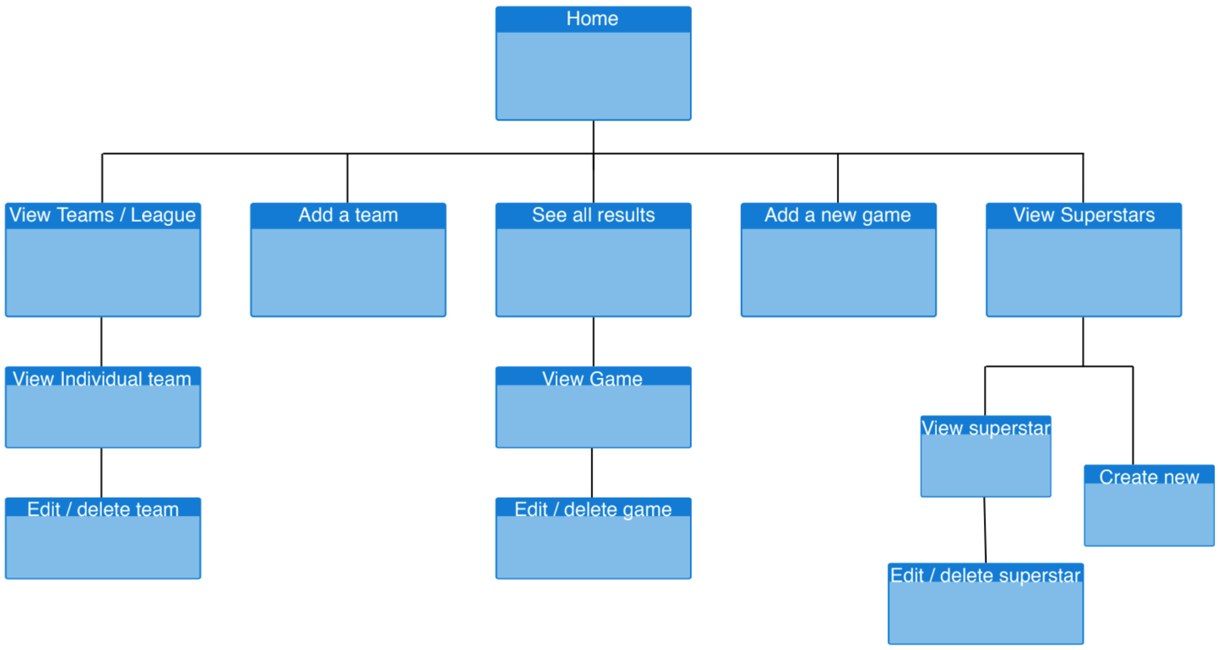
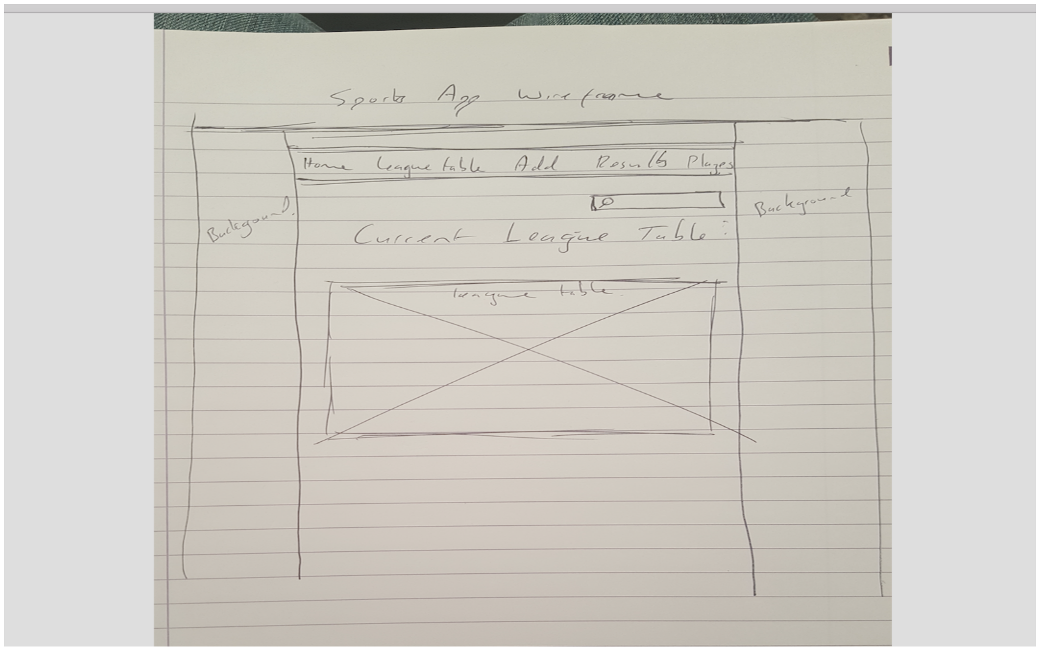
P 5.

Example of user sitemap

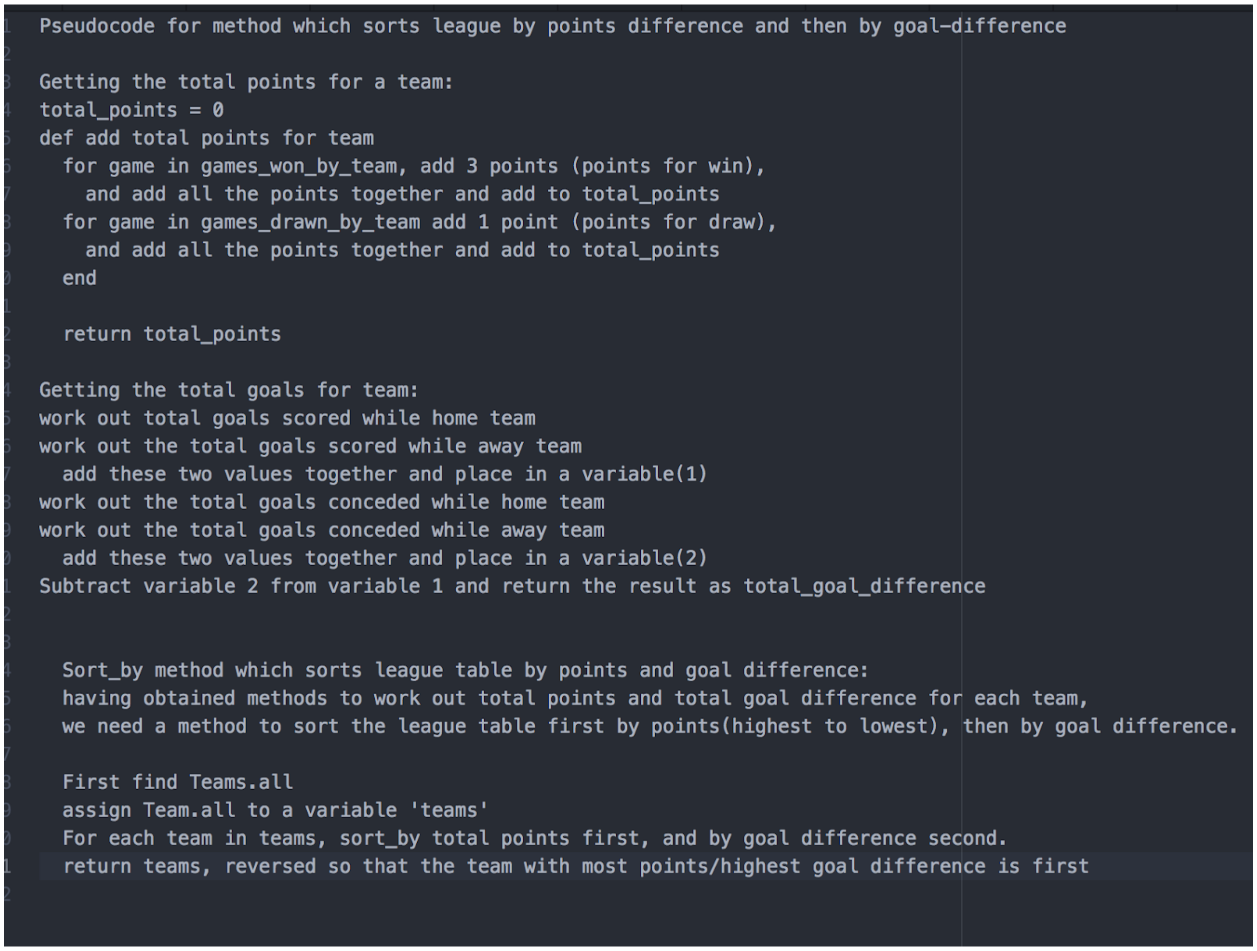


P 6.

Example of wireframe design:

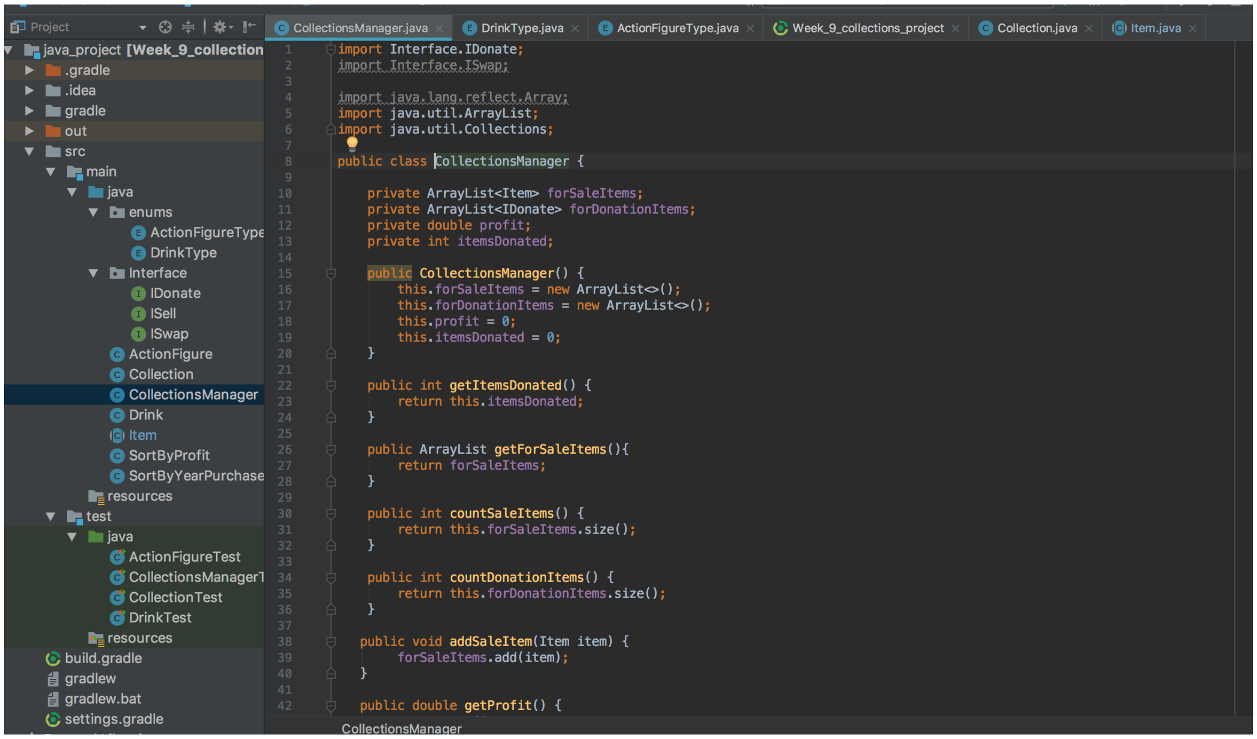


P 10.
Example of pseudocode used in method to sort teams in league by points (highest to lowest) and then by goal difference (highest to lowest)



```
Pseudocode for method which sorts league by points difference and then by goal-difference

Getting the total points for a team:
total_points = 0
def add total points for team
  for game in games_won_by_team, add 3 points (points for win),
    and add all the points together and add to total_points
  for game in games_drawn_by_team add 1 point (points for draw),
    and add all the points together and add to total_points
  end

  return total_points

Getting the total goals for team:
work out total goals scored while home team
work out the total goals scored while away team
  add these two values together and place in a variable(1)
work out the total goals conceded while home team
work out the total goals conceded while away team
  add these two values together and place in a variable(2)
Subtract variable 2 from variable 1 and return the result as total_goal_difference


  Sort_by method which sorts league table by points and goal difference:
  having obtained methods to work out total points and total goal difference for each team,
  we need a method to sort the league table first by points(highest to lowest), then by goal difference.

  First find Teams.all
  assign Team.all to a variable 'teams'
  For each team in teams, sort_by total points first, and by goal difference second.
  return teams, reversed so that the team with most points/highest goal difference is first
```

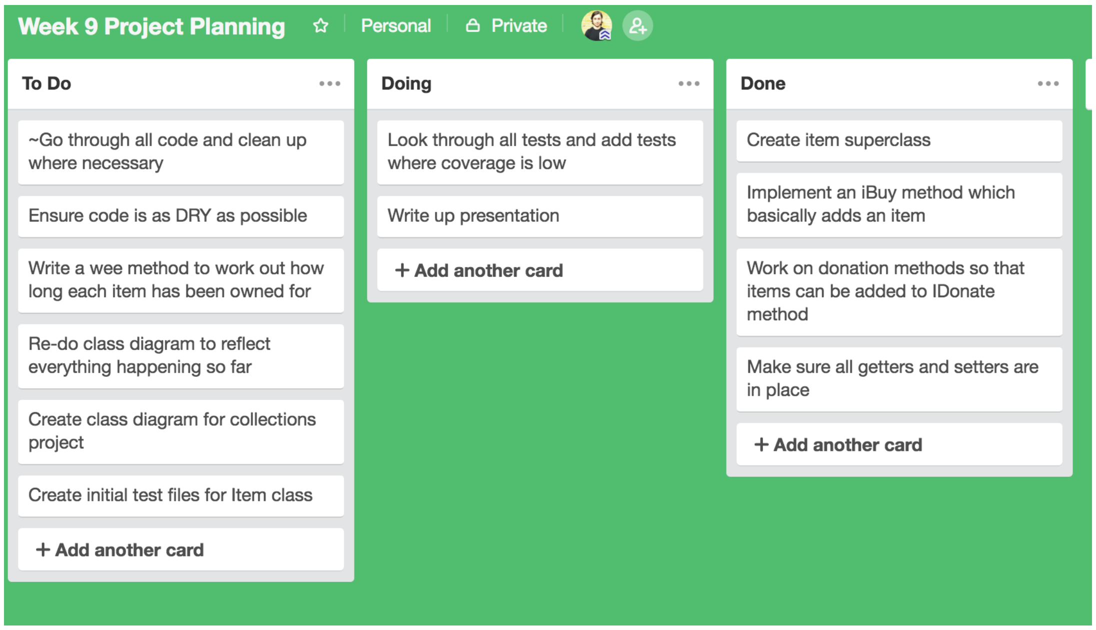
P.11. An example screenshot from a project on which I have worked alone, and the Github link:



```java
import Interface.IDonate;
import Interface.ISwap;

import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Collections;

public class CollectionsManager {

    private ArrayList<Item> forSaleItems;
    private ArrayList<IDonate> forDonationItems;
    private double profit;
    private int itemsDonated;

    public CollectionsManager() {
        this.forSaleItems = new ArrayList<>();
        this.forDonationItems = new ArrayList<>();
        this.profit = 0;
        this.itemsDonated = 0;
    }

    public int getItemsDonated() {
        return this.itemsDonated;
    }

    public ArrayList getForSaleItems(){
        return forSaleItems;
    }

    public int countSaleItems() {
        return this.forSaleItems.size();
    }

    public int countDonationItems() {
        return this.forDonationItems.size();
    }

   public void addSaleItem(Item item) {
        forSaleItems.add(item);
   }

   public double getProfit() {
```

https://github.com/Andylaugh1/week9_java_collections_project

P.12. Screenshots and photos of the project planning process showing changes and updates



To Do

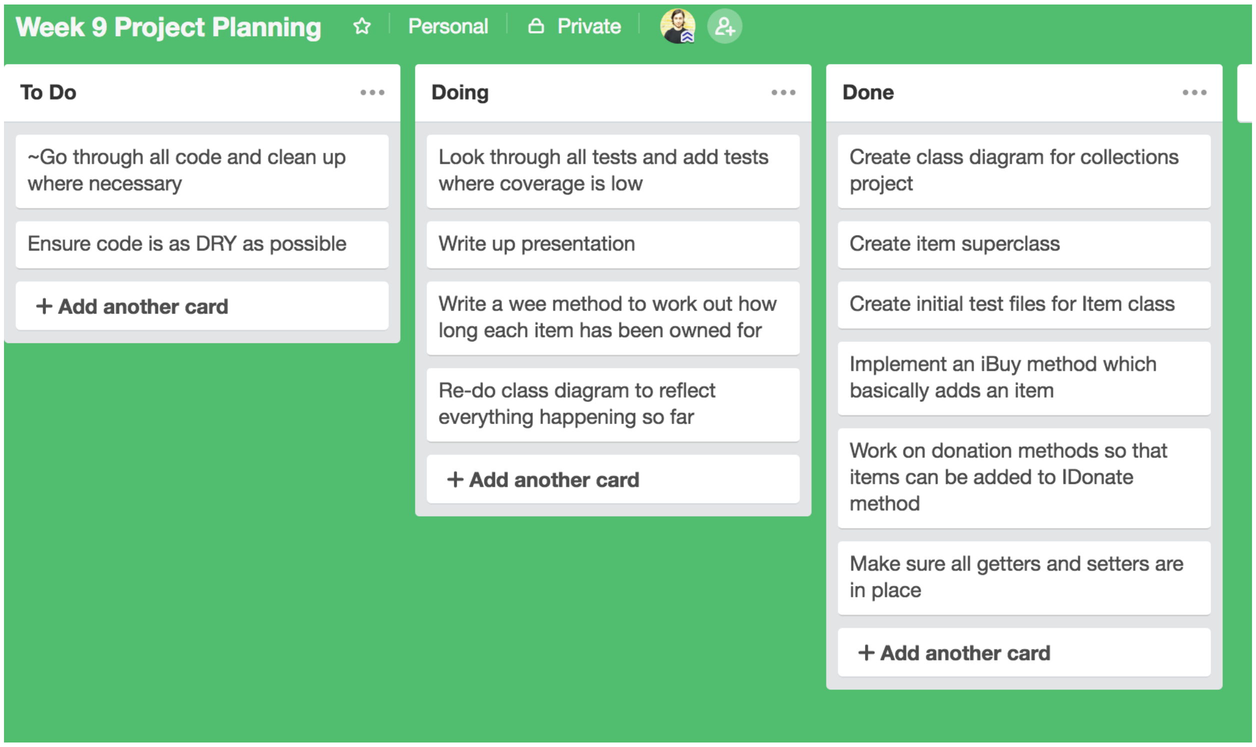
- ~Go through all code and clean up where necessary
- Ensure code is as DRY as possible
- Write a wee method to work out how long each item has been owned for
- Re-do class diagram to reflect everything happening so far
- Create class diagram for collections project
- Create initial test files for Item class

Doing

- Look through all tests and add tests where coverage is low
- Write up presentation

Done

- Create item superclass
- Implement an iBuy method which basically adds an item
- Work on donation methods so that items can be added to IDonate method
- Make sure all getters and setters are in place



To Do

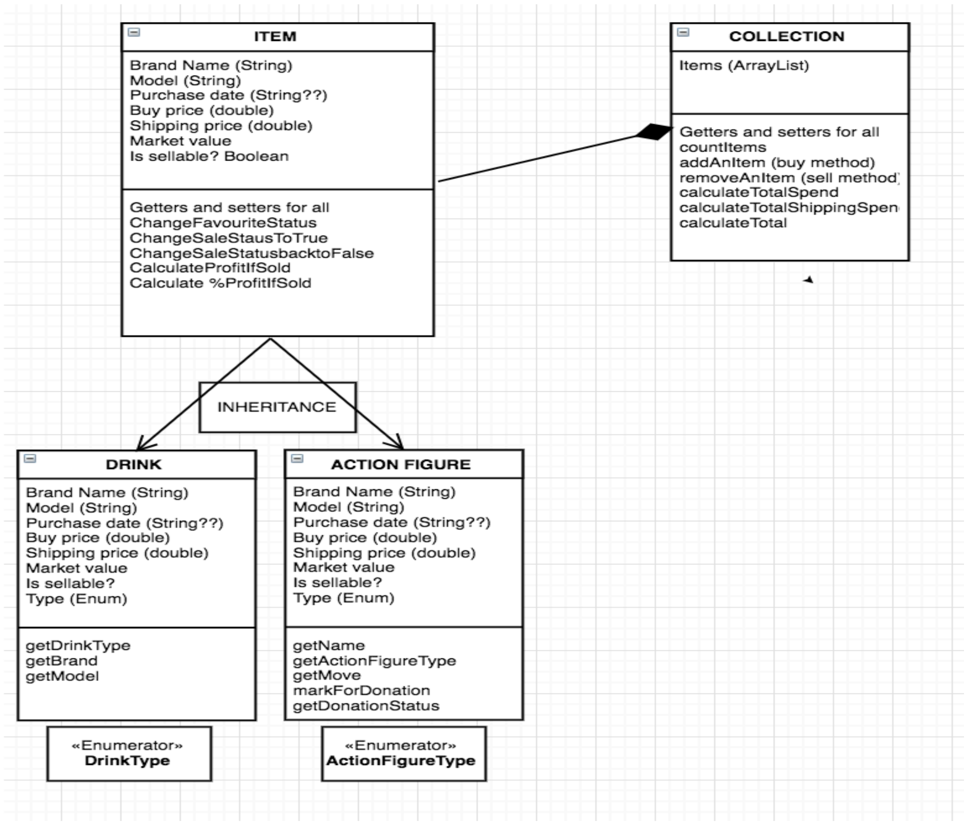
- ~Go through all code and clean up where necessary
- Ensure code is as DRY as possible
- + Add another card

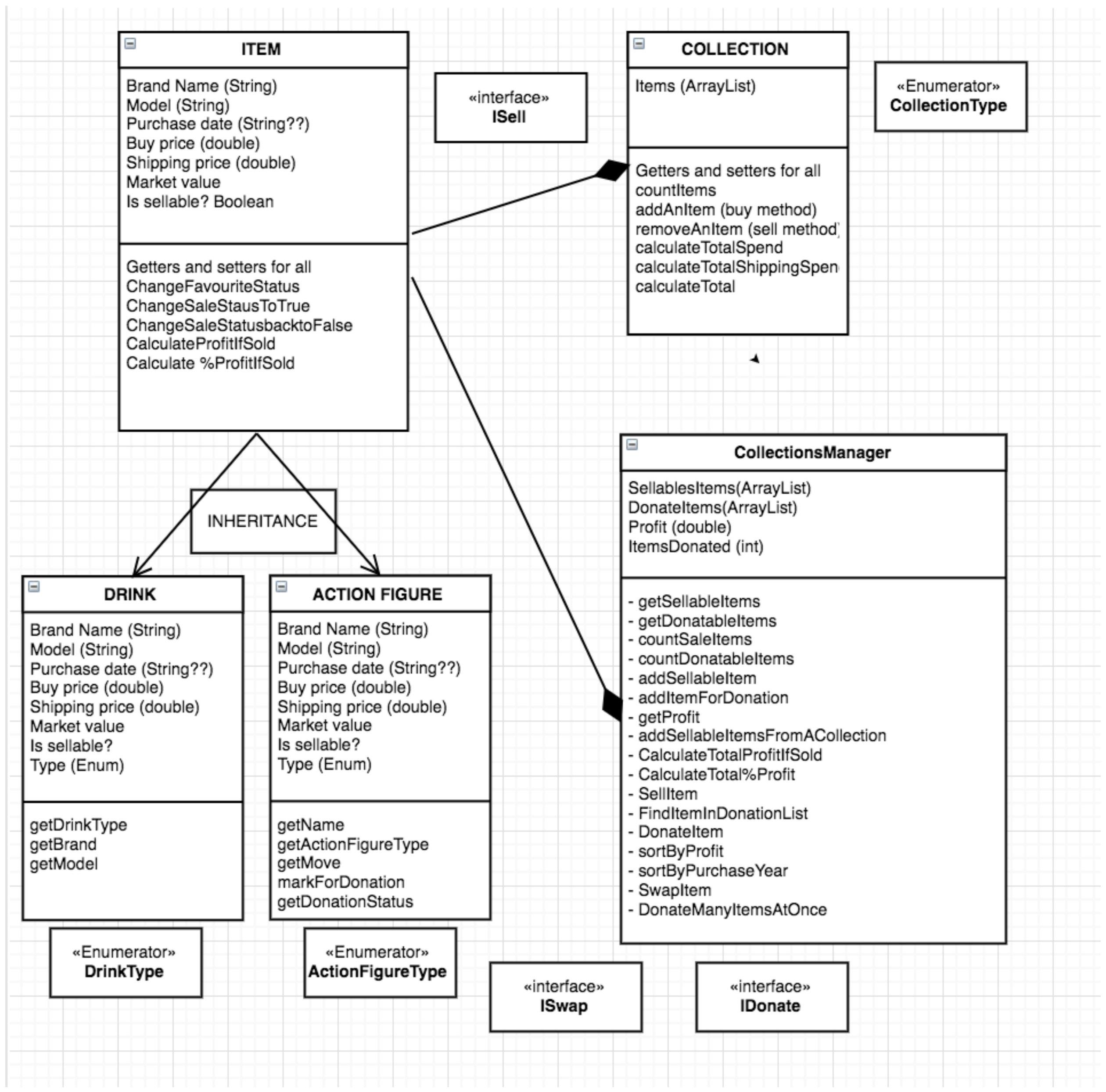
Doing

- Look through all tests and add tests where coverage is low
- Write up presentation
- Write a wee method to work out how long each item has been owned for
- Re-do class diagram to reflect everything happening so far
- + Add another card

Done

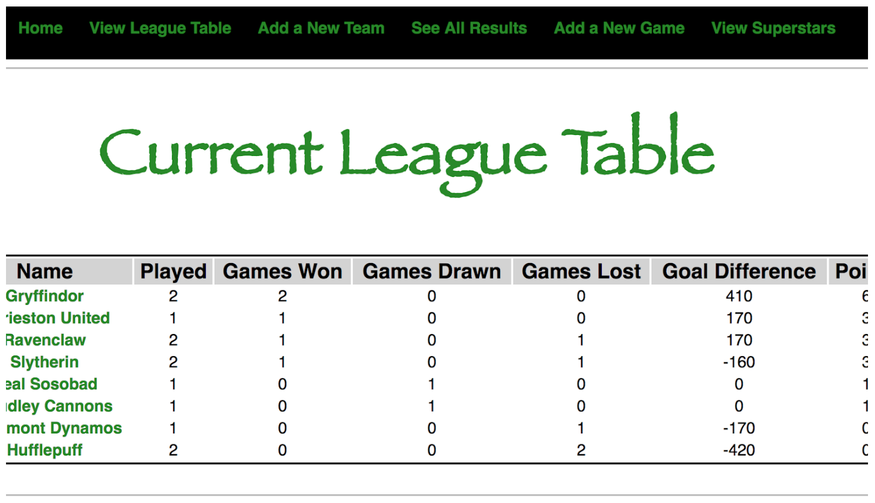
- Create class diagram for collections project
- Create item superclass
- Create initial test files for Item class
- Implement an iBuy method which basically adds an item
- Work on donation methods so that items can be added to IDonate method
- Make sure all getters and setters are in place
- + Add another card





P.13. Example of user input being saved or used

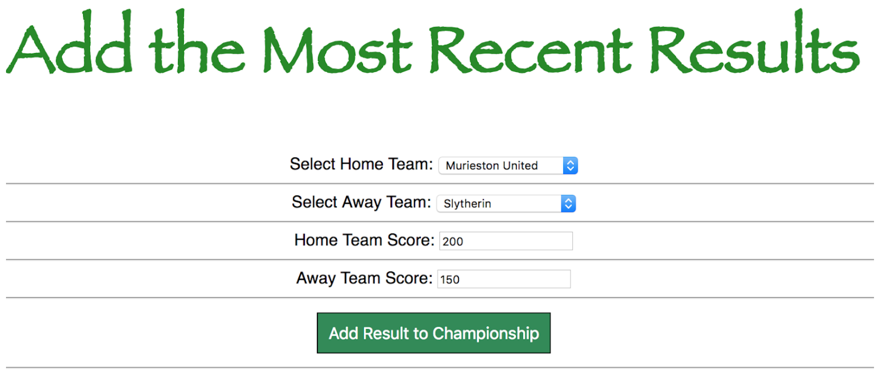
Screenshot 1: player clicks to add a new game



Current League Table

Name	Played	Games Won	Games Drawn	Games Lost	Goal Difference	Poi
Gryffindor	2	2	0	0	410	6
ieston United	1	1	0	0	170	3
Ravenclaw	2	1	0	1	170	3
Slytherin	2	1	0	1	-160	3
eal Sosobad	1	0	1	0	0	1
dley Cannons	1	0	1	0	0	1
mont Dynamos	1	0	0	1	-170	0
Hufflepuff	2	0	0	2	-420	0

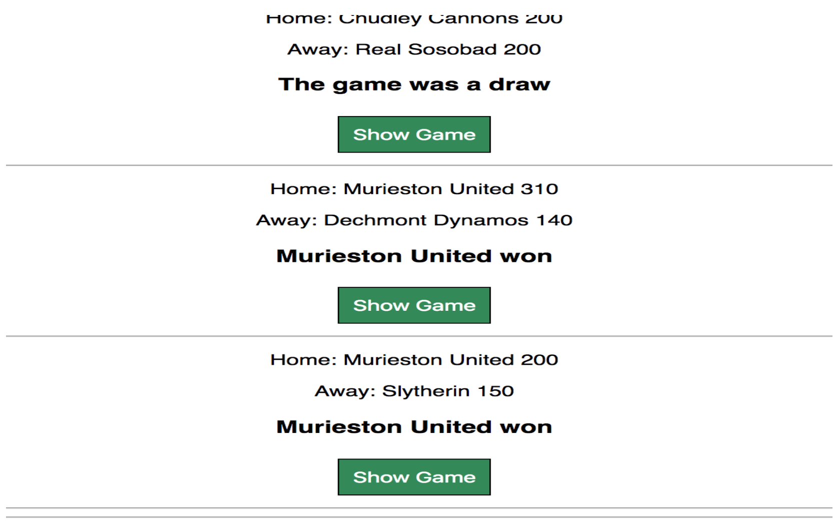
Screenshot 2: user enters new game details:



Add the Most Recent Results

Select Home Team:	Murieston United
Select Away Team:	Slytherin
Home Team Score:	200
Away Team Score:	150

Screenshot 3: User can see the new game which has been added, under the see all results tab:



Home: Chudley Cannons 200

Away: Real Sosobad 200

The game was a draw

Home: Murieston United 310

Away: Dechmont Dynamos 140

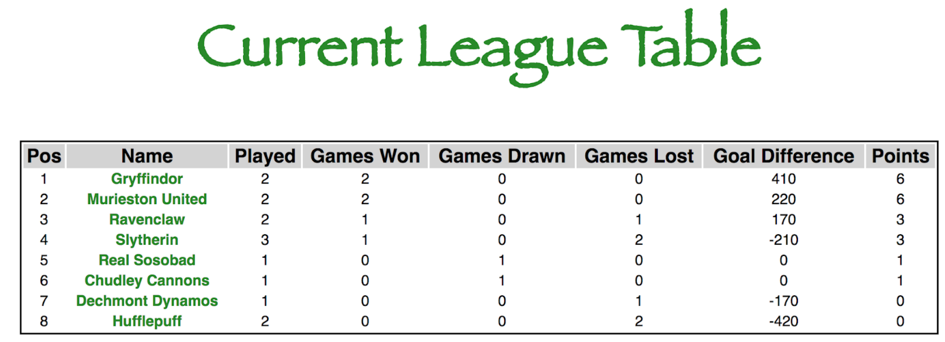
Murieston United won

Home: Murieston United 200

Away: Slytherin 150

Murieston United won

Screenshot 4: Updated league table following the processing of the new result:

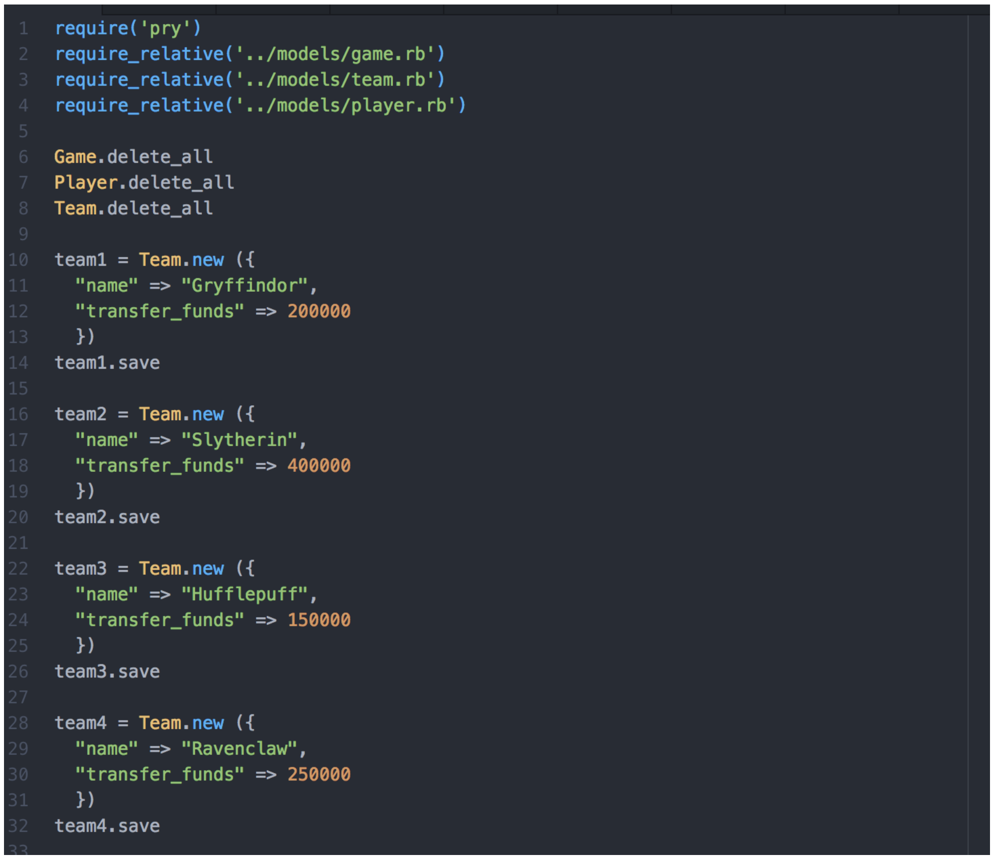


Current League Table

Pos	Name	Played	Games Won	Games Drawn	Games Lost	Goal Difference	Points
1	Gryffindor	2	2	0	0	410	6
2	Murieston United	2	2	0	0	220	6
3	Ravenclaw	2	1	0	1	170	3
4	Slytherin	3	1	0	2	-210	3
5	Real Sosobad	1	0	1	0	0	1
6	Chudley Cannons	1	0	1	0	0	1
7	Dechmont Dynamos	1	0	0	1	-170	0
8	Hufflepuff	2	0	0	2	-420	0

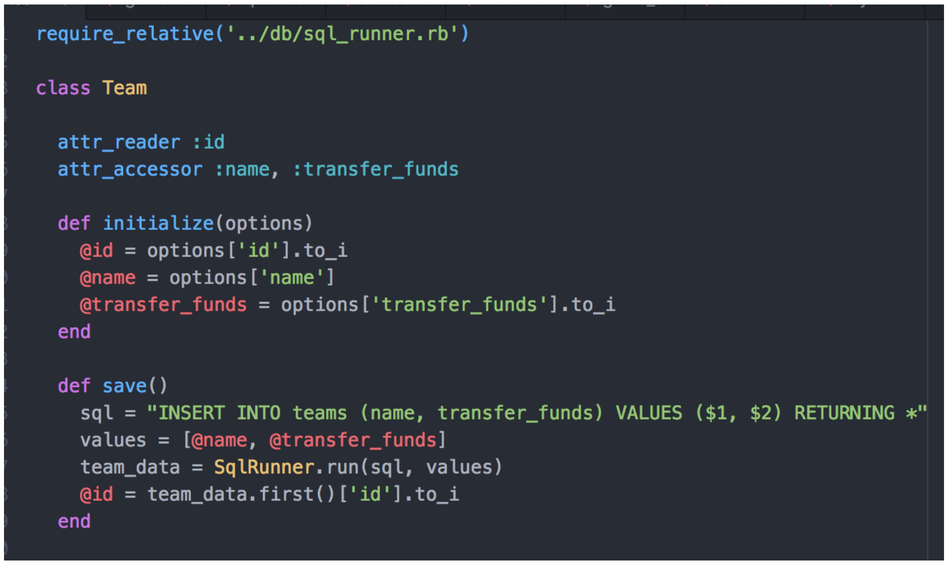
P.14: Interaction with database persistence:

Picture 1: Seed file which adds new teams to the database and saves them:



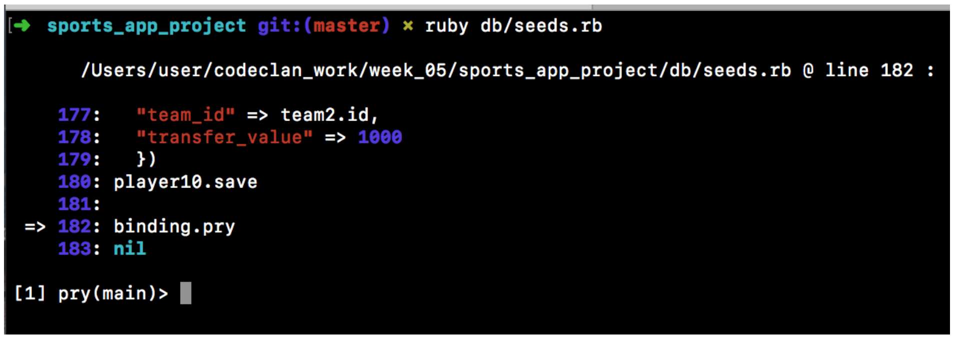
```ruby
require('pry')
require_relative('../models/game.rb')
require_relative('../models/team.rb')
require_relative('../models/player.rb')

Game.delete_all
Player.delete_all
Team.delete_all

team1 = Team.new ({
  "name" => "Gryffindor",
  "transfer_funds" => 200000
  })
team1.save

team2 = Team.new ({
  "name" => "Slytherin",
  "transfer_funds" => 400000
  })
team2.save

team3 = Team.new ({
  "name" => "Hufflepuff",
  "transfer_funds" => 150000
  })
team3.save

team4 = Team.new ({
  "name" => "Ravenclaw",
  "transfer_funds" => 250000
  })
team4.save
```

Screenshot 2: Showing Team class file, initialised, requiring SQL runner and showing the .save function.



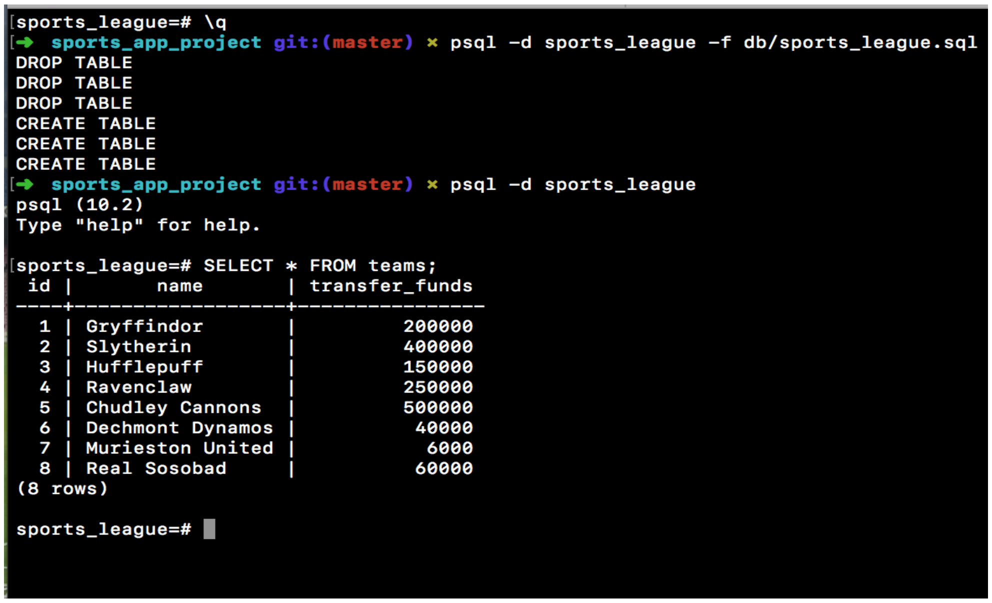
```ruby
require_relative('../db/sql_runner.rb')

class Team

  attr_reader :id
  attr_accessor :name, :transfer_funds

  def initialize(options)
    @id = options['id'].to_i
    @name = options['name']
    @transfer_funds = options['transfer_funds'].to_i
  end

  def save()
    sql = "INSERT INTO teams (name, transfer_funds) VALUES ($1, $2) RETURNING *"
    values = [@name, @transfer_funds]
    team_data = SqlRunner.run(sql, values)
    @id = team_data.first()['id'].to_i
  end
```

Screenshot 3: The seeds file being run to populate the database with the team data:



```
➜ sports_app_project git:(master) ✗ ruby db/seeds.rb

    /Users/user/codeclan_work/week_05/sports_app_project/db/seeds.rb @ line 182 :

    177:     "team_id" => team2.id,
    178:     "transfer_value" => 1000
    179:     })
    180: player10.save
    181:
 => 182: binding.pry
    183: nil

[1] pry(main)>
```

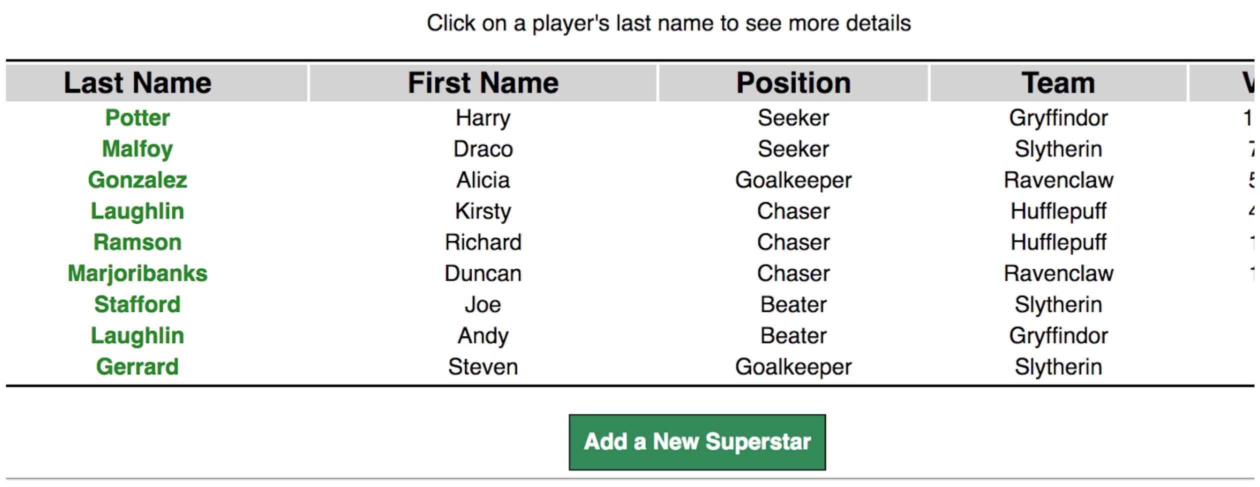
Screenshot 4: Database run showing all of the saved team data:



```
sports_league=# \q
➜ sports_app_project git:(master) ✗ psql -d sports_league -f db/sports_league.sql
DROP TABLE
DROP TABLE
DROP TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
➜ sports_app_project git:(master) ✗ psql -d sports_league
psql (10.2)
Type "help" for help.

sports_league=# SELECT * FROM teams;
 id |        name        | transfer_funds
----+--------------------+----------------
  1 | Gryffindor         |         200000
  2 | Slytherin          |         400000
  3 | Hufflepuff         |         150000
  4 | Ravenclaw          |         250000
  5 | Chudley Cannons    |         500000
  6 | Dechmont Dynamos   |          40000
  7 | Murieston United   |           6000
  8 | Real Sosobad       |          60000
(8 rows)

sports_league=#
```

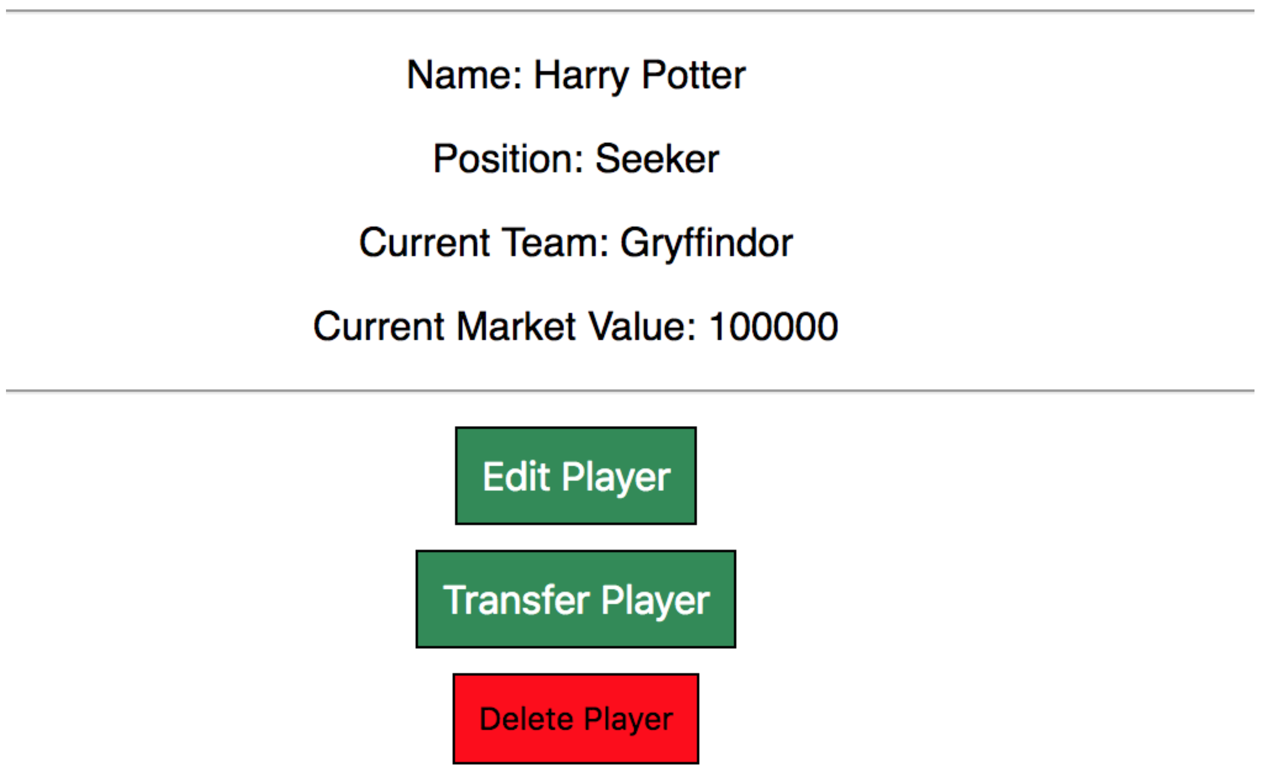
P. 15: Example of the current output of results and feedback to the user

Screen1 showing all players in list


Click on a player's last name to see more details

Last Name	First Name	Position	Team	V
Potter	Harry	Seeker	Gryffindor	1
Malfoy	Draco	Seeker	Slytherin	7
Gonzalez	Alicia	Goalkeeper	Ravenclaw	5
Laughlin	Kirsty	Chaser	Hufflepuff	4
Ramson	Richard	Chaser	Hufflepuff	1
Marjoribanks	Duncan	Chaser	Ravenclaw	1
Stafford	Joe	Beater	Slytherin	
Laughlin	Andy	Beater	Gryffindor	
Gerrard	Steven	Goalkeeper	Slytherin	

Screenshot 2 shows Harry Potter having been clicked on:



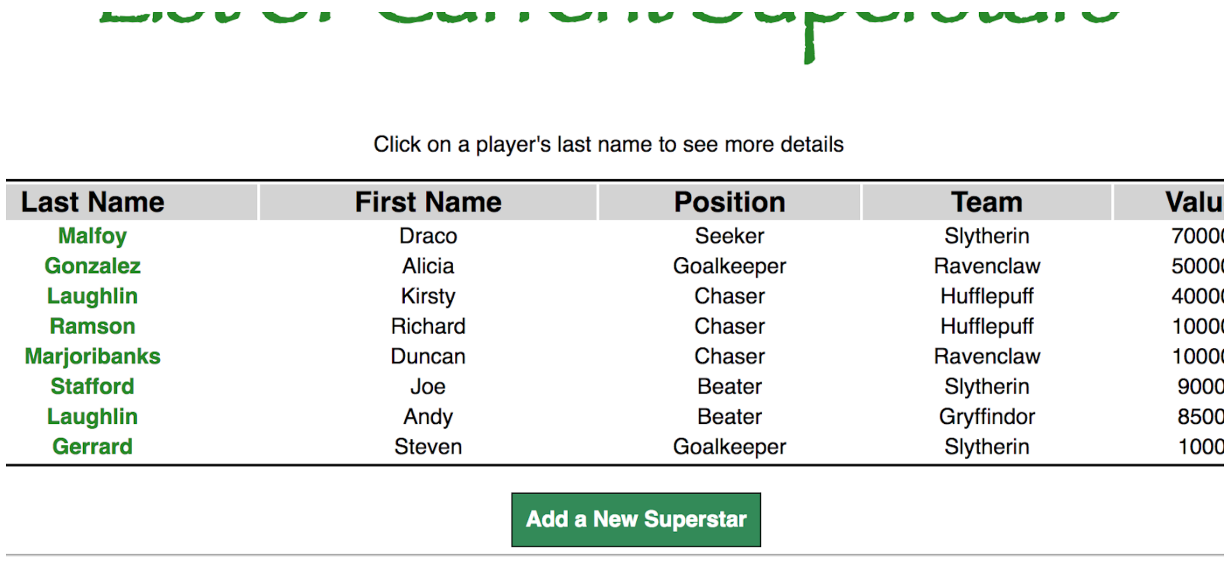
Name: Harry Potter

Position: Seeker

Current Team: Gryffindor

Current Market Value: 100000

Screen 3 shows the player list after clicking the delete button on Harry Potter; the player has been deleted from the database:



Click on a player's last name to see more details

Last Name	First Name	Position	Team	Valu
Malfoy	Draco	Seeker	Slytherin	7000
Gonzalez	Alicia	Goalkeeper	Ravenclaw	5000
Laughlin	Kirsty	Chaser	Hufflepuff	4000
Ramson	Richard	Chaser	Hufflepuff	1000
Marjoribanks	Duncan	Chaser	Ravenclaw	1000
Stafford	Joe	Beater	Slytherin	9000
Laughlin	Andy	Beater	Gryffindor	8500
Gerrard	Steven	Goalkeeper	Slytherin	1000