Grade 12
Key 3

PAT Design Document

Five a Side Football

Nikhar Ramlakhan
TRINITYHOUSE HIGH SCHOOL 2020

Table of Contents

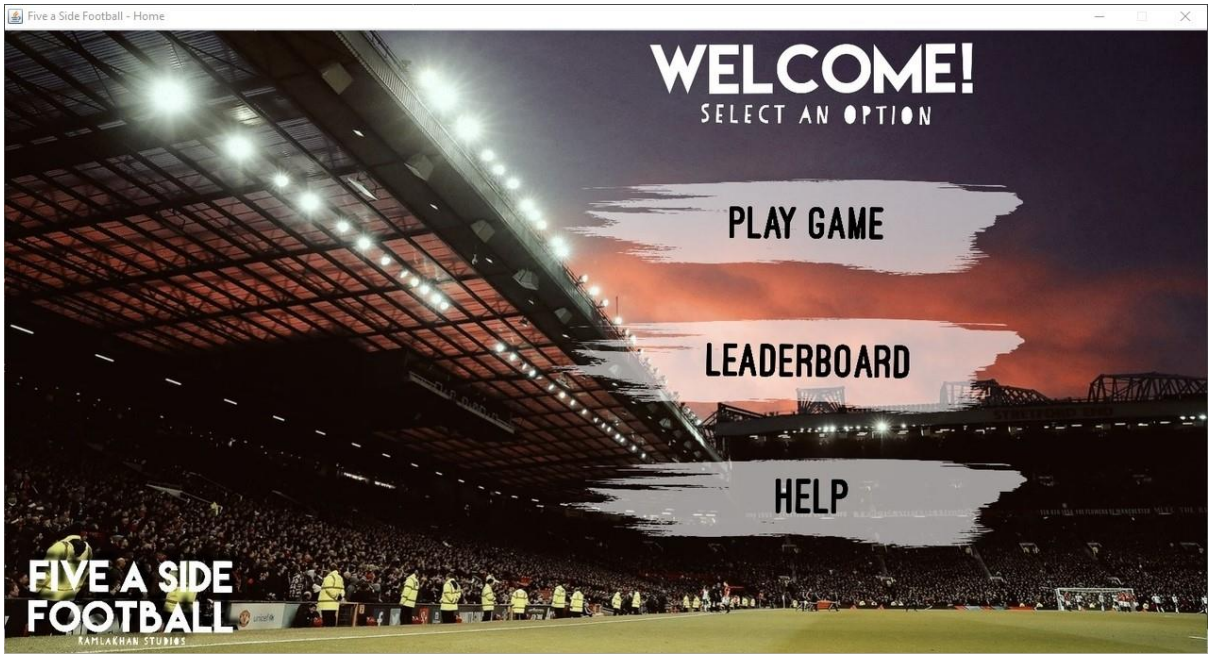
UI Design

Start

This is the first screen of the program and the main class of the program that starts the program.

Component	Action Elements and Data Validation
'Start Game' *Button*	Clicked: Goes to the **Home** screen.
'QUIT' *Button*	Clicked: **Exit** the program.

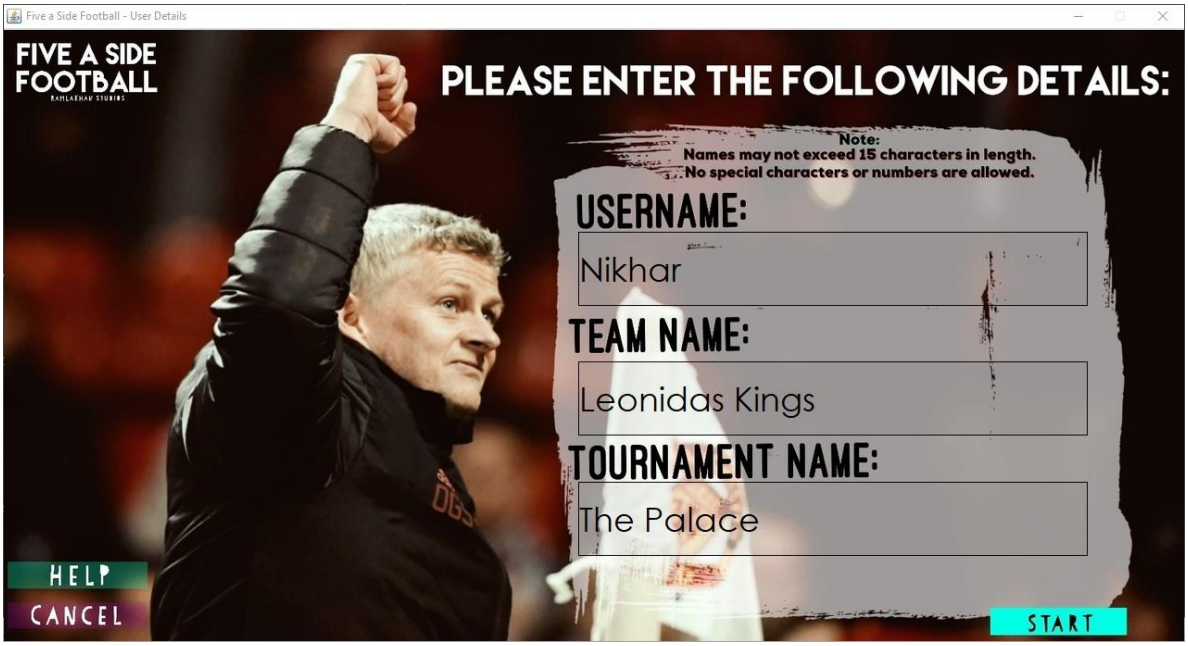
Home



This is the home screen of the program where the user is able to start the game, view the leaderboard or access the help menu.

Component	Action Elements and Data Validation
'Play Game' *Button*	Clicked: Goes to the **User Details** screen.
'Leaderboard' *Button*	Clicked: Goes to the **Leaderboard** screen.
'Help' *Button*	Clicked: **Opens the Help** screen as an external screen.

UserDetails

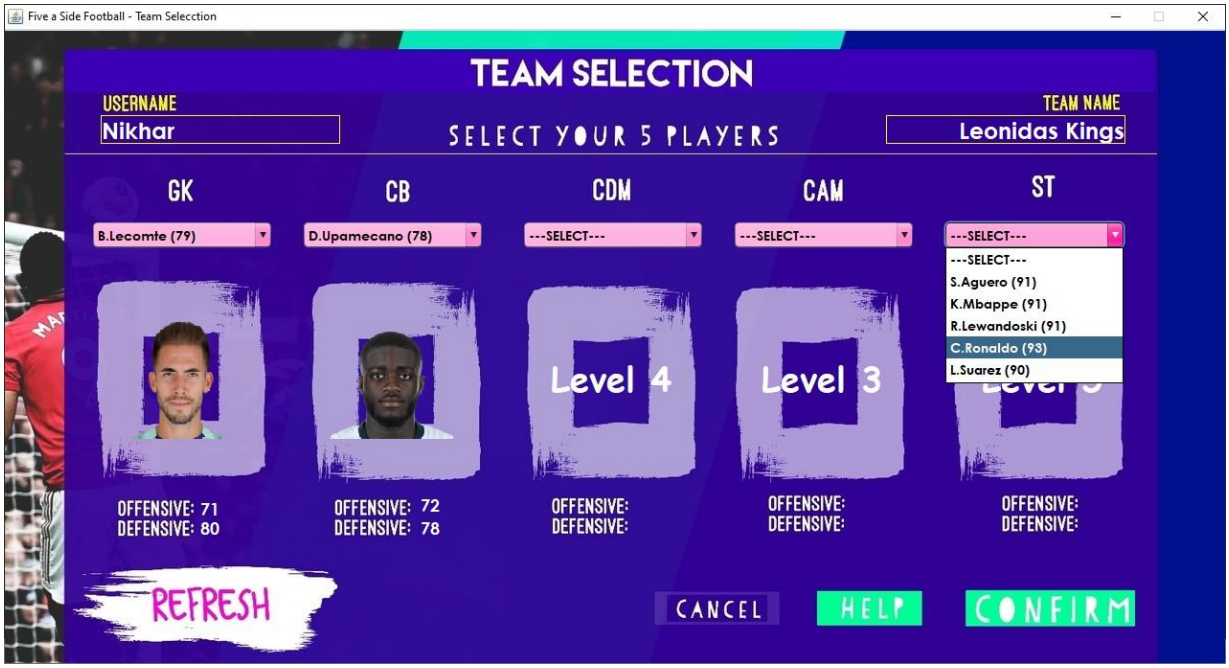


This is the screen where the program will capture the user's details and team and tournaments name.

Component	Action Elements and Data Validation
'Help' *Button*	Clicked: **Opens the Help** screen as an external screen.
'Cancel' *Button*	Clicked: Goes back to the **Home** screen.
'Username' *Text field*	User **inputs** their **Username**. **Validation**: - Presence Check (Data must be input) - Is Letter Check (No numbers or special characters allowed) - Length Check (Not more than 15 characters)
'Team Name' *Text field*	User **inputs** their **Team Name**. **Validation**: - Presence Check (Data must be input) - Is Letter Check (No numbers or special characters allowed) - Length Check (Not more than 15 characters)
'Tournament Name' *Text field*	User **inputs** their **Tournament Name**. **Validation**: - Presence Check (Data must be input)

	- Is Letter Check (No numbers or special characters allowed) - Length Check (Not more than 15 characters)
'Start' *Button*	Clicked: **Performs validation** as described above and proceeds to the **Team Selection** screen if data is valid or displays an error message if data is invalid.

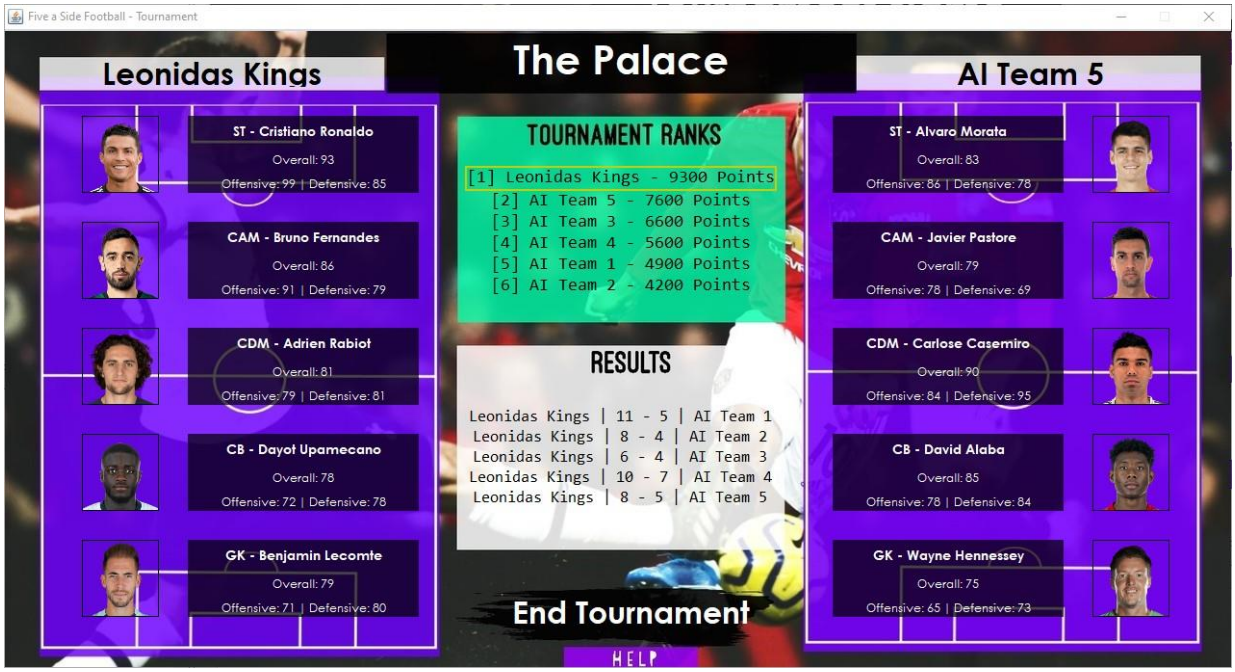
TeamSelection



This is the screen where the user will select their team.

Component	Action Elements and Data Validation
Select Players *Combo Boxes*	5 Combo Boxes are used for each position. When an item in the combo box has been **selected**, the face image, offensive and defensive stat of the player must be **updated**. **Validation:** - Presence Check (a player selected in each position) NOTE: The **players are stored in the Players Table of the Database** and are accessed through the PlayersArray class which is used in the TeamSelection class.
'Refresh' *Button*	Clicked: **Reopens** the **Team Selection** screen and displays new possible players.
'Cancel' *Button*	Clicked: Goes back to the **Home** screen.
'Help' *Button*	Clicked: **Opens the Help** screen as an external screen.
'Confirm' *Button*	Clicked: **Performs the validation** required by the combo boxes, **adds the team to the Teams Table of the Database** and proceeds to the **Tournament** screen if data is valid. If data is invalid, an error message appears.

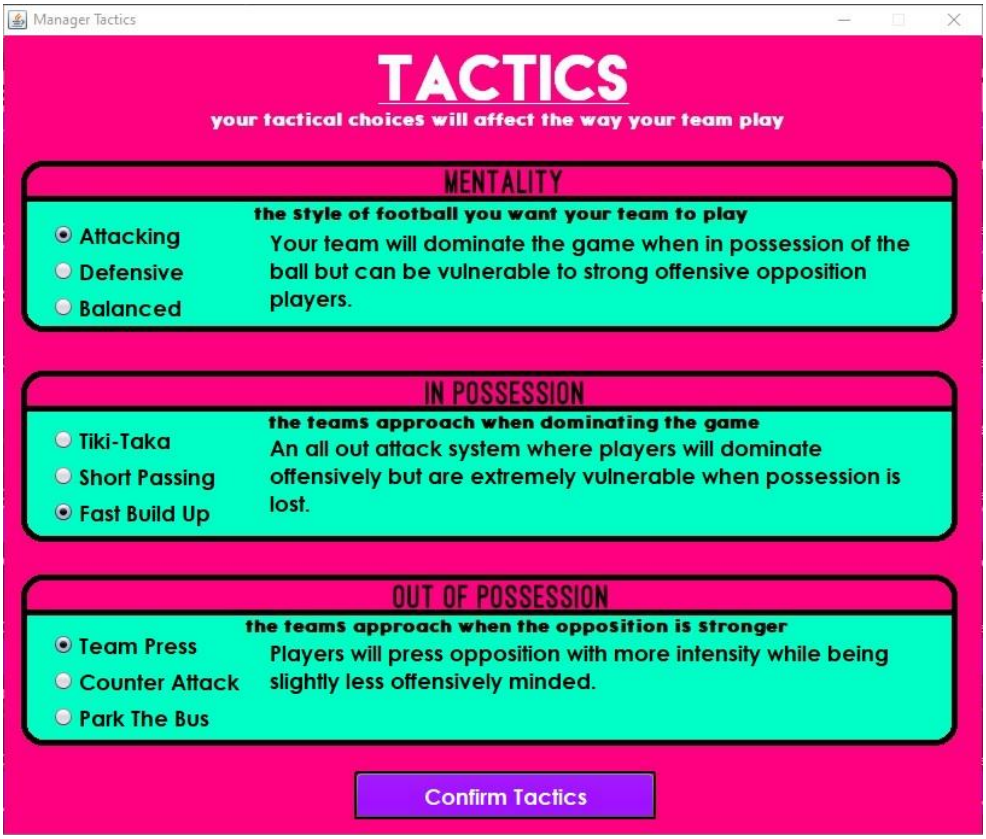
Tournament



This is the screen where the user will face off against the 5 other teams.

Component	Action Elements and Data Validation
'Set Tactics' *Button*	This button is visible before a match takes place in the same location as the 'End Tournament' button. Clicked: **Opens the Tactics screen** and temporarily hides.
'Play Match' *Button*	This button is only visible after the user has confirmed their tactics. Clicked: **Plays the match** by calling the playMatchWeek method of the **League** class.
'End Tournament' *Button*	This button text will read 'End Tournament' after all 5 matches have been played. Clicked: Proceeds to the **Results** screen.
'Help' *Button*	Clicked: **Opens the Help** screen as an external screen.

Tactics



This is the screen where the user will select their tactics before each match. This displayed on top of the Tournament screen.

Component	Action Elements and Data Validation
Attacking *Radio Button*	Belongs to the button group: **Mentality**. Selected: **Sets Mentality enum** to '**Attacking**'.
Defensive *Radio Button*	Belongs to the button group: **Mentality**. Selected: **Sets Mentality enum** to '**Defensive**'.
Balanced *Radio Button*	Belongs to the button group: **Mentality**. Selected: **Sets Mentality enum** to '**Balanced**'.
Tiki Taka *Radio Button*	Belongs to the button group: **InPossession**. Selected: **Sets InPossession enum** to '**TikiTaka**'.
Short Passing *Radio Button*	Belongs to the button group: **InPossession**. Selected: **Sets InPossession enum** to '**ShortPassing**'.
Fast Build Up *Radio Button*	Belongs to the button group: **InPossession**. Selected: **Sets InPossession enum** to '**FastBuildUp**'.
Team Press *Radio Button*	Belongs to the button group: **OutOfPossession**. Selected: **Sets OutOfPossession enum** to '**TeamPress**'.
Counter Attack *Radio Button*	Belongs to the button group: **OutOfPossession**. Selected: **Sets OutOfPossession enum** to '**CounterAttack**'.
Park the Bus	Belongs to the button group: **OutOfPossession**.

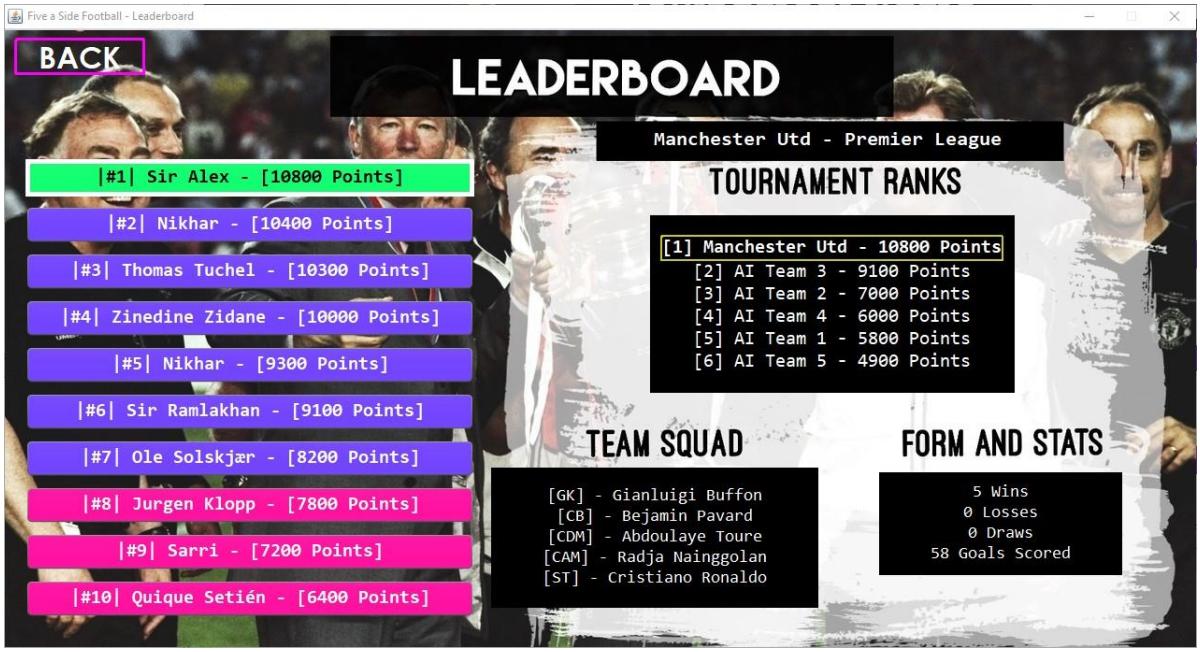
Radio Button	Selected: **Sets OutOfPossession enum** to '**ParkTheBus'**.
'Confirm Tactics' *Button*	Clicked: **Validates** that a radio button has been selected for each tactic (**Presence Check**) and sets the **visibility** of the **PlayMatch button** on the Tournament screen to true.

Results

This is the screen where the user will be showed a summary of their game such as their team squad, stats and tournament table.

Component	Action Elements and Data Validation
'Save and Proceed' *Button*	Clicked: **Adds the tournament to the Opponents and Leaderboard Tables of the Database** and proceeds to the **Leaderboard** screen. The user is prompted if saving to the database was successful or not.

Leaderboard



The Leaderboard lists the Top 10 User Teams sorted in descending order of their points acquired.

Data is accessed from the Teams, Opponents and Leaderboard Tables from the Database which is done using the Rankings class.

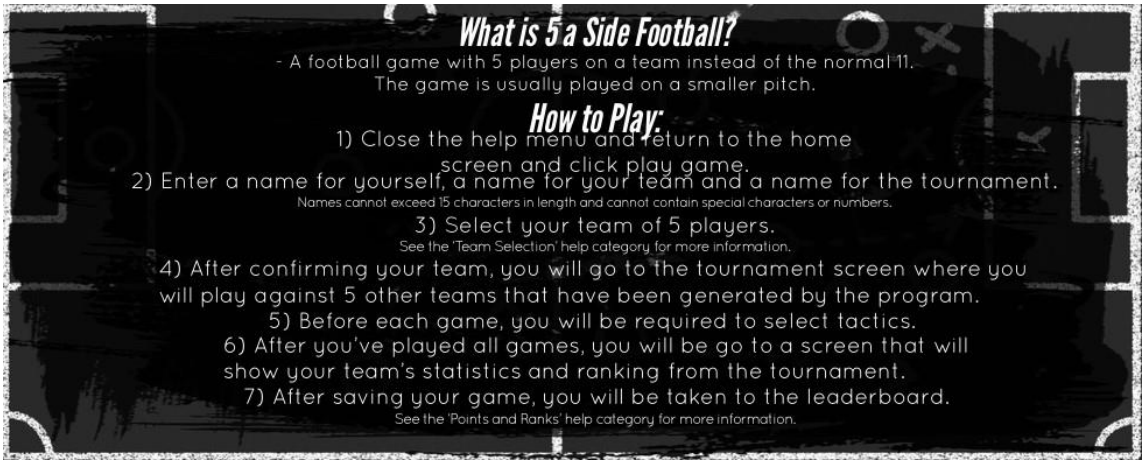
Component	Action Elements and Data Validation
#1 *Button*	Clicked: **Displays** the tournament information for the team that ranks **first**.
#2 *Button*	Clicked: **Displays** the tournament information for the team that ranks **second**.
#3 *Button*	Clicked: **Displays** the tournament information for the team that ranks **third**.
#4 *Button*	Clicked: **Displays** the tournament information for the team that ranks **fourth**.
#5 *Button*	Clicked: **Displays** the tournament information for the team that ranks **fifth**.
#6 *Button*	Clicked: **Displays** the tournament information for the team that ranks **sixth**.
#7 *Button*	Clicked: **Displays** the tournament information for the team that ranks **seventh**.
#8 *Button*	Clicked: **Displays** the tournament information for the team that ranks **eighth**.
#9 *Button*	Clicked: **Displays** the tournament information for the team that ranks **nineth**.
#10 *Button*	Clicked: **Displays** the tournament information for the team that ranks **tenth**.
'Back' *Button*	Clicked: Goes back to the **Home** screen.

Help

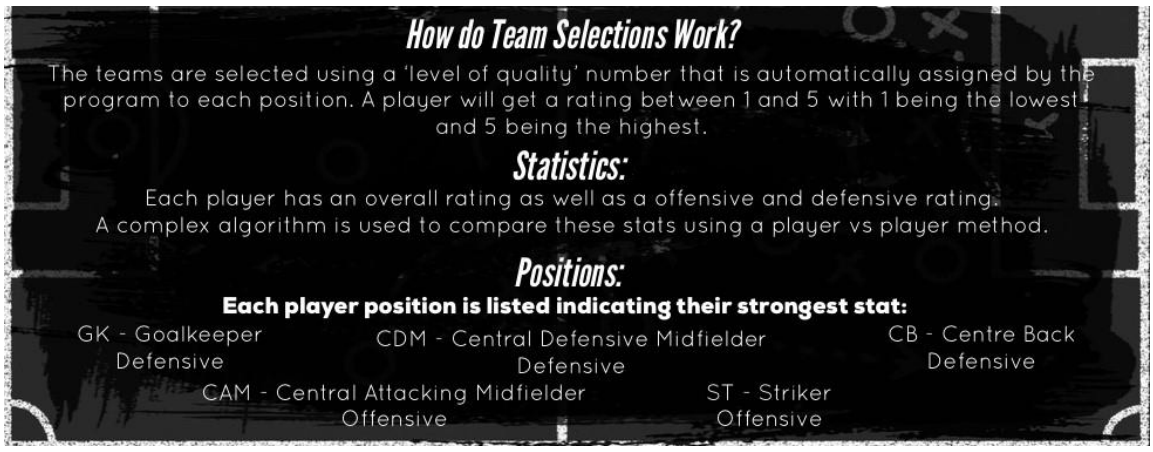
The Help screen is always a secondary screen that appears over any current class. There are 3 categories for the help menu.

Component	Action Elements and Data Validation
'Basics' *Button*	Clicked: **Displays** the **Basics Help Image stored** in the project folder.
'Team Selection' *Button*	Clicked: **Displays** the **Team Selection Help Image** stored in the project folder.
'Points and Ranks' *Button*	Clicked: **Displays** the **Points and Ranks Help Image** stored in the project folder.

Basics Help:

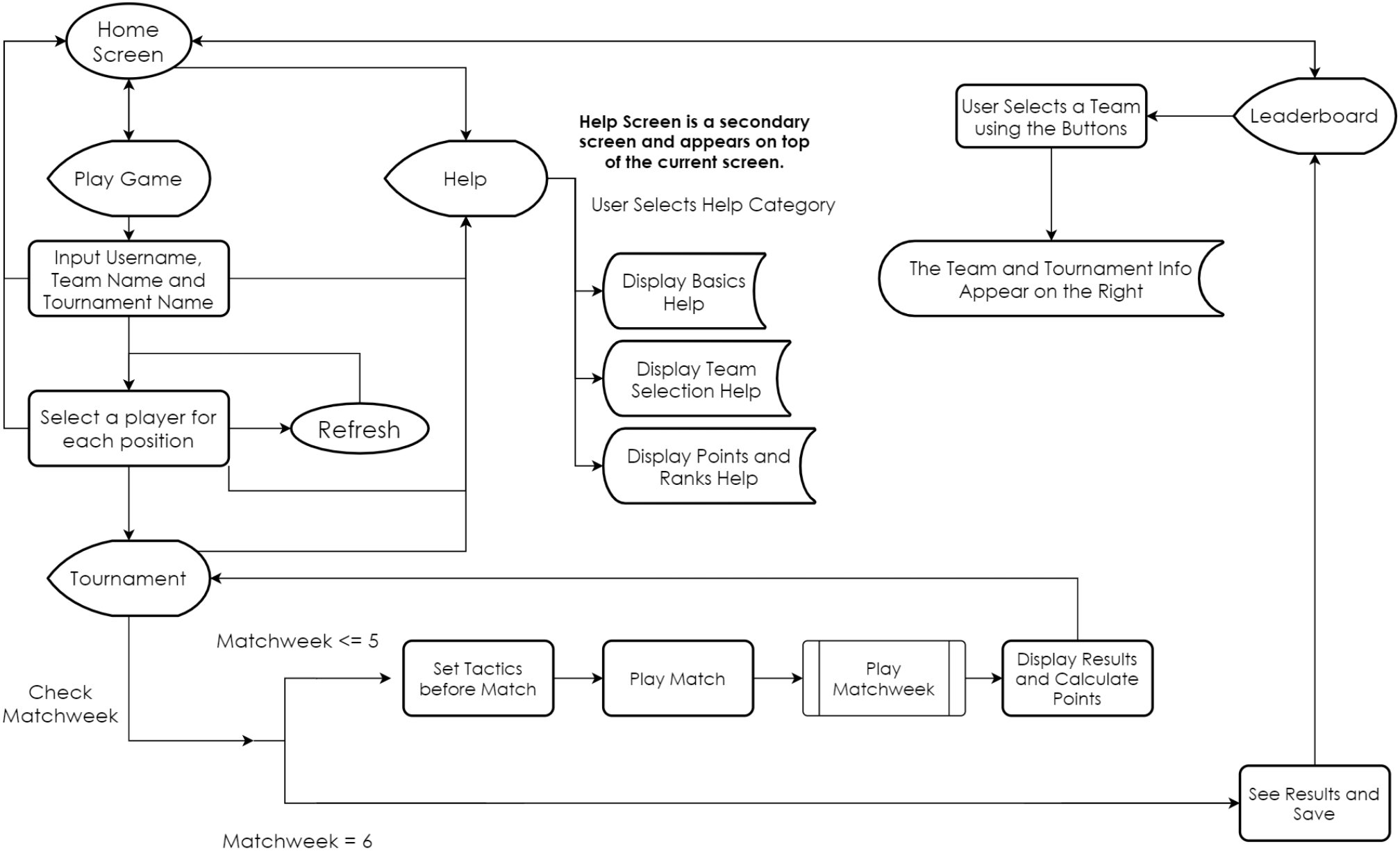


Team Selection Help



Points and Ranks Help:

Sequencing


Home Screen
Play Game
Input Username, Team Name and Tournament Name
Select a player for each position
Refresh
Tournament
Help
Help Screen is a secondary screen and appears on top of the current screen.
User Selects Help Category
Display Basics Help
Display Team Selection Help
Display Points and Ranks Help
Leaderboard
User Selects a Team using the Buttons
The Team and Tournament Info Appear on the Right
Check Matchweek
Matchweek <= 5
Set Tactics before Match
Play Match
Play Matchweek
Display Results and Calculate Points
Matchweek = 6
See Results and Save

Class Diagrams

***All Class Diagrams are working classes and do not have any GUI Properties.**

Database	
- connection: Connection	The connection with the Database.
+ Constructor ()	Connects to the database to allow read and write access.
+ getConnection ()	Return the connection with the Database.
+ runTeamInsertQuery (user: Team)	Adds the user team to the Team Table in the Database.
+ getTop1StringRecord (field: String, table: String, where: String, order: String)	Accesses a single string value from the Database based on parameters.
+ addOpponentsToDatabase (tournament: Rank)	Adds the points acquired by the 5 opposition teams to the Opponents Table in the Database.
+ addTeamResultsToDatabase (tournament: Rank)	Adds the team's results to the Leaderboard Table to the Database.

Player	
- playerID: String	The Player ID of a Player.
- position: String	The Position of a Player.
- firstName: String	The First Name of a Player.
- surname: String	The Surname of a Player.
- loq: String	The Level of Quality Number of a Player.
- overall: Integer	The Overall of a Player.
- offStat: Integer	The Offensive Stat of a Player.
- defStat: Integer	The Defensive Stat of a Player.
- filePath: String	The Image File Path of a Player.
+ Constructor (id: String, pos: String, fN: String, sN: String, lvl: String, ove: Integer, off: Integer, def: Integer, fP: String)	Creates a Player object using the fields from the Database Table "Players".
+ getPlayerID (): String	Player's Player ID Accessor.
+ getPosition (): String	Player's Position Accessor.
+ getLOQ (): String	Player's Level of Quality Number Accessor.
+ getOverall (): Integer	Player's Overall Accessor.

+ getOffStat (): Integer	Player's Offensive Stat Accessor.
+ getDefStat (): Integer	Player's Defensive Stat Accessor.
+ getFilePath (): String	Player's Image File Path Accessor.

PlayersArray	
- playerArray: Player [125]	An array of 125 Players.
- db: Database	Database communication.
- connection: Connection	Database connection.
+ Constructor ()	Creates an array of Player objects from the Players Table in the Database.
+ findPlayer (searchID: String): Integer	Returns a Player's index in the array.
+ getPlayer (index: Integer): Player	Returns a Player object.
+ formatSelectablePlayer (index: Integer): String	Returns a Player's details in a format that can be used to extract their details and file path.
+ _pullPlayerDetails_ (playerDetails: String): String	Returns a Player's details formatted to be presented in the Combo Boxes.
+ _pullPlayerFilePath_ (playerDetails: String): String	Returns a Player's Image File Path from the formatted String.
+ _getPlayerFilePath_ (filepath: String): Player	Returns a Player object using the Player's file path.

Team	
- tournamentID: String	The Tournament ID of a Team.
- teamName: String	The Team Name.
- gkID: String	The Player ID for the Team's GK.
- cbID: String	The Player ID for the Team's CB.
- cdmID: String	The Player ID for the Team's CDM.
- camID: String	The Player ID for the Team's CAM.
- stID: String	The Player ID for the Team's ST.
- username: String	The Username of the User.
- tournamentName: String	The Tournament Name of the Team.
- points: Integer	The Points acquired by the Team.
- wins: Integer	The number of Wins of a Team.
- draws: Integer	The number of Draws of a Team.
- losses: Integer	The number of losses of a Team.
- goalsScored: Integer	The number of Goals Scored of a Team.
- db: Database	Access to the Database for inserting.
- _playerArr_: PlayersArray	The PlayersArray object.

+ Constructor (un: String, tN: String, toN: String, gk: String, cb: String, cdm: String, cam: String, st: String)	Creates a Team for the User by receiving the properties.
+ Constructor (tID: String, tN: String)	Creates an Opponent Team that links with User Team Tournament ID.
+ updatePoints ()	Updates the Points acquired by a Team.
+ updateForm (wns: Integer, drws: Integer, los: Integer, gs: Integer)	Updates the number of wins, draws, losses and goals scored by a Team.
+ getTournamentID (): String	Tournament ID Accessor.
+ getUsername (): String	Username Accessor.
+ getTeamName (): String	Team Name Accessor.
+ getTournamentName (): String	Tournament Name Accessor.
+ getWins (): String	Number of Wins Accessor.
+ getDraws (): String	Number of Draws Accessor.
+ getLosses (): String	Number of Losses Accessor.
+ getPoints (): Integer	Points acquired Accessor.
+ getGoalsScored (): String	Number of Goals Scored Accessor.
+ getGK (): Player	Goalkeeper Accessor.
+ getCB (): Player	Centre Back Accessor.
+ getCDM (): Player	Central Defensive Midfielder Accessor.
+ getCAM (): Player	Central Attacking Midfielder Accessor.
+ getST (): Player	Striker Accessor.
+ getTeamPlayers (): String []	Returns an array of the Team Players formatted to show their name and position.
+ generateLOQ (): Integer []	Returns numbers 1 to 5 in a random order.
+ getPossiblePlayers (position: String, loq: String, format: Boolean): String []	Returns a formatted list of players that are selectable based on their position and level of quality.
+ getOpponentPlayers (position: String, loq: String): String	Randomly selects and returns the Player ID of a Player based on the parsed position and loq.
+ addTeamToDatabase ()	Adds the Team to the Database.

League	
- teamsLeagueArr: Team [6]	An array of 6 Teams.
- rankingsTable: String [6]	Used to store the rankings table.
- sortedPoints: Integer [6]	Used to store teams' points.
- rankTeamNames: String [6]	Used to store teams' names.
enum Mentality	Contains all possible Mentality tactic selections.
enum InPossession	Contains all possible In Possession tactic selections.
enum OutOfPossession	Contains all possible Out of Possession tactic selections.
enum GameOutcomes	Contains all possible Game Outcomes.
- aiMentality: Mentality	AI Team's Mentality.
- userMentality: Mentality	User Team's Mentality.
- aiInPossession: InPossession	AI Team's In Possession Tactic.
- userInPossession: InPossession	User Team's In Possession Tactic.
- aiOutOfPossession: OutOfPossession	AI Team's Out of Possession Tactic.
- userOutOfPossession: OutOfPossession	User Team's Out of Possession Tactic.
- userResult: String	The User Team's Result.
- game2Result: String	The 2nd Game Result.
- game3Result: String	The 3rd Game Result.
- t1Goals: Integer	Rounded off Team 1 Goals.
- t2Goals: Integer	Rounded off Team 2 Goals.
- t1RunGoals: Double	Team 1 Running Total Goals.
- t2RunGoals: Double	Team 2 Running Total Goals.
- t1OffBoost: Double	The Offensive Boost of Team 1.
- t2DefBoost: Double	The Defensive Boost of Team 2.
- team1Tactics: Integer [6]	The tactic boosts of Team 1.
- team2Tactics: Integer [6]	The tactic boosts of Team 2.
+ Constructor (uT: Team, t1: Team, t2: Team, t3: Team, t4: Team, t5: Team)	Creates a League object by receiving all 6 teams.
+ updateTable ()	Updates the Rankings Table.
+ getRankTable (): String []	Returns the Rankings Table.
+ getTeam (index: Integer): Team	Returns a specific team in the League.
+ playMatchWeek (matchweek: Integer, selMentality: Mentality, selInPossession InPossession, selOutOfPossession: selOutOfPossession)	Runs all the methods required to simulate 3 matches of a matchweek.
+ matchWeek1 (): String	Simulates Matchweek 1.
+ matchWeek2 (): String	Simulates Matchweek 2.
+ matchWeek3 (): String	Simulates Matchweek 3.
+ matchWeek4 (): String	Simulates Matchweek 4.

+ matchWeek5 (): String	Simulates Matchweek 5.
+ playMatch (t1: Team, t2: Team, user: Boolean): String	Runs all the methods required to simulate a match between 2 Teams.
+ teamVSteam (offTeam: Team, defTeam: Team, offTeamTactics: Integer [], defTeamTactics: Integer [], offTeamNumber: Integer, defTeamNumber: Number)	Runs the methods required for the player vs player.
+ playerVSplayer (p1Off: Integer, p2Def: Integer, t1Tac: Integer [], t2Tac: Integer [], offTeam: Integer, defTeam: Integer)	Compares an offensive player to a defensive player.
+ displayMatchOutcome (outcome: GameOutcomes)	Displays the match result.
+ determineBoosts (mentality: Mentality, inPossession: InPossession, outOfPossession: OutOfPossession, tactics: Integer [])	Determines the boosts of a Team's Tactics.

Rank	
- userPlayers: String [5]	The User Team's Players.
- userName: String	The User's Username.
- tournamentName: String	The Tournament Name.
- tournamentID: String	The Tournament ID.
- points: Integer	The User Team's Points.
- wins: String	The User Team's Wins.
- draws: String	The User Team's Draws.
- losses: String	The User Team's Losses.
- goalsScored: String	The User Team's Goals Scored.
- leagueTableRankings: String [6]	The League Rankings Table.
- teamPoints: Integer [6]	The Points of all Teams.
- db: Database	Database communication.
+ Constructor (rankedLeague: League)	Creates a Rank using a League.
+ Constructor (uN: String, tN: String, toN: String, gkID: String, cbID: String, cdmID: String, camID: String, stID: String, pts: Integer, wns: String, ls: String, gS: String, t1Pts: Integer, t2Pts: Integer, t3Pts: Integer, t4Pts: Integer, t5Pts: Integer)	Creates a Rank from the Database.

+ generateLeagueTable (t1Pts: Integer, t2Pts: Integer, t3Pts: Integer, t4Pts: Integer, t5Pts: Integer): String []	Creates the Rankings Table of a Rank.
+ getUserPlayer (index: Integer): String	Returns a Player's details based on the index.
+ getUserTeamName (): String	User Team Name Accessor.
+ getTournamentName (): String	Tournament Name Accessor.
+ getWins (): String	Number of User Team Wins Accessor.
+ getDraws (): String	Number of User Team Draws Accessor.
+ getLosses (): String	Number of User Team Losses Accessor.
+ getGoalsScored (): String	Number of User Team Goals Scored Accessor.
+ getLeagueTableRankings (position: Integer): String	Returns the formatted Rankings Table.
+ getTeamPoints (team: Integer): Integer	Return the points of any Team.
+ getRankTitle (): String	Return a formatted Rank Title that will be displayed on the Leaderboard buttons.
+ addTournamentToDatabase ()	Add the Tournament to the Opponents and Leaderboard Table in the Database.

Rankings	
- db: Database	Database communication.
- connection: Database	Database connection.
- leaderboard: Rank [10]	An array of 10 Ranks.
+ Constructor ()	Creates an array of the top 10 user teams from the Database.
+ getRank (position: Integer): Rank	Returns a Rank based on the array index.

Persistent Storage Design

Players Table

Players: Has fields related to a Football Player as well as a level of quality and an index number for generating a random team. (Linked to Player class)

Field Name	Data Type	Field Size	Description
Player ID	Short Text	3	The unique field used to identify each player
Position	Short Text	3	The position of the player
First Name	Short Text	15	The first name of the player
Surname	Short Text	20	The surname of the player
LOQ	Short Text	1	The level of quality of the player
LOQN	Short Text	1	The LOQ index number of the player
Overall	Number	Byte	The overall of the player
Off Stat	Number	Byte	The player's offensive stat
Def Stat	Number	Byte	The player's defensive stat
Face Image	Short Text	255	The pathway of the player image

Teams Table

Teams: Has fields related to a user made team. (Linked to Team class and Rank class)

Field Name	Data Type	Field Size	Description
Tournament ID	AutoNumber	Long Integer (Format 000)	The unique field to identify each tournament
Username	Short Text	15	The name of the user
Team Name	Short Text	15	The name of the user's team
Tournament Name	Short Text	15	The name of the user's league
GK ID	Short Text	3	The Player ID of the goalkeeper
CB ID	Short Text	3	The Player ID of the centre back
CDM ID	Short Text	3	The Player ID of the defensive midfielder
CAM ID	Short Text	3	The Player ID of the attacking midfielder
ST ID	Short Text	3	The Player ID of the striker

Opponent Table

Opponents: Stores the points acquired by the 5 opponent teams in order to rank the teams. (Linked to Rank class and Rankings class)

Field Name	Data Type	Field Size	Description
Tournament ID	Number	Long Integer (Format 000)	The unique field that links the opponents to the leaderboard
Team1 Points	Number	Integer	The points acquired by "AI Team 1"
Team2 Points	Number	Integer	The points acquired by "AI Team 2"
Team3 Points	Number	Integer	The points acquired by "AI Team 3"
Team4 Points	Number	Integer	The points acquired by "AI Team 4"
Team5 Points	Number	Integer	The points acquired by "AI Team 5"

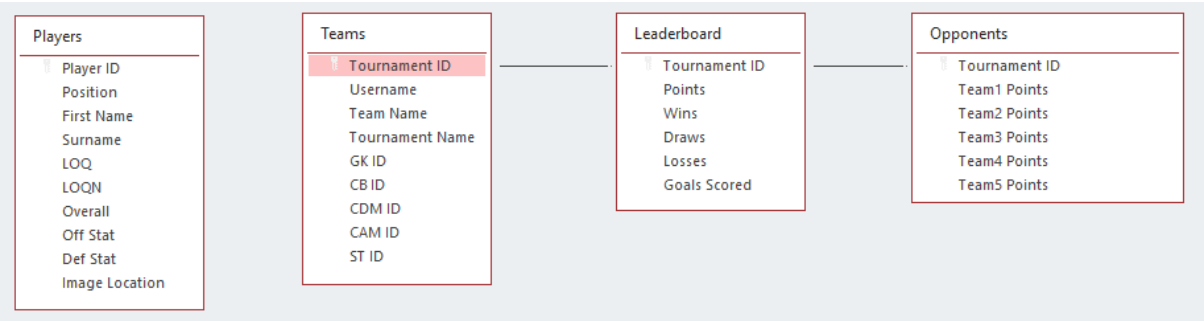
Leaderboard Table

Leaderboard: Has fields that link to the Team but would only be added after the game has finished. (Linked to Rank class and Rankings class)

Field Name	Data Type	Field Size	Description
Tournament ID	Number	Long Integer (Format 000)	The unique field that links the opponents' team to the user's team.
Points	Number	Integer	The amount of points obtained by the user's team.
Wins	Short Text	1	The number of wins recorded for the user's team.
Losses	Short Text	1	The number of losses recorded for the user's team.
Draws	Short Text	1	The number of draws recorded for the user's team.
Goals Scored	Short Text	2	The number of goals scored for the user's team.

Primary Keys are bolded and underlined.

Database Relationships:



Tournament ID is a common primary key for the tables Teams, Leaderboard and Opponents. This is because different data in a tournament is received at different times. The Teams table gets a record before the matches start where the Tournament ID is an AutoNumber. The program then gets the Tournament ID and stores it to be used as the Tournament ID for the Leaderboard and Opponents table when data is added after all the games have finished.

The object Rank from the program is a combination of fields from the Teams, Leaderboard and Opponents table making it essential that all of these tables are linked with the same Tournament ID.

Explanation of Storage Design

Five a Side Football integrates heavily with a Database to make storing and accessing data efficient. A database has been chosen primarily for its effectiveness in managing and organising data as there is a lot of data to store in the database tables. With a database, fields can be stored in specific tables to keep data organised and easy to find. It also makes editing, removing and adding records easier and safer. Using queries also allows data to be combined to get specific fields from specific tables. Queries are also able to sort and limit the data that is displayed making it easier to generate a leaderboard of only the top 10 teams.

A text file would have complicated this process and required more code to gather specific parts of a record and sort items accordingly. The database connection in Java is done with ease and can get any field from a table without having to use a delimiter and store every single value. A text field would have required that every field is stored in the program and then the program would require to have more code to find specific fields or limit the records based on their values. Multiple text fields would have also had to be used with no easy way to link the data together.