.

First line: User's username
Second line: User's password
Third line: User's balance

Past Season Data

The Past Season Data is stored in a .txt file called pastSeason.txt. It records all the results of games and contains team data of team name, amount of games played, amount of games won, and amount of games lost in the 19/20 premier league season. Each line of the file represents a team. It is stored in the format:

Team_name, Games_won, Games_lost, Games_tied

Current Season Data

The Current Season Data is stored in a .txt file called ~~currentSeason.txt~~ pySoccer.txt. It contains the results of every game played between two teams in the 20/21 premier league season. It is stored in the format:

LineNumber, Season, HomeTeam, AwayTeam, FTR