Workout with Friends
Health Monitoring for Fitness Applications

16:332:567 – Software Engineering
Report #2

Mateus Santos Abdul Hassan
Kevin Kobilinski Daihou Wang
Brien Range Sujana Gangadharbatla

https://workoutwithfriends.wikispaces.com/

Fall 2012

Table of Contents

1. Interaction Diagrams

1.1 Sequence Diagrams

In this section, the system sequence diagrams of some most important use cases defined above will be illustrated.

1.1.1 UC-1: MonitorExercise

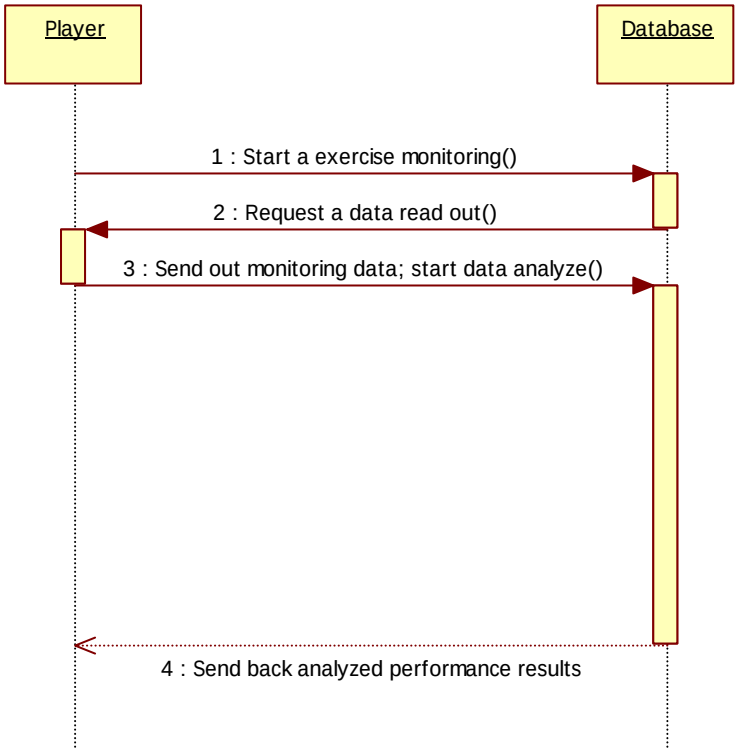


Figure 1-1: Sequence Diagram of UC-1: MonitorExercise

1.1.2 UC-2: ChallengeFriend

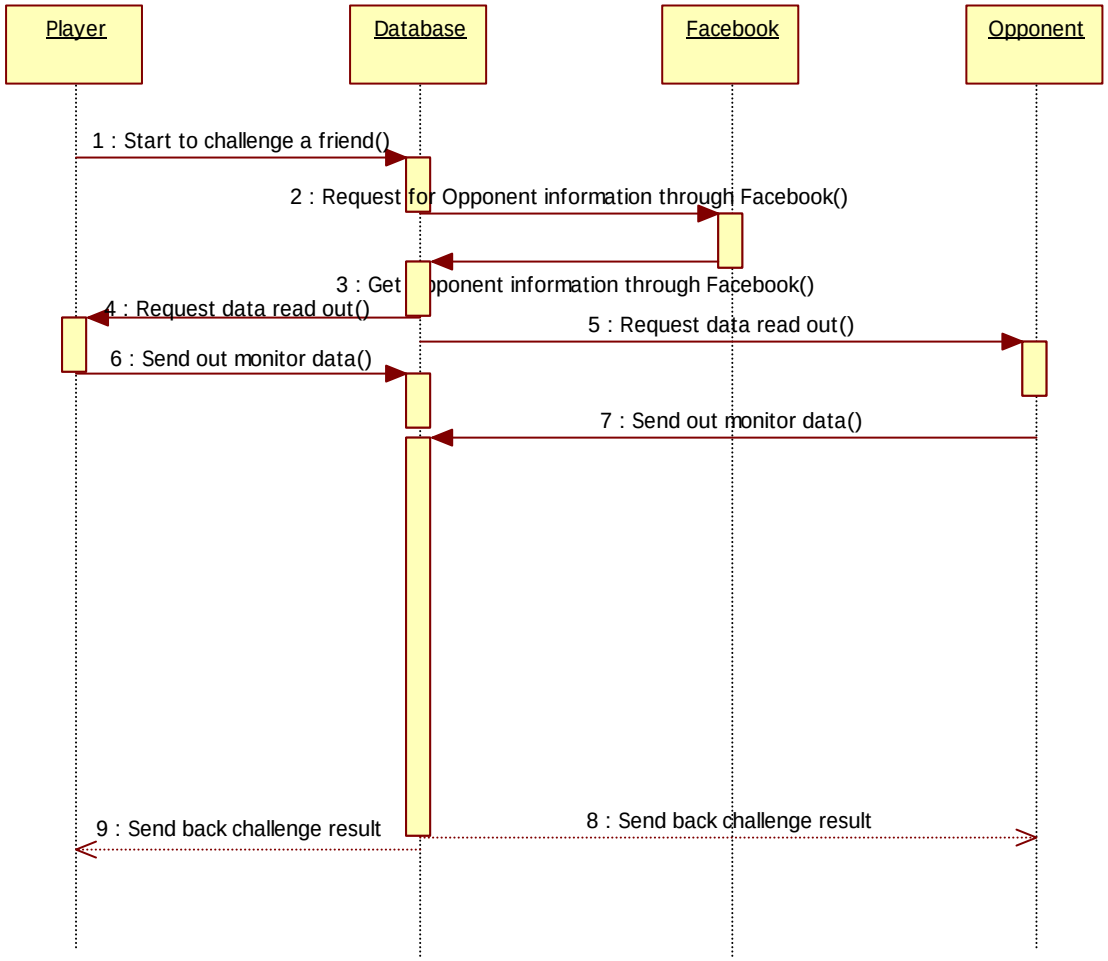


Figure 1-2: Sequence Diagram of UC-2: ChallengeFriend

1.1.3 UC-3: ChallengeOneself

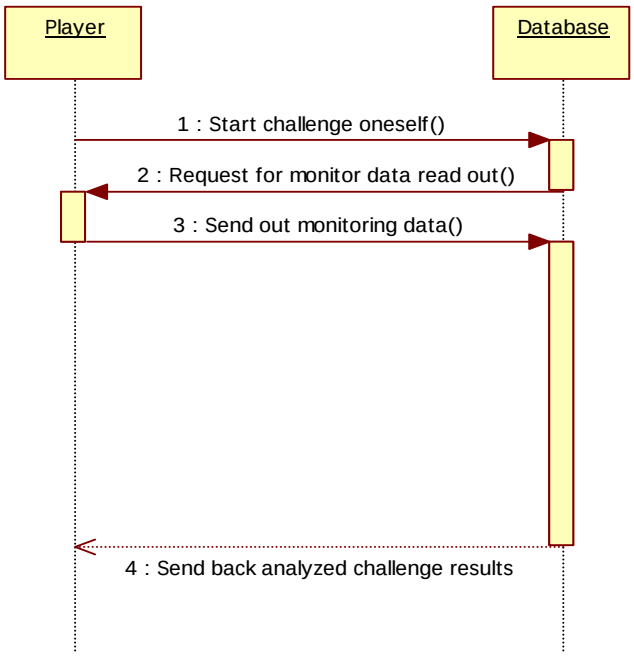


Figure 1-3: Sequence Diagram of UC-3: ChallengeOneself

1.1.4 UC-4: ViewRecord

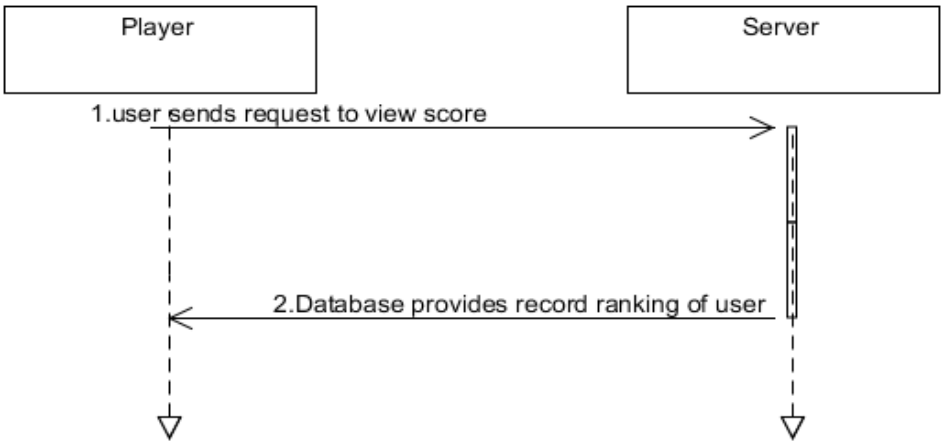


Figure 1-4: Sequence Diagram of UC-4:View Record

1.1.5 UC-5: Send Message

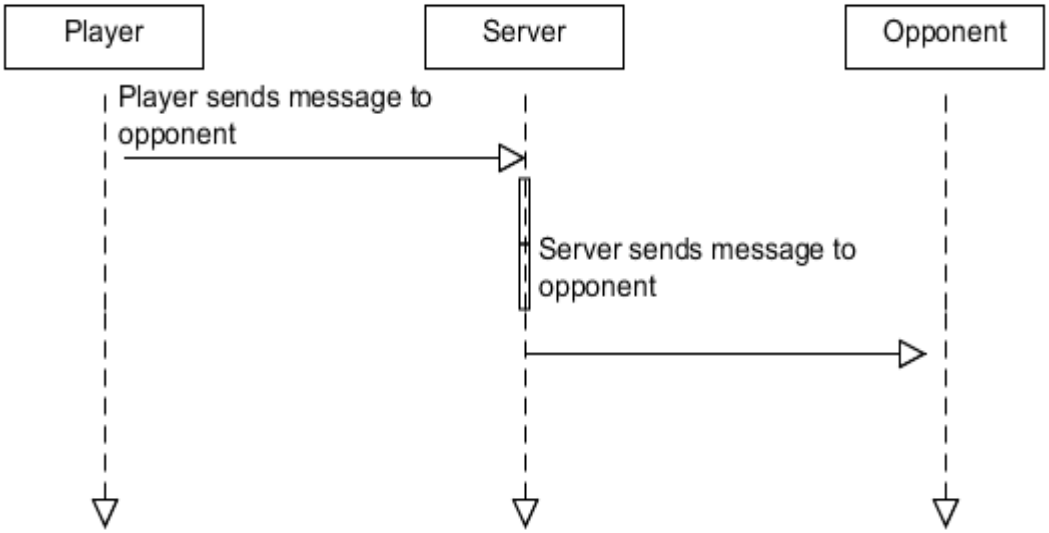


Figure 1-5: Sequence Diagram of UC-5:Send Message

1.1.6 UC -6:Register Profile

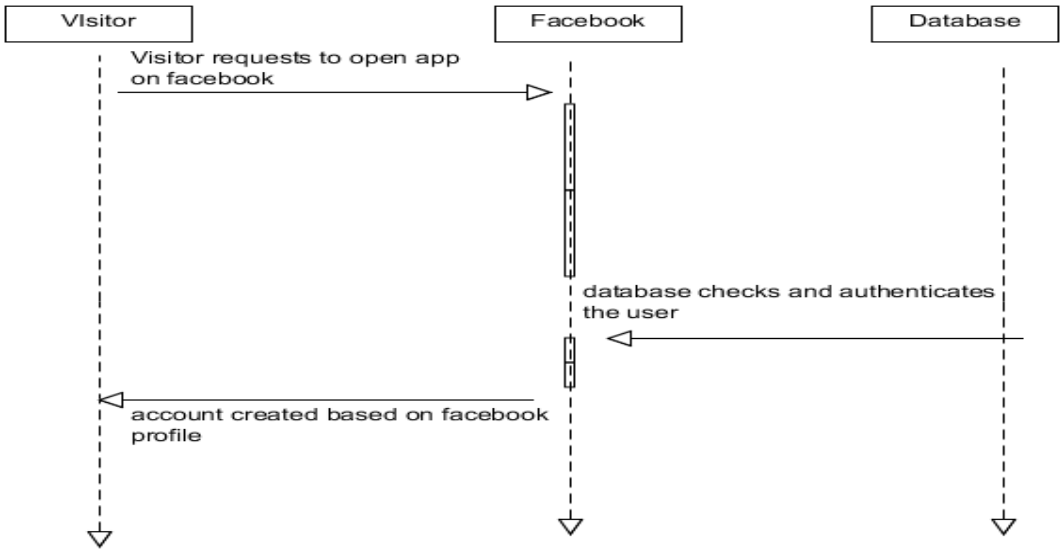


Figure 1-6: Sequence Diagram of UC-6: Register Profile

1.1.7 UC-7:Post Result

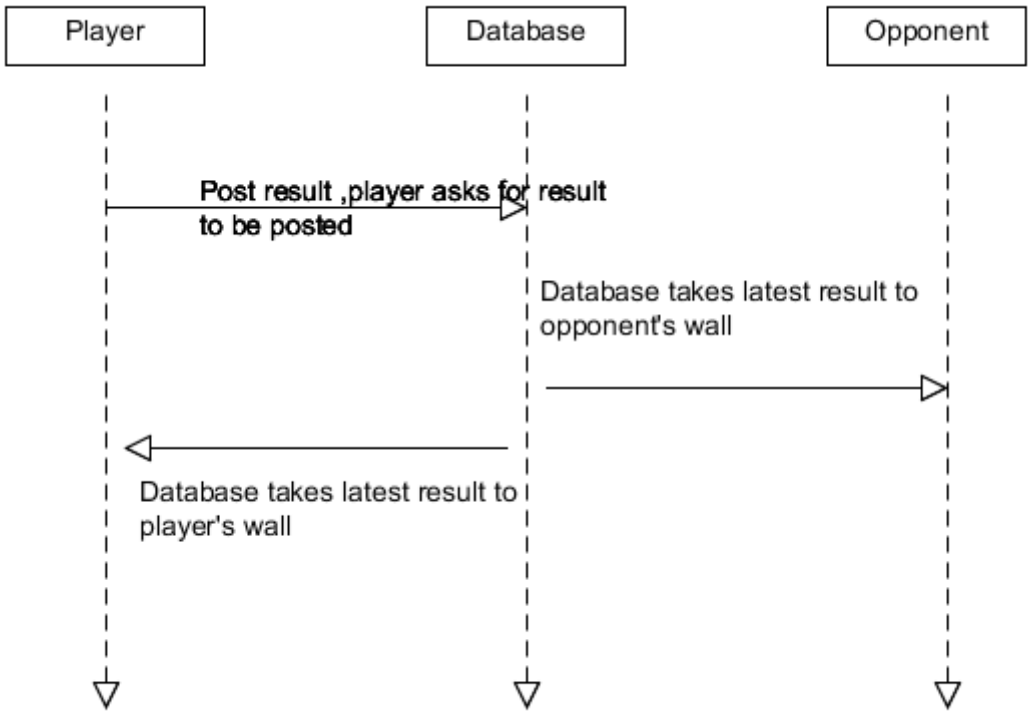


Figure 1-7: Sequence Diagram of UC-7:Post Result

1.1.8 UC-8:Search Friend

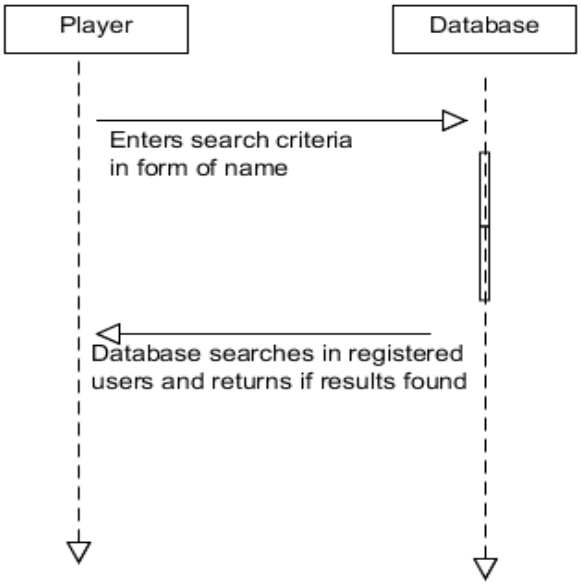


Figure 1-8: Sequence Diagram of UC-8:Search Friend

1.1.9 UC-10: ChallengeRandom

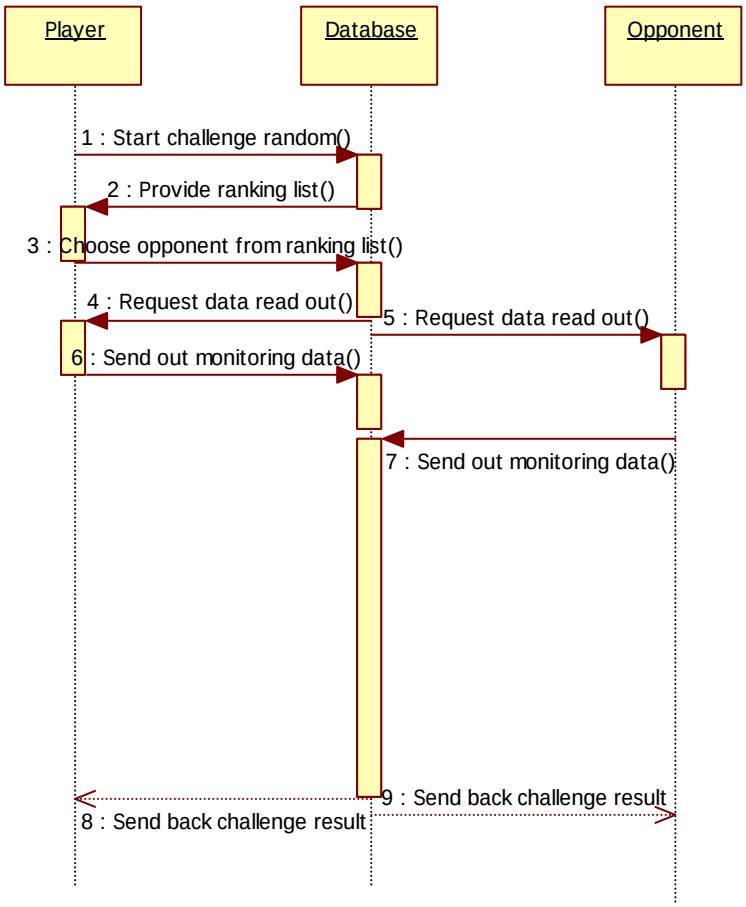


Figure 1-4: Sequence Diagram of UC-10: ChallengeRandom

2. Class Diagrams and Interface Specification

2.1 Class Diagrams

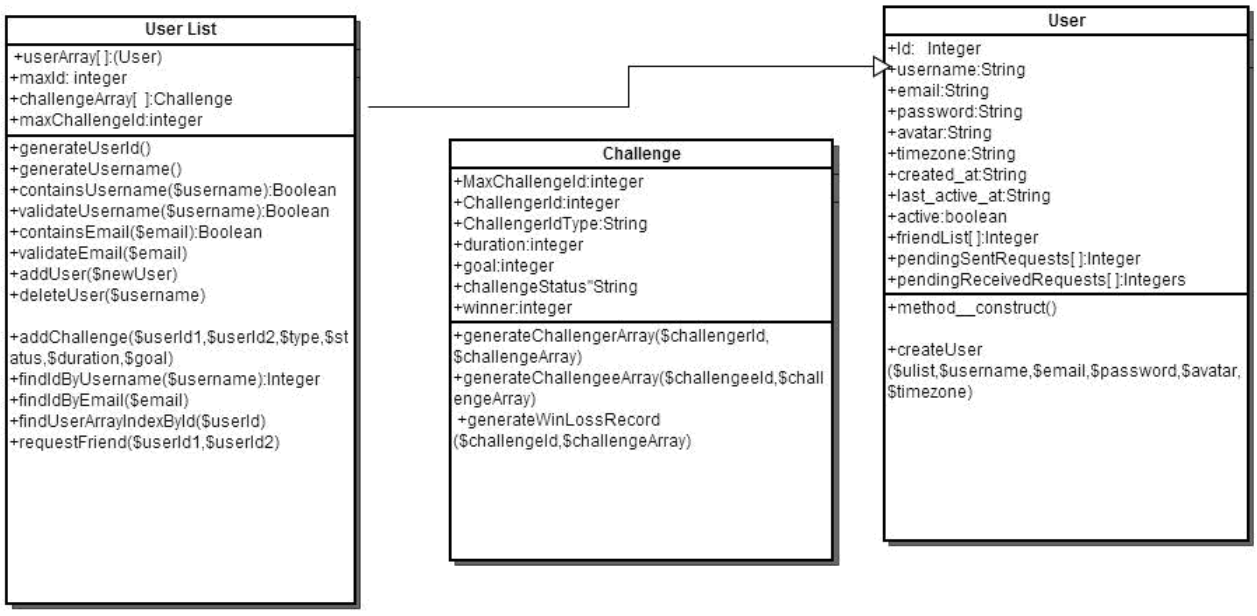


Figure 2-1: Class Diagram

2.1.1 Classes Specifications :

1.Challenge():
The Challenge () class manages the various challenges incorporated by all users .
The variables used in Challenge () and their brief descriptions :

public maxChallengeId:Integer largest challenge id
public challengerId:Integer challenger's id
public type:String The two ways you can play " ("STRENGTH", "ENDURANCE")
public duration: Integer Duration of workout in minutes

*The default value for duration ,if not entered by the user ,is 5 minutes .
public goal:integer Approximate amount of cals.the player wants to burn in calories.
*The default value for goal ,if not entered by the user ,is 5 cals .

public challengeStatus:String ("OPEN","CLOSED")
public winner:Integer User id of winner

Functions in Challenge():

function__construct(maxChallengeId,challengerId,challengeeId,type,status,duration,goal):
The explicitly defined constructor which defines all variables to the user's inputs.

public displayWinner(): A function which prints the id and name of the player who has won .

public generateChallengerArray(challengerId,challengeArray)
return type:array of challenges.
takes the user_list challenge array, and generates all the challenges from an id.

public generateChallengeeArray(challengeeId,challengeArray): for user terminated challenges
return type: array of challenges

function generateWinLossRecord(challengeId,challengeArray)
generates array of integers [W][L][T]

2.User():

The User class gives all details of a user when registering for "workout with friends ".Quite simply , it provides all the details which the user finds when they login and open their dashboard .

The variables used in User() and their brief descriptions :

public id: integer user's unique user id of exactly 10 digits
public username:String user's unique username (should start with alpha)
public email:string The email id of user
public password:string should be 8-20 chars ,should include one upper case,one lower case ,one special character and one digit .
public avatar:undefined (string)
public timezone:string
public created_at:string time when the user registered in "workout " .
public last_active_at:string time when the user last accesses "workout ".
public active:Boolean to check if user is presently active or not .
public friendList[ ]: userid collection of all user ids .

public pendingSentRequests[ ]: userid collection of userids whose requests who have been sent out by the user , but no action has been taken by reciepients .

public pendingReceivedRequests[ ]: userid collection of userids whose requests have not been taken any action by the user .

Functions in User():
method__construct():
Explicitly defined constructor to initialize instance of user, we need to use createUser to add information. This function only initializes arrays-friendslist[],pendingSentRequests[] and pendingRecieved Requests[].

createUser(ulist,username,email,password,avatar,timezone):
createUser() is used to initialize the basic parameters while registering .
$ulist is userlist object.

3.UserList():

The UserList() is a class , quite simply storing the details of “workout with friends” – including the validation functions, and the various operations which can be performed by the user .

The variables used in UserList() and their brief descriptions :

public userArray: (User)	array of user objects
public maxId:integer	largest issued userid???
public challengeArray: (Challenge)	array of challenge objects
public maxChallengeId:integer	largest issued challengeid????

Functions in UserList():
Public generateUserId():
This function generates a unique user id when the user registers into the site for the first time .(it works by incrementing the userid by one for every user)
function generateUsername():
The username given by the user is set in this function .
Public containsUsername(String):
return type:boolean
this function checks if username exists in the database of UserArray[ ] in the UserList() class .
Public validateUsername(String):
return type :Boolean
the function validates and checks if username is of valid type,The username must :contain only alpha numeric characters and start with an alpha character .

public containsEmail(String):
return type:Boolean
the function checks if emailid exists in the database of UserArray[ ] in the UserList() class .

public validateEmail(String):
return type:Boolean
the function validates and checks if email is of valid type.
Public addUser(User type):
adds user object to userArray[].
Public deleteUser(String-userName):
this function only deactivates an account , the user can come back and activate it any time later on.
public addChallenge($userId1,$userId2,$type,$status,$duration,$goal):
adds challenge to challengeArray[ ].
function findIdByUsername($username):
This function enables a user to find a friend by searching for the username .The function returns -1 of username doesn't exist, or is not found .
function findIdByEmail($email):

This function enables a user to find a friend by searching for the emailid .
function findUserArrayIndexById(userId):
This function enables a user to find a friend by searching for the userid .
function requestFriend($userId1,$userId2):
This function ,enables a user to invite a friend for a challenge .When the user sends a request to a friend ,the function adds userid2 to userId1's object's pending friend request array[].

2.2 Tractability Matrix

Use Case	Challenge()	User()	UserList()
Compare analyzed results between Player and its Opponent.	×		
Compare analyzed results between Player and his/ her former record.			×
Show challenge result.		×	
Send a message to its Opponent during a challenge.	×		
Allow Visitor to register through a Facebook account.		×	
Allow Player/ Opponent post the challenge result on Facebook wall.	×		
Allow Player to search for a friend through Facebook.			×
Allow Player to invite a friend to join the App through Facebook.			×
Provide a ranked list of all the Player' s results from Database, as well as the Player' s ranking in the list.			×
Allow Player to choose a random Player with certain performance rank to challenge.			×
Compare analyzed results between Player and another random Player with certain performance rank.	×		

The most difficult thing to do was to trace the actual classes we had obtained to the domain concepts we had recognized earlier and decided to implement :

Concepts implemented in Challenge ():

1. Compare analyzed results between Player and its Opponent.
2. Compare analyzed results between Player and another random Player with certain performance rank.
3. Allow Player/ Opponent post the challenge result on Facebook wall.

The main reason of a challenge class was to for the interaction between the user and any other friend with whom a challenge has been accepted .
The results of a challenge , is given once a challenge is over and a winner is clearly given .
Then , the analysis of the challenge is present , giveingthe user details of who has won , by how much and also the improvement since the beginning .

The option given to the user to post the challenge result on one's facebook wall could have been given to either the user() class or the challenge() class .However , it iseasy to seethat the result can be posted only once a particular challenge has been completed ,and the results are also present with the challenge() class .Hence , the concept was implemented in the challenge() class .

Concepts implemented in UserList():
Easily the most important class , the user() class helda lot of domain concepts and many were implemented using the UserList() class , as it held all the data of the user ().
1.Compare analyzed results between Player and his/ her former record.
2.Allow Player to search for a friend through Facebook.
3.Allow Player to invite a friend to join the App through Facebook.
4.Provide a ranked list of all the Player' s results from Database, as well as the Player' s ranking in the list.
5.Allow Player to choose a random Player with certain performance rank to challenge.

The UserList () has the data of all previous challenges of the user , and can hence show the improvement the user has been making .

3. System Architecture and System Design

3.1 Architectural Stytles

In order to simplify the design process, we reviewed various architectures and chose the most appropriate one for our system. Our project has the need for centralized data storage, and a scalable amount of users accessing and providing data. Thus, our project is a great candidate for the Client/Server model. In this model, the system is segregated into two applications, where the client makes requests to the server. Each user will have access to a client program, which offers a user interface to interact with the system. The server is a database with application logic represented as stored procedures. Our main motivation for this style was mainly driven by the centralized data feature. However, this architecture also offers the benefit of increased security as well, since all data is stored on the server as opposed to more vulnerable client machines.

3.2 Identifying Subsystems

The UML package diagram for Workout with Friends can be seen below.

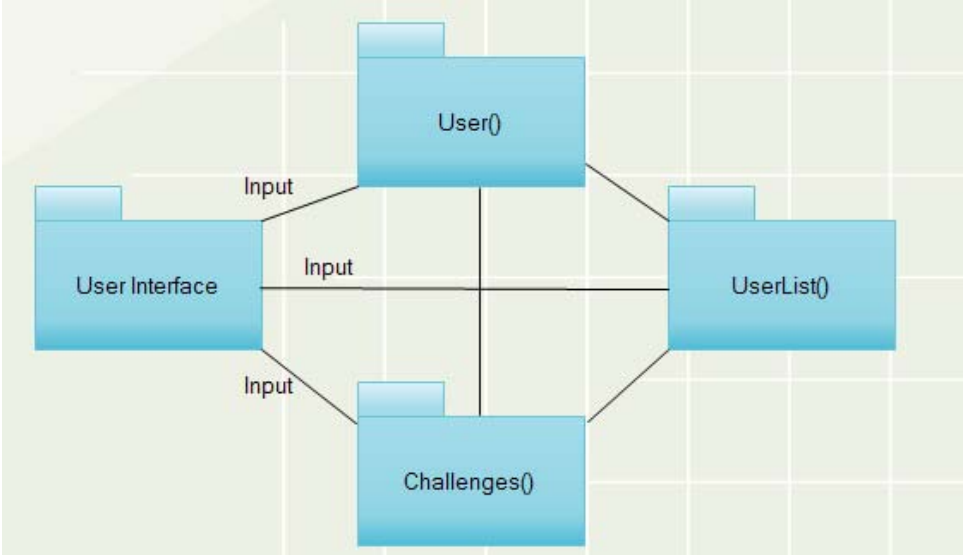


Figure 3-1: UML package diagram

The User Interface is the gateway into the system. It is through this that a user will create his profile, edit said profile, create challenges, view results, and more. The list of users is maintained through the user list package, which is used to create individual user classes. It is also provides the userId information to the challenge package, for match information between two challengers. The user class can be accessed through the user interface to edit a particular user's personal information as necessary; it can also access the challenge class when creating and maintaining existing challenges. The challenge class is used to create challenges between two users, and must be able to access the initiators information as well as the UserList to inform the challenged user that there is a workout request waiting for them.

3.3 Mapping Subsystems to Hardware

As our system follows the client/server architecture, it is designed to hold a database on a host server (one PC), and handle any number of clients.

3.4 Persistent Data Storage

Our server must store all data that the system may use. This includes all user profile information, existing matches, and match data. Our system accomplishes this by way of relational database.

3.5 Network Protocol

Our system has a central server, which is hosted through Apache HTTP. Requests to access information in the databases, which are MySQL, are done through php.

3.6 Global Control Flow

Workout with Friends is an event driven system. This means that users can generate actions in different order. Once a User has created his profile, he can, at any time, challenge another user to a match. The challenged user may accept or decline, and either user can complete the workout at their convenience. Once both users have completed their workout challenge, they can view the results of the particular challenge at any time. In addition, that data can also be observed as part of a broader data.

3.7 Hardware Rerquirement

Workout with Friends requires the use of a MOTO Heart Rate monitor. In addition, the client PC must have an internet connection to be able to access the server.

4. Algorithms and Data Structures

4.1 Algorithms

4.1.1. Basic Scoring Algorithm

Workout with Friends, is a challenge based application, and the challenge result is based solely on the score of each work out. The workout intensity score is computed based on the weighted score of three different inputs: workout time spent in HRZ 5, HRZ 4, and calories burned.

For the simplistic model:
1. Score1 : Time (in seconds) spent in heart rate zone 5 (90-100% of target max HR) x factor of 10 (Weighting = 40%)
- Example: 1:45 min = 105 seconds x 10 = 1050
- Score1 = 1050 x 40% = 420 points

2. Score2 : Number of calories burned – (Weighting = 25%)
- Example: 850 calories
- Score2 = 850 x 25% = 212.5 points

3. Score3 : Time (in seconds) spent in heart rate zone 4 (80-89% of target max HR) – (Weighting = 35%)
- Example: 21:45 min = 1305 seconds
- Score3 = 1305 x 35% = 456.75 points

4. Final Score = Score1 + Score2 + Score3
- Example: Final Score = 420 + 212.5 + 456.75 = 1089.25 points

Based on this model, when calories and HRZ 4 and 5 are known, the calculation is straightforward and the process to determine a winner is simple.

In the embedded Facebook web-app case, where no specific calories burned and HRZ are known, we will use a formula to calculate the calories burned. The starting conditions (known variables) are: gender, weight, age, exercise duration and average heart rate of the workout. The equations derived by LR Keytel, JH Goedecke, TD Noakes, H Hiiloskorpi, R Laukkanen, L van der Merwe, and EV Lambert for their study titled "Prediction of energy expenditure from heart rate monitoring during submaximal exercise" are shown below:

⇒ Male: $C = ((-55.0969 + (0.6309 \times HR) + (0.1988 \times W) + (0.2017 \times A))/4.184) \times 60 \times T$
⇒ Female: $C = ((-20.4022 + (0.4472 \times HR) - (0.1263 \times W) + (0.074 \times A))/4.184) \times 60 \times T$
Where:
C = Calories burned
HR = Heart rate (in beats/minute)
W = Weight (in kilograms)
A = Age (in years)
T = Exercise duration time (in hours)

4.1.2. Improved Scoring Algorithms

Strength

The strength workouts are based on research of mathematical models that normalize powerlifting abilities based on weight and repetition. The first underlying formula is the Wilks Formula,

developed by Robert Wilks, is used for scoring by the International Powerlifting Federation (IPF). The formula to determine the Wilks Coefficient is as follows (from Wikipedia):

$$Coeff = \frac{500}{a + b \cdot x + c \cdot x^2 + d \cdot x^3 + e \cdot x^4 + f \cdot x^5}$$

x is the body weight of the lifter in kilograms

Coefficients for *men* are:
a=-216.0475144
b=16.2606339
c=-0.002388645
d=-0.00113732
e=7.01863E-06
f=-1.291E-08

Coefficients for *women* are:
a=594.31747775582
b=-27.23842536447
c=0.82112226871
d=-0.00930733913
e=0.00004731582
f=-0.00000009054

The second formula is the Wathan 1 Rep max formula, used to estimate a person's 1RM based on repetitions at a given weight (from Wikipedia).

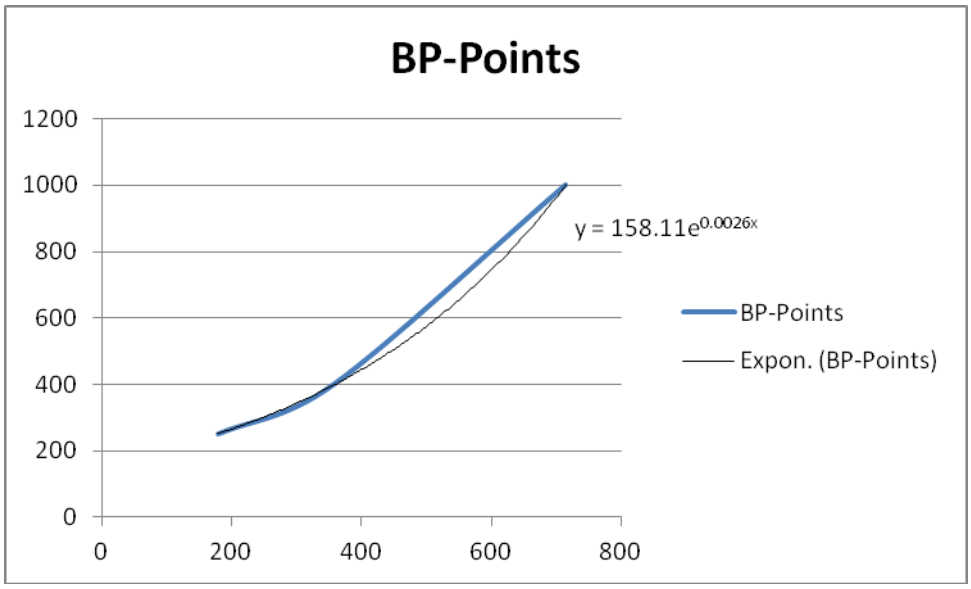
$$1RM = \frac{100 \cdot w}{48.8 + 53.8 \cdot e^{-0.075 \cdot r}}$$

Where w is weight (dimensions doesn't matter) and r is reps

The normalized lifted weight is determined as follows:

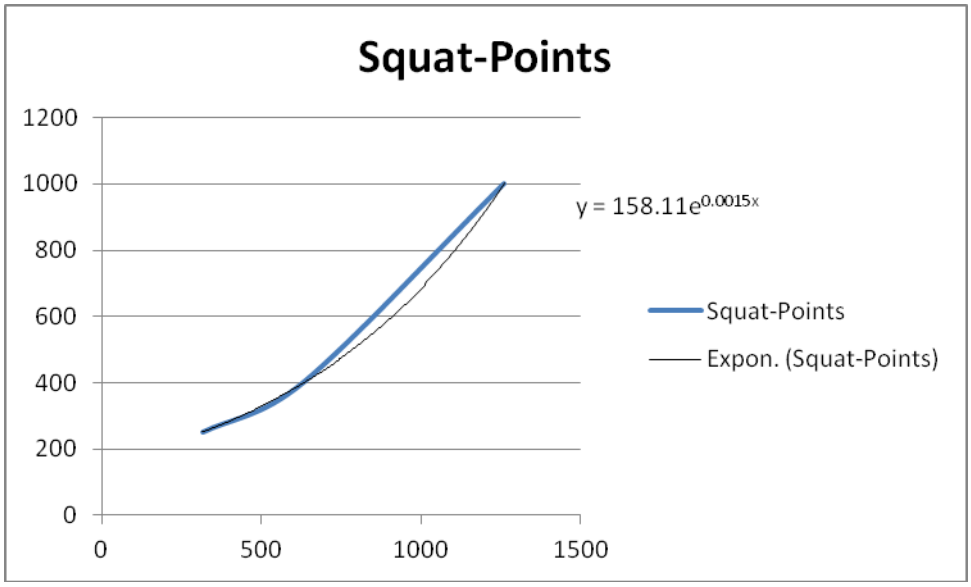
$$x(w, r) = Coeff(0.453592 \cdot w) \cdot \frac{100 \cdot w}{48.8 + 53.8 \cdot e^{-0.075 \cdot r}}$$

(Bench Press): The x derived in the previous equation is used in the equation in the following chart to determine the score for the challenge:

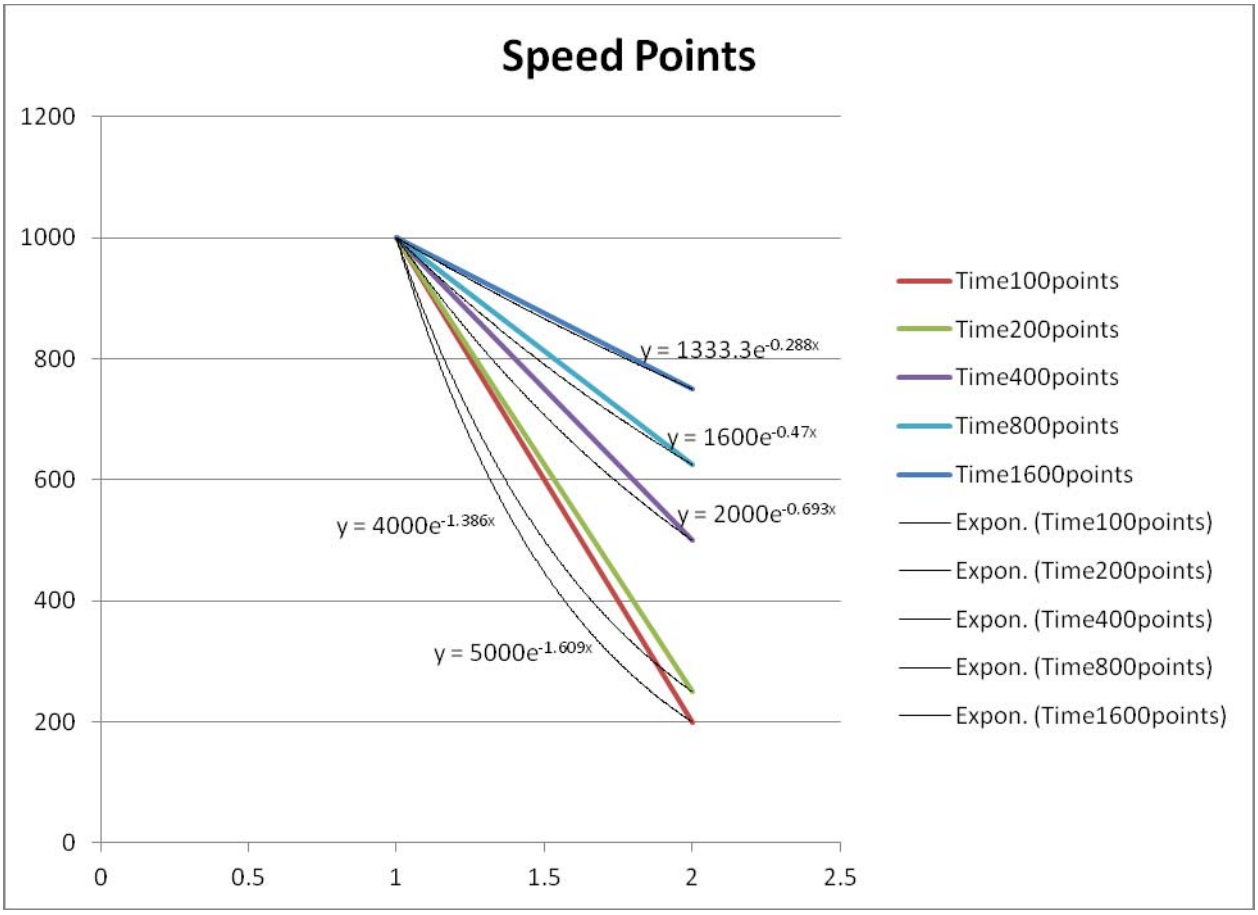


This equation was designed to score the maximum 1000 points to the person that can perform the maximum bench press rep performed by the record holder, Scot Mendelson, at 715lb. Using an exponential model allows the challenger to be rewarded even for smaller incremental gains during the plateau period. The following shows the squat equivalent:

(Squat):

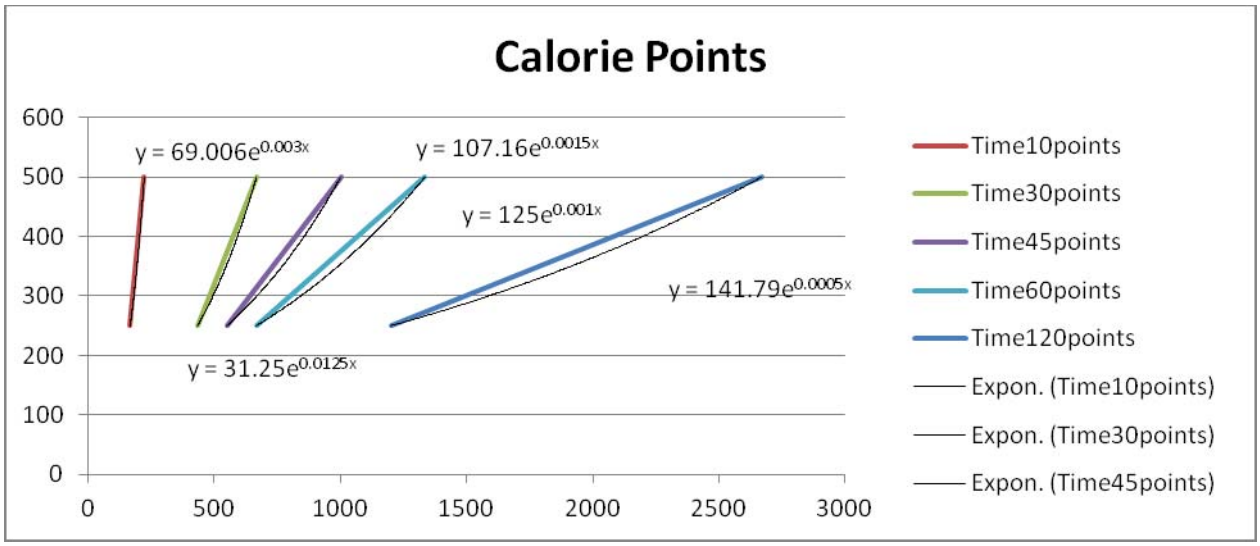


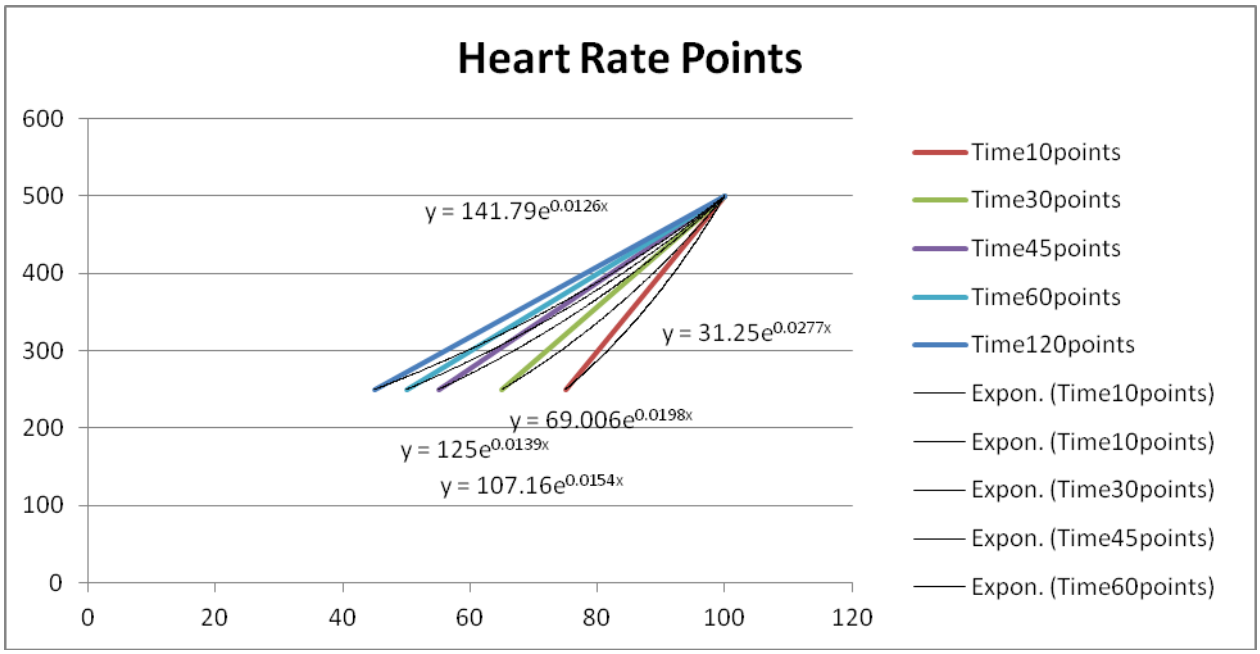
(Speed): The following chart shows the exponential algorithm used to determine the points in Speed games. Each line depicts a different distance chosen by the challenger/player. The choices are 100m, 200m, 400m, 800m, and 1600m.



These graphs are based on current world best times in these events, and getting these scores would yield the full 1000 points. The x-axis is how many times greater that player's distance is off from the world best in that event, or User_Time/World_Record_Time. This scheme rewards those players who do comparatively worse on a longer distance event than the short distance counterparts, because the difference between average and elite in short distance and long distance is not linear.

(Endurance): The endurance challenge mathematical model is arguably the most complicated, as there is no real metric to which we can give the title "world best". As such, this model calls a 100% heart rate "perfect" and calls the calories burned at 100% heart rate "perfect." Thus, it is more likely that the scores in these challenges will be lower in practice. Below shows the charts depicting the point allocation of both heart rate and calories, and their points are added together to give the overall score.





Again, the challenger has a finite amount of options to choose from: 10minutes,30,45,60, and 120. As with the Speed challenges, it was deemed unfair to reward people more for being able to keep a higher heart rate on shorter events, and the model was adjusted accordingly, by making it exponential.

What makes this model more complicated is the method used to determine average heart rate. In the previous model, points were divvied based on how long one was in a particular zone. This method was open to cheating, where people can essentially play for any length of time and win without having to push himself or herself. The new method takes the maximum moving average for a period the length of the challenge chosen. This method would require the user to push himself for the entire duration of the challenge, as working out longer will more likely result in lower moving averages and not higher ones.

4.2 Data Structures

The server of Work out with friends, has two main databases built, one for user login and administration, and the other for store user performances. Both of them are build using MySQL on the server disk.

4.2.1. User Login Information store

The login information was stored as table entries in a separate table.

DATABASE	TABLE	Values	Description
dbweb	user	Vchar(10) uid	User id used when login.
		Vchar(10) key	Password.
		Vchar(20) email	Email.
		Char(1) gender	M/F
		Int(11) age	Age.
		Char(1) access	U- user; A- system administrator.

4.2.2. User performance store

The user performance database was built with the following ideas:

- Construct one data for each use; during registration time.
- In each use there' re tables stores friends info and performance information for challenge.

DATABASE	**TABLE**	**Values**	**Description**
Uid(uer id used to login)	friends	Int(11) id	Index number starts from 1.
		Vchar(10) uid	User id used when login.
	perfor	Int(11) id	Index number starts from 1.
		Double tz4	Time spent in time zone 4.
		Double tz5	Time spent in time zone 5.
		Double eng	Energy consumption of the workout.
		Double scoreSum	Total score calculated from the math model.

5. User Interface Design and Implementation

5.1 User Interface Design

The main concept of user design interface was to provide for an easy , clear picture of wha the user can do on the website application .From the beginning , we have kept it clear that the user should be comfortable with the on screen appearance ,the working of the application.Hence forth , we had identified that the main points in keeping up with the user interface design were(given in the first report) :

1. Landing page: on Facebook Market App.

2.e – Login with Facebook account.

2.d – Registration: for new and 1st time users (some info should also be available in About Screen)
- 2.d.1 - part 1 - game description and how-to-play
- 2.d.2 - part 2 - disclaimers & permissions
- 2.d.3 - part 3 - user info (name, user name, gender, age, resting heart rate, weight, sports preferences, share stores (y/n), share location (y/n), alerts..)

2.a - Home Screen major building blocks:
1) high level status bar: (number of challenges: completed, waiting for opponent, won; weekly points, delta of wk-to-wk) - button takes users to "Performance Tracker screen" (2.j)
2) About
3) Start a new challenge
4) Open challenges
5) Challenge History
6) Help

2.b - Start a new challenge screen
- 2.b.1 - Search for facebook friends, random local opponent, search by username, select previous workout (self-trainer mode)
- 2.b.2 - (from select facebook friends) - populates list of FB friends, allow to select 1 or more opponents
- 2.b.3 - Pre-challenge screen: shows selected friends, asks users to input: challenge nickname, workout duration, options (web or mobile), message opponents

2.c - Open Challenge: function residing in Home Screen, lists open challenges waiting for opponents turn

2.f – Help: FAQ's, report user, send feedback to dev team, upgrade SW.

2.g – About: users can find more info (links) on HR, training and tips. Users can also access how-to-play and game description available at registration, Help (takes users to help screen) and Account (takes users to Account Management screen)

2.i – Account: users can review and/or edit account attributes such as: nickname, weight, resting HR, alerts, sharing options, age

2.j - Performance tracker: shows numerical values for challenges completed, winning %, margin won/lost, calories counter (toggle monthly vs weekly vs all-time view). Also users may select to graph such attributes

The main editions were made when we actually started implementing the project , and found that some of the requirements were making the project cumbersome , or difficult to implement with . We also found new ideas to implement , that would make the user interface much more easy .

The main thing was to start implementing the website ,before jumping to the facebook application .In the website , we recognized that the most important part of easy user interface was to have a "dashboard" or home screen with all the data the user would require as as soon as they logged in .The Dashboard consists of :
1.Home
2.User details
3.Challenges
4.Help
5.Find Friends
6.Contact us

Below is a brief description of each :
1.Home : the home screen reiterates to the dashboard , where the user can find all details pertaining to his/her use of the application .

2.User Details :
The user details give all details of the user, which had been inputted during the time of registration .These details can be edited at any point of time .

3.Challenges :
The challenges block provides for all the challenges the user has been in yet . It will include the scores ,who has won the challenge ,who has initiated the challenge ,and also which challenges are yet to be completed .

4.Help:
The help button on the dashboard will provide one with basic details on how to edit one's information , how to send out a challenge ,and how to invite new friends .

5.Find Friends :
The find friends button can help the user access any friend / or any member through their respective email .
The user can send invites to anyone to register in " workout with friends " ,and further on we will also make a way to invite friends present in one's facebook account .

Main Editions in User Interface Implementation:

TO FIND FRIENDS :
At the beginning the idea was to incorporate the idea of facebook friends , and keep the application within the limitations of facebook .However ,as we started working on the concept ,we decided that it would be better not to limit the application , and we started working on a website ,which we would later integrate onto a application .
Now , a user can not only add friends from one's facebook account , but also add anyone with an email id .

5.2 User Interface Implementation

As a web based application, the user interface is built using PHP/ HTML/JavaScript, and following are some main page from the application.

Figure 5-1: The home page

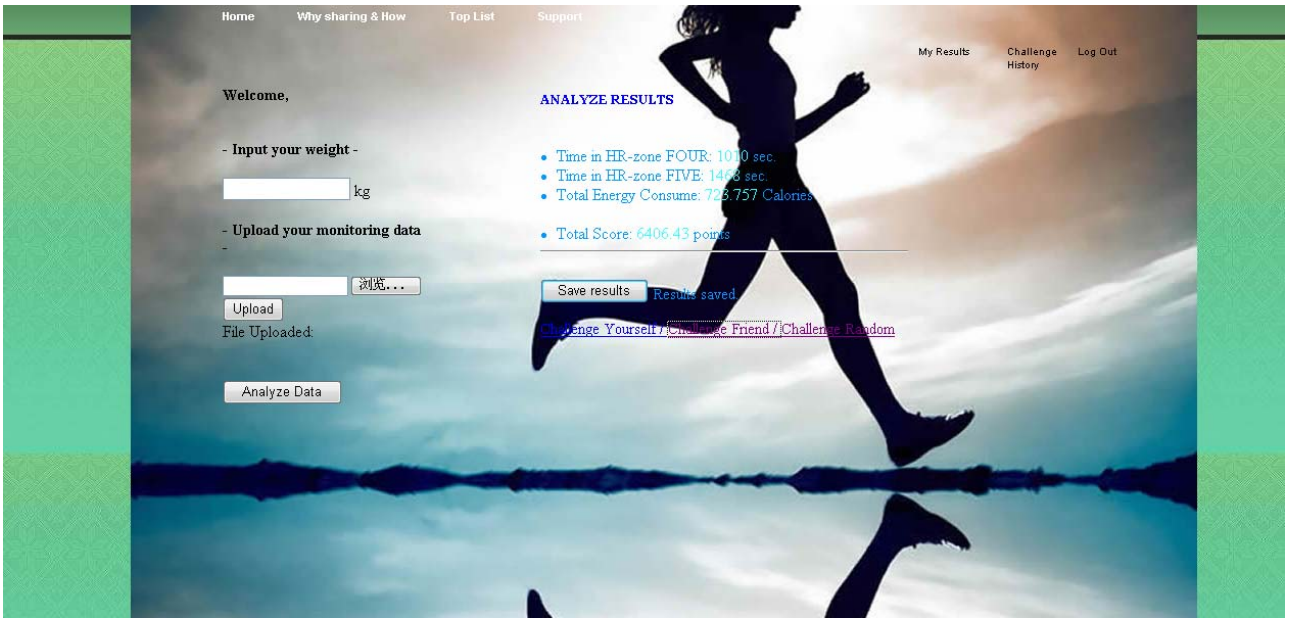


Figure 5-2: The data analyze page

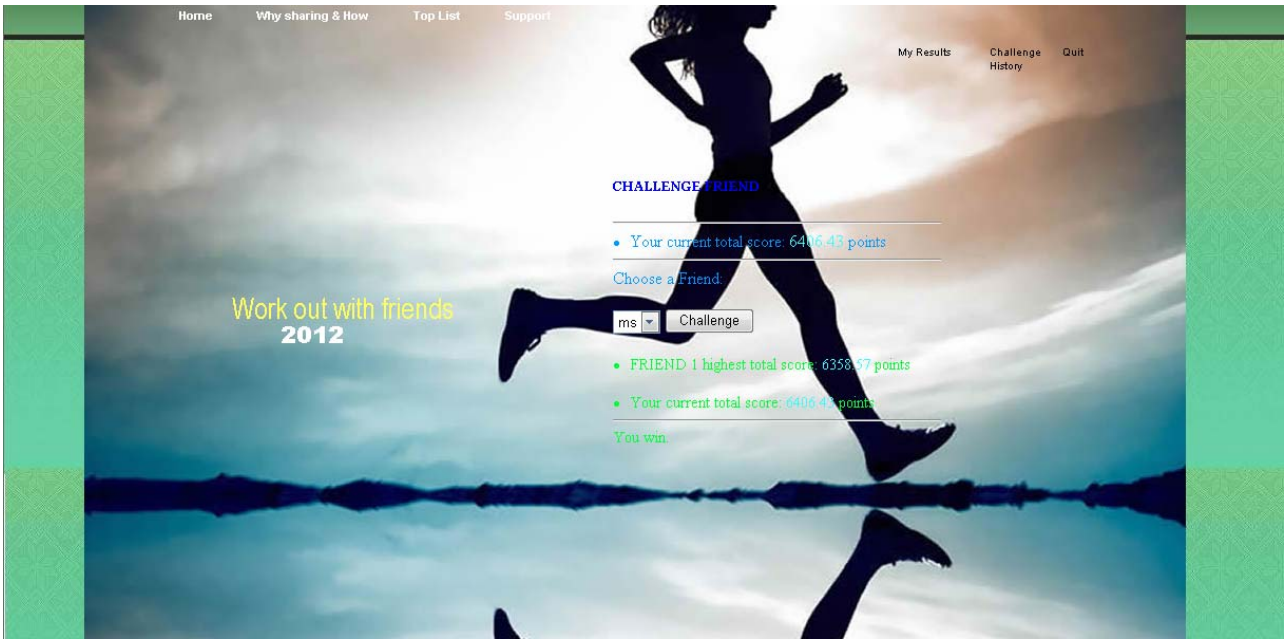


Figure 5-3: The friend challenge page

6. Design of Tests

6.1 System Framework Tests

Goal: To test the basic function of the whole application, the aim is to ensure all the non-critical functions, including user registration, log in, system administration, works smoothly in the system.

Testing approach: Register new users in the system and log in.

Results: All the parts worked well in the system.

6.2 Functional Units Tests

Goal: To test the running condition of all the central parts of the application: the data analysis and challenge parts.

6.2.1. Data analysis

Testing approach: Upload a real workout monitor data and analyze the data, save the analysis results.

Results: All the parts worked well in the system.

6.2.2. Challenge

Testing approach: Try to upload a different set of workout results through a different user account, analyze the data and start a challenge with the user from last section, and compare the results.

Results: All the parts worked well in the system.

7. Project Management and Plan of Work

7.1 Project Progresses and Plan of Work

Task \ Week		40	41	42	43	44	45	46	47	48	49	50
Website Maintenance												
Web Interface Design												
Mobile Interface Design												
Database Structure Design												
Algorithm Design												
Algorithm Test												
Debugging												
System Test												

From the above table, the group has finish the first step of the project: building a basic functional application. And the following focus would be on improving the further analysis of the data, and developing a mobile interface for the application.

7.2 Contribution Breakdown(Report 2)

Task	Mateus	Kevin	Brien	Abdul	Daihou	Sujana
1) Interaction Diagrams					X	X
2) Class Diagram And Interface Specification						
a. Class Diagram						X
b. Data Types and Operation Signatures						X
c. Tracebility Matrix						X
3) System Architecture and System Design						
a. Architectural Styles		X				
b. Identifying Subsystems		X				
c. Mapping Systems to Hardware		X				X
d. Persistance Data storage		X				X
e. Network Protocol		X				X
f. Global Control Flow		X				X
g. Hareware Requirement						
4) Algorithms and Data Structures						
a. Algorithms			X	X	X	
b. Data Structures					X	
5) User Interface And Design and implementation						
a. UI Design						X
b. UI Implementation					X	
6) Design Of Tests					X	
7) Project Management					X	
8) References		X			X	
Editing		X			X	X

8. References

1. Software Architecture Styles
http://msdn.microsoft.com/en-us/library/ee658117.aspx
2.