Institute of Technology Tallaght

Project Year 3

Golf Handicap

Final Document

Project Supervisor

John Cardiff

Submitted by

Jurijs Cicelimovs – X00116645

1st May 2017

Table of Contents

[Detailed Proposal 4](#_Toc481335521)

[Track Handicap 4](#_Toc481335522)

[Track Statistics 5](#_Toc481335523)

[Timetable 7](#_Toc481335524)

[Research Document 8](#_Toc481335525)

[Introduction 8](#_Toc481335526)

[Comparison with similar systems 9](#_Toc481335527)

[Hole19 9](#_Toc481335528)

[Keepmygolfscore.com 9](#_Toc481335529)

[Technical aspects of the project 10](#_Toc481335530)

[Languages 10](#_Toc481335531)

[Frameworks 11](#_Toc481335532)

[Databases 11](#_Toc481335533)

[Conclusion 11](#_Toc481335534)

[Exploration and Requirements 12](#_Toc481335535)

[Architecture Overview 12](#_Toc481335536)

[Three components of MVC: 13](#_Toc481335537)

[Story Card 14](#_Toc481335538)

[Release Plan 21](#_Toc481335539)

[Iteration 1 22](#_Toc481335540)

[Use Case Diagram 22](#_Toc481335541)

[Database Tables 23](#_Toc481335542)

[Hole 23](#_Toc481335543)

[Course 23](#_Toc481335544)

[Round 24](#_Toc481335545)

[User 24](#_Toc481335546)

[Handicap 24](#_Toc481335547)

[Iteration 2 25](#_Toc481335548)

[Iteration 3 27](#_Toc481335549)

[ERD - entity relationship diagram 28](#_Toc481335550)

[Class Diagram 29](#_Toc481335551)

[Testing conducted 30](#_Toc481335552)

[References 31](#_Toc481335553)

Development and Progress of the project can be reviewed at the GitHub repository: <https://github.com/yyyura/Project3>

# Detailed Proposal

The project will be developed using the java play framework and the use of an H2 database. It will have the following functionalities.

## Track Handicap

• When a user is created, they will have the option of inputting a handicap that already has or to generate a new one. A user who chooses to generate a new handicap will have their initial handicap calculated from the first 3 rounds that they play.

• When a user plays “round”, their handicap is either deducted from or added to their gross score, depending on whether the handicap is “plus” or “minus” to give their net score (“plus” and “minus” handicaps will be explained later), which will use to alter their handicap as they play more rounds (e.g. If a user with a handicap of 16 plays a par 72 course and returns a gross score of 86, their net score will be 86 - 16 = 70. Since this is below the Par of the course, their handicap will be lowered. If they returned a net score of above the Par of the course, their handicap would be raised)

• A user’s handicap will be given to a single decimal point (e.g. 16.7), and rounded to the nearest number, which will be their playing handicap (e.g. A user with an exact handicap of 16.5 will have a playing handicap of 17, and a user with a handicap of 16.4 will have a playing handicap of 16)

• After the user, has played their first 3 rounds, their handicap will be calculated by subtracting the Par of the course from their gross score for each round (keeping in mind that different courses can have different Pars), and calculating the average of these 3 final scores (eg. if a user returns a gross score of 89 on a par 72, 88 on a par 72, and 90 on a par 71, the final scores would be 17, 16 and 19 respectively, the average of these scores giving the user an initial handicap of 17.3, which would be rounded down to give a playing handicap of 17)

• As the user plays more rounds, their net score will be compared with the par of the course to determine how their handicap should be adjusted. A net score that is equal to, or up to 2 shots worse than the Par of the course will not increase or decrease the user’s handicap. A net score that is at least 3 shots worse than the Par of the course will add 0.1 to their handicap (this is to take into consideration that it is very difficult for a user to consistently return scores that are as good as or better than their handicap. By only increasing their handicap when they play a bad round, this will avoid the handicap from fluctuating wildly). If a user returns a net score that is below par, their handicap will be reduced according to what category their handicap falls into (this is explained below)

• There will be 3 categories a user’s handicap can fall into.

1. Category 1: any playing handicap better than 9 inclusive (including “plus” handicaps). A user’s handicap that falls into this category will have their handicap reduced by 0.1 for each shot better than par on any round that they play.

2. Category 2: any playing handicap between 10 - 19 inclusive. A user’s handicap that falls into this category will have their handicap reduced by 0.2 for each shot better than par on any round that they play.

3. Category 3: any playing handicap between 20 - 28 inclusive. A user’s handicap that falls into this category will have their handicap reduced by 0.3 for each shot better than par on any round that they play.

• The maximum handicap a user may have is 28.0, and there is no minimum handicap (this is because it is physically impossible for a user to have a gross score better than 18 on an 18-hole course)

• For calculating how a player’s handicap affects the gross score of a Round to give the net score, there will be 2 categories that handicaps fall into.

1. “plus”: a plus handicap is any playing handicap below 0. A plus handicap will be added to the gross score of a Round to give the net score. A plus handicap will be indicated using the “+” symbol just before the handicap (e.g. +2, or +1.6).

2. “minus”: a minus handicap is any playing handicap equal to or above 0. A minus handicap will be deducted from the gross score of a Round to give the net score. A minus handicap will not be indicated using any explicit symbol, all handicaps without a “+” symbol are implicitly minus handicaps.

## Track Statistics

• When inputting a Round, the user will be given the option to either record the gross score from a Round, or to input the statistics for the round (which will include recording the gross score of the round)

• When recording the statistics for a Round, the user will be asked to input the statistics for each hole. These statistics will proceed as follows

1. Did the user hit the fairway? (yes/no answer, any par 3 will not have this question as there is no fairway on a par 3)

2. If the user missed the fairway, did the user miss the fairway left/right?

3. Did the user hit the green in regulation? (yes/no answer, hitting a green in regulation is defined as the ball staying on the green after several shots such that the user has a putt for a birdie or better, a birdie is defined as a score that is 1 shot better than the Par of the hole)

4. If the user did not hit the green in regulation, did the user miss the green left/right/long/short?

5. If the user did not hit the green in regulation, how many shots did the user take before the ball stayed on the green?

6. After hitting the green, how many putts did the user take before finishing the hole?

7. Did the user miss any putts shorter than 2 feet? (yes/no answer)

8. Did the user take any penalty strokes on the hole? (yes/no answer)

9. If the user took any penalty strokes on the hole, how many did they take?

• While recording the statistics for a Round, the user will have the option of returning to previous holes or proceeding to the next hole if the current hole’s statistics have been completed.

• After recording the statistics for a hole, the score on that hole will be calculated.

• After recording the statistics for each hole, the gross score for the round will be calculated by adding up the scores for each hole. The user will be presented with an on-screen scorecard, which will display the scores for each hole, the par on each hole, and the final score gross, and net scores for the round. The user will be asked to confirm that these scores are correct.

• If the scores are not correct, the user will be able to go back through each hole to adjust their score.

• If the user confirms that the scores are correct, they will be presented with several graphs that show some of their statistics in a visual format. These will be as follows:

1. Driving accuracy, a half pie chart that will show the percentage of fairways that they hit, missed left, and missed right

2. Greens in regulation, a pie chart that will show the percentage of greens in regulation that they hit, missed left, missed right, missed short, or missed long.

3. Putting, a bar graph that will display the number of 0-putts, 1-putts, 2-putts, and 3+ putts in the round (a 0-putt means that the user managed to get the ball in the hole without needing to putt)

4. Scoring, a bar graph displaying a number of eagles (2 shots better than par), birdies (1 shot better than par), pars, bogeys (1 shot worse than par), double-bogeys (2 shots worse than par, etc.), triple-bogeys (or worse) during the round

• after recording statistics for a Round, the data will be saved in a database, and the user will be able to view their average statistics for their last 5 rounds, last 10 rounds, and their all-time average statistics (i.e. The average statistics for all of the rounds that they have recorded)

## Timetable

• the user will be able to record their planned rounds of golf. They will be asked to input several details for the round, which will be as follows:

1. date

2. tee time

3. golf course

4. playing partners (these can be other users, people who do not use the application, or unknown)

• the user may record that they do not have any playing partners. If they do so, they can be suggested other users within their skill range (their skill range being ±4 shots on their handicap), if the user wishes, they can be given the contact details of any of these suggestions.

# Research Document

This document prepared to determine which technologies to use in the project.

This project going to provide users with golf handicap tracker.

## Introduction

This research document is the basis for the third-year project. It will study technologies which are going to be considered for this project.

In this research, there will be a comparison of already existing systems on the market. And highlighting the aspects which are going to be better or different in the new project.

After the thorough consideration, team members going to choose most suitable software and technologies.

## Comparison with similar systems

### Hole19

Hole19 is an app created for android and iOS systems, it features the ability to track your scores throughout a “round”, and displays your statistics after the round is completed. This app is notable for having a very user-friendly design, being very simple to use and having very clear displays for your round statistics. Due to the nature of this being an application designed for phones, it is limited in the statistics it can track, and as such only covers the essential areas of the user's game, and even then, not in the most detail. Since this project will be accessed mostly using a computer, it can cover these areas in more detail, and as such give a complete overview of the user's game.

### Keepmygolfscore.com

Like this project, keepmygolfscore is a web-based system that tracks the user's statistics and handicap after rounds. While this allows it to be detailed about the user's statistics, it is not presented in a very clear manner. The handicap system it has is also unfriendly to use, as it requires 12 rounds to be input before the user receives a handicap. While this has the benefit of giving a user a more accurate initial representation of their skill level (i.e. The system used by the project could give an initial handicap several shots above or below), it also means the user must wait a long time before they can get a handicap (assuming the user only plays 1 full round a week, this would mean they should use the system for 3 months before they get their handicap).

## Technical aspects of the project

Intended users going to use this online web application over a web browser. The performance of this web application is not going to depend on platform or language in which it will be written. Thus, our team going to choose framework and language which is more comfortable for us, and give less trouble. Below is considered some possibilities for our project.

### Languages

#### C++

C ++ is widely used for software development, being one of the most common, the most commonly used programming languages. The area of application includes the creation of operating systems, a variety of applications, device drivers, applications for embedded systems, high-performance servers, and entertainment applications (games).

The biggest advantage of C++ language, that it is faster than other languages, it has direct access to hardware without a virtual machine in the middle. Due to the nature of the application not having any major performance requirements this advantage of C++ is not necessary.

Difficulties:

A memory leak is an uncontrollable process of reducing the amount of free RAM. This could be a big challenge when using C++. Therefore, it will need some additional code to control memory.

#### Java

Disadvantages:

Per shootout.alioth.debian.org site for seven different tasks, the Java averages runtime two times slower than C++.

Benefits:

Java applications are usually compiled in a special bytecode, so they can run on any computer architecture, using Java Virtual Machine. Another important feature of Java technology is a flexible security system, in which the execution of the program is fully controlled by the virtual machine. Additionally, Java has a special process called garbage collection which periodically frees the memory by removing objects that are no longer in demand applications. Thus, comparing with C++, Java is preferable for our project.

### Frameworks

#### AngularJS

AngularJS - JavaScript framework with open source. Designed for single-page application development. The goal of AngularJS is an expansion of browser-based applications based on MVC-pattern, as well as simplification of testing and development.

#### Play Framework

Play - the development of open source framework written in the Java and Scala, uses a design pattern Model-View-Controller (MVC). It aims to increase productivity by using the coding by convention, display the error in the browser and the hot reloading of the code. Play 2 is stateless and is fully RESTful.

### Databases

#### MySQL

MySQL open-source relational database management system. Development and support by Oracle Corporation, which received the rights to the trademark, together with the absorbed Sun Microsystems. It is a solution for small to medium-sized applications.

#### H2

H2 cross-platform open relational database management system written entirely in Java. Despite its small size (just over 1 MB), H2 supports many features. Including built in client-server, data encryption, browser management console.

## Conclusion

Considering all the choices of frameworks, languages and databases, and since we have a lot of experience with some technologies in the second-year project, we concluded that the best choice for our third-year project will be Java language, Play framework and H2 database management system.

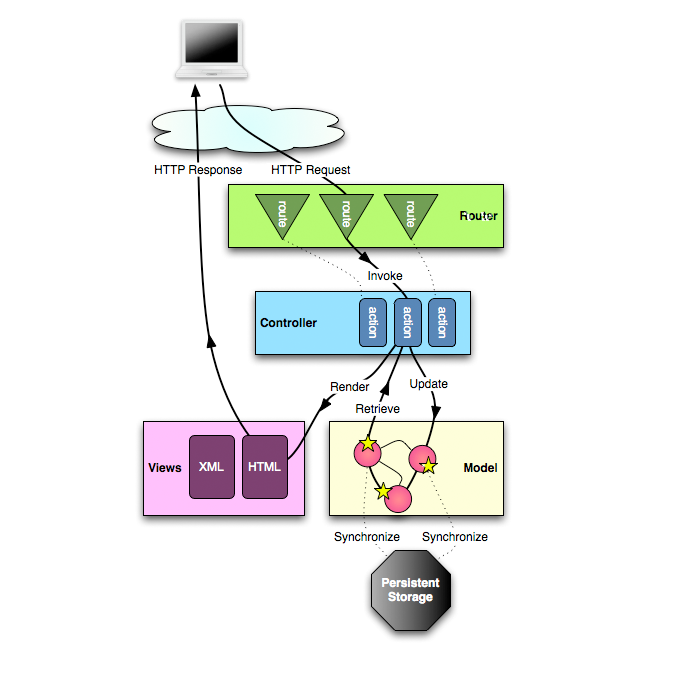
# Exploration and Requirements

## Architecture Overview

In this project is going to be used Play Framework. Framework splits the application into layers: Model – View – Controller. This design pattern splits data access and logic from presentation.

So, if there are some changes in user interface they will not affect any data management. And data can be handled without changing presentation layer. To make this functionality in MVC used controller component.

MVC is modular. Separation easy to support different users, and components easy to replace. Reduce complexity in large applications, increase maintainability, testability, and reusability.



### Three components of MVC:

#### Model

Model is directly managing Data, Logic and Rules of application. Record state of the App. Model updates View. Model links to the database. Independent of View, one Model can link to different views.

#### View

View (presentation layer) main functionality is to present Data to the user, it allows user interaction, but there is no any processing. It renders the model, request updates from the model. The view uses the model to generate the new user interface.

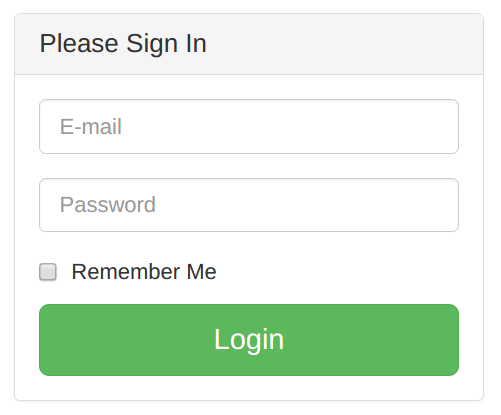
#### Controller

The controller defines application behaviour. Accept input, it is how user interface reacts to user input(event), receive messages from view, and send a message to model (tells what data to display). The controller handles user input, and update the model. Like when receiving a request for some page it loads the view of according page.



# Story Card

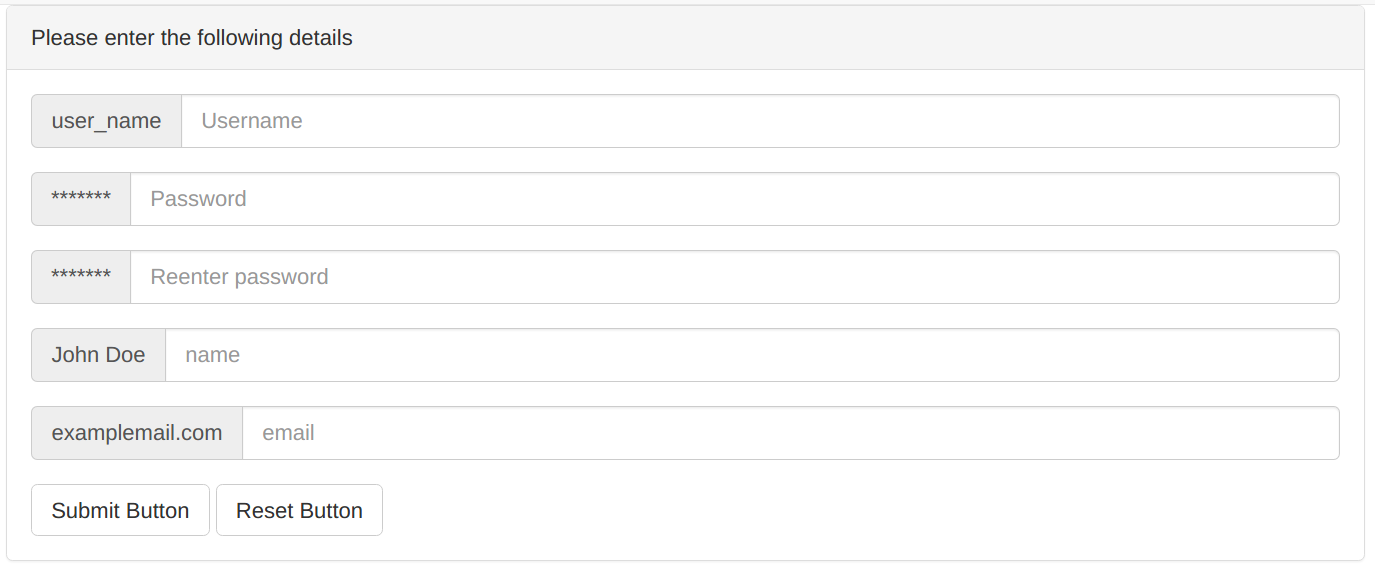
|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 1** | **Story Title: Sign In** | **Priority: 1** | **Estimate: 2 h** |
| Sign In, so the user gets access to user account. | | | |



**Confirmation:**

1. **Success** – valid user sign in and directed to home page
   1. User introduced with the homepage, and have all access to options of golf handicap app.
2. **Failure** – display message
   1. E-mail and user password did not match against stored data in the database.
   2. Failed message displayed.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 2** | **Story Title:**  **Sign Up** | **Priority: 1** | **Estimate: 2 h** |
| Sign Up, add a new user to the system. | | | |



**Confirmation:**

1. **Success** – new user sign up and added to the system and can sign into account
   1. User added to the system, and have access to options of golf handicap app.
2. **Failure** – display message
   1. Users detail not passed validity test and cannot be stored in the database.
   2. Failed message displayed.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 3** | **Story Title:**  **Update User** | **Priority: 4** | **Estimate: 2 h** |
| Update user details per new requirements. | | | |

**Confirmation:**

1. **Success** – details of user updated and stored in database
   1. User details i.e. username, is updated and successfully stored in database
2. **Failure** – display message
   1. Users detail not passed validity test and cannot be stored in the database.
   2. Failed message displayed.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 4** | **Story Title:**  **Delete User** | **Priority: 4** | **Estimate: 2 h** |
| Remove user from the system. | | | |

**Confirmation:**

1. **Success** – user deleted from system per request or violation of rules.
   1. User does not wish to keep his details in the system and successfully deleted from system and database
2. **Failure** – display message
   1. Users detail have not been found in the current database.
   2. Failed message displayed.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 5** | **Story Title:**  **List User Data** | **Priority: 5** | **Estimate: 2 h** |
| List the detail of the user. | | | |

**Confirmation:**

1. **Success** – admin has access to view detail of the particular user.
   1. All data stored on some user is displayed before admin.
2. **Failure** – display message
   1. Users detail have not been found in the current database.
   2. Failed message displayed.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 6** | **Story Title:**  **Log Out** | **Priority: 2** | **Estimate: 2 h** |
| User logs out from the system. | | | |

**Confirmation:**

1. **Success** – user finish work and logs out from system
   1. User finish work clicks on log out button and logs out.
2. **Failure** – display message
   1. The user currently not logged in.
   2. Failed message displayed.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 7** | **Story Title:**  **Recording a “round”** | **Priority: 2** | **Estimate: 3h** |
| User records a “round” | | | |

**Confirmation:**

1. **Success** – round is recorded to the database
   1. User is shown the details of their round
2. **Failure** – display message
   1. Error occurred, round not recorded, please try again
   2. Detail failures & messages

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 8** | **Story Title:**  **Update handicap** | **Priority: 3** | **Estimate: 2h** |
| System calculates changes to handicap after a “round” and makes the update | | | |

**Confirmation:**

1. **Success** – handicap is updated successfully
   1. User is shown their new handicap
2. **Failure** – display message
   1. Error occurred, round not recorded, please try again
   2. Detail failures & messages

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 9** | **Story Title:**  **Calculate initial handicap** | **Priority: 3** | **Estimate: 2h** |
| Calculate the user’s initial handicap | | | |

**Confirmation:**

1. **Success** – handicap is successfully calculated
   1. User is shown their new handicap
2. **Failure** – display message
   1. Error occurred, handicap not calculated, please try again
   2. Detail failures & messages

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 10** | **Story Title:**  **Display statistics** | **Priority: 5** | **Estimate: 2h** |
| Display round statistics and past statistics for the user | | | |

**Confirmation:**

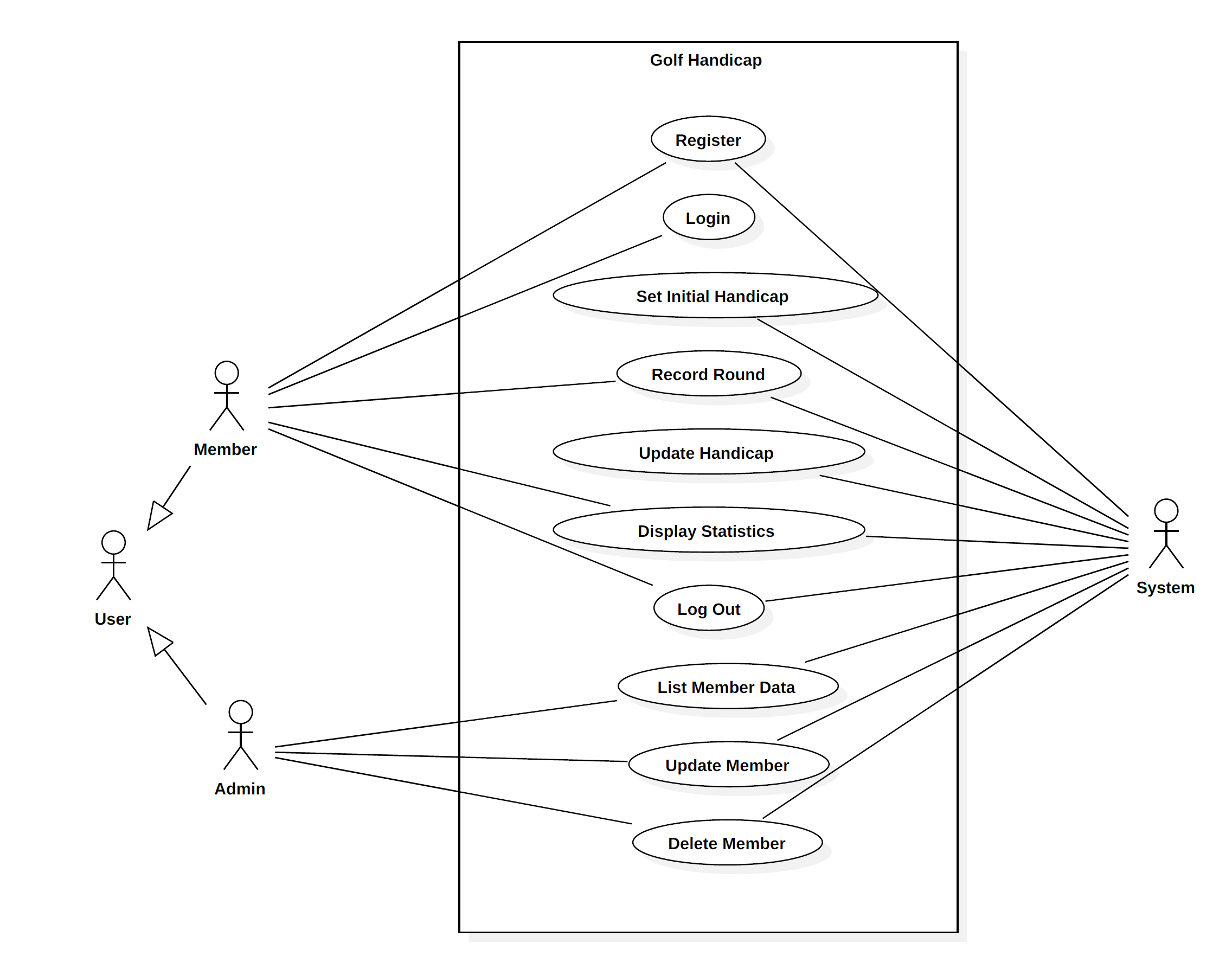
1. **Success** – statistics successfully displayed
   1. User is can view the statistics for their past rounds
2. **Failure** – display message
   1. Error occurred, round cannot be displayed, please try again
   2. Detail failures & messages

# Release Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | Stories | Duration | Start Date | End Date |
| 1 | 1, 2 | 4h | 17/03/17 | 19/03/17 |
| 2 | 6, 7 | 5h | 29/03/17 | 02/04/17 |
| 3 | 8, 9 | 4h | 25/04/17 | 30/04/17 |
| 4 | 3, 4 | 4h | 30/04/17 | 31/04/17 |
| 5 | 5, 10 | 4h | 31/04/17 | 01/05/17 |

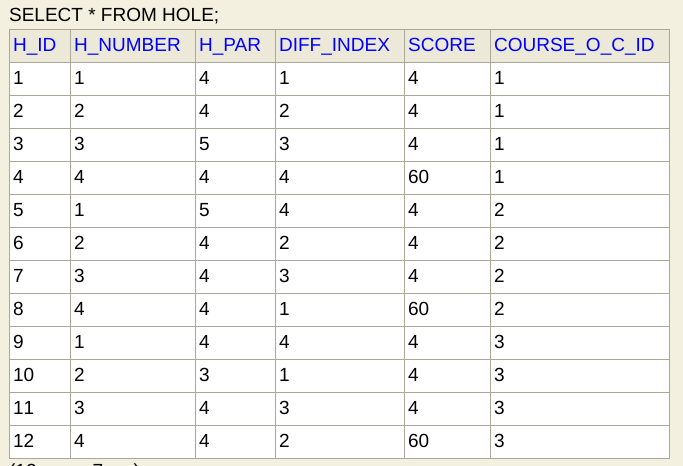
# Iteration 1

## Use Case Diagram

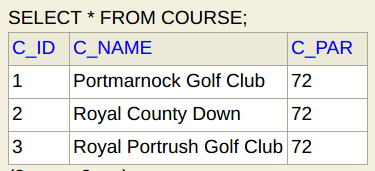


## Database Tables

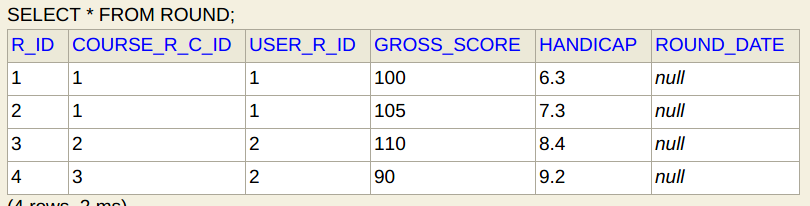
### Hole



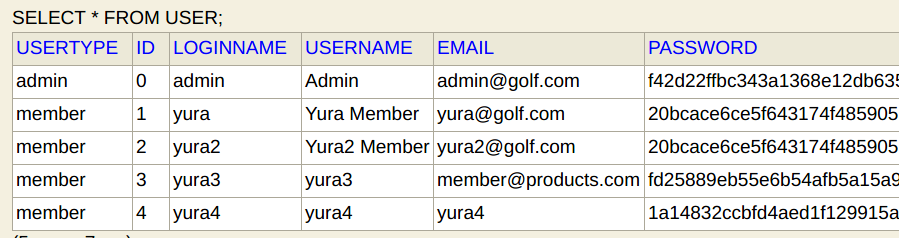
### Course



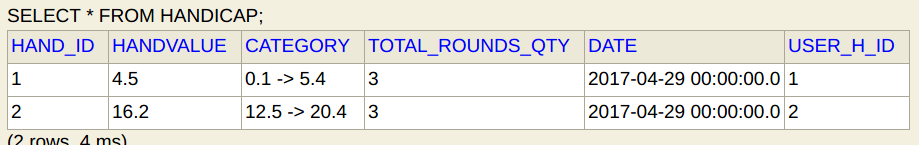
### Round



### User



### Handicap



# Iteration 2

Since the few changes in new version of the Play Framework, I encountered some difficulties with configuring E-bean and database. To fix connection problem the following code was added to the Project.

|  |  |
| --- | --- |
| play.db { | |
|  | | # The combination of these two settings results in "db.default" as the |
|  | | # default JDBC pool: |
|  | | config = "db" |
|  | | default = "default" |
|  | |  |
|  | | # Play uses HikariCP as the default connection pool. You can override |
|  | | # settings by changing the prototype: |
|  | | prototype { |
|  | | # Sets a fixed JDBC connection pool size of 50 |
|  | | #hikaricp.minimumIdle = 50 |
|  | | #hikaricp.maximumPoolSize = 50 |
|  | | } |
|  | | } |
| db { |
|  | # You can declare as many datasources as you want. | |
|  | # By convention, the default datasource is named `default` | |
|  |  | |
|  | # https://www.playframework.com/documentation/latest/Developing-with-the-H2-Database | |
|  | default.driver = org.h2.Driver | |
|  | #default.url = "jdbc:h2:mem:play" | |
|  | default.url = "jdbc:h2:file:./data/golfDB" | |
|  | default.username = sa | |
|  | default.password = "" | |
|  |  | |
|  | # You can turn on SQL logging for any datasource | |
|  | # https://www.playframework.com/documentation/latest/Highlights25#Logging-SQL-statements | |
|  | #default.logSql=true | |
|  | } | |
|  |  | |
|  |  | |
|  | ebean.default = ["models.\*"] | |

Test on Mapping class to the database went successfully.

To the Project was added, two new classes Course and Hole. There were problems encountered in the relationships between these two classes, causing compilation errors, for which we are still seeking solutions.

The entire development and progress of the project you can follow at the GitHub repository: https://github.com/yyyura/Project3

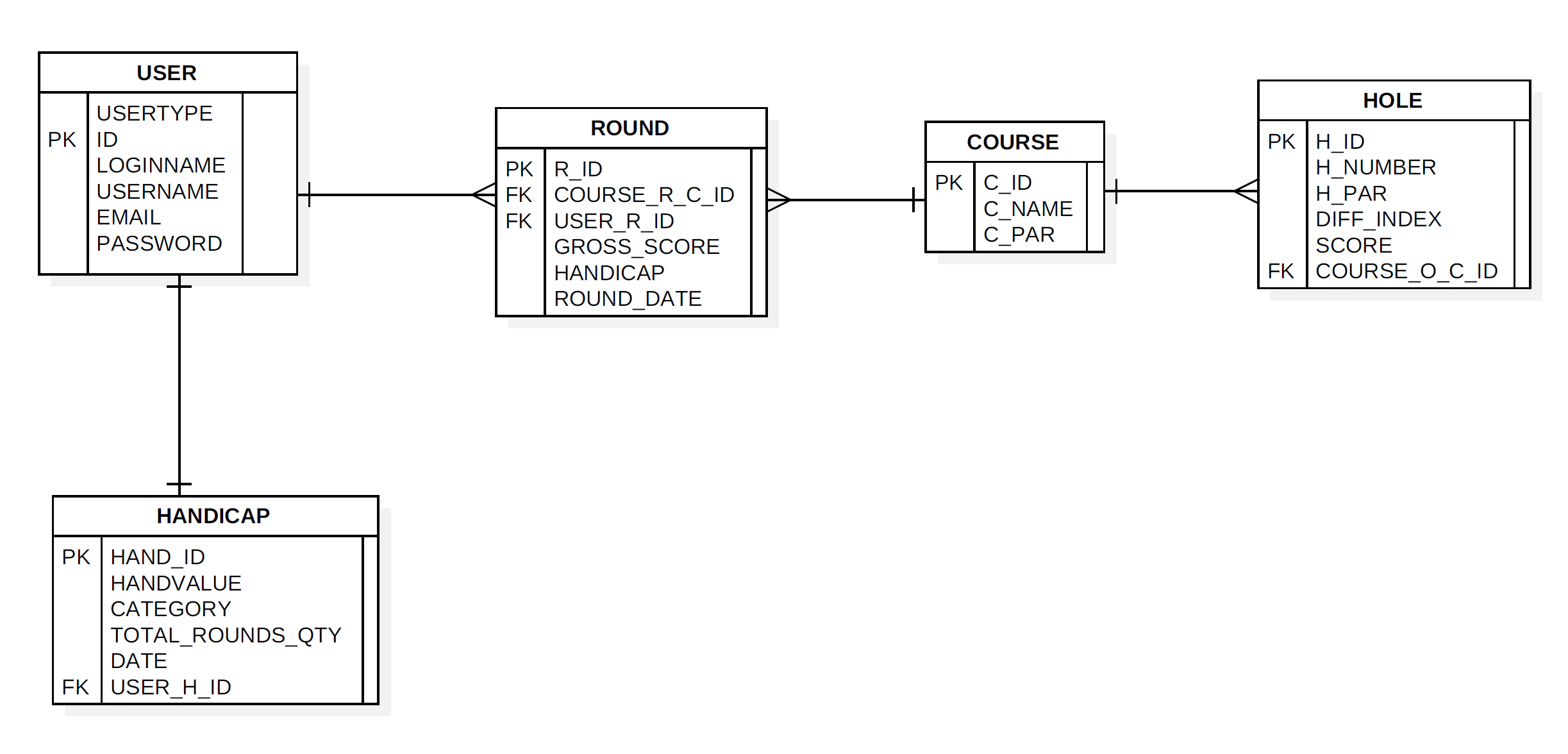
# Iteration 3

In this iteration reset Handicap, Handicap calculation was added and Form to save to the database. Submit round was altered and only if handicap exist case added. JUnit test was created to check the correctness of golf handicap calculation. SHA 256 cryptographic hash functions for passwords was added as well salt value created. Class diagram had major changes. User and Handicap classes become one to one relationships. Some changes in entity relationship diagram.

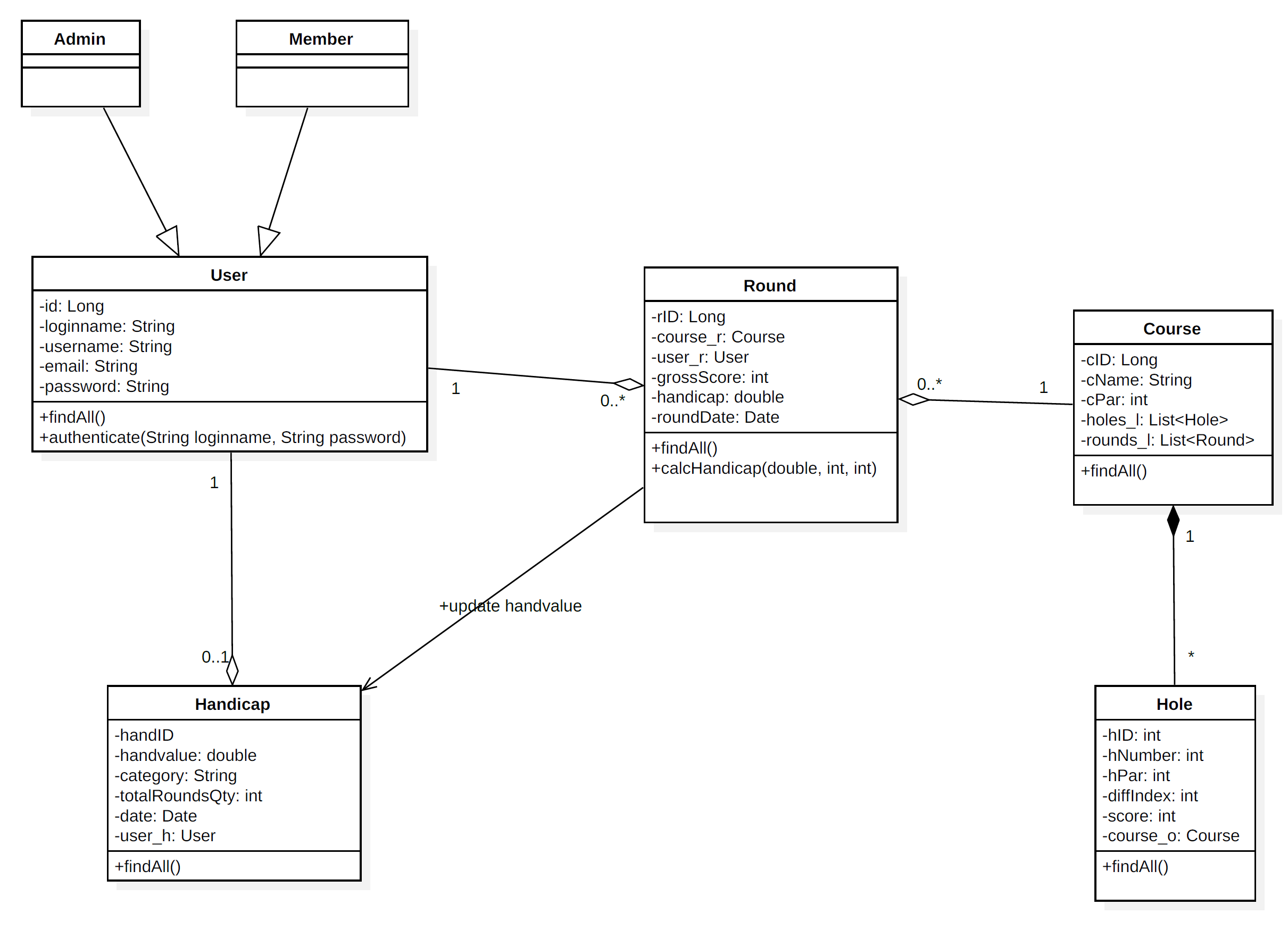
For the detailed development of the project please check GitHub repository: <https://github.com/yyyura/Project3>.

When creating a cryptographic hash function, a new value was added for password fields in 2.sql. The value was formed by adding a salt value to password and generated on <http://www.xorbin.com/tools/sha256-hash-calculator>.

## ERD - entity relationship diagram



## Class Diagram

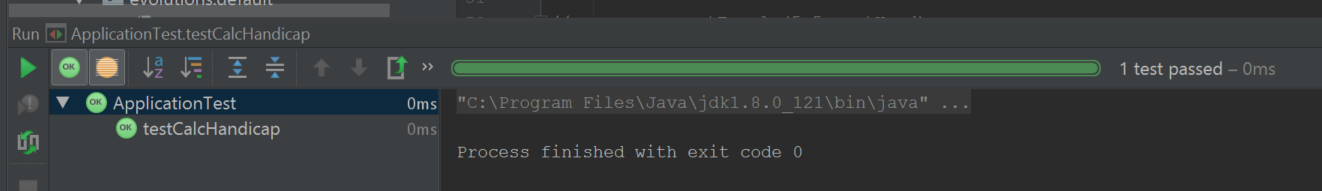


## Testing conducted

JUnit test was conducted to check the correctness of handicap calculation.

Initial values of initial handicap, total score and course par was inserted, and expecting value was added in assertion statement, delta value was added for double value in the range of +- 0.01.

@Test  
 public void testCalcHandicap(){  
 double outHand;  
 outHand = *calcHandicap*(6.3, 70, 72);  
 *assertEquals*(5.5, outHand, 0.01);  
 }



Test passed successfully.

# References

<https://www.playframework.com/documentation/1.4.x/main>

<https://www.playframework.com/documentation/1.4.x/images/diagrams_path>

https://upload.wikimedia.org/wikipedia/commons/thumb/a/a0/MVC-Process.svg/500px-MVC-Process.svg.png